Rule Induction for Adaptive Sport Video Characterization Using MLN Clause Templates

David Windridge[1], Josef Kittler[1], Teofilo de Campos[1],
Fei Yan[1], William Christmas[1] and Aftab Khan[2]
1. CVSSP, University of Surrey
Guildford, GU2 7XH, UK
2. Culture Lab, School of Computing Science,
Newcastle University, Newcastle upon Tyne, UK

***Abstract*—The grounding of high-level semantic concepts is a key requirement of video annotation systems. Rule induction can thus constitute an invaluable intermediate step in characterizing protocol-governed domains, such as broadcast sports footage. We here set out a novel "clause grammar template" approach to the problem of rule-induction in video footage of court games that employs a second-order meta-grammar for Markov Logic Network construction. The aim is to build an adaptive system for sports video annotation capable, in principle, both of learning *ab initio* and also adaptively transferring learning between distinct rule domains.**

The method is tested with respect to both a simulated game predicate generator and also real data derived from tennis footage via computer-vision based approaches including HOG3D based player-action classification, Hough-transform-based court detection, and graph-theoretic ball-tracking.

Experiments demonstrate that the method exhibits both error resilience and learning transfer in the court domain context. Moreover the clause template approach naturally generalizes to any suitably-constrained, protocol-governed video domain characterized by feature noise or detector error.

keywords: Video Annotation, Markov processes, Stochastic Logic, Markov logic network (MLN), Action Recognition, Behavior discovery, Statistical Relational Reasoning

I. Introduction

Automated video annotation is a well-established research problem within computer vision and includes various challenges such as event detection and action recognition [12]. Application domains range from surveillance to sign-language recognition, with sports videos having proved particularly popular and a range of methodologies having been implemented. Within tennis video annotation, research has focused on shot classification [6], within-shot event detection [13], stroke-type classification [10], analysis of player tactics [18] and scene retrieval [16]. In general, court-games may be characterized in terms of their *low-level visual events* (e.g. ball and player movements within the court) and their *high-level events* incorporating all the rule-salient contextual details such as the match-score.

High-level semantic concepts this play a key role in video annotation in general, and sport-video annotation in particular [6] [1]. However, in order to function effectively, it is crucial that concepts are properly grounded in order to overcome the 'semantic gap'. High-level conceptual information must thus correspond appropriately to low-level features extracted via computer-vision and computer-audio techniques. Within protocol-governed domains such as broadcast sports footage, semantic concepts will thus typically correlate with *rule-relevant structures* within the game such as players, court-locations etc. The utilization of high-level rules in sport video annotation consequently has a long history [5], and while generally these rules are specified *a priori*, rule induction has been found to be useful in this context [2] [5]. Generally, game protocols are expressed by first-order logical rules (or at least logics equipped with variables and universal quantification). However, induction of first-order logical rules has (to date) been accomplished only via the processes of *rule-mining* and *Inductive Logic Programming (ILP)*. Both of these have disadvantages in an annotation-based context; the former typically does not establish a full set of rules and the latter typically has too large a search space to be computationally tractable for complex domains. It will be the task of this paper to rectify this by proposing a novel inductive methodology that employs MLNs (which are generally assumed to be incapable of rule inference) to provide a complete, and computationally tractable solution.

Thus, while sports video annotation is a well established problem area in machine vision, for which manifestly successful techniques are available, most existing machine annotation systems tend to be crafted for individual domains and have little or no adaptability, and very limited capacity to extend abilities/cope with new environments. However, it is clear that many sports domains are intrinsically similar; tennis and badminton, for instance share many common rules and visual primitives (court-lines, nets etc). Our aim in the following is thus to develop a high-level mechanism for generic court-based sports video annotation capable of autonomously adapting rule-bases, transferring knowledge, and acquiring new competences as the problem demands, one that is thus capable of complementing existing low-level adaptation/transfer-learning approaches.

As a step toward achieving this, we set out an approach that employs a *meta-grammar* for Markov Logic Network (MLN) construction (Markov Logic Networks being the increasingly predominant approach to applying logical reasoning in the context of stochastic uncertainty, with widespread application throughout the field of pattern recognition [15] [4]). However, rather than induce clauses directly from predicatized representations of computer vision and computer audio based features (which tends to produce poor results), we seek instead to define a representative set of clause *grammars*, which can be exhaustively enumerated to create large number of clauses for individual weighting. This typically enables greater grammatical flexibility than alternative MLN-based induction methods that are limited to inducing conjunctions of literals.

Our proposed method of game-rule learning has some similarities to other second-order MLN transfer-learning-approaches in the literature (specifically [8] and [3], both of which substitute predicates applicable in one domain by second-order variables to be appropriately instantiated within the novel domain). However, our approach is based on *a priori* knowledge of generic game-clause structures, enabling us to pre-emptively exhaust the clausal possibilities of the novel domains via an appropriate choice of templates. Moreover, because we choose predicate structures that are universal to all court games (e.g., predicates indicating court-lines, projectile tracks etc), we do not need to consider the re-mapping/re-instantiation of second-order variables over predicates (other than to expand the template). Rule adaptation is consequently achieved in our approach via clause *re-weighting*, greatly reducing the magnitude of the search space and therefore the risk of over-fitting. The clause template approach thus naturally generalizes to any protocol-governed computer-vision domain affected by stochastic detector noise.

