Combining Video and Sequential Statistical Relational Techniques to Monitor Card Games

Laura Antanas LAURA.ANTANAS@CS.KULEUVEN.BE
Bernd Gutmann BERND.GUTMANN@CS.KULEUVEN.BE
Ingo Thon INGO.THON@CS.KULEUVEN.BE
Kristian Kersting KRISTIAN.KERSTING@IAIS.FRAUNHOFER.DE
Luc De Raedt LUC.DERAEDT@CS.KULEUVEN.BE
Katholieke Universiteit Leuven, Department Computerwetenschappen, Celestijnenlaan 200a, 3001 Heverlee, Belgium

Fraunhofer IAIS, Department of Knowledge Discovery, Schloß Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

Games are a multi-billion dollar industry and a driving force behind technology. The key to make computer games more interesting is to create intelligent artificial game agents. A first step is teaching them the protocols to play a game. To the best of our knowledge, most systems which train AI agents are used in virtual environments. In this work we train a computer system in a real-world environment by video streams. First, we demonstrate a way to bridge the gap between low-level video data and high-level symbolic data. Second, using the high-level, yet noisy data, we show that state-of-the-art statistical relational learning systems are able to capture underlying concepts in video streams. We evaluate the selected methods on the task of detecting fraudulent behavior in card games.

1. Introduction

Games are a multi-billion dollar industry and a driving force behind technology. Computer games were one of the main reasons for the spread of home computers in the 1980s. Also, since the advent of the world wide web, online games have gained popularity and created new challenges for the developers. At the other pole, the gambling industry generates a large volume of revenues and plays a non-negligible role from an economic point of view. This pushes technical innovation in several directions – an important one is smart visual surveillance systems.

Appearing in *Proceedings of the ICML Workshop on Machine Learning and Games*. Copyright 2010 by the author(s)/owner(s).

In this work, we are interested in monitoring people playing games in real-world environments by sensorial observation. This can be useful in several ways. One is to create artificial game agents that learn by interacting with humans in a natural environment – for instance playing card games. As a first step, agents need to learn the rules of the game by observing humans playing it. Currently, most computer-controlled agents are trained in virtual environments, where it is assumed that the state of objects is directly available to the agent. Another useful task is fraud detection in real-world casinos based on sensor information. Not only does fraudulent behavior lower the game experience for players, it can also cause serious economic threats.

The difficulty of the tasks is due to several aspects. Firstly, it depends, besides the challenges raised by sensor information, on the richness of the game protocols. Games can be arbitrarily complex due to the number of actions and objects or stochastic aspects. Still, common characteristics between them are their sequential behavior and inherent structure – given by relations between objects, which can elegantly be represented using *relational sequences*. While complex scenes are best described by high-level, logical representations, video data consists out of noisy low-level numerical values. Bridging the gap between the two types of representation is complex and is the first problem to solve. While this question has been studied before (Tran & Davis, 2008; Needham et al., 2005), there does not yet exist a generally accepted framework that is flexible enough to extract rich symbolic representations from video streams in a general setting.

Secondly, one needs to learn models of dynamic scenes based on logical representations in order to reason about different aspects of the scene. Previous work has approached learning from sensor data aspects of games – such as their strategies – in a purely relational setting (Needham et al., 2008; Needham et al., 2005; Bennett & Magee, 2007; Fern, 2005). Efficient reasoning about real-world activities requires logical representations, however due to the inherent noise in video streams purely logical rules will not suffice. *Statistical* relational learning (SRL) techniques (De Raedt, 2008) combine hard logical information with noisy uncertain knowledge. This makes them a good fit for our task. Different SRL systems exist that can handle logical sequences (Kersting et al., 2008; Thon et al., 2008).

This paper significantly extends the earlier work (Antanas et al., 2009)[1] by (1) a new problem setting – namely detecting fraud in card games – and (2) the use of discriminative models – namely TildeCRF. We applied the selected techniques on the popular card game Uno.

