Combining Logic and Probability
Languages, Algorithms and Applications

Pedro Domingos
University of Washington, USA

Kristian Kersting
Fraunhofer IAIS & Univ. of Bonn, Germany
Acknowledgements

- Statistical Relational Learning (SRL) and AI (StarAI) are a synthesis of ideas of many individuals who have participated in various SRL/StarAI events, workshops and classes.

- Thanks to all of you!
General Take-Away Message

- Graphs are not enough
- We need logic
Roadmap

1. Motivation

2. Statistical Relational Learning / AI: a short overview

3. Markov Logic Networks
MOTIVATION
Rorschach Test
Etzioni’s Rorschach Test for Computer Scientists
Moore’s Law?
Storage Capacity?
Number of Scientific Publications?
Number of Facebook Users?
Number of Web Pages?
The World-Wide Mind

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
So, Tasks Are Often Structural

- Objects are not just feature vectors
  - They have parts and subparts
  - Which have relations with each other
  - They can be trees, graphs, etc.
- Objects are seldom i.i.d. (independent and identically distributed)
  - They exhibit local and global dependencies
  - They form class hierarchies (with multiple inheritance)
  - Objects’ properties depend on those of related objects
- Deeply interwoven with knowledge

How do computer systems deal with structural problems?
(First-order) Logic handles Structures

- Main theoretical foundation of computer science
- General language for describing complex structures and knowledge: trees, graphs, hierarchies, etc.
- Inference algorithms (satisfiability testing, resolution, theorem proving, etc.)

More compact knowledge representation. Consider e.g. classical examples such as chess or wumpus:
FOL << PL << atomic

$$\forall x, y \text{father-of}(x, y) \land \text{female}(y) \iff \text{daughter-of}(y, x)$$

Explicit enumeration

daugther-of(cecily,john)
daugther-of(lily,tom)
...

Many types of entities
Relations between them
Arbitrary knowledge

Logic
true/false

atomic
propositional
first-order/relational

5th C B.C.

19th C

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Tasks are also often Statistical

- Information are ambiguous
- Our information is always incomplete
- Our predictions are uncertain

How do computer systems deal with uncertainty?

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Probability handles Uncertainty

Mixture models
Hidden Markov models
Bayesian networks
Markov random fields
Maximum entropy models
Conditional random fields

Many types of entities
Relations between them
Arbitrary knowledge

Sensor noise
Human error
Inconsistencies
Unpredictability

Explicit enumeration

atomic
propositional
first-order/relational

17th C
20th C
5th C B.C.
19th C

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Many types of entities
Relations between them
Arbitrary knowledge

Sensor noise
Human error
Inconsistencies
Unpredictability

Explicit enumeration

atomic
propositional
first-order/relational

17th C
20th C
5th C B.C.
19th C

Probability
Logic
true/false

So, will traditional (U)AI scale?
Propositional vs. Relational Data

- Traditional work in robotics, machine learning and knowledge discovery assume data instances form a single table.
- Traditional statistical models assume independence among instances (rows) and find associations among the values of multiple variables within a single instance.
- Relational models assume dependence among instances in different rows and tables and find associations among these values.

[slide adapted from David Jensen]
Let’s consider a simple relational domain: Reviewing Papers

- The grade of a paper at a conference depends on the paper’s quality and the difficulty of the conference.
  - Good papers may get A’s at easy conferences
  - Good papers may get D’s at top conference
  - Weak papers may get B’s at good conferences
  - ...

[inspired by Friedman and Koller]
\[ P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1}, \ldots, X_1) \]

<table>
<thead>
<tr>
<th>Qual</th>
<th>Diff</th>
<th>P(Grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>0.2, 0.5, 0.3</td>
</tr>
<tr>
<td>low</td>
<td>middle</td>
<td>0.1, 0.7, 0.2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The real world, however, ... 

... has inter-related objects

These ‘instance’ are not independent!

[Slide inspired by Friedman and Koller]
So, will traditional (U)AI scale? **No!**

“Scaling up the environment will inevitably overtax the resources of the traditional (U)AI architecture.”

Probability

17th C

Sensor noise
Human error
Inconsistencies
Unpredictability

Logic

true/false

Explicit enumeration

19th C

Many types of entities
Relations between them
Arbitrary knowledge

5th C B.C.

atomic

propositional

first-order/relational

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Let’s deal with **uncertainty**, **objects**, **relations**, and **learning** jointly.