The paper is formatted as follows: the following section will describe MLN theory prior to setting out our novel "clause template" approach to rule induction. In section III we describe a simulated game predicate generator that will parallel the computer vision methods employed for real data. Section IV then sets out a series of evaluation protocols and Section VI looks at the application of the clause template method to the real and simulated data. We conclude in Section VII with experimental discussions/conclusions.

II. Methodology

A. Markov Logic Networks (MLNs)

Markov Logic Networks, first proposed by Domingos and Richardson [4], are an amalgam of *first-order logic* and probabilistic graphical models *Markov Networks* (also known as Markov random fields). They allow first-order logic clauses to be treated in *probabilistic* terms [14] by relaxing the strict boundaries of first-order logic clauses via the association of logical formulas with real number weights.

For a given set of first-order logic clauses with quantifiers $\forall, \exists$ scoped over the variables of predicates connected by conjunction and disjunctions (e.g. of the form $\forall x\ \forall y\ \forall z\ (A(x,y) \wedge B(y,z)) \vee C(x,z) \Rightarrow D(x,y)$ for predicates A, B, C, D), it is possible to build a Markov network graph in which vertices are variables and edges are derived from the the logical connectives used to construct formulae. Each formula thus constitutes a *clique* within the Markov random field: the Markov blanket of a variable is the set of cliques in which it appears. A *Ground Markov Network* is one in which vertices are grounded atomic predicates that can be collectively associated with a concrete Herbrand interpretation (i.e. a 'possible world' generated by predicate term instantiations).

A Markov logic network L is formally defined (cf [14]) as a set of pairs (F_i, w_i), where F_i is a first-order logical formula and w_i is a real number that defines the weight of F_i. Given a finite set of constants *C*, a Markov network $M_{L,C}$ consists of a single binary node for each possible grounding of each predicate appearing in L, and one grounded feature per formula F_i in L, which has a value of 1 for *true* ground formula, and 0 otherwise. ω_i affects the degree to which inconsistency with the i th formula is tolerated, and affects the probability of possible world x accordingly.

A potential function, $\phi_k(x_{\{k\}})$, is associated with each formula such that $\phi_k = 1$ when true and $\phi_k = 0$ when false (i.e. when the formula variable instantiation $x_{\{k\}}$ is realized as part of possible world x).

If $n_i(x)$ is number of true groundings of the i'th formula in the possible (Herbrand) world x , then the world's probability is given by the exponential expansion:

$$\begin{aligned} P(X=x) &= \frac{1}{Z}\prod_i \phi_i(x_{\{i\}})^{n_i(x)} \\ &= \frac{1}{Z}\cdot \exp\left(\sum_i w_i \cdot n_i(x)\right) \\ &= \frac{\exp(\sum_i w_i \cdot n_i(x))}{\sum_{x'\in\chi}\exp(\sum_i w_i \cdot n_i(x'))} \end{aligned} \qquad (1)$$

(constituting the Gibbs measure and partition function for the MLN)

Weights are obtained via a maximum likelihood gradient descent of equation 1 with respect to a relational database (equation 1 is straightforwardly differentiable). Several possibilities exist; in the following, the Box-constrained Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS-B) Quasi-Newtonian method is employed to attain the minimum.

The goal of inference in a Markov network is then to find the *stationary distribution* (i.e the most likely assignment, given the clause weights, of probabilities to the vertices of the ungrounded network graph given a particular knowledge base and query formula). To solve this marginalization problem, several approaches are possible; in our case the solution is achieved through MaxWalksat sampling.

A straightforward generative mode of weight learning is used for our purposes (discriminative methods are also possible). Each frame of data (i.e. feature vector) is thus converted into feature predicates via a set of computer vision processes and added to a *relational* database for test and training purposes. We follow an implicit open-world assumption (i.e. any ground predicates not included in the training datasets are considered as potentially either *true* or *false*)).

B. MLN Meta-Structure for Game Rule Induction

Structure learning (clause induction) via processes similar to Inductive Logic Programming is possible for MLNs, but is generally prohibitively computationally expensive, and limited in practice to the induction of conjunctive formulae only. We require something substantially more flexible and appropriate to the domain.

Our approach to rule induction is therefore to utilize the (relatively efficient) MLN weight learning processes in conjunction with a very large number of clauses that are combinatorially-exhaustive with respect to a particular clause "meta-grammar" (to be defined). In this way, depending on the generality of the meta-grammar, all significant rule-like behaviors can be extracted from a knowledge-base, with irrelevant rules being weighted to near zero. This approach also has the potential for combining large numbers of weakly-weighted partially-accurate rules in ways that are advantageous (e.g. due to error decorrelation in the manner of *bagging*, or cooperative support as in *boosting*).

We thus employ *a priori* generic court rules that are applicable to all games (such as e.g. "a player hitting a ball must be proximate to the ball") and combine these with the unweighted clauses generated by the meta-grammar. Clause weighting may then be used to learn specific game-rules according to the standard MLN approach. We finally obtain predicate likelihoods via exhaustive inference over each atomic predicate at a particular event-interval.