The rest of this paper is organized as follows. In Section 2 we formulate the problem settings and show how to obtain logical descriptions from video streams. In Section 3 we discuss the sequential learning systems we used. Before concluding we present our experiments in Section 4.

2. From video streams of games to relational representations

Uno is a card game for two to seven players. The game objective is to be the first to get rid of all the cards in one's hand to a discard pile. The Uno deck (Fig. 1(a)) consists of 'common' cards of 4 colors with ranks from `0` to `9` in each color. There are 'action' cards in each color (e.g `skip`) and special action cards or jokers (e.g. `wild`). At any point in time only one exposed card is on the table. Each turn, a player may play a card from its hand that matches either the color or number of the top exposed card, or a (special) action card.

We approach the subtask of translating videos of Uno games into *relational sequences*, therefore bridging the gap between low-level data and high-level representations. Uno games can be naturally described using sequences of played cards. One major difference in representing sequences is given by the complexity of the underlying language – namely the individual sequence elements. Uno games can be described by sequences of propositional identifiers where each identifier represents a played card (as in Example 2.1).

Example 2.1 *A sequence of moves in an Uno game:* `2 − red`, `1 − red`, `red − draw2`, `wild`, `blue − 6`, `blue − skip`, `wild4`, ...

These sequences are atomic and applying propositional models to them requires one to explicitly enumerate all possible states in the game (all possible combinations number-colors). For complex problems propositional representations can lead to a combinatorial explosion in the number of parameters. Instead, we use *relational representations* (De Raedt, 2008) – more precisely ground atoms – to describe sequences of elements (as in Example 2.2). This allows one to generalize over similar situations.

[1]presented as poster at the 19th International Conference on Inductive Logic Programming (ILP 2009)

(a) Standard Uno cards

(b) Cards with markers *(red, one)*; *(one, green)*

Figure 1. The Uno game domain

A logical *atom* is an expression of the form $\mathtt{p}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ where `p` is a *predicate symbol* with arity `n` and the $\mathtt{t}_i$ are *terms*. We assume a functor-free language, hence terms are only built from constants and variables. *Constants* are denoted in lower case and *variables* in upper case. *Ground* expressions do not contain variables and ground atoms are called *facts*. In our examples the symbols `card` and `joker` are predicates, while `blue`, `red`, `2`, etc are constants. `card(red, 2)` is a predicate which does not contain any variables. Common cards are represented as `card(red, 2)`, and action cards as either `card(red, draw2)` (colored action card) or `joker(wild4)` (special action card). Each relational atom in the sequence represents the top exposed card on the discard pile.

Example 2.2 *The same sequence of moves in a relational form:*

`card(red, 2)`, `card(red, 1)`, `card(red, draw2)`, `joker(wild)`, `card(blue, 6)`, `card(blue, skip)`, `joker(wild4)`, ...

We propose a simple and efficient method to obtain relational sequences from video streams by making use of *tags* for object recognition. We associate with each (previously trained) tag a symbol that represents the object that we want to detect. As an example, a common card contains two tags: one for *color* and one for *number* (action cards have special symbols – e.g. `skip`). In Fig. 1(b), two different cards with tags are shown together with their associated symbols. We use the ARToolKit framework (Kato et al., 2000) to generate and recognize markers. It uses 2D planar tags and has been employed in augmented reality applications.

The introduction of tags for object detection avoids the difficult task of applying feature extraction and image processing. Instead of doing complex object recognition, we can analyze scenes by looking for known markers. This enables us to focus on the machine learning task. Still, the approach is realistic in that similar results could be obtained by applying more advanced state-of-the-art results in vision (Bay et al., 2008). In addition, the use of tags offers a general framework for symbol detection across different games. With each tag one can associate any symbol, therefore the same set of markers can be used to represent different symbols, depending on the cards of the game (e.g. a tag with associated symbol *one* for Uno can be used to represent symbol *ace* for Poker). In previous work (Needham et al., 2005; Needham et al., 2008) similar relational sequences were obtained from video and audio data by clustering extracted video features. However, the disadvantages of this approach are that feature clustering can give much redundancy and objects can easily be misclassified.

