# Application of Markov Logic Networks to Entity Resolution

## An Introduction

Anup Kulkarni and Prashanth Kamle

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

April 30, 2009

# Outline

# Markov Network

- A Markov network is a model of the joint probability distribution of a set $X$ of random variables having the Markov property.
- Constituents
  - An undirected graph $G$
  - A set of potentials functions $f_k$
  - Every node is associated with a random variable
  - A potential function is associated with every clique

# Joint distribution

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

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

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

# Markov Network as a log-linear model

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

- Exact inference is *NP*-complete
- Approximation techniques such as Markov Chain Monte Carlo and loopy belief propagation used
- MAP estimates found using standard gradient-based or quasi-Newton optimization methods (L-BFGS).

# 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.
- Greater the weight, more important is the constraint.
- In MLNs, if a constraint is violated by a world, it simply becomes less probable.

# Formal definition

A Markov logic network $L$ is a set of pairs $(F_i, w_i)$

- $F_i$ is a formula in first-order logic
- $w_i$ is a real number

# A Markov Logic Network

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

# MLN as a Markov Network

With a finite set of constants $C = \{c_1, c_2, ..., c_{|C|}\}$ , it defines a Markov Network $M_{L,C}$

- $M_{L,C}$ 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.
- $M_{L,C}$ 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 $L$.

# Ground Markov Network $M_{L,C}$

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

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 $\phi_i \left(x_{\{i\}}\right) = e^{w_i}$.

# Graphical structure of ground markov network

- There is an edge between 2 nodes if the corresponding ground atoms appear together in at least one grounding of a formula in $L$.
- Atoms in each ground formula form a clique in $M_{L,C}$

# Example

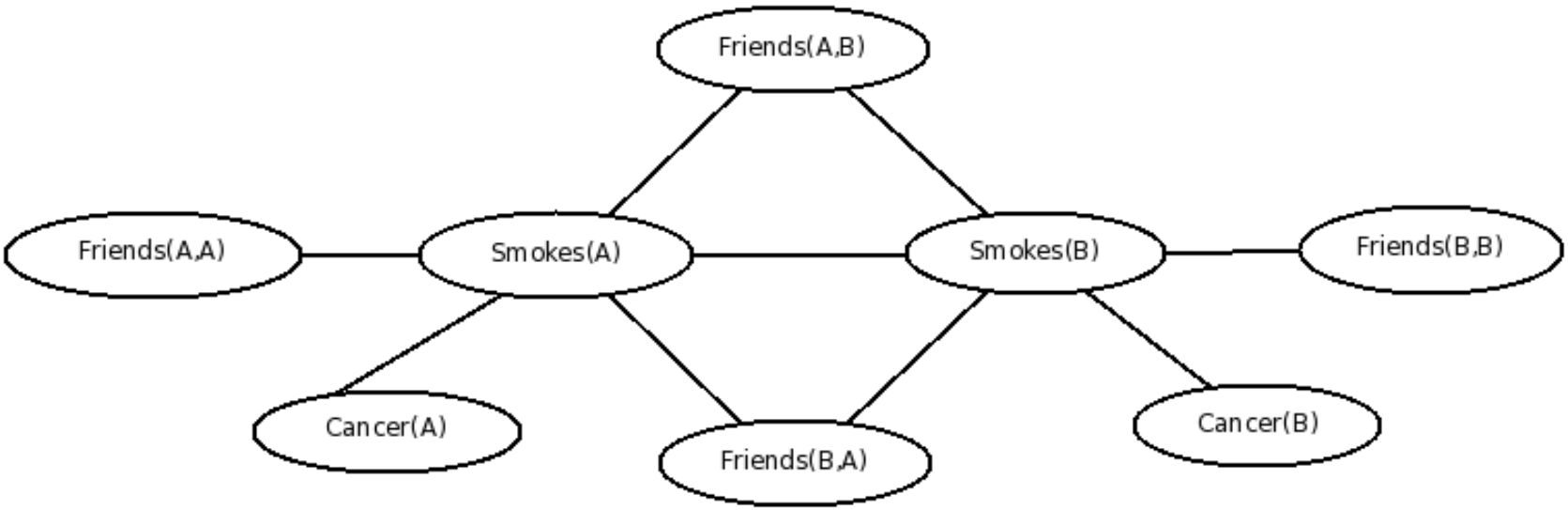


Figure: Ground Markov network obtained by applying the last two formulas in Figure 1 to the constants Alice($A$) and Bob($B$)

# Assumptions

- Each $M_{L,C}$ denotes a possible world.
- 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 $C$.

# Entity Resolution

1. Entity resolution is the problem to determine which records refer to same real world entity.

- Same authors can be refered differently: Andrew McCallum, McCallum A. and A. McCallum refers to same person
- Same conference can be refered differently: ICML, International Conference of Machine Learning
- Same city name: Los Angeles and LA

# Unique Name Assumption

**Unique Name Assumption** different constants refer to different objects in the domain

**Predicate Form** Equals(x, y) or $x = y$

# Unique Name Assumption

**Unique Name Assumption** different constants refer to different objects in the domain

**Predicate Form** Equals(x, y) or $x = y$

**Reflexivity**

$$\forall x, x = x$$

**Symmetry**

$$\forall x, y \; x = y \Longrightarrow y = x$$

**Transitivity**

$$\forall x, y, z \; x = y \wedge \; y = z \Longrightarrow \; a = z$$

# Predicate Equivalence

**Predicate Equivalence** for each binary relation R is given by

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

if two objects are same if their some of fields are same

**Reverse Predicate Equivalence** for each binary predicate R

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

if two objects are in the same relation to the same object, this is evidence that they may be the same object
two papers appearing in same conference with same author name are likely to be same

# Problem Formulation

**Binary Relations** n-ary relation is represented using n binary relations
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).

**Typed Fields** All the fields are assumed to be typed
the first argument of the predicate HasAuthor(paper, author) is of type Paper, and the second is of type Author

# Problem Formulation

**Binary Relations** n-ary relation is represented using n binary relations
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).

**Typed Fields** All the fields are assumed to be typed
the first argument of the predicate HasAuthor(paper, author) is of type Paper, and the second is of type Author

**Goal** Goal is to find two author names corresponds to same person or not.
if $x_1$ and $x_2$ have same type then is $x_1 = x_2$ ?

# Deduplication

**HasWord predicate** object to be deduplicated is citation which is mainly in the form of string tokens

HasWord(field,word) checks if "word" occurs in "field"

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

if fields have same words then those fields are same.

**Putting in Probabilistic Model**

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

w is weight of formula and n is the number of times $x_1 = x_2$.

# Deduplication

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

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

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

# Issues

**Misspellings** defining the predicate HasEngram(word, engram), which is true iff engram is a substring of word

**Fellegi-Sunter Model**

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

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

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

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

# Model 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): reverse predicate equivalence rules
3. MLN(B+T): adding transitive closure rules to MLN(B)
4. MLN(B+C+T): both reverse predicate equivalence and transitive closure rules
5. MLN(B+C+T+S): rules added using structure learning algorithm
6. MLN(B+N+C+T): two-level learning/inference step

# Performance Analysis

- 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.
- 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.

# Experiments

- Experimented with Alchemy
- Constructed and trained an MLN for entity resolution using CORA citation dataset
- Used B+N+C+T model

# Conclusion

- Logic and probability can be combined into a graphical model for inferencing on a probabilistic knowledge base
- A small number of axioms in Markov logic capture the essential features of citation records
- Markov logic enables us to easily build a sophisticated entity resolution system