Our approach to clause meta-generation can be separated into two grammatical classes: *Contemporaneous implication* (the inference of rules concerning configuration correlations) and *Successive implication* (the inference of rules concerning causal relations). Examples of both of these classes are given below (with P_x designating an arbitrary predicate with second-order label x and the symbol $\bigwedge^x$ designating a conjunctive combination over all predicate indices x of the logical formulae to the right-hand side of the symbol:

Contemporaneous implication:

$\bigwedge^x \bigwedge^y P_x([T^{\{n(x)\}}], t_1) \Rightarrow P_y([T^{\{n(y)\}}, t_1)$

$\bigwedge^x \bigwedge^y \bigwedge^z P_x([T^{\{n(x)\}}], t_1) \quad \wedge \quad P_y([T^{\{n(y)\}}], t_1) \quad \Rightarrow P_z([T^{\{n(z)\}}], t_1)$

Successive implication:

$\bigwedge^x \bigwedge^y Succ(t_1, t_2) \wedge P_x([T^{\{n(x)\}}], t_1) \Rightarrow P_y([T^{\{n(y)\}}, t_2)$

$\bigwedge^x \bigwedge^y \bigwedge^z Succ(t1, t2) \quad \wedge \quad P_x([T^{\{n(x)\}}], t_1) \quad \wedge P_y([T^{\{n(y)\}}], t_1) \Rightarrow P_z([T^{\{n(z)\}}], t_2)$

where $Succ$ indicates a succession relationship between temporal instants t_1 and t_2 (various other temporal logic relations are also possible).

Within this generalized high-level format, $[T^{\{n(t)\}}]$ is an arity $n(t)$ list of terms such that: $[T^{\{n(t)\}}] = [R^{\{n^1(t)\}}, N^{\{n^2(t)\}}]$ where $R^{\{n^1(t)\}}$ are *functionally-relatable terms*, and $N^{\{n^2(t)\}}$ are *non-functionally-relatable terms*. The reason for separation of the functionally and non-functionally relatable terms is that additional rule relations can thereby be incorporated within clause meta-grammar. Thus, if a sole function $y = f(x)$ exists, then the term list $T^{\{n(t)\}} = (x, z, y)$ separates as $R^{\{n^1(t)\}} = \{x, y\}$ and $N^{\{n^2(t)\}} = \{z\}$. *Additional* clause structures may consequently be extracted from the above meta-relations by exhaustively enumerating all of the possible substitutions (in this case just $y = f(x)$ for x) available within each functionally-relatable term-list. These will, in general, be $2^{n^1(t)}$ in number for each predicate. Permitting such substitutions is potentially useful in a court-game context because it allows for the induction of symmetric relations in the rule structure via the of use opposition functions of the type *FARSIDE=opposite(NEARSIDE)*, which allows for e.g., serves on opposite sides to be described by a single clause.

Terms themselves may be grounded or ungrounded using the format-symbol: $+$ (according to the Alchemy usage: *http://alchemy.cs.washington.edu/*). Where grounding is selected, terms are grounded *independently*, giving rise to very large numbers of individually-weighted clauses. Thus, for the predicate term list of the form $N^{\{n^2(t)\}} = (A+, B+, C+, D+ ..)$, the clause base for weighting is expanded by a factor: $|(A, B, C, D..)| = |(a1, a2, a3..)| \times |(b1, b2, b3..) \times ...|$ where the $a1, a2..$, $b1, b2...$ are logical constants (i.e., possible instantiations of A, B etc). There are additionally $2^{n(t)}$ grounded/non-grounded decisions to make for each predicate, which expands the number of clause construct possibilities considerably further, potentially to the point of intractability. Throughout the following, we therefore opt for fully grounded terms. Note that negative weights constitute a possible solution, so that negations of any given clause-template are also implicitly incorporated; for our problem domain, generated clauses typically number in the $10,000$'s (i.e. of a magnitude that can be processed efficiently via the weight learning process described earlier).

Application of this approach in practice requires that variable instantiations and identity relations between variables (so as to enable them to fall under the same quantifier's scope) are given in advance. These may either specified *a priori* or derived directly from the training data. In latter case, the sets of variable instantiations can be derived greedily from the training and input data sets; variable identity relations can then be made on the basis of the degree of overlap between instantiation sets (e.g. such that variables with identical, or very similar, instantiation sets have the option of being scoped under the the same quantifier within the generated clauses).

The five specific grammar templates utilized in our domain tests are listed below; however we require more than the clauses generated above in order to build a MLN; additional header information embodying mutual exclusion (mutex) principles and function/constant declarations also need to be incorporated into the weight learning process.

C. Grammars Employed

The grammar templates employed for clause generation are listed below (for brevity, the *a priori* grammar is not listed). Here P_n represents an arbitrary atomic predicate drawn greedily from a relational database; $\bigwedge^n$ and $\bigvee^n$ are conjunctive/disjunctive combinations, over all predicate indices n, of any logical formula to its right-hand side (the symbol "\" indicates set- difference); a is a predicate term-list.

Grammar 1: simultaneous likelihoods, no conjunction or implication (with individual weightings of non-temporal variable instantiations)

$\bigwedge^n P_n(+a,t)$

Grammar 2: simultaneous likelihoods, implications, no conjunction (with individual weightings of non-temporal variable instantiations)

$[\bigwedge^m \bigwedge^n P_n(+a,t) \Rightarrow P_m(+b,t_1)] \setminus [\bigwedge^m P_m(+a,t) \Rightarrow P_m(+b,t_1)]$

henceforth, for terminological compression, we will write:

$[\bigwedge^m \bigwedge^n logform(P_m,P_n)] \setminus [\bigwedge^m logform(P_m,P_m)]$

as

$\bigwedge^{m:m\neq n} \bigwedge^{n:n\neq m} logform(P_m,P_n)$

where $logform(P_m,P_n)$ is a well-formed (quantified) first-order formula (i.e. disjunctive or conjunctive combination of P_m, P_n).