The study and design of intelligent agents that act in noisy worlds composed of objects and relations among the objects.
The Big Picture on AI

Commonsense reasoning
- Domain Knowledge
- Inference and Learning

Natural Language Processing
- Language Knowledge
- Inference and Learning

Robotics
- Domain & Robot Knowledge
- Inference and Learning

Vision
- Objects & Optics Knowledge
- Inference and Learning

Lifted Inference and Learning

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Why the Tutorial?

- **A very active, multi-disciplinary research area**
  - Involves all sub-disciplines of AI: *reasoning and acting under uncertainty, knowledge representation, constraint satisfaction, machine learning, ...*
  - Unfortunately, can be hard to follow: *they all speak a different language*

- **A success story**
  - Often outperforms state-of-the-art
  - Novel ways of *using the structure* for faster and/or more robust solutions
  - Growth path for (U)AI in general
STATISTICAL RELATIONAL LEARNING / AI: A SHORT OVERVIEW
Applications to Date

- Natural language processing
- Information extraction
- Entity resolution
- Link prediction
- Collective classification
- Social network analysis
- Robot mapping
- Activity recognition
- Scene analysis
- Computational biology
- Probabilistic Cyc
- Personal assistants
- Etc.
Information Extraction

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Entity Resolution

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Relations are at the heart of entity resolution.
Gene Localization

- Predict the localization of a given gene in a cell among 15 distinct positions
- Relations important as sequence similarity does not help

Relational Kernels better than Hayashi et al.’s KDD Cup 2001 winning approach
Semantic Labeling of 3D Scan Data

- Neighbouring pixels/voxels have the same semantic label

[Anguelov et al. CVPR05, Triebel et al. ICRA06, ...]
Relational approaches outperform traditional ranking approaches.
Social Recommendation / Collaborative Filtering

- Predict whether a user **likes** a movie given attributes of users and movies, as well as known ratings and complex link structures

Relational approaches outperform set-based recommendation systems
What is the world talking about?
Topic Models

Relational approaches estimate better low-dimensional embeddings
How do you spend your spare time?

YouTube like media portals have changed the way users access media content in the Internet. Every day, millions of people visit social media sites such as Flickr, YouTube, and Jumpcut, among others, to share their photos and videos, ...

while others enjoy themselves by searching, watching, commenting, and rating the photos and videos; what your friends like will bear great significance for you.
How do you efficiently broadcast information?

Lifted inference faster than belief propagation
Cardiovascular disease cost the EU €169 billion in 2003 and the USA about €310.23 billion in direct and indirect annual costs.

By comparison, the estimated cost of all cancers is €146.19 billion and HIV infections €22.24 billion.

### Table: Algorithm Performance

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
<th>AUC-ROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>J48</td>
<td>0.667</td>
<td>0.607</td>
</tr>
<tr>
<td>SVM</td>
<td>0.667</td>
<td>0.5</td>
</tr>
<tr>
<td>AdaBoost</td>
<td>0.667</td>
<td>0.608</td>
</tr>
<tr>
<td>Bagging</td>
<td>0.677</td>
<td>0.613</td>
</tr>
<tr>
<td>NB</td>
<td>0.75</td>
<td>0.653</td>
</tr>
<tr>
<td>RPT</td>
<td>0.669*</td>
<td>0.778</td>
</tr>
<tr>
<td>RFGB</td>
<td>0.667*</td>
<td>0.819</td>
</tr>
</tbody>
</table>

### So, what are relations?

Relational models provide new insights
What are Relations?

- There are several types of relations and in turn there are several views on what (statistical) relational learning is

1. **Relations provide additional correlations/regularization**
   - If two words occur frequently in the same context (page, paragraph, sentence, ...) then there must be some semantic relation between them

2. **Often extensional (data) only, for one relation**
   - Covariance function, distance functions, kernel functions, graphs, tensors, random walks with restarts...
What are Relations?

