Study of Markov Logic Networks in Entity Resolution

Prashanth Kamle (08305006) and Anup Kulkarni (08305045)

May 20, 2009

Contents

1 Introduction

A Markov logic network (or MLN) is a probabilistic logic which applies the ideas of a Markov network to first-order logic. Markov logic networks generalize first-order logic, in the sense that all unsatisfiable statements have a probability of zero, and all entailed formulas have probability one.

An MLN is obtained by attaching weights to clauses in a first order knowledge base. This can be viewed as a template for constructing normal Markov networks. Every possible grounding of a clause in the KB is taken as a feature in the constructed network. Inference can be performed by grounding the minimal subset of the network required for answering the query and running a Gibbs sampler over this subnetwork, with initial states found by MaxWalkSat. Weights can be learnt from relational databases by iteratively optimizing a pseudo-likelihood measure (say, using the L-BFGS algorithm). Optionally, clauses can be learnt using ILP techniques (such as CLAUDIEN).

Entity resolution is the problem to determine which records refer to same real world entity. The real world entity may be anything like name of author, name of conference, venue etc.. These same entities can be written by different ways like name of author can be given by A. McCaullum and Andrew McCaullum refers to same entity. The task is to identify such entries correctly. This report gives is a survey of the MLN based method for entity resolution.

2 Markov Network

A Markov network, Markov random field, or undirected graphical model is a model of the joint probability distribution of a set X of random variables having the Markov property. It is made up of an undirected graph G and a set of potentials functions f_k. Every node in the graph is associated with a random variable, and a potential function is assigned to every clique in the graph. The joint distribution represented by a Markov network is given by

$$P(X = x) = \frac{1}{Z} \prod_k f_k(x_{\{k\}}) \tag{1}$$

where $x_{\{k\}}$ is the state of variables that appear in the k^{th} clique. Z, called the partition function, is given by

$$Z = \sum_{x \in \mathcal{X}} \prod_k f_k(x_{\{k\}}). \tag{2}$$

Usually, Markov networks are represented as log-linear models, with each clique potential represented as an exponentiated weighted sum of features of the state, as below

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_k w_k^\top \phi_k(x_{\{k\}}) \right) \tag{3}$$

Exact inference is a $\#P$-complete problem, and thus computationally intractable in the general case. Approximation techniques such as Markov chain Monte Carlo and loopy belief propagation are often more feasible in practice. MAP estimates of Markov network weights cannot be computed in closed form, but, because the log-likelihood is a concave function of the weights, they can be found efficiently using standard gradient-based or quasi-Newton optimization methods (L-BFGS).

3 First Order Logic

A first order logic knowledge base is made up of constants, variables, functions and predicates. Constant symbols represent objects in the domain of interest (Eg: $Alice, Bob, Charles$). Variable symbols range over the objects in the domain. Function symbols represent mappings from tuples of objects to objects (Eg: $Alice = motherOf(Bob)$). Predicate symbols represent relations among objects in the domain or attributes of objects (Eg: $friends(Bob, Charles)$). An interpretation maps symbols to objects, functions and relations in the domain. Variables and constants may be typed. In this case, variables range only over objects of the corresponding type (Eg: A variable of type *people*, x may range over $Alice, Bob, Charlesetc.$, and constants can represent only objects of the corresponding type.

A term is any expression representing an object in a domain. A term can be a constant, a variable or a function. For example, $Alice, MotherOf(Alice), x$ are all terms. An atom is a predicate symbol applied to a tuple of terms (Eg: $friends(x, y)$). Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers.

The set of well-formed formulas(wff) is recursively defined by the following rules:

1. Simple and complex predicates If P is a relation of arity n and $a1, ..., an$ are terms then $P(a1, ..., an)$ is a well-formed formula. If equality is considered part of logic, then $(a1 = a2)$ is a well-formed formula.