Grammar 3: successive likelihoods, implications, no conjunction (individual weightings)

$[\bigwedge^m \bigwedge^n Succ(t_1,t_2) \wedge P_n(+a,t_1) \Rightarrow P_m(+b,t_1)]$

Grammar 4: simultaneous likelihoods, implications, conjunction (individual weightings)

$[\bigwedge^{m:m\neq n,o} \bigwedge^{n:n\neq m,o} \bigwedge^{o:o\neq m,n} P_m(+a,t_1) \wedge P_n(+b,t_1) \Rightarrow P_o(+c,t_1)]$

Grammar 5: successive likelihoods, implications, conjunction (individual weightings)

$[\bigwedge^{m:m\neq n} \bigwedge^{n:n\neq m} \bigwedge^{o} Succ(t_1,t_2) \wedge P_m(+a,t_1) \wedge P_n(+b,t_1) \Rightarrow P_o(+c,t_1)]$

Grammar 6: conjunction over all grammars i.e.:

$\bigwedge^n P_n(+a,t) \wedge$
$\bigwedge^m \bigwedge^n P_n(+a,t) \Rightarrow P_m(+b,t_1)] \setminus [\bigwedge^m P_m(+a,t) \Rightarrow P_m(+b,t_1)] \wedge$
$[\bigwedge^m \bigwedge^n Succ(t_1,t_2) \wedge P_n(+a,t_1) \Rightarrow P_m(+b,t_1)] \wedge$
$[\bigwedge^{m:m\neq n,o} \bigwedge^{n:n\neq m,o} \bigwedge^{o:o\neq m,n} P_m(+a,t_1) \wedge P_n(+b,t_1) \Rightarrow P_o(+c,t_1)] \wedge$
$[\bigwedge^{m:m\neq n} \bigwedge^{n:n\neq m} \bigwedge^{o} Succ(t_1,t_2) \wedge P_m(+a,t_1) \wedge P_n(+b,t_1) \Rightarrow P_o(+c,t_1)]$

III. PREDICATE GENERATION

A. Computer Vision Predicate Generation

In the following paragraphs we briefly detail the computer vision processing required to generate tennis configuration predicates (detailing player actions and locations, and ball trajectories and locations) in a manner appropriate for input into the MLN rule induction process (i.e. as a set of predicatized detections). Computer audio based predication can be similarly appended, although beyond the scope of this paper; ground-truthing is achieved via hand-annotation.

a) Low-level Pre-Processing: Image frames are initially de-interlaced into *fields* (the tennis videos employed in our experiments are interlaced when captured). Fields thus eliminate temporal aliasing as required for the ball tracking procedure. Following de-interlacing, camera lens geometric distortion is corrected for. It is assumed that the camera position on the court is fixed, with the global transformation between frames defined as a *homography*. The homography is determined by corner-tracking throughout the sequence (RANSAC [7] is applied to the detected corners to find a robust estimate of the homography, with a Levenberg-Marquardt optimizer also applied to improve the homography).

b) Shot Analysis: A broadcast tennis video is presumed to be composed of *shots*, consisting in e.g. gameplay, close-ups, crowd-scenes and commercials. Shot boundaries are detected via a breakdown in color histogram intersection between adjacent frames. Shots are then classified into the aforementioned categories using the histogram mode and, in the case of overhead gameplay, the presence of corner-point continuity (it is the latter category of shot that provides the basis for the predicate generation).

c) Court Detection, Ball Tracking, and Player Tracking: For shots classified as gameplay, we thus determine the court lines, player location (relative to the court lines) and actions, as well as the ball trajectory. Tennis court lines are determined via Hough transformation, while player detection is carried out by background subtraction, incorporating geometric spatio-temporal consistency-checking with prior-based masking. Player tracking (as distinct from detection) is carried out via a particle filter, with player actions classified via a nearest neighbor approach using a bag of HOG3D features. For ball tracking, background subtraction is employed to generate initial ball candidates. A feature vector is computed for each candidate, incorporating size, color and edge-contour information. SVM classification is then used to eliminate all but the strongest ball candidates.

Ball tracks themselves are established in two stages. First, "tracklets" are built from sets of extracted strong object candidates in the form of second-order (roughly parabolic) trajectories. A graph-theoretic data-association technique is then used to link the tracklets into complete ball tracks [17]. Gameplay events are indicated by significantly above-threshold ball-trajectory changes, which are then correlated with other event-label indicators such as player action-class in order to obtain an event characterization. These characterizations constitute one of the key predicatized outputs of the computer vision stage, with the possible event-label instantiations *hit*, *bounce*, *net*, and *serve*. Alongside these event-labels, predicatized output is also generated for player and ball positions (in terms of fine-grained/coarse-grained court-box geometry), as well as for player actions. The encoding scheme is depicted in Fig. 1; thus the predicate ball $ballloclattice(ONE,ONE,t)$ indicates that the the ball is in court box $(+1,+1)$ at time t.

Predicatizing the Ball and Player location requires that a 3D position is determined relative to the detected court lines. Since the preceding computer vision processes only generate screen-relative locations, we utilize the homography, along with an appropriate set of *a priori* assumptions in order to project the screen-coordinates of events into the court ground-plane. In particular, we assume a constant height for the player, and that the lower edge of the player bounding box is in contact with the ground plane ($z = 0$). The three key ball events: *serves*, *hits* and *bounces* are thus presumed to occur at a typical player's height, a typical player's shoulder-height and the ground plane ($z = 0$), respectively. Nets are assumed to occur at half the regulation net height.

