AN INTRODUCTION TO MARKOV LOGIC NETWORKS AND APPLICATION IN VIDEO ACTIVITY ANALYSIS

Guangchun Cheng, *Yiwen Wan, Bill P. Buckles, and Yan Huang*

University of North Texas, Computer Science and Engineering, Denton, TX 76207, USA

ABSTRACT

A Markov logic network (MLN) is a compact combination of logic representation of knowledge and probabilistic reasoning in Markov networks. We have seen its applications in different domains, however, few tried to explain or demonstrate the underneath reasons why MLN works. This paper gives an introduction to MLN using examples in the hope to help understand its elegance and booster the application. Application in video activity analysis was designed to demonstrate how MLN can be used in a specific domain, including feature extraction, logic predicate/formula design, and activity recognition through probabilistic reasoning.

Index Terms— Markov logic networks; computer vision; activity analysis; action recognition

1. INTRODUCTION

Artificial intelligent systems have long been pursued in societies of cognitive science, computer science and engineering among others. Over the years, different tools have been developed for the knowledge representation and reasoning. Two of the most influential are logic and probabilistic methods. The former is based on formal representation of knowledge while the later admits the incomplete and uncertain property of information. In this paper, we examine Markov logic networks (MLN), a statistical relational learning model which combines the logic representation and uncertainty reasoning.

Markov logic networks are a probabilistic logic proposed by Richardson and Domingos [1], which combines the probabilistic Markov network model with first-order logic (FOL). First-order logic is utilized for expressing knowledge in different application domains, and Markov network is used to handle reasoning of events with uncertainty. To do that, MLNs first treats the first-order logic as *templates* to construct Markov networks from the instantiation of the FOL formulas, and then performs probabilistic reasoning in Markov networks based on weights MLN assigns to FOL formulas.

*Corresponding author (guangchuncheng@my.unt.edu). Emails: Y. Wan (yiwenwan@my.unt.edu), B.P. Buckles (bbuckles@cse.unt.edu), Y. Huang (huangyan@unt.edu)

1.1. Related work

Markov logic networks have been applied for several video analysis tasks [2, 3, 4, 5]. For instance, Tran and Davis [2] modeled and recognized events in a parking lot using MLNs. Morariu et. al [3] presented a framework for automatic recognition of complex multi-agent events. They automatically detect and track players, their hands and feet, and the ball, generating a set of trajectories which are used in conjunction with spatio-temporal relations to generate event observation. These observation is used to instantiate the predicates and rules which are designed based on domain knowledge. Due to its domain-independent attribute, MLN can potentially be applied to different areas. It is beneficial to understand how and why it works from the standpoint of application.

Richardson and Domingos provide the most comprehensive discussion in [1], which, however, does not show the procedure for probability computation in Markov logic network and is not suitable for tutorial purpose. We take an introductory method to explain MLN using examples and experiments in this paper. The focus is on the conversion from FOL to Markov networks and probabilistic reasoning in Markov networks. To fulfill this purpose, we design our examples and experiments in the context of video activity recognition, and discuss the key issues involved in an incremental way. Section 2 gives preliminaries and how Markov networks are constructed, and Section 3 discusses the probabilistic reasoning in Markov networks step by step under different scenarios. Application in activity recognition on video data is shown in Section 4. We conclude in Section 5.

2. PRELIMINARY

First-order logic is used for knowledge representation in MLNs. FOL formulas are abstract and compact representation of knowledge. For activity recognition, it is usually in the form of $L_1 \wedge ... \wedge L_n \Rightarrow A$ to express that lower-level actions $L_1, ..., L_n$ together define activity A. Universal or existential quantifier may apply. For example,

$$(\forall x)\, MoveF(x) \Rightarrow Walk(x) \tag{1}$$
$$(\forall x)\, Walk(x) \wedge Near(x, y) \Rightarrow Walk(y) \tag{2}$$

stand for the knowledge that a person x is walking if he is moving forward, and y is walking if it's close to x. However, such rigorous representation of knowledge has its limits when dealing with inference from uncertain and inconsistent observation. Many solutions have been proposed, such as fuzzy logic [6] and Markov logic [1]. Markov logic "softens" the rigorous constraints in First-order logic but still keeps its powerful representation capacity.