2. Inductive Clause I: If φ is a wff, then $\neg\varphi$ is a wff.

3. Inductive Clause II: If φ and ψ are wffs, then $(\varphi \rightarrow \psi)$ is a wff.

4. Inductive Clause III: If φ is a wff and x is a variable, then $\forall x, \varphi$ and $\exists x, \varphi$ are wffs.

English	First-Order Logic	Clausal Form	Weight
Friends of friends are friends.	$\forall x \forall y \forall z\ \mathtt{Fr}(x,y) \wedge \mathtt{Fr}(y,z) \Rightarrow \mathtt{Fr}(x,z)$	$\neg\mathtt{Fr}(x,y) \vee \neg\mathtt{Fr}(y,z) \vee \mathtt{Fr}(x,z)$	0.7
Friendless people smoke.	$\forall x\ (\neg(\exists y\ \mathtt{Fr}(x,y)) \Rightarrow \mathtt{Sm}(x))$	$\mathtt{Fr}(x,\mathtt{g}(x)) \vee \mathtt{Sm}(x)$	2.3
Smoking causes cancer.	$\forall x\ \mathtt{Sm}(x) \Rightarrow \mathtt{Ca}(x)$	$\neg\mathtt{Sm}(x) \vee \mathtt{Ca}(x)$	1.5
If two people are friends, either	$\forall x \forall y\ \mathtt{Fr}(x,y) \Rightarrow (\mathtt{Sm}(x) \Leftrightarrow \mathtt{Sm}(y))$	$\neg\mathtt{Fr}(x,y) \vee \mathtt{Sm}(x) \vee \neg\mathtt{Sm}(y)$,	1.1
both smoke or neither does.		$\neg\mathtt{Fr}(x,y) \vee \neg\mathtt{Sm}(x) \vee \mathtt{Sm}(y)$	1.1

Figure 1: Example of a first-order knowledge base and MLN. $Fr()$ is short for $Friends()$, $Sm()$ for $Smokes()$, and $Ca()$ for $Cancer()$

5. Closure Clause: Nothing else is a wff.

For example, $\forall x \forall y(P(f(x)) \rightarrow \neg(P(x) \rightarrow Q(f(y), x, z)))$ is a well-formed formula, if f is a function of arity 1, P a predicate of arity 1 and Q a predicate of arity 3. $\forall x, x \rightarrow$ is not a well-formed formula.

A positive literal is an atomic formula; a negative literal is a negated atomic formula. The formulas in a KB are implicitly conjoined, and thus a KB can be viewed as a single large formula. A ground term is a term containing no variables. A ground atom or ground predicate is an atomic formula all of whose arguments are ground terms. A possible world or Herbrand interpretation assigns a truth value to each possible ground atom.

A formula is satisfiable iff there exists at least one world in which it is true. The basic inference problem in first-order logic is to determine whether a knowledge base KB entails a formula F , i.e., if F is true in all worlds where KB is true (denoted by $KB \models F$). This is often done by refutation: KB entails F iff KB $\cup\neg F$ is unsustainable. It is to be noted that inference in first order logic is only semi decidable.

4 Markov Logic Networks

A first order knowledge base is a set of hard constraints. A Markov Logic Network intends to make these constraints soft by associating a weight to every constraint. The weight indicates how influential a constraint is. Greater the weight, more important is the constraint. In a first order knowledge base, if a world violates a formula, then it is termed impossible. But in MLNs, if a constraint is violated by a world, it simply becomes less probable.

As defined in [2], a Markov logic network L 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. Together with a finite set of constants $C = c_1, c_2, ..., c_{|C|}$, it denes a Markov network $M_{L,C}$ as follows:

1. $M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground atom is true, and 0 otherwise.

2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.

An MLN can looked at as a template for constructing Markov networks. For a given set of constants, a particular Markov network is produced. Markov networks produced by different sets of constants are different, but they contain certain regularities in structure and parameters. For example, all groundings of the same formula will have the same weight. Each of these networks is called a Ground Markov