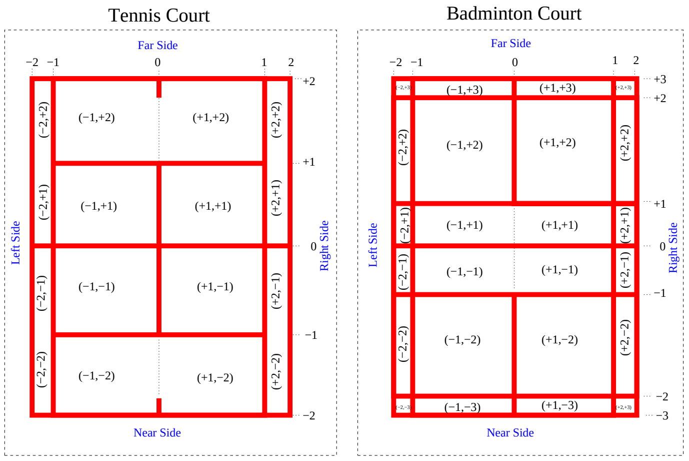

Fig. 1: Court-box labelling scheme for Ball and Player detections (fine-grained in black, coarse-grained in blue

A typical predicate output for event n would thus be as follows:

$eventlabel(SERVE, n)$ - the label of the event
$serveside(NEARSIDE, n)$ - the side from which the ball was served
$eventside(NEAR, n)$ - the side on which the current event takes place
$ballloc(NEAR, LEFT, n)$ - the ball's location
$p1loc(NEAR, LEFT, n)$ - player 1's location
$p2loc(NEAR, RIGHT, n)$ - player 2's location
$p3loc(FAR, RIGHT, n)$ - player 3's location (if present)
$p4loc(FAR, LEFT, n)$ - player 4's location (if present)
$balllloclattice(ONE, ONE, n)$ - fine-grained ball-location
$p1loclattice(ONE, ONE, n)$ - fine-grained player 1 location
$p2loclattice(ONE, ONE, n)$ - fine-grained player 2 location
$p3loclattice(ONE, ONE, n)$ - fine-grained player 3 location (if present)
$p4loclattice(ONE, ONE, n)$ - fine-grained player 4 location (if present)
$p1action(SERVING, n)$ - player 1 action
$p2action(IDLE, n)$ - player 2 action
$p3action(IDLE, n)$ - player 3 action (if present)
$p4action(IDLE, n)$ - player 4 action (if present)
$Succ(n + 1, n)$ - asserts topological continuity of temporal indices.

B. Simulated Game Predicate Generation

As well as the predicatization of computer-vision data, we wish to create a ready source of simulated game predicates for evaluation purposes. At an abstract level, an observed game of tennis can be implicitly modelled by a non-deterministic push-down automata (note, not a context free grammar, since modelling the game's progression requires a memory of, for instance, the current score and the serve side from which the current play-event originated). If score progression is omitted, game-play sequences from *serve* through to *point* can be modelled as a non-deterministic finite-state machine (which requires only memory of the original serve side, rather than a memory stack for scoring events).

A non-deterministic finite state machine is defined as a 5-tuple: $(Q, X, T, q0, F)$ with Q a finite set of states, X a finite set of input symbols X, T a transition function mapping $Q \times X$ to 2^Q, $q0$ an initial (or start) state $q0 \in Q$, and F a set of states distinguished as *accepting* (or *final*) states $F \subseteq Q$. Utilizing the power set 2^Q permits stochastic transitional ambiguity to be modelled.

For a generic court game, we have thus have the following set relations:

$X = \{nearside, farside\}$
$F = \{Point\}$
$Q = \{Pre\_event, Point, Serve, Hit, Bounce\_Out, Bounce, Net\}$
$q0 = \{Serve(X)\}$
$T : Q \times X \rightarrow 2^Q$

We also have the following transition probabilities (fully constraining T)):

A) Chance of player on side X not returning ball $= p([Serve(\neg X)|Hit(\neg X)] \rightarrow Bounce(X) \rightarrow Point(\neg X)) = 0.2$
B) Chance of ball bouncing on side X before player hits it $= p([Serve(\neg X)|Hit(\neg X)] \rightarrow Bounce(X) \rightarrow Hit(X)) = 0.7$
C) Chance of player on side X losing point after hit $= p(Hit(X) \rightarrow [Bounce\_Out|Net(X)|Bounce(\neg X)]) = 1 - p(Pre\_event) = 0.3$
D) Chance of bounce out following losing hit $= p(Bounce\_Out|C) = 0.2$
E) Chance of 'Net' following losing hit $= p(Net|C) = 0.5$

(Note that $p([X|Y])$ denotes the probability of either X or Y occurring, while $p(X|Y)$ denotes the probability of X occurring given that Y has occurred in accordance with standard statistical notation.)

These transition probabilities are sufficient to parameterize all macro-configurations of sub-events within an individual point (all other transition probabilities are derivable from the above).

We also have the following opposition relations:

$\neg(nearside\_serve) = farside\_serve$

$\neg(farside\_serve) = nearside\_serve$

We thus represent the non-deterministic finite state machine appropriate to Tennis as a state diagram in figure 2 (note that states differ from those in a Markov Model in having an input variable within parentheses). The game of *Badminton* can then be modelled within this framework by omission of the *Bounce* possibility from the *PreEvent* to *PlayerAct* transition; we shall utilize this in our evaluation of high-level transfer/adaptation.

The non-deterministic finite state machine of figure 2 thus encompasses all basic game transitions for the game of tennis in a way that permits enrichment via an additional set of non-deterministic microstates so as to give a full predicate description of the game of tennis in terms of the major rule-relevant entities (i.e the entities in terms of which the rules of the game are specified in official language-based material). We thus, for example, delineate positions only at the 'grain' of court-boxes.