In order to obtain the data in the format shown in Example 2.2 from video streams, a pre-processing phase from tags to logical atoms is required, as described in the following steps.

Step 1: Using ARToolKit, we first obtain a description of each video frame in terms of tags:

$\mathtt{tag}(1,2)$, $\mathtt{tag}(1,\mathtt{red})$, $\ldots$, $\mathtt{tag}(102,2)$, $\mathtt{tag}(102,\mathtt{red})$, $\mathtt{tag}(103,1)$, $\mathtt{tag}(103,\mathtt{red})$, $\ldots$, $\mathtt{tag}(179,1)$, $\mathtt{tag}(179,\mathtt{red})$, $\mathtt{tag}(180,\mathtt{red})$, $\ldots$, $\mathtt{tag}(186,\mathtt{red})$, $\mathtt{tag}(187,1)$, $\mathtt{tag}(187,\mathtt{red})$, $\ldots$, $\mathtt{tag}(205,4)$, $\mathtt{tag}(206,4)$, $\mathtt{tag}(207,\mathtt{draw2})$, $\mathtt{tag}(207,\mathtt{red})$.

The atom $\mathtt{tag}(1,2)$ – for instance – corresponds to observing the tag 2 in video frame 1, similarly $\mathtt{tag}(1,\mathtt{red})$ stands for observing tag $\mathtt{red}$ in frame 1.

Step 2: We compress this sequence by merging tags with the same frame number into one atom and replacing sets of identical consecutive atoms with one atom. The compressed variant of the sequence above is:

$\mathtt{card}(\mathtt{red},2,102)$, $\mathtt{card}(\mathtt{red},1,77)$, $\mathtt{joker}(\mathtt{red},7)$, $\mathtt{card}(\mathtt{red},1,18)$, $\mathtt{joker}(4,2)$, $\mathtt{card}(\mathtt{red},\mathtt{draw2},36)$.

The atom $\mathtt{card}(\mathtt{red},2,102)$ has as arguments the color, the number (or special action) and the number of identical video frames, respectively. The atom $\mathtt{joker}(\mathtt{wild},7)$ has as arguments the joker symbol and the number of identical video frames.

Step 3: We filter very short sequences with length $\mathtt{S}_1 < 5$ and replace the states where the symbols are senseless with the tags $\mathtt{unknown}$ for jokers, $\mathtt{unknown}_\mathtt{c}$ for colors and $\mathtt{unknown}_\mathtt{n}$ for numbers[2]. For instance, $\mathtt{joker}(4,2)$ does not make sense as jokers cannot be numbers, therefore it is replaced by $\mathtt{joker}(\mathtt{unknown})$. Also, the ground atom $\mathtt{card}(\mathtt{yellow},\mathtt{green})$ is substituted by $\mathtt{card}(\mathtt{unknown}_\mathtt{c},\mathtt{unknown}_\mathtt{n})$ since a card cannot contain two colors. The resulting relational sequence is:

$\mathtt{card}(\mathtt{red},2)$, $\mathtt{card}(\mathtt{red},1)$, $\mathtt{joker}(\mathtt{unknown})$, $\mathtt{card}(\mathtt{red},1)$, $\mathtt{card}(\mathtt{red},\mathtt{draw2})$.

After pre-processing, the noise-free sequence from Example 2.2 is in fact the one in Example 2.3.