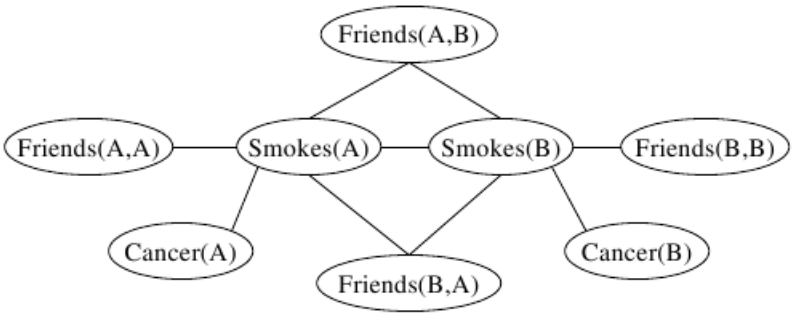

Figure 2: Ground Markov network obtained by applying the last two formulas in Figure 1 to the constants Alice(A) and Bob(B)

Network. From Equations 1 and 3, the probability distribution over possible worlds x specified by ground Markov network $M_{L,C}$ is given by

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) = \frac{1}{Z} \prod_i \phi_i \left(x_{\{i\}}\right)^{n_i(x)} \tag{4}$$

where $n_i(x)$ is the number of true groundings of F_i in x, $x_{\{i\}}$ is the truth values (state) of the atoms appearing in F_i, and $\phi_i \left(x_{\{i\}}\right) = e^{w_i}$.

The graphical structure of $M_{L,C}$ follows from the above definition. Explicitly, there is an edge between 2 nodes of $M_{L,C}$ iff the corresponding ground atoms appear together in at least one grounding of a formula in L. Thus, atoms in each ground formula form a clique in $M_{L,C}$

Figure 2 shows the ground Markov network define by the last 2 formulas in Figure 1. Each node in this graph is a ground atom (Eg. $Friends(Alice, Bob)$). The graph contains an arc between each pair of atoms that appear together in some grounding of one of the formulas. $M_{L,C}$ can now be used to infer the probability that Alice and Bob are friends given their smoking habits, the probability that Bob has cancer given his friendship with Alice and whether she has cancer, etc.

Each $M_{L,C}$ denotes a possible world. A world is a set of objects, a set of functions which map a tuple of objects to an object, and the relations that hold between the objects. With an interpretation, they determine the truth value of each ground atom. To ensure that the set of possible worlds is finite, the following assumptions are made.

1. Unique names: Different constants refer to different objects
2. Domain closure: The only objects in the domain are those representable using the constant and function symbols in (L, C).
3. Known functions: For each function appearing in L, the value of that function applied to every possible tuple of arguments is known, and is an element of C. (This allows us to replace function calls by their values while grounding formulas)

Figure 3 shows how the groundings of a formula are obtained under the assumptions.

4.1 Example

To see how MLNs generalize first order logic, let us consider an MLN consisting of the single formula $\forall x, R(x) \Rightarrow S(x)$ with weight w, and $C = A$. This leads to 4 possible worlds.

function Ground(F, C)
  inputs: F, a formula in first-order logic
    C, a set of constants
  output: G_F, a set of ground formulas
  calls: $CNF(F, C)$, which converts F to conjunctive normal form, replacing existentially quantified formulas by disjunctions of their groundings over C
$F \leftarrow CNF(F, C)$
$G_F = \emptyset$
for each clause $F_j \in F$
  $G_j = \{F_j\}$
  for each variable x in F_j
    for each clause $F_k(x) \in G_j$
      $G_j \leftarrow (G_j \setminus F_k(x)) \cup \{F_k(c_1), F_k(c_2), \ldots, F_k(c_{|C|})\}$,
        where $F_k(c_i)$ is $F_k(x)$ with x replaced by $c_i \in C$
  $G_F \leftarrow G_F \cup G_j$
for each ground clause $F_j \in G_F$
  repeat
    for each function $f(a_1, a_2, \ldots)$ all of whose arguments are constants
      $F_j \leftarrow F_j$ with $f(a_1, a_2, \ldots)$ replaced by c, where $c = f(a_1, a_2, \ldots)$
  until F_j contains no functions
return G_F

Figure 3: Algorithm to construct all grounding of a formula under the assumptions

1. $\neg R(A), \neg S(A)$
2. $\neg R(A), S(A)$
3. $R(A), \neg S(A)$
4. $R(A), S(A)$

From Equation 4, we obtain $P(R(A), \neg S(A)) = \frac{1}{3e^w+1}$ and the probability of each of the other 3 worlds is $\frac{1}{e^{-w}+1}$. If $w > 0$, the effect of the MLN is to make the world that is inconsistent with $\forall x, R(x) \Rightarrow S(x)$ less likely than the other three. When $w \to \infty$, $P(S(A)|R(A)) \to 1$, recovering the logical entailment.