The stochastically-selected microstates adding additional fine-grained locations for the Ball and Players are thus instantiated from the following sets:

$lateral\_side\_indicator = \{Left\_hand, Right\_hand\}$
$courtbox\_vertical\_offset = \{0, 1, 2, 3\}$

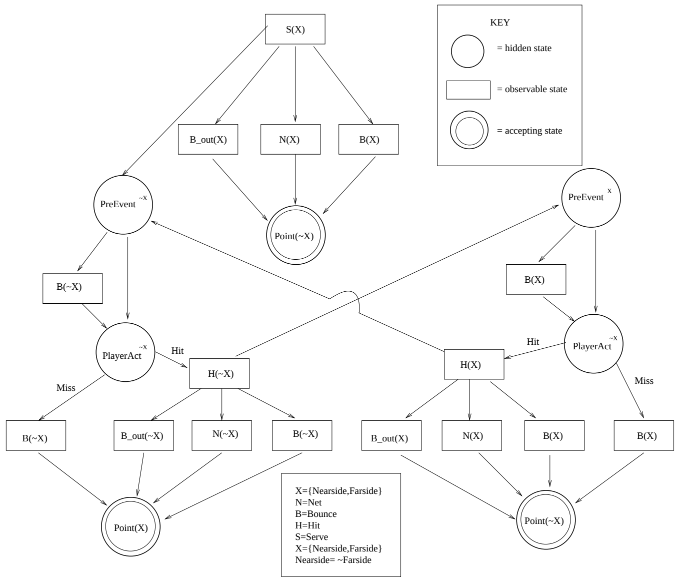

Fig. 2: Tennis state diagram

$courtbox\_horizontal\_offset = \{0, 1, 2, 3\}$

Adopting this format for position-specification makes it straightforward to encode symmetries within predicates (thus we don't consider the absolute offsets $\{-3, -2, -1, 0, 1, 2, 3\}$, but rather the conjunction of the $lateral\_side\_indicator$ predicate with the relative $court - box\_horizontal\_offset$ predicate). To a large extent, this avoids the necessity of adopting relational functions with the clause meta-grammar.

The non-deterministic finite state machine of figure 2, along with the additional micro-structuring, is implemented via recursive (stochastic) function calling. (This has the benefit of being extensible; we can straightforwardly add additional microstates if necessary, such as ball velocities, correlated multi-modal representations such as line-calls etc).

IV. Evaluation Domains

We select four distinct areas of evaluation for the proposed meta-MLN approach for rule-induction; *Domain Adaption/High-level Transfer Learning*, *Noise Resilience*, *Structured Output Learning*, and *Accuracy of Prediction*. These are evaluated using the simulated predicate generator (except for prediction accuracy, which is less meaningful for a finite state machine) and on real predicates obtained from footage of the *Australian Open mens' singles 2003* tennis championship for the latter two evaluation scenarios. The domains are characterized as follows:

1) Scenario 1: Accuracy of Prediction: Accuracy of Prediction is evaluated in terms of total predicate accuracy with respect to an immediately prior sequence of predication assertions. Thus, if we have a set of predicate groundings of the form $p_X(A_X, t)$, such that A_X is a sequence of grounded terms associated with event-number instantiations of t, then we test the prediction accuracy of the MLN with respect to the ground-truthed predicates $p_X(B_X, t+1)$ associated with the subsequent event. (The MLN generates conditional likelihoods for the entire Herbrand base, so that the assertion of the existence of the temporal variable grounding: $t+1 \in T$ is sufficient to ensure that all possible predictive groundings are generated for the subsequent event). Predications are then hardened on the assumption of a clausally-constrained *post facto* 'mutex' (mutual exclusion) principle with respect to individual variables.

As well as the set of *a priori* rules generic to all court games, five candidate clause grammar templates are generated in the manner set out in Section III and evaluated both in terms of the overall average hardened predicate accuracy, and also the hardened event-label predicate accuracy (as this is generally the configuration predicate we are most interested in in an annotation scenario). A combination of all of the template-generated clauses constitutes a sixth grammar. This latter grammar requires substantial computation time for the inference process (though not the weighting process), and so only a single experiment is carried out[1].

In addition to evaluating the grammars individually and in composition, we also apply the Inductive Logic Method (ILP) approach of Muggleton et al. [9] as a baseline method (this being the only comparable rule-induction baseline evaluation method in the literature). To apply this approach comparably, 'Head' and 'Body' mode predicates are exactly as the specification given above, and all variable instantiations (types) are declared at the outset. ILP works by generalizing most specific Horn clauses obtained from the training observations and then uses resolution theorem-proving to establish clause generality, retracting any redundant clauses. Following exploratory experimentation, ILP meta-parameters were set as follows: the maximum number of combinations in the clause lattice (nodes) searched is 1000, the maximum number of resolutions is set to 3000, the maximum clause length is set to 100, the maximum variable depth was set to 3, predicates are set to be positive only; all other parameters were the defaults. (Note inductive convergence only occurs with respect to certain predicates -it is thus possible that exhaustive searching would give a more optimized set of ILP parameters; however, this is prohibitively time-consuming).

Accuracy is again determined as percentage of correctly predicted event labels after hardening.