As to representation, Markov logic simply adds a weight on each FOL formula to indicate the confidence in that knowledge. Therefore, a Markov logic network is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. For the formula above, its corresponding Markov logic form is

$$\begin{array}{ll} w_1 & (\forall x)\, MoveF(x) \Rightarrow Walk(x) \\ w_2 & (\forall x)\, Walk(x) \wedge Near(x,y) \Rightarrow Walk(y) \end{array}$$

The value of w_i can be any positive values as long as larger values are assigned to formulas with high confidence. In some implementations, negative values are used to indicate the negation of the formula. The weights can also be learned from the data as in most applications.

Given a set of constants $C = \{c_1, c_2, ..., c_{|C|}\}$, a Markov network is defined as follows:

1) The nodes in the Markov network are groundings of predicates in $\{F_i\}$. Each node has a binary value dependent on the value of the grounding;
2) All nodes whose corresponding predicates appear in the same formula form a clique in Markov network;
3) Each clique is associated with a feature. It is 1 if the ground formula F_i is true, and 0 otherwise. The weight of the feature is the weight of the formula, w_i.

Therefore, each first-order logic formula corresponds to one or more cliques in Markov networks because each formula F_i can have more than one instantiation. For formula (1), there could be multiple cliques in the Markov network if multiple constants exist for x. If $x \in \{Alice, Bob\}$, then there are two cliques in the Markov network (as shown in Fig. 1) from the following two ground formulas

$$\begin{array}{ll} w_1 & MoveF(Alice) \Rightarrow Walk(Alice) \\ w_1 & MoveF(Bob) \Rightarrow Walk(Bob) \end{array}$$

This illustrates that an MLN (via its formulas) can be viewed as a template for constructing Markov networks. The two cliques in dashed lines are generated from the same template formula. This is the key to understanding MLNs and the computation of the probability we will discuss later. As the number of constants and formula groundings increases, the Markov network grows, but the number of the formulas or the templates (i.e. representation for knowledge) stays the same.

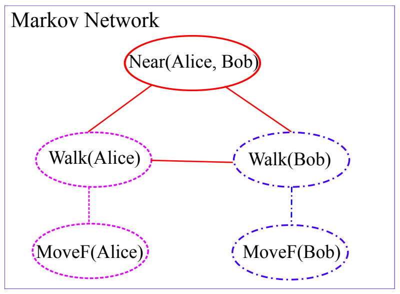

Fig. 1. Illustration of a Markov network. There are two instances for FOL formula (1) and one instance for formula (2).

Once the Markov network is constructed, a common task is to determine the truth values of all predicate grounds, such as $Walk(Alice)$ and $MoveF(Bob)$. The goal is to assign a value for a variable X which contains the truth values for each predicate. For the Markov network in Fig. 1, $X = (MoveF(Alice), MoveF(Bob), Walk(Alice), Walk(Alice), Near(Alice, Bob))$. Some of the components may be determined from the observation while the others need inference. One classic inference task is done through maximizing the probability of $p(y|x)$ where x is known evidence and y represents the predicates whose values are to be determined. The problem is solved by modelling the joint distribution $p(x, y)$ according to Bayes' theorem as shown in the equation below. It gives the satisfiability of the FOL formulas from evidence.

$$P(X = x) = \frac{1}{Z} exp\left(\sum_i w_i n_i(x)\right) \tag{3}$$

where $n_i(x)$ is the number of true groundings of FOL formula F_i in x. X may contain multiple parts which correspond to the same template formula F_i with different truth assignments, and $n_i(x)$ only counts the assignments which make F_i true. For example, given $x = (1, 1, 1, 0, 1)$ where 1 is true and 0 is false, there are two groundings for formula (1), but $n_1(x) = 1$ because only $MoveF(Alice) \Rightarrow Walk(Alice)$ gives true value while $MoveF(Bob) \Rightarrow Walk(Bob)$ does not.