4.2 Inferencing

MLNs can answer arbitrary queries of the form "What is the probability the formula F_1 holds given that formula F_2 does?". If F_1 and F_2 are 2 formulas in first order logic, C is a finite set of constants and L is an MLN, then

$$P(F_1|F_2, L, C) = P(F_1|F_2, M_{L,C}) \tag{5}$$

$$= \frac{P(F_1 \wedge F_2|M_{L,C})}{P(F_2|M_{L,C})} \tag{6}$$

$$= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2}} P(X = x|M_{L,C})}{\sum_{x \in \mathcal{X}_{F_2}} P(X = x|M_{L,C}} \tag{7}$$

where $\mathcal{X}_{\mathcal{F}_)}$ is the set of worlds where F_i holds, and $P(x|M_{L,C})$ is given by Equation 4. To check whether a KB entails a formula F ins first order logic, we need to check whether $P(F|L_{KB}, C_{KB,F}) = 1$, where

function ConstructNetwork(F_1, F_2, L, C)
  inputs: F_1, a set of ground atoms with unknown truth values (the "query")
    F_2, a set of ground atoms with known truth values (the "evidence")
    L, a Markov logic network
    C, a set of constants
  output: M, a ground Markov network
  calls: $MB(q)$, the Markov blanket of q in $M_{L,C}$
$G \leftarrow F_1$
while $F_1 \neq \emptyset$
  for all $q \in F_1$
    if $q \notin F_2$
      $F_1 \leftarrow F_1 \cup (MB(q) \setminus G)$
      $G \leftarrow G \cup MB(q)$
    $F_1 \leftarrow F_1 \setminus \{q\}$
return M, the ground Markov network composed of all nodes in G, all arcs between them in $M_{L,C}$, and the features and weights on the corresponding cliques

Figure 4: Algorithm for network construction for inference in MLNs

L_{KB} is the MLN obtained by assigning infinite weight to all formulas in the KB, and C_{KBF}) is the set of all constants appearing in the KB or F. This can be done by computing $P(F|L_{KB}, C_{KB,F})$ using Equation 7, with $F_2 = True$.

But computing Equation 7 is intractable. So, a 2 phase process is adapted for inferencing. This works only for the case where F_1 and F_2 are conjunctions of ground literals. While less general that Equation 7, this is the most frequent type of query in practice, and the following algorithm provides answers to it far more efficiently than a direct application of Equation 7.

Phase 1: This returns the minimal subset M of the ground Markov network required to compute $P(F_1|F_2, L, C)$. The algorithm for this is shown in Figure 4. In the worst case, the network contains $\mathcal{O}(|C|^a)$ nodes, where a is the largest predicate arity in the domain.

Phase 2: This phase performs inference on the network obtained from Phase 1, with nodes in F_2 set to their values in F_2. Gibbs sampling can be employed here. The probability of a ground atom X_l when its Markov blanket B_l is in state b_l is

$$P(X_l = x_l|B_l = b_l) = \frac{\exp\left(\sum_{f_i \in F_l} w_i f_i(X_l = x_l, B_l = b_l)\right)}{\exp\left(\sum_{f_i \in F_l} w_i f_i(X_l = 0, B_l = b_l)\right) + \exp\left(\sum_{f_i \in F_l} w_i f_i(X_l = 1, B_l = b_l)\right)} \quad (8)$$

where F_l is the set of ground formulas that X_L appears in, and $f_i(X_l = x_l, B_l = b_l)$ is the value of the feature corresponding to the ith ground formula when $X_l = x_l$ and $B_l = b_l$.