3. Relations are symmetries/redundancies in the model
   - E.g. lifted inference based on bisimulation

4. Hypergraph representations of data
   - Multiple (extensional) relations
   - Random walks with restarts as similarity measure or to produce path features

5. Full-fledged relational (or logical) knowledge as considered in this tutorial
   - Multiple (often typed) relations
   - Intensional formulas (often Horn clauses)
     \[
     \text{ancestor}(X,Z) \land \text{parent}(Z,Y) \Rightarrow \text{ancestor}(X,Y)
     \]
The SRL Alphabet Soup

- Relational Gaussian Processes
- Infinite Hidden Relational Models
- Relational Markov Networks
- Object-Oriented Bayes Nets
- IBAL
- Figaro
- Probs. Horn Abduction: Poole
- Prob. Horn Abduction: Poole
- PLP: Haddawy, Ngo
- PRISM: Kameya, Sato
- DAPER
- Curch
- Prob. CLP: Eisele, Riezler
- Probabilistic Entity-Relationship Models
- SLPs: Cussens, Muggleton
- BUGS/Plates
- 1BC(2): Flach, Lachiche
- Multi-Entity Bayes Nets
- BLPs: Kersting, De Raedt
- PRMs: Friedman, Getoor, Koller, Pfeffer, Segal, Taskar
- Markov Logic: Domingos, Richardson
- CLP(BN): Cussens, Page, Qazi, Santos Costa
- SPOOK
- Logical Bayesian Networks: Blockeel, Bruynooghe, Fierens, Ramon
- LOHMMs: De Raedt, Kersting, Raiko
- RMMs: Anderson, Domingos, Weld
- Prob. Horn Abduction: Poole
- First KBMC approaches: Bresse, Bacchus, Charniak, Glesner, Goldman, Koller, Poole, Wellmann
- PRISM: Kameya, Sato
- ´90, ´93, ´94, ´95, ´96, ´97, ´99, ´00, ´02, ´03
- ´10 PSL: Broecheler, Getoor, Mihalkova
- ´07 RDNs: Jensen, Neville
Key Dimensions with some prototypes

directed

- PRISM
- PRMs
- CLP(BN)
- BUGS
- BLPs
- SLPs
- LPAD
- RDN
- ProbLog
- ICL
- BLOG
- SLPs

undirected

- MLNs
- RMNs
- RGPs
- IHRM
- LPAD
- BLOG
- RBNs
- RBNs
- RGNs
- BLOG
- PRISM

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Directed: Probabilistic Relational Models (PRMs)  
Bayesian logic Programs (BLPs)

\[\forall x \ author(x, p) \land smart(x) \Rightarrow high\_quality(p)\]
\[\forall x \ high\_quality(p) \Rightarrow accepted(p)\]

Macro for conditional probability table

<table>
<thead>
<tr>
<th>high_quality(Y)</th>
<th>smart(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.9, 0.1)</td>
<td>yes</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Inference on BN constructed by instantiating the rules/ macros using back-or forward chaining

But what happens if instead we have author(bob,p1)?

So, we can deal with a variable number of objects. The induced BN depends on the domain elements and the background knowledge we have.
Directed: Aggregate Dependencies

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Directed: Aggregate Dependencies

Still, the induced model is assumed to be acyclic
**Option 1: Relational Dependency Networks (RDNs)**

\[
\forall x \ author(x, p) \land \ high\_quality(x) \Rightarrow high\_quality(p)
\]

\[
\forall x \ high\_quality(p) \Rightarrow accepted(p)
\]

\[
\forall x, y \ co\_author(x, y) \land \ smart(x) \Rightarrow smart(y)
\]

\[
\forall x, y \ \exists p \ author(x, p) \land \ author(y, p) \Rightarrow co\_author(x, y)
\]

---

**cyclic dependency**

Run approximate Gibbs sample
Relational Dependency Networks

Run approximate Gibbs sample
Option 2: Markov Logic Networks

Suppose we have constants: alice, bob and p1

1.5 $\forall x \text{ author}(x, p) \land \text{smart}(x) \implies \text{high_quality}(p)$

1.1 $\forall x \text{ high_quality}(p) \implies \text{accepted}(p)$

1.2 $\forall x, y \text{ co_author}(x, y) \implies (\text{smart}(x) \iff \text{smart}(y))$

$\infty$ $\forall x, y \exists p \text{ author}(x, p) \land \text{author}(y, p) \implies \text{co_author}(x, y)$

Compile to an undirected model
Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
ProbLog

- Label of a clause/fact \( c \) is the probability that \( c \) belongs to the target program; Facts/clauses independent of each other
- Defines a distribution over programs

\[
P(L|Program) = \prod_{c_i \in L} p_i \prod_{c_j \notin L} (1 - p_j)
\]

\[
P(path(x\textunderscore gene,disease2)) = \text{sum of probs of all programs that entail the query}
\]

\[
P = 0.1 \cdot 0.66 \cdot 0.39 + P = (1 - 0.1) \cdot 0.66 \cdot 0.39 + P = 0.1 \cdot 0.66 \cdot (1 - 0.39)
\]

Exponentially many subprograms! To avoid explosion, consider proofs/paths only + store them in a BDD in order to count correctly
Many other approaches!!