2) Scenario 2: Domain Adaption/High-level Transfer Learning: This evaluation regime is intended to complement low-level transfer-learning approaches, in which a *feature*-transform is typically sought between learning domains [11]). We here look explicitly at the potential for inter-domain *re-weighting* of clauses to act as a mechanism for learning-transfer for the higher-level aspects of the domain (i.e. the domain's protocols). The idea is thus that weight-learning in one domain will, for many clauses, still have validity in another domain, such that a pre-weighted MLN will achieve a more generally accurate (i.e more truly global) minimum than an unweighted MLN applied in the new domain. (Outcomes are again evaluated in terms of total predication accuracy).

[1]One possible efficiency measure involves filtering of weak (i.e. low-weighted) rules according to a weight-threshold. However this latter approach results in substantial degradation of performance (perhaps due to a boosting-like effect from the very large numbers of weakly-weighted clauses being lost)

Note that other mechanisms have been proposed for MLN transfer-learning, in particular [8] and [3]; however, both of these approaches are tuned to learning new predicates from lifted rules, and so are distinct from our approach. (In fact the two classes of approaches are complementary, and could, in principle, be used together. We do not need to consider this option here, however, given that predicates are designed to have universal-validity between court domains).

The average fraction of total predicates accurately-predicted is thus determined for the respective configurations: Tennis train/tennis test; Badminton train/badminton test; Tennis train/badminton test; Badminton train/tennis test using 100 simulated events (the predicates are those listed in Section IV - we thus employ a total of $18 \times 100 = 1800$ training predicates with a total of 2700 variable instantiations over 2 experimental runs). A performance baseline for comparison with these figures is determined by randomly instantiating predicates to give an accurately-predicted fraction of 0.32.

3) Scenario 3: Noise Resilience: Within the microstate-supplemented non-deterministic finite state machine we can introduce an additional "noise factor", such that a proportion of variables are randomly instantiated with a probability proportional to a set threshold. This allows for the simulation of detector failure/error.

Overall MLN noise robustness is thus established with respective noise levels sampled at $[0.25, 0.50, 0.75, 1.0]$, such that 0.0 represents no noise and 1.0 represents completely random instantiation of predicate variables. (Again, outcomes are evaluated in terms of total predication accuracy).

4) Scenario 4: Structured Output Learning: Finally, we conduct a test of the proposed methodology's capability of determining a full event *sequence* when given predicatized detector input (i.e. a test of the ability of the method to classify *structured output* rather than discrete configuration predicate instantiations as in the above case). The input is thus the entire sequence of non-event predicates, such that the first-order query addressed to the trained MLN now concerns the full temporal sequence of event predicate instantiations. (For efficiency in the case of the simulated data we use the single best performing grammar from the first test).

In all of the following, 100 events are used for MLN training, selected from a temporally-distant portions of play (i.e., non-contiguous data-sets). The Alchemy system for MLN construction is employed throughout: "http://alchemy.cs.washington.edu/".

a) Average predicate prediction accuracy (averaged over all temporal training instances)	
A priori grammar	
all predicate accuracy	event label accuracy
0.4572 ± 0.0022	0.4718 ± 0.0026
Grammar 1	
all predicate accuracy	event label accuracy
0.2527 ± 0.0049	0.2631 ± 0.0298
Grammar 2	
all predicate accuracy	event label accuracy
0.5112 ± 0.0006	0.7879 ± 0.181
Grammar 3	
all predicate accuracy	event label accuracy
0.4474 ± 0.0057	0.6579 ± 0.0819
Grammar 4	
all predicate accuracy	event label accuracy
0.4665 ± 0.0213	0.4831 ± 0.0610
Grammar 5	
all predicate accuracy	event label accuracy
0.4618 ± 0.0153	0.4474 ± 0.0074
Composite grammar	
all predicate accuracy	event label accuracy
0.4710	0.4470
ILP Baseline	
all predicate accuracy	event label accuracy
0.14	0.43

b) Structured Output Learning Results (Gr. 2)	
Absolute Performance	Multiple of Chance
0.6800	2.7200

TABLE I: Evaluation of a) Performance for Individual/Composite Grammars and ILP baseline and b) Structured Output Learning Results using best performing grammar using predicates from observed match data

a) Transfer Learning/Domain Adaptation Results: Time-averaged Predicate Prediction Accuracy			
		Test Environment	
		Tennis	Badminton
Training Environment	Tennis	0.47	0.48
	Badminton	0.46	0.48

b) Noise Resilience Results				
Noise level	0.25	0.50	0.75	1.0
Prediction accuracy	0.62	0.53	0.45	0.32

c) Structured Output Learning Results		
Instantiation Noise	Abs. accuracy	Multiple of Chance
0%	0.6818	2.7273
20%	0.5741	2.2963

TABLE II: Evaluation of a) Transfer Learning/Domain Adaptation, b) Noise Resilience, and c) Structured Output Learning using generated predicates

V. RESULTS AND DISCUSSION

Results for the evaluation scenarios listed above are set-out (where relevant) for the *Australian Open mens' singles 2003* data in Table 1 and for the generated data in Table 2.

The games of tennis and badminton differ with respect to just a few critical rules; examining the *Transfer Learning* results (Table 1), it is clear that the MLN results demonstrate significantly greater than chance performance (0.32) on all transfer scenarios (recall that while rules differ, transition likelihoods are retained within the simulator). Overall configuration accuracy figures for the finite-state simulated data are dominated by ball and player position predicates (as we would expect given the finite-state machine domain model).

As regards the *Noise Resilience* evaluation, a very nearly linear tail-off is observed with respect to the sampled noise levels on the best performing grammar. (Convergence failure occurs with any non-zero noise value using ILP).