The estimated probability of a conjunction of ground literals is simply the fraction of samples in which the ground literals are true, after the Markov chain has converged. Because the distribution is likely to have many modes, the Markov chain algorithm has to be run multiple times. When the MLN is in clausal form, the burn-in time is minimized by starting each run from a mode found using MaxWalkSat, a local search algorithm for the weighted satisability problem (i.e., finding a truth assignment that maximizes the sum of weights of satisfied clauses). When there are hard constraints (clauses with infinite weight), MaxWalkSat finds regions that satisfy them, and the Gibbs sampler then samples from these regions to obtain probability estimates.

4.3 Training

In training phase, weights w_i associated with every formula are learned. The weights are learned by gradient descent algorithm by maximizing log-likelihood of y given x.

$$\frac{\partial}{\partial w_i} log \Pr_w(y|x) = n_i(x,y) - E_w[n_{i(x,y)}] \tag{9}$$

This equation gives derivative of conditional log-likelihood with respect to the weight of the ith clause. Here $n_i(x,y)$ is the number of true groundings of the ith clause in the data $E_w[n_{i(x,y)}]$ is the clause's expected number of true groundings, averaged over all possible worlds, weighted by their probabilities according to the current weights. Thus while training if actual count of the clause in the KB is more than calculated the weight of the clause should increase in the next iteration and vice versa.

Since computation of E involves computation of Z which is intractable. So E is approximated by the counts $n_i(x, y_w^*)$ in the MAP state $y_w^*(x)$. If most of the probability mass of $\Pr_w(y|x)$ is concentrated around $y_w^*(x)$. This approximation is essence of voted perceptron algorithm which initializes all weights to zero and after T steps of gradient descent returns weights

$$w_i = \sum_{t=1}^{T} w_{i,t}/T$$

5 Entity Resolution

Entity resolution is the problem to determine which records refer to same real world entity. The real world entity may be anything like name of author, name of conference, venue etc. These same entities can be written by different ways like name of author can be given by A. McCaullum and Andrew McCaullum refers to same entity. The task is to identify such entries correctly.

5.1 Knowledge Base and Rules

For object deduplication (also called as entity resolution) application we use unique name assumption. Unique name assumption says that: different constants refer to different objects in the domain. This can be written in predicate form as Equals(x, y) or x = y

Unique name assumption have three axioms as:

1. Reflexivity:
$$\forall x, \mathrm{x} = \mathrm{x}$$
2. Symmetry:
$$\forall x, y \ \mathrm{x} = \mathrm{y} \implies \mathrm{y} = \mathrm{x}$$
3. Transitivity:
$$\forall x, y, z \ \mathrm{x} = \mathrm{y} \wedge \ \mathrm{y} = \mathrm{z} \implies \ \mathrm{a} = \mathrm{z}$$

Predicate equivalence is given as

Predicate equivalence: for each binary predicate R:

$$\forall x_1, x_2, y_1, y_2 \ x_1 = x_2 \wedge y_1 = y_2 \implies (R(x_1, y_1), R(x_2, y_2))$$

Thus we can use this formula to see if two objects have are same if their some of fields are same. Similarly we can add Reverse Predicate Equivalence as

Predicate equivalence: for each binary predicate R:

$$\forall x_1, x_2, y_1, y_2 (R(x_1, y_1) \wedge R(x_2, y_2)) \implies (x_1 = x_2 \wedge y_1 = y_2)$$

These formula are best explained as: if two objects are in the same relation to the same object, this is evidence that they may be the same object.

It is obvious that some relations provide better evidence than others. For example, when deduplicating citations, two papers having the same title is stronger evidence that they are the same paper than them having the same author, which in turn is stronger evidence than them having the same venue. However, even the latter is quite useful: two papers appearing in the same venue are much more likely to be the same than two papers about which nothing is known.

Thus the problem for citation deduplication is formulated using these definitions.

5.2 Problem formulation

[3] describes predicates used for citation deduplication. Here, n-ary relation is represented using n binary relations. For example, if a citation database contains ground atoms of the form Paper(title, author, venue), they can be replaced by atoms of the form HasTitle(paper, title), HasAuthor(paper, author) and HasVenue(paper, venue). Each real-world entity (e.g., each paper, author or venue) is represented by one or more strings appearing as arguments of ground atoms in the database. For example, different atoms could contain the strings ICDM-2006, Sixth ICDM and IEEE ICDM'06, all of which represent the same conference.