**directed**
- CLP(BN)
- PRISM
- PRMs
- BLPs
- SLPs
- BUGS
- ICL
- ProbLog
- RDN
- BLOG
- IHRM
- LPAD
- RBNs
- BLOG
- PRISM
- RDN
- RGPs
- RBNs
- NP-BLOG
- MEBNs
- BLOG

**undirected**
- MLNs
- RMNs
- RDN
- BLOG
- PRISM
- LPAD
- IHRM
- RGPs
- ICL
- BLOG
- NP-BLOG
- MEBNs
- BLOG

**macro**
- MLNs
- BLPs
- PRMs
- IHRM
- BLOG
- RBNs
- PRISM
- RDN
- RGPs
- BLOG
- NP-BLOG
- MEBNs
- BLOG

**proofs**
- MLNs
- BLPs
- PRMs
- IHRM
- BLOG
- RBNs
- PRISM
- RDN
- RGPs
- BLOG
- NP-BLOG
- MEBNs
- BLOG

**parametric**
- MLNs
- BLPs
- SLPs
- PRMs
- IHRM
- BLOG
- RBNs
- PRISM
- RDN
- RGPs
- BLOG
- NP-BLOG
- MEBNs
- BLOG

**non-parametric**
- MLNs
- BLPs
- SLPs
- PRMs
- IHRM
- BLOG
- RBNs
- PRISM
- RDN
- RGPs
- BLOG
- NP-BLOG
- MEBNs
- BLOG

CWA
- MLNs
- BLPs
- SLPs
- PRMs
- IHRM
- BLOG
- RBNs
- PRISM
- RDN
- RGPs
- BLOG
- NP-BLOG
- MEBNs
- BLOG

OWA
- MLNs
- BLPs
- SLPs
- PRMs
- IHRM
- BLOG
- RBNs
- PRISM
- RDN
- RGPs
- BLOG
- NP-BLOG
- MEBNs
- BLOG

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
And actually they span the whole AI spectrum

- Relational topic models
- Mixed-membership models
- Relational Gaussian processes
- Relational reinforcement learning
- (Partially observable) MDPs
- Systems of linear equations
- Kalman filters
- Declarative information networks

No, this is very much like in the early days of UAI!

So, should we worry about the soup?
The early days of UAI

Maximum entropy inference
Odds-likelihood updating
Dempster-Shafer Belief Functions

Mycin's Certainty Factors
Bayesian Networks

Expert-rating
Decision-theoretic metrics

Belief Maintenance System
Bayes’ Theorem

Prospector
Probabilistic Logic

Fuzzy Set Theory
Incidence Calculus

[A. Bundy. Incidence Calculus: A Mechanism for Probabilistic Reasoning. UAI-85]
[D. Hunter. Uncertain Reasoning Using Maximum Entropy Inference. UAI-85]
[D. Heckerman. Probabilistic Interpretations for MYCIN's Certainty Factors. UAI-85]
[S. Ursic. Generalizing Fuzzy Logic Probabilistic Inferences. UAI-86]
[B. Falkenheiner. Towards a General-Purpose Belief Maintenance System. UAI-86]
[D. Heckerman. An Empirical Comparison of Three Inference Methods. UAI-88]

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
This soup boiled down to Graphical Models as intermediate representation

Distributions can naturally be represented as Factor Graphs

- There is an edge between a circle and a box if the variable is in the domain/scope of the factor

\[
p(x) = f_a(x_1, x_2)f_b(x_1, x_2)f_c(x_2, x_3)f_d(x_3)\]

\[
p(x) = \prod_{s} f_s(x_s)\]

unnormalized!
Factor Graphs from Graphical Models

\[ p(x) = p(x_1)p(x_2) \]
\[ p(x_3|x_1, x_2) \]

\[ f_a(x_1) = p(x_1) \]
\[ f_b(x_2) = p(x_2) \]
\[ f_c(x_1, x_2, x_3) = p(x_3|x_1, x_2) \]

\[ \psi(x_1, x_2, x_3) \]
\[ = \psi(x_1, x_2, x_3) \]