3. AN EXAMPLAR INTRODUCTION TO MLNS

In this section, we continue to discuss how the probability in equation (3) is computed through three examples. Section 3.1 discuss the case with one first-order formula and only one constant, Section 3.2 deals with more constants, and the last section extends the discussion to multiple first-order formulas.

3.1. Basis case: single formula with single constant

In this simple example, we suppose first-order formula (1) is the only rule we have, and its weight is w_1. There is

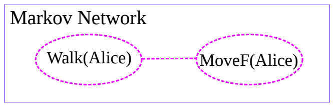

Fig. 2. Illustration of Markov logic network with a single formula (1) and a single constant (Alice).

only one constant $\{A\}$. Fig. 2 shows the corresponding Markov network. The X in equation (1) is a 2D variable $(MoveF(A), Walk(A))$. Thus there are totally 4 different probabilities in this world, or 4 different possible worlds, or 4 different states of the world: $x \in \{(0,0),(0,1),(1,0),(1,1)\}$.

Since we only have one first-order formula, i can only be $\{1\}$ in equation (3), and the summation is only for one item. Following this and the discussion in previous section, we have

$$P(0,0) = \frac{1}{Z}e^{w_1 * 1} = \frac{e^{w_1}}{Z}, \qquad P(0,1) = \frac{1}{Z}e^{w_1 * 1} = \frac{e^{w_1}}{Z}$$

$$P(1,0) = \frac{1}{Z}e^{w_1 * 0} = \frac{1}{Z}, \qquad P(1,1) = \frac{1}{Z}e^{w_1 * 1} = \frac{e^{w_1}}{Z}$$

where $Z = e^{w_1} + e^{w_1} + e^{0} + e^{w_1} = 3e^{w_1} + 1$ is the partition factor. $P(1,0) = \frac{1}{3e^{w_1}+1}$ is because when $MoveF(A) = 1$, $Walk(A) = 0$, the truth value of formula (1) is false and thus $n_1 = 0$. Except for this assignment, the formula is true for each of the other three groundings, which gives $n_1 = 1$. This finishes the MLN modeling, i.e. getting the joint distribution of $P(MoveF(A), Walk(A))$.

We verify whether MLNs generalize (or soften) first-order logic. First, it is obvious that the probability for $(MoveF(A) = 1, Walk(A) = 0)$ is not zero but $P(1,0) = \frac{1}{3e^{w_1}+1}$. So it "softens" the classic first-order logic requirement of either true or false. Moreover, the more confidence we have for the formula (i.e. larger w_1), the smaller the probability and thus the less likely this observation. If w_1 is small, it is likely that $(MoveF(A) = 1, Walk(A) = 0)$ happens because the knowledge expressed in the formula is not solid. Second, when given fact that A moves forward $(MoveF(A) = 1)$, the probability that A walks is

$$P(Walk(A) = 1 | MoveF(A) = 1) = \frac{P(Walk(A) = 1, MoveF(A) = 1)}{P(MoveF(A) = 1)}$$

where the marginal probability

$$P(MoveF(A) = 1) = \frac{1}{Z} + \frac{e^{w_1}}{Z} = \frac{e^{w_1}+1}{Z},$$

and the joint probability

$$P(Walk(A) = 1, MoveF(A) = 1) = \frac{1}{Z}e^{w_1 * 1} = \frac{e^{w_1}}{Z}.$$

Therefore,

$$P(Walk(A) = 1 | MoveF(A) = 1) = \frac{e^{w_1}}{e^{w_1}+1} \qquad (4)$$

From (4), as we become confident in the formula, the probability that A walks increases to 1,

$$\lim_{w_1 \to +\infty} P(Walk(A) = 1 | MoveF(A) = 1) = \lim_{w_1 \to +\infty} \frac{e^{w_1}}{e^{w_1}+1} = 1,$$

which is consistent with first-order logic. The same conclusion can be verified for $P(Walk(A) = 1 | MoveF(A) = 0)$ and $P(Walk(A) = 0 | MoveF(A) = 0)$.