In terms of the *Accuracy of Prediction* evaluation, *grammar 2* proves the most effective clause template for prediction of real-world events, and outperforms the weighted combination of all grammars.

The *Structured Output Learning* evaluation using the single best-performing grammar on the simulated tennis obtains an event prediction accuracy of 0.6818, corresponding to a near-equivalent accuracy of 0.6800 on the Australian Singles data. This drops to 0.5741 (2.2963 chance accuracy) with 20% instantiation noise, suggested random instantiation is unrepresentative of the computer vision predicate generation.

Collectively, these evaluations demonstrate a key advantage of first-order logical techniques; namely, flexibility with respect to arbitrary querying, with structured output learning potentially representing the most typical usage scenario.

VI. Conclusions

The utility of the proposed clause meta-template approach to rule-induction is demonstrated in the context of sport-video annotation, where the domain as a whole can be characterized via a set of very generic spatio-temporal grammatical constraints. The grammar templates are thus capable of exhausting the main classes of relation that exist between detectable entities in a sport-based environment, such that high-level domain learning and adaptation can take place purely in terms of MLN-based clause-weighting, which can efficiently accommodate the large numbers of rules so generated (entity predication is sufficiently universal as to apply between different sport domains).

The MLN-based method, furthermore, has the advantage of noise-resilience, exhibiting a linear degradation of performance (contrasting sharply with deductive/ILP-based methods). Experiments on real data also indicate relatively little performance degradation in relation to simulated game data.

Ensemble grammars can be constructed via template aggregation; however, only marginal performance improvement was obtained in the tested annotation context relative to the best single-performing grammar.

The presented clause grammar template method is thus a flexible and adaptive approach to MLN building, suitable in particular for high-level semantic classification. Other possible modes of application, besides annotation and learning-transfer, include providing *high-level priors* for detector modules (so that e.g. the Ball Tracking Module might call on the MLN Module to establish whether “ball on far-side of net” is a viable hypothesis during graph-theoretic tracklets pruning). A further possibility enabled by top-down feedback is the *re-specification of visual primitives* in novel domains, such that high-level rule inductions are used to re-tune detectors in order to remove non-rule-salient detections, effectively bootstrapping visual capabilities from scratch.

The proposed MLN clause-induction strategy can thus potentially form the basis for a fully-adaptive annotation system that combines both high- and low-level transfer-learning; this is the objective of ongoing research.

Acknowledgement

We gratefully acknowledge funding from the UK Engineering and Physical Sciences Research Council (grant E/F0694 21/1).

References

[1] J. Assfalg, M. Bertini, C. Colombo, and A. Del Bimbo. Semantic annotation of sports videos. *MultiMedia, IEEE*, 9(2):52–60, 2002. 1

[2] L. Ballan, M. Bertini, A. Del Bimbo, and G. Serra. Video annotation and retrieval using ontologies and rule learning. *MultiMedia, IEEE*, 17(4):80–88, 2010. 1

[3] J. Davis and P. Domingos. Deep transfer: A markov logic approach. *AI Magazine*, 32(1):51–53, 2011. 2, 7

[4] P. Domingos and M. Richardson. Markov logic: A unifying framework for statistical relational learning. In *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its connections to other fields*, pages 49–54, 2004. 2

[5] A. Dorado, J. Calic, and E. Izquierdo. A rule-based video annotation system. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(5):622–633, 2004. 1

[6] L.-Y. Duan, M. Xu, Q. Tian, C.-S. Xu, and J. Jin. A unified framework for semantic shot classification in sports video. *Multimedia, IEEE Transactions on*, 7(6):1066–1083, 2005. 1

[7] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 4

[8] L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, AAAI’07, pages 608–614. AAAI Press, 2007. 2, 7

[9] S. Muggleton. Learning from positive data. In S. Muggleton, editor, *Inductive Logic Programming Workshop*, volume 1314 of *Lecture Notes in Computer Science*, pages 358–376. Springer, 1996. 6

[10] T. Ogata, W. Christmas, J. Kittler, and S. Ishikawa. Tennis stroke detection and classification based on boosted activity detectors and particle filtering. In *Proc. Joint 3rd Int. Conf. on Soft Computing and Intelligent Systems and 7th Int. Symposium on Advanced Intelligent Systems*, pages 2035–2040, 2006. 1

[11] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, Oct. 6

[12] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, June 2010. 1

[13] N. Rea, R. Dahyot, and A. Kokaram. Classification and Representation of Semantic Content in Broadcast Tennis Videos. In *IEEE International Conference on Image Processing (ICIP 2005)*, September 2005. 1

[14] M. Richardson and P. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, 2006. 2

[15] S. Satpal, S. Bhadra, S. Sellamanickam, R. Rastogi, and P. Sen. Web information extraction using markov logic networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD ’11, pages 1406–1414, New York, NY, USA, 2011. ACM. 2

[16] M. Tien, Y. Wang, and C. Chou. Event detection in tennis matches based on video data mining. In *IEEE Int. Conf. on Multimedia and Expo*, pages 1477–1480, 2008. 1

[17] F. Yan, A. Kostin, W. Christmas, and J. Kittler. A novel data association algorithm for object tracking in clutter with application to tennis video analysis. In *CVPR*, volume 1, pages 634–641, 2006. 4

[18] G. Zhu, C. Xu, Q. Huang, W. Gao, and L. Xing. Player action recognition in broadcast tennis video with applications to semantic analysis of sports game. In *Proc of the ACM Multimedia*, pages 431–440, 2006. 1