Example 2.3 *'Noisy' relational sequence – the same as in Example 2.2) – obtained from video streams:*

$\mathtt{card}(\mathtt{red},2)$, $\mathtt{card}(\mathtt{red},1)$, $\mathtt{joker}(\mathtt{unknown})$, $\mathtt{card}(\mathtt{red},1)$, $\mathtt{card}(\mathtt{red},\mathtt{draw2})$, $\mathtt{joker}(\mathtt{wild})$, $\mathtt{joker}(\mathtt{unknown})$, $\mathtt{card}(\mathtt{blue},6)$, $\mathtt{card}(\mathtt{yellow},6)$, $\mathtt{card}(\mathtt{unknown}_\mathtt{c},\mathtt{unknown}_\mathtt{n})$, $\mathtt{card}(\mathtt{yellow},2)$, $\mathtt{card}(\mathtt{blue},\mathtt{skip})$, $\mathtt{joker}(\mathtt{wild4})$, $\ldots$

Tags simplify the recognition task, yet there is uncertainty in the recognition process, due to lighting conditions and occlusion. ARToolKit deals with this by providing confidence values for detected markers. In this work we only consider the markers detected with a confidence factor above 0.5. Although this removes a considerable amount of noise, ARToolKit still introduces non-negligible inter-marker confusion and false positive rates. Added to the temporary occlusion of markers when cards are manipulated, this translates into a significant source of noise (as shown in Example 2.3). We approach the sequential, relational and noisy aspects of this kind of data by employing sequential SRL techniques.

3. Employing statistical relational techniques for relational video sequences

There are several learning tasks that can be identified when learning from sequences. In this work we focus on learning to detect fraudulent game sequences based on video streams. This is done by considering the task of *sequence classification*, that is to label sequences of Uno game moves as *legal* or *illegal*. Because our domain is best represented using *sequences of relational atoms* and even though there exist several SRL techniques for relational sequences (Kersting et al., 2008), in this work we employ r-grams (Landwehr & De Raedt, 2007) and Tilde-CRF (Gutmann & Kersting, 2006). These two models are representatives of very different classes of learning algorithms. The former is trained using a generative learner, whereas the latter employes a discriminative one.

3.1. R-grams: n-grams for relational sequences

The r-gram model lifts propositional n-grams (Manning & Schütze, 1999) to logical representations. It estimates the

[2] ARToolKit introduces inter-marker confusion (e.g. it may recognize $\mathtt{green}$ instead of the correct tag 6).

probability of a sequence $X = \langle x_1 \ldots x_m \rangle$ as smoothed Markov chains, a finite mixture of Markov distributions of different orders. A Markov chain of order $n-1$ estimates the probability of X as follows

$$\begin{aligned} P(X) &= \prod_{i=1}^{m} P\big(x_i | x_{i-n+1} \ldots x_{i-1}\big) \\ &= \prod_{i=1}^{m} \frac{C(x_{i-n+1} \ldots x_i)}{C(x_{i-n+1} \ldots x_{i-1})} \end{aligned}$$

where the conditional probabilities are estimated from a set S of training sequences using 'gram' counts: $C(x_1 \ldots x_k)$ is the number of times $\langle x_1 \ldots x_k \rangle$ appeared as a subsequence in any $X \in S$. To avoid the overfitting of the model for a large gram order n, models of different orders can be combined and consequently the conditional probabilities are defined as

$$P(x_i | x_{i-n+1} \ldots x_{i-1}) = \sum_{k=1}^{n} \alpha_k P_k\big(x_i | x_{i-k+1} \ldots x_{i-1}\big)$$

where $\alpha_1, \ldots, \alpha_n$ are positive weights with $\sum_{k=1}^{n} \alpha_k = 1$ and P_k is the conditional distribution defined by a k-order gram. An r-gram model is obtained by generalizing sequence elements x_i to first-order logical atoms, such as $x_i = \texttt{card}(\texttt{blue}, 2)$. They exploit the relational structure by considering relational generalizations of grams and estimating conditional probabilities for non-ground atoms. The generalized gram $\texttt{card}(\texttt{blue}, \texttt{X})$ – for instance – stands for an arbitrary blue card and the probability $P(\texttt{card}(\texttt{blue}, \texttt{X}) \mid \texttt{card}(\texttt{blue}, \texttt{Y}))$ is the probability that a blue card is followed by another blue card. This way, by relational generalization they upgrade n-grams with smoothed probability estimates (compared to modeling sequences by considering all data at the ground level). Similar to n-grams, the r-gram model can consider grams of different orders. In r-grams the conditional distribution of a relational sequence $X = \langle x_1 \ldots x_m \rangle$ is defined as