3.2. More subjects: single formula with multiple constants

We move on to discuss the probability computation with multiple constants. Suppose that one more person Bob (B) joins in the system and thus $x \in \{A, B\}$. The only formula (1) now produces two grounding formulas, as shown in Fig. 3. It is a better example to demonstrate that MLN is a template for constructing Markov networks.

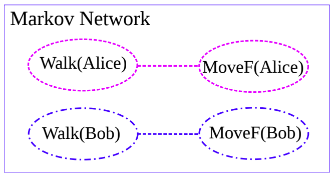

Fig. 3. Illustration of Markov logic network with a single formula (1) and multiple constants (Alice and Bob).

The world we are modelling in this case is 4D: $X = (MoveF(A), Walk(A), MoveF(B), Walk(B))$. We have to obtain the probability $P(MoveF(A), Walk(A), MoveF(B), Walk(B))$ through (3) given different truth value assignments to each of the four components.

What is the range of i in equation (3)? Is it $\{1, 2\}$ because we now have two grounding formulas and thus two cliques? The answer is no, and $i \in \{1\}$ instead of $i \in \{1, 2\}$. Careful readers may have noticed that true groundings have already been characterized in $n_i(x)$, or in order words, the equation (3) can be written as

$$P(X = x) = \frac{1}{Z}exp\left(\sum_{i}^{f}\sum_{j}^{NG_i} w_i \mathbf{1}_{F_i(x)}\right) \equiv \frac{1}{Z}exp\left(\sum_{i}^{f}\sum_{j}^{n_i(x)} w_i\right) \qquad (5)$$

where f is the number of original first-order formulas in Markov logic, NG_i is the number for groundings of F_i, and $\mathbf{1}_{F_i(x)}$ equals to 1 when $F_i(x)$ is true and 0 otherwise. This reorganized formula can be read as "for each first-order formula, add the weights whenever its grounding is true for given constants."

Since it involves a 4D binary world, i.e. each component of X can only be true or false, so we totally have $2^4 = 16$ different possible states of the world. When computing $P(X)$, we assume that we do not have only facts or evidence (such as $MoveF(A) = 0$). This is equivalently to assume that each of the 16 different states have equal occurrence probability. In this way, the computation method we discuss here can be easily extended for inference in MLNs, where the only difference is states have different occurrence probabilities given some evidence.

First, let's look at how the probability $P(X = (0, 0, 0, 0))$ is computed as an example, i.e. $MoveF(A) = 0$, $Walk(A) = 0$, $MoveF(B) = 0$, $Walk(B) = 0$. According to equation (3) or (5), how many true groundings of first-order formula (1) in (0,0,0,0)?

- MoveF(A)=0 and Walk(A)=0, so $MoveF(A) \Rightarrow Walk(A)$ is true;
- MoveF(B)=0 and Walk(B)=0, so $MoveF(B) \Rightarrow Walk(B)$ is true;

Therefore, there are $n_i(x) = 2$ true groundings of formula F_i ($i = 1$), i.e. formula (1). We only have one formula, so

$$\begin{aligned} P(0,0,0,0) &= \frac{1}{Z} exp\left(\sum_{i=1}^{1}\sum_{j}^{n_i(x)} w_i\right) \\ &= \frac{1}{Z} exp\left(\sum_{j=1}^{2} w_1\right) \\ &= \frac{1}{Z} e^{2w_1} \end{aligned}$$

Similarly, we can get the probabilities of different "possible worlds", shown in Table 1 (Z is omitted). From Table 1, we can get $Z = 9e^{2w_1} + 6e^{w_1} + 1$.

Once joint distribution $P(X)$ is obtained, different probabilities such as conditional probability and marginal probability can be easily computed. For example (MF short for MoveF, Wk for Walk)

$$\begin{aligned} &P(Wk(A) = 1, Wk(B) = 1|MF(A) = 1, MF(B) = 1) \\ &= \frac{P(MF(A) = 1, Wk(A) = 1, MF(B) = 1, Wk(B) = 1)}{P(MF(A) = 1, MF(B) = 1)} \end{aligned} \quad (6)$$

where the joint probability in the numerator is in Table 1, and the marginal probability in denominator can be obtained by adding items in Table 1 where (MF(A)=1 and MF(B)=1):

$$P(S(A) = 1, S(B) = 1) = \frac{e^{2w_1} + 2ew_1 + 1}{Z}$$

Table 1. Probabilities of different worlds. (MF stands for MoveF and Wk for Walk.)