Similar “boiling down” process is going on in SRL!
Boiled-Down SRL Alphabet Soup

- Given a relational model in your language of choice, a set of constants and a query, compile everything into an intermediate representation
  - Factor graphs
  - BDDs, Arithmetic Circuits, d-DNNFs, ...
  - Weighted CNFs
- Run inference
Rules + Potential: Logically Parameterized Factors

∀X. φ₁(popular, attends(X))

∀X. φ₂(attends(X), series)

Atoms represent a set of random variables

Parfactors parameterized factors

There can also be constraints to logical variables such as X=/=UAI11
Rules + Weights: Weighted CNF

- Weighted MAX-SAT as mode finding for log-linear distributions
- Each configuration has a cost: the sum of the weights of the unsatisfied (ground) clauses.
- An infinite cost gives a ‘hard’ clause.
- Goal: find an assignment with minimal cost.

Factor Graph:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$z$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\beta$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$f_\alpha(x, y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$w_1$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$w_1$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$w_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x$</th>
<th>$z$</th>
<th>$f_\beta(x, z)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$w_2$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$w_2$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$w_2$</td>
</tr>
</tbody>
</table>
Find a set of general rules

mutagenic(X) :- atom(X,A,c), charge(X,A,0.82)
mutagenic(X) :- atom(X,A,n), ...

Examples E

pos(mutagenic(m_1))
neg(mutagenic(m_2))
pos(mutagenic(m_3))
...

Background Knowledge B

molecule(m_1)
molecule(m_2)
atom(m_1,a_{11},c)
atom(m_2,a_{21},o)
atom(m_1,a_{12},n)
atom(m_2,a_{22},n)
bond(m_1,a_{11},a_{12})
bond(m_2,a_{21},a_{22})
charge(m_1,a_{11},0.82)
charge(m_2,a_{21},0.82)
...
...
Example ILP Algorithm: FOIL
[Quinlan MLJ 5:239-266, 1990]

- mutagenic(X) :- atom(X,A,n), charge(A,0.82)
  Coverage = 0.5, 0.7

- mutagenic(X) :- atom(X,A,c), bond(A,B)
  Coverage = 0.6

- :- true
  Coverage = 0.6, 0.3

Some objective function, e.g. percentage of covered positive examples

Pedro Domingos, Kristian Kersting
Combining Probability and Logic: Languages, Algorithms and Applications
Vanilla SRL Approach [De Raedt, K ALT04]

- Traverses the hypotheses space a la ILP
- Replaces ILP’s 0-1 covers relation by a “smooth”, probabilistic one [0,1]

\[
\text{mutagenic}(X) :\text{- atom}(X,A,n),\text{charge}(A,0.82)
\]

\[
\text{mutagenic}(X) :\text{- atom}(X,A,c),\text{bond}(A,B)
\]

\[
\text{Vanilla SRL Approach}
\]

\[
\begin{align*}
\text{cover}(e, H, B) &= P(e|H, B) \\
\text{cover}(E, H, B) &= \prod_{e \in E} \text{cover}(e, H, B)
\end{align*}
\]
MARKOV LOGIC
MARKOV LOGIC
Overview

- Representation
- Inference
- Learning
- Applications
- Discussion
Propositional Logic

- **Atoms**: Symbols representing propositions
- **Logical connectives**: ¬, ∧, ∨, etc.
- **Knowledge base**: Set of formulas
- **World**: Truth assignment to all atoms
- Every KB can be converted to **CNF**
  - **CNF**: Conjunction of clauses
  - **Clause**: Disjunction of literals
  - **Literal**: Atom or its negation
- **Entailment**: Does KB entail query?
First-Order Logic

- **Atom**: Predicate(Variables, Constants)
  
  E.g.: $Friends(Anna, x)$

- **Ground atom**: All arguments are constants

- **Quantifiers**: $\forall, \exists$

- **This talk**: Finite, Herbrand interpretations
Markov Networks

- **Undirected** graphical models

![Graphical Model Diagram]

- Potential functions defined over cliques

\[
P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)
\]

\[
Z = \sum_x \prod_c \Phi_c(x_c)
\]

<table>
<thead>
<tr>
<th>Smoking</th>
<th>Cancer</th>
<th>$\Phi(S,C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>4.5</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>4.5</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>2.7</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Markov Networks