$$P(x_i | x_{i-n+1} \ldots x_{i-1}) = \sum_{r \in R} \alpha_r P_r(x_i | x_{i-k+1} \ldots x_{i-1})$$

where the x_i are logical atoms, R is the set of all generalized relational grams, P_r is the conditional distribution defined by a particular gram. Learning an r-gram model from data involves choosing the set of relational grams, estimating their corresponding probabilities (cf. Figure 2) and define weights for every r-gram in the selected set.

Sequence classification is performed by building an r-gram model R_C for each class C and labeling unseen sequences X with the class that maximizes $P_C(X) \cdot P(C)$, where $P(C)$ is the prior probability of the class C. More details can be found in (Landwehr & De Raedt, 2007; Kersting et al., 2008).

$$\left.\begin{array}{ll} 0.40 & \texttt{card(C,B)} \\ 0.51 & \texttt{card(A,C)} \\ 0.08 & \texttt{joker(C)} \\ 0.01 & \texttt{card(C,D)} \end{array}\right\} \longleftarrow \texttt{card(A,B)}$$

Figure 2. Rules extracted from a relational bigram model for the class *legal*. The first two rules show that the next card should have either the same color A with probability $P_1 = 0.4$, or the same number B with probability $P_2 = 0.51$, while the third shows that a joker can be played next with a probability $P_3 = 0.08$. The last rule models noise.

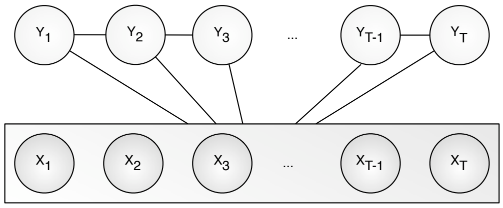

Figure 3. Graphical representation of a linear-chain CRF. The nodes labeled with Y_i represent the output sequence, and the X_i's represent the input. As one can see, every node element depends on the complete input.

3.2. TildeCRF: CRFs for relational sequences

Conditional Random Fields (Lafferty et al., 2001) are a state-of-the art model for sequence labeling and tagging. They define a probability distribution $P(Y|X)$ as follows

$$\frac{1}{Z(X)} \exp \sum\nolimits_{t=1}^{m} F(y_{t-1}, y_t, X)$$

where $X = \langle x_1 x_2 \ldots x_n \rangle$ is the observed sequence, $Y = \langle y_1 \ldots y_n \rangle$ is the sequence of labels assigned to the observed sequence, $F(y_{t-1}, y_t, X)$ is a potential function, and $Z(X)$ is a normalization factor over all possible state sequences $Y \in \mathcal{Y}$ defined as

$$\sum_{Y \in \mathcal{Y}} \exp \sum\nolimits_{t=1}^{m} F(y_{t-1}, y_t, X)$$

A *potential function* is a real-valued function that captures the degree to which the assignment y_t to the output variable fits the transition from y_{t-1} and X. Due to the global normalization by $Z(X)$, each position t influences the overall probability. In the Uno domain, X is the sequence of cards played in one game and Y labels every move either as *legal* or *illegal*.

TildeCRF[3] (Gutmann & Kersting, 2006) is a relation extension of CRFs where the potential function $F(y_{t-1}, y_t, X)$ is represented as sums relational regression trees (cf. Figure 4). TildeCRF employs Gradient Tree Boosting (Friedman, 2001; Dietterich et al., 2004) to learn the potential