X				$n_i(x)$	$e^{n_i(x)*w_i}$
MF(A)	Wk(A)	MF(B)	Wk(B)		
0	0	0	0	2	e^{2w_1}
0	0	0	1	2	e^{2w_1}
0	0	1	0	1	e^{w_1}
0	0	1	1	2	e^{2w_1}
0	1	0	0	2	e^{2w_1}
0	1	0	1	2	e^{2w_1}
0	1	1	0	1	e^{w_1}
0	1	1	1	2	e^{2w_1}
1	0	0	0	1	e^{w_1}
1	0	0	1	1	e^{w_1}
1	0	1	0	0	1
1	0	1	1	1	e^{w_1}
1	1	0	0	2	e^{2w_1}
1	1	0	1	2	e^{2w_1}
1	1	1	0	1	e^{w_1}
1	1	1	1	2	e^{2w_1}

Therefore, the probability in (6) is

$$\begin{aligned} &P(Wk(A) = 1, Wk(B) = 1|MF(A) = 1, MF(B) = 1) \\ &= \frac{e^{2w_1}}{e^{2w_1} + 2e^{w_1} + 1} \end{aligned}$$

It becomes one when $w_1 \rightarrow +\infty$. Similarly, we can get $P(Wk(A) = 0, Wk(B) = 0|MF(A) = 1, MF(B) = 1) = \frac{1}{e^{2w_1}+2e^{w_1}+1}$, and it goes to zero when $w_1 \rightarrow +\infty$.

3.3. More knowledge: multiple formulas with multiple constants

The discussion above establishes a basis on which more complex cases can be solved. We can easily extend the probability computation from knowledge base with a single formula to that with multiple ones. Suppose one more knowledge is extracted and expressed with formula (2), i.e. any person near a walking person is likely walking too. The corresponding Markov network with two subjects (Alice and Bob) is illustrated in Fig. 1. In this example, formula (1) has two groundings while formula (2) has one. The feature X in this world is 5D: (MF(A), Wk(A), MF(B), Wk(B), Near(A,B)).

The procedure to calculate the probability is the same as the previous two cases, and the only difference is that we have to enumerate i over 1,...,N=2 when computing the probability, where N is the total number of first-order formulas. For each i, the procedure is exactly the same as it is shown in section "single formula with multiple constants." When computing $P(x)$, $\sum_j^{n_1(x)} w_1$ and $\sum_j^{n_2(x)} w_2$ can be performed separately, so the problem is decomposed into the cases discussed in Section 3.2 and Section 3.1, respectively. No matter how many formulas the knowledge base has, the computation of $P(X)$ can always be expressed as a product of single formula cases. This becomes clearer from the equation (5). Therefore, we omit the computation procedure for simplicity.

4. APPLICATION ON ACTIVITY ANALYSIS

In this section, we show an application of Markov logic network for video action recognition. Firs-order logic is used to represent the relationship among low-level action features. The temporal relationships are often employed in logic systems for action recognition, as in [7][8]. They are frequently expressed using Allen temporal predicates [9]. To our best knowledge, no experimental results have been shown on widely used public dataset. Since there is no source code available for those existing approaches, we implemented our logic rules for actions in Weizmann dataset according to the guideline described in [7]. We used Alchemy[1] as our inference engine as used in [8].

Dataset

We used Weizmann dataset in this demonstration application. The Weizmann dataset consists of 90 low-resolution (144x180 pixels) video sequences showing nine different persons, each performing 10 natural actions: bending, jumping, jumping-in-place (pjump), jacking, running, gallop sideways (side), skipping, walking, waving one hand (wave1) and waving two hands (wave2), as shown in Figure 4.