- **Undirected** graphical models

Log-linear model:

\[
P(x) = \frac{1}{Z} \exp \left( \sum_i w_i f_i(x) \right)
\]

\[
f_1(\text{Smoking}, \text{Cancer}) = \begin{cases} 
1 & \text{if } \neg \text{Smoking} \lor \text{Cancer} \\
0 & \text{otherwise}
\end{cases}
\]

\[w_1 = 0.51\]
Probabilistic Knowledge Bases

**PKB** = Set of formulas and their probabilities
  + Consistency + Maximum entropy
  = Set of formulas and their weights
  = Set of formulas and their potentials
  (1 if formula true, $\phi_i$ if formula false)

$$P(\text{world}) = \frac{1}{Z} \prod_i \phi_i^{n_i(\text{world})}$$
Markov Logic

- A Markov Logic Network (MLN) is a set of pairs \((F, w)\) where
  - \(F\) is a formula in first-order logic
  - \(w\) is a real number

- An MLN defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula \(F\) in the MLN, with the corresponding weight \(w\)
Example

\[ \neg \text{Friends}(\text{Anna, Bob}) \]

\[ \text{Friends}(\text{Anna, Bob}) \]

\[ \neg \text{Happy}(\text{Bob}) \quad \text{Happy}(\text{Bob}) \]
Example

\[ \neg \text{Friends}(\text{Anna}, \text{Bob}) \]

\[ \text{Friends}(\text{Anna}, \text{Bob}) \]

\[ \neg \text{Happ}y(\text{Bob}) \]

\[ \neg \text{Happ}y(\text{Bob}) \]

\[ \text{Happ}y(\text{Bob}) \]

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications
Example

\[ P(\neg \text{Friends}(\text{Anna}, \text{Bob}) \lor \text{Happy}(\text{Bob})) = 0.8 \]
Example

$$\Phi(\neg \text{Friends(Anna, Bob)} \lor \text{Happy(Bob)}) = 1$$
$$\Phi(\text{Friends(Anna, Bob)} \land \neg \text{Happy(Bob)}) = 0.75$$

<table>
<thead>
<tr>
<th></th>
<th>Friends(Anna, Bob)</th>
<th>\neg Friends(Anna, Bob)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\neg Happy(Bob)</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>Happy(Bob)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications
Example

\[ w(\Phi(\neg \text{Friends}(Anna, Bob) \lor \text{Happy}(Bob))) \]

\[ = \log(1/0.75) = 0.29 \]
Overview

- Representation
- Inference
- Learning
- Applications
- Discussion
Theorem Proving

\[ \text{TP}(KB, \text{Query}) \]

\[
\begin{align*}
KB_Q & \leftarrow KB \cup \{\neg \text{Query}\} \\
\text{return} & \neg \text{SAT}(\text{CNF}(KB_Q))
\end{align*}
\]
Satisfiability (DPLL)

\[ \text{SAT}(CNF) \]

if \( CNF \) is empty return \( True \)
if \( CNF \) contains empty clause return \( False \)
choose an atom \( A \)
return \( \text{SAT}(CNF(A)) \lor \text{SAT}(CNF(\neg A)) \)
First-Order Theorem Proving

- Propositionalization
  1. Form all possible ground atoms
  2. Apply propositional theorem prover

- Lifted Inference: Resolution
  - Resolve pairs of clauses until empty clause derived
  - Unify literals by substitution, e.g.: $x = Bob$ unifies $\text{Friends}(Anna, x)$ and $\text{Friends}(Anna, Bob)$

\[
\neg \text{Friends}(Anna, x) \lor \text{Happy}(x)
\]

\[
\text{Friends}(Anna, Bob)
\]

\[
\text{Happy}(Bob)
\]
Probabilistic Theorem Proving

**Given** Probabilistic knowledge base $K$
Query formula $Q$

**Output** $P(Q|K)$
Weighted Model Counting

- $\text{ModelCount} (\text{CNF}) = \# \text{ worlds that satisfy CNF}$
- Assign a weight to each literal
- $\text{Weight(world)} = \prod \text{weights(true literals)}$
- Weighted model counting:
  
  **Given** CNF $C$ and literal weights $W$
  
  **Output** $\sum \text{weights(worlds that satisfy } C)$

PTP is reducible to lifted WMC
Example

Friends(Anna, Bob)

\[ \neg \text{Friends}(\text{Anna}, \text{Bob}) \]

<table>
<thead>
<tr>
<th>\text{Friends}(\text{Anna}, \text{Bob})</th>
<th>\neg \text{Happy}(\text{Bob})</th>
<th>\text{Happy}(\text{Bob})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications
Example

\[ P(\text{Happy}(\text{Bob}) \mid \text{Friends}(\text