[3] `http://www-kd.iai.uni-bonn.de/index.php?page=software_details&id=17`

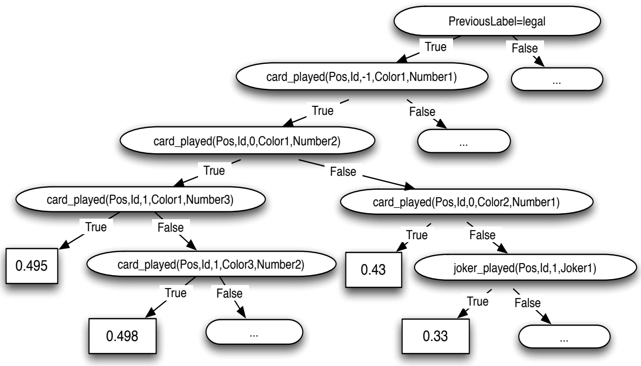

Figure 4. A learned regression tree by TildeCRF representing the gradient in the first iteration. Internal nodes represent tests – queries in Prolog form – and leafs represent the output.

function. This is a functional gradient search, where one approximates the true gradient by a regression tree. While it is not possible to determine the gradient analytically, the value of the gradient can be calculated for every position in the training data. By evaluating the gradient for all positions and fitting a relation regression tree to this data set, one obtains an implicit representation of the true gradient. The potential after the i-th iteration is thus the sum of i regression trees $F(y_{t-1}, y_t, X) = \Delta_1 + \ldots + \Delta_i$.

There are several ways for getting a classifier from a trained CRF. We can predict the output sequence Y with the highest probability: $H(X) = \arg\max_Y P(Y|X)$. The Viterbi algorithm (Rabiner, 1989) can be used for this. Another option is to predict every atom y_t in the output sequence individually. This makes sense when we want to maximize the number of correctly tagged input atoms

$$H_t(X) = \arg\max_{k \in K} P(y_t = k|X).$$

There are several ways to use a CRF for sequence classification, i.e. to predict a single label for the entire sequence X. The easiest one – similar to r-grams – is to calculate the likelihood $P(Y|X)$ for the label sequence $Y = \langle ccc \ldots c \rangle$ where c is a possible label. The predicted class is the one with the highest likelihood. We refer to this as global label rule. Another possibility is to use majority vote. That is, one first predicts $H(X)$. Next, one counts the number of times each class atom was predicted, i.e.

$$\mathit{count}(c, Y) := |\{i \in \{1, \ldots, T\} \mid y_i = c\}| \,.$$

Then, the sequence X is assigned to class c with probability $P(c|X) = T^{-1} \cdot \mathit{count}(c, H(X))$. For binary classification problems, one can also predict the class as positive, if there is at least one position labeled as positive. We refer to this as single rule mode. Majority vote and rule mode can be combined with forward backward and Viterbi respectively.

4. Experiments

We set up experiments to answer the following questions:

(Q1) Does a generative statistical relational model, such as r-grams, perform well when dealing with limited real-world video data?

(Q2) Can a discriminative statistical relational model, such as TildeCRF, be used for sequence classification tasks even when it is trained as a model for tagging?

Experimental data was collected from video sequences of people playing the game with the special tagged cards, using a subset of the Uno cards (without the doubles). The camera was mounted on the ceiling so that it captured the playing deck at any moment. The illegal games were played by 2 players – a fair player and a fraudulent one, while the legal ones by 2 honest players. In order to make sure that the fraudulent player is performing illegal moves during the game, the real players reproduced simulated games with the special tagged cards. For experiments a set of 50 complete Uno games were recorded as example sequences. Each of the examples are labeled with one general label (*legal* or *illegal*) per sequence.

We used stratified 5-fold cross validation. The folds were built by randomly assigning the examples to folds such that the number of legal and illegal examples are evenly distributed. For both legal and illegal examples we randomly sampled from examples with high and low level of noise and for each of the these, in the case of illegal examples, we sampled from the distribution of the low and high number of incorrect moves per sequence, while in the case of legal examples from the distribution of the low and high sequence lengths. The absence of such a stratification can give an uneven distribution of noisy, low-level illegal examples and noise free, high-level illegal examples, which results in a standard deviation often higher than 10%.