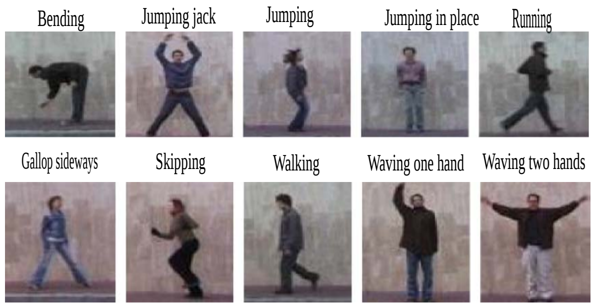

Fig. 4. Sample frames from Weizmann dataset.

Feature extraction

Preprocessing is needed to describe actions using logic rules, including feature (object-based trajectories) extraction, predicates design and grounding, and action rule formalization. We semi-automatically extracted trajectories of human body parts from the Weizmann dataset. The actions analyzed include bending, jacking, jumping, jumping in place, skipping, walking, single-hand waving and two-hand waving. We excluded running because the tracker could not give reasonable trajectories. Two videos from each action category were selected. Each video may contain a different number of action occurrences. For each video, we first manually marked the body parts (head, hands and feet) to track, and then ran TLD tracker[2] to generate trajectories for them. The first row of Figure 5 shows examples of the trajectories. Depending on the chosen object trackers, the extracted trajectories may not always be accurate. In Figure 5(d), for example, the TLD tracker confuses the two hands when they are close to each other. The trajectories were then segmented according to the moving directions, as illustrated in Figure 6 for the right-hand trajectory in skipping.

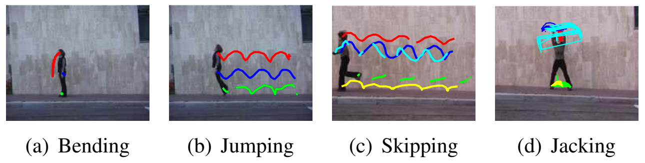

Fig. 5. Examples of trajectories from object-based tracking.

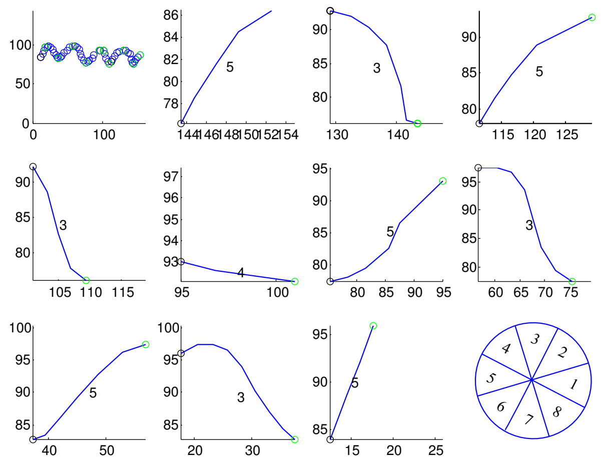

Fig. 6. Segmentation of trajectory of a body part. The top left is the original trajectory for right hand, the bottom right is the direction assignment, and the others are the segmentation results. The x and y axes are the coordinates of trajectory points.

Logic representation design

Based on these trajectory segments, we generated candidate time intervals during which an action happens. The relationships between two intervals were described by Allen interval predicates. To describe the spatio-temporal change of a trajectory, we defined predicate Move($\boldsymbol{P}$, $\boldsymbol{BP}$, $\boldsymbol{I}$, $\boldsymbol{D}$) and MoveInSX($\boldsymbol{P}$, $\boldsymbol{BP}$, $\boldsymbol{I}$) where SX is a quantized movement

[1]http://alchemy.cs.washington.edu/

[2]http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html

Table 2. Example predicates and rules to define actions.