All the fields are assumed to be typed. for example, the first argument of the predicate HasAuthor(paper, author) is of type Paper, and the second is of type Author. Our goal is to find if two author names corresponds to same author or not. This problem reduces to finding: for each pair of constants of the

same type: (x1, x2), to determine whether they represent the same entity: is x1 = x2? The most likely truth assignment to the query atoms given the evidence is computed using MaxWalkSAT. Conditional probabilities of query atoms given the evidence are calculated using Gibbs sampling. This model includes unit clause for each query predicate. The weight of unit clause used to capture the marginal distribution of the corresponding predicate. The non unit clauses (mainly binary clauses) capture the relationship between the predicates.

5.3 Field Comparison: deduplication

The object to be deduplicated is citation which is mainly in the form of string tokens. For which [3] defines a predicate HasWord(field, word) which is true iff field contains word. Using this predicate and reverse predicate equivalence we can say:

$$\forall x_1, x_2, y_1, y_2(\text{HasWord}(x_1, y_1) \wedge \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \Longrightarrow x_1 = x_2$$

This means, if fields have same words then those fields are same. Now this predicate can be put in probabilistic model as,

$$\Pr(x_1 = x_2|n) = \frac{exp(w \times n)}{Z}$$

Where w is weight of formula and n is the number of times $x_1 = x_2$.
The negative version of reverse equivalence is also added as

$$\forall x_1, x_2, y_1, y_2(\ \text{HasWord}(x_1, y_1) \wedge \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \Longrightarrow x_1 \neq x_2$$

$$\forall x_1, x_2, y_1, y_2(\text{HasWord}(x_1, y_1) \wedge\ \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \Longrightarrow x_1 \neq x_2$$

$$\forall x_1, x_2, y_1, y_2(\ \text{HasWord}(x_1, y_1) \wedge\ \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \Longrightarrow x_1 = x_2$$

5.4 Issues in this approach

This approach has the disadvantage that it treats misspellings, variant spellings and abbreviations of a word as completely. different words. Since these are often a significant issue in entity resolution, it would be desirable to account for them. One way to do this efficiently is to compare word strings by the engrams they contain. This can be done in our framework by defining the predicate HasEngram(word, engram), which is true iff engram is a substring of word.

This model is further enhanced by The Fellegi-Sunter Model in which relation R is used as the relation between a ?field and the record it appears in and applied in reverse predicate equivalence as

$$\forall x_1, x_2, y_1, y_2(\text{HasWord}(x_1, y_1) \wedge \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \wedge R(z_1, x_1) = R(z_2, x_2) \Longrightarrow z_1 = z_2$$

$$\forall x_1, x_2, y_1, y_2(\ \text{HasWord}(x_1, y_1) \wedge \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \wedge R(z_1, x_1) = R(z_2, x_2) \Longrightarrow z_1 \neq z_2$$

$$\forall x_1, x_2, y_1, y_2(\text{HasWord}(x_1, y_1) \wedge\ \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \wedge R(z_1, x_1) = R(z_2, x_2) \Longrightarrow z_1 \neq z_2$$

$$\forall x_1, x_2, y_1, y_2(\ \text{HasWord}(x_1, y_1) \wedge\ \text{HasWord}(x_2, y_2)) \wedge y_1 = y_2 \wedge R(z_1, x_1) = R(z_2, x_2) \Longrightarrow z_1 = z_2$$

This means if fields x_1 and x_2 are same and they are related to some objects through same relation then those objects are also likely to be same.