For r-grams we trained two models, one for each of the classes *legal* and *illegal*. We used both models to classify a sequence as described in Section 3.1. For TildeCRF we considered the classifiers described in Section 3.2.

The experimental results are shown in Table 1. Most classification methods perform well when used with TildeCRF, except FB rule which give the poorest results and also a high standard deviation. Viterbi majority gives the best performance. Both systems perform well on the sequence classification task with respect to the predicted accuracy, answering positively to the questions **Q1** and **Q2**. Discriminative models perform slightly better than generative ones.

Model	Setting	Accuracy
r-grams	Length 2	0.84 ± 0.12
	Length 3	$\mathbf{0.94 \pm 0.05}$
	Length 4	$\mathbf{0.94 \pm 0.05}$
	Length 5	0.92 ± 0.04
TildeCRF	Vi majority	$\mathbf{0.96 \pm 0.06}$
	Vi rule	0.92 ± 0.07
	FB majority	$\mathbf{0.96 \pm 0.06}$
	FB rule	0.87 ± 0.09
	Global label	0.90 ± 0.07

Table 1. Classification results on the Uno data set. The bold notation shows the best accuracy scores.

However, due to the size of the data set, the result is not statistically significant. The advantage of generative models is that the learned models are easier to understand.

5. Conclusions

This work is a first step to solve the fraud detection problem in games from video data. We present a method to obtain relational descriptions from video streams using markers, bridging the gap between low-level video information and high-level representations. We successfully employed r-gram and TildeCRF models with relational descriptions of sequences to show that they perform well to detect illegal game sequences in Uno. As future work we plan to address the detection of more complex and less obvious fraudulent behaviors, games with richer protocols as application and the use of multiple and different types of sensors. Another interesting problem is fraud detection in real-time, after each move is played.

Acknowledgements

Bernd Gutmann is supported by the Research Foundation Flanders (FWO). Kristian Kersting was supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM.

References

Antanas, L.-A., Thon, I., van Otterlo, M., Landwehr, N., & De Raedt, L. (2009). Probabilistic logical sequence learning for video. *ILP*.

Bay, H., Ess, A., Tuytelaars, T., & Gool, L. J. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, *110*, 346–359.

Bennett, A., & Magee, D. R. (2007). Learning sets of sub-models for spatio-temporal prediction. *SGAI Conf.* (pp. 123–136).

De Raedt, L. (2008). *Logical and relational learning*. Springer.

Dietterich, T. G., Ashenfelter, A., & Bulatov, Y. (2004). Training conditional random fields via gradient tree boosting. *ICML*.

Fern, A. (2005). A simple-transition model for relational sequences. *IJCAI-05* (pp. 696–701).

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Gutmann, B., & Kersting, K. (2006). Tildecrf: Conditional random fields for logical sequences. *ECML* (pp. 174–185).

Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., & Tachibana, K. (2000). Virtual object manipulation on a table-top AR environment. *International Symposium on Augmented Reality* (p. 111).

Kersting, K., De Raedt, L., Gutmann, B., Karwath, A., & Landwehr, N. (2008). Relational sequence learning. In *Probabilistic inductive logic programming*, vol. 4911/2008, 28–55. Springer.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML* (pp. 282–289).

Landwehr, N., & De Raedt, L. (2007). r-Grams: Relational grams. *IJCAI* (pp. 907–912).

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Massachusetts: The MIT Press.

Needham, C. J., Santos, P. E., & Magee, D. R. (2008). Inductive learning spatial attention. *Controle and Automation*, *19*, 316–326.

Needham, C. J., Santos, P. E., Magee, D. R., Devin, V., Hogg, D. C., & Cohn, A. G. (2005). Protocols from perceptual observations. *AI*, *167*, 103–136.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE* (pp. 257–285).

Thon, I., Landwehr, N., & De Raedt, L. (2008). A simple model for sequences of relational state descriptions. *ECML* (pp. 506–521). Springer.

Tran, S. D., & Davis, L. S. (2008). Event modeling and recognition using markov logic networks. *ECCV* (pp. 610–623).