I. Predicates

Move(***P***, ***BP***, ***I***, ***D***)	Person ***P***'s body part ***BP*** moves in direction ***D*** during interval ***I***
MoveInS1(***P***, ***BP***, ***I***)	Person ***P***'s body part ***BP*** moves ***S1*** (length) during interval ***I***
Bend(***P***, ***I***)	Person ***P*** performs bending during interval ***I***
. . .	. . .
Jump(***P***, ***I***)	Person ***P*** performs jumping during interval ***I***
Meet(***I1***,***I2***),Before(***I1***,***I2***), Finish(***I1***, ***I2***), Start(***I1***,***I2***), During(***I1***,***I2***),· · ·	Allen temporal predicates

II. Rules

Move(***P***, Head, ***I1***, D6) $\bigwedge$ MoveInS1(***P***, Head, ***I1***) $\bigwedge$ Move(***P***, Head, ***I2***, D2) $\bigwedge$
MoveInS1(***P***, Head, ***I2***) $\bigwedge$ Start(***I1***,***I3***) $\bigwedge$ Finish(***I2***, ***I3***) $\bigwedge$ Meet(***I1***, ***I2***)
=> Bend(***P***, ***I3***)
. . .

Table 3. Accuracy from Markov logic on Weizmann dataset.

Action	bend	jack	jump	pjump
Accuracy	100%	80%	80%	100%
Action	skip	walk	wave1	wave2
Accuracy	86%	40%	75%	83%

scale. All the rules for actions were in first-order logic forms, as shown in Table 2 for some examples. For a complete list of predicates and rules, refer to appendix at http://students.cse.unt.edu/~gc0115/mln-ar-icccnt5.pdf.

Results

Table 3 shows the recognition accuracy of the logic-based approach. The average accuracy for the logic-based system is 76.7%. The accuracy depends on the rules to express the actions, but the obtained accuracy is similar to the results reported in [7] though different test videos were used. We noticed that the accuracy for some actions such as walking is lower. This may result from the fact that multiple similar actions (jumping and walking) exist in the dataset, and the trajectory detection extracts erroneous evidence.

5. CONCLUSIONS

We provided an introduction to Markov logic networks, which focuses on the computation of probability and explains it using examples and an application for video action recognition. While Markov logic networks have been proposed and applied for different tasks, this work demonstrates the computation of probability in Markov logic network, and shows how Markov logic combines logic representation and probabilistic reasoning. It also shows how MLNs could be used for video action analysis. This paper provides preparation for MLN practitioners in the hope to promote its application.

While we only show the results on the dataset with only one subject, it is straightforward to extend the work into scenarios where multiple subjects present as long as successful labeling of objects is ready. This can be done in the same way we deal with different body parts, and is our future work.

6. REFERENCES

[1] Matthew Richardson and Pedro Domingos, "Markov logic networks," *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[2] Son D. Tran and Larry S. Davis, "Event modeling and recognition using Markov logic networks," in *Proc. European Conference on Computer Vision*, Marseille France, 2008, pp. 610–623.

[3] Vlad I. Morariu and Larry S. Davis, "Multi-agent event recognition in structured scenarios," in *IEEE Computer Vision and Pattern Recognition*, Colorado Springs CO, 2011, pp. 3289–3296.

[4] Chen Wu and H. Aghajan, "User-centric environment discovery with camera networks in smart homes," *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 2, pp. 375–383, march 2011.

[5] Mimaanshu Gupta, Li Yu, Asaad Hakeem, Tae Eun Choe Niels Haering, and Michael Locasto, "Multimodal complex event detection framework for wide area surveillance," in *IEEE Workshop on Camera Networks and Wide Area Scene Analysis*, Colorado Springs CO, 2011, pp. 47–54.

[6] Stuart J. Russell and Peter Norvig, *Artificial intelligence: A modern approach*, Prentice hall, Upper saddle River, New Jersey, 2 edition, 2003.

[7] Vlad Morariu and Larry Davis, “Multi-agent event recognition in structured scenarios,” in *IEEE Conf Computer Vision Pattern Recognition*, 2011, pp. 3289–3296.

[8] Son D. Tran and Larry S. Davis, “Event modeling and recognition using Markov logic networks,” in *European Conf Computer Vision*, 2008, pp. 1–14.

[9] James Allen, “Maintaining knowledge about temporal intervals,” *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.