System	Citation		Author		Venue	
	CLL	AUC	CLL	AUC	CLL	AUC
NB	−0.637±0.010	0.913±0.000	−0.133±0.021	0.986±0.000	−0.747±0.017	0.738±0.002
MLN(B)	−0.643±0.010	0.915±0.000	−0.131±0.022	0.987±0.000	−0.760±0.017	0.736±0.002
MLN(B+C)	−0.809±0.012	0.891±0.000	−0.386±0.064	0.968±0.000	−1.163±0.034	0.741±0.001
MLN(B+T)	−0.369±0.003	0.949±0.000	−0.213±0.036	0.994±0.000	−1.036±0.029	0.745±0.002
MLN(B+C+T)	−0.597±0.007	0.964±0.000	−0.171±0.043	0.984±0.000	−0.704±0.023	0.828±0.002
MLN(B+C+T+S)	−0.503±0.006	0.988±0.000	−0.100±0.033	0.992±0.000	−0.874±0.027	0.807±0.002
MLN(B+N+C+T)	−0.879±0.008	0.952±0.000	−0.096±0.032	0.992±0.000	−0.781±0.023	0.817±0.002
MLN(G+C+T)	−0.394±0.004	0.973±0.000	−0.263±0.053	0.980±0.000	−1.196±0.031	0.743±0.002

6 Accuracy and Efficiency of learnt model

6.1 Model Variations

Model is tested considering following variations:

1. MLN(B): It is most basic model. It has the four reverse predicate equivalence rules connecting each word to the corresponding field/record match predicate.

2. MLN(B+C). This is obtained by adding reverse predicate equivalence rules for the various fields to MLN(B).

3. MLN(B+T). This is obtained by adding transitive closure rules to MLN(B).

4. MLN(B+C+T). This model has both reverse predicate equivalence and transitive closure rules.

5. MLN(B+C+T+S). This model is obtained by adding rules learned using the structure learning algorithm by [1]. [1] proposes method to refine KB generated by human experts or a way to design own KB from scratch. For building KB from scratch [1] uses addition of literal as basic operator to build MLN from scratch. When adding a literal to a clause, [1] considers all possible ways in which the literal's variables can be shared with existing ones, subject to the constraint that the new literal must contain at least one variable that appears in an existing one. The size of the search space will increase exponentially with this. So it is limited by limiting the number of distinct variables in a clause. While modifying hand coded rules, literals are removed only when it leaves at least one path of shared variables between each pair of remaining literals. An example of a rule added by structure learning is: if two papers have the same title and the same venue, then they are the same paper.

6. MLN(B+N+C+T). This model has a two-level learning/inference step. The first step involves learning a model to predict if two word mentions are the same based on the n-grams they have in common (we used n=3). This step incorporates word-level transitivity and reverse predicate equivalence rules. The second step involves learning an MLN(B+C+T) model on the words inferred by the first stage.

7. MLN(G+C+T). This model is similar to MLN(B+C+T) except that it does not have per word reverse predicate equivalence rules. Rather, it has four global rules, each with the same weight for all words, and these weights are learned discriminatively, like the rest.

The performance is given in table 6.1. The dataset used was CORA.

6.2 Analysis of performance

For venue best performing model is MLN(B+C+T). Performance gradually increases from basic model to BCT model. This trend shows how adding each feature in turn enhances the performance, giving the

largest improvement when all the features are added to the model. For venues conditional log-likelihood is max for again BCT model.

MLN(B+C+T+S) helps improve the performance of MLN(B+C+T) on citations and authors.

MLN(B+N+C+T) improves performance on authors but not on citations and venues. This shows that while stemming can be a good way of identifying word duplicates in most cases, the n-gram model is quite helpful when dealing with fields such as authors, where initials are used for the complete word and can not be discovered by stemming alone.

The global model MLN(G+C+T) performs well for citations but less so for authors and venues. In general, per-word rule models seem to be particularly helpful when there are few words from which to infer the relationship between two entities.

7 Conclusion

This report studied a unifying framework for entity resolution. We studied how logic and probability can be combined into a graphical model for inferencing on a probabilistic knowledge base. We then saw how a small number of axioms in Markov logic capture the essential features of citation records. Experiments on two citation databases evaluate the contribution of this approach, and illustrate how Markov logic enables us to easily build a sophisticated entity resolution system.

References

[1] Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 441–448, New York, NY, USA, 2005. ACM.

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

[3] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *In ICDM*, pages 572–582. IEEE Computer Society Press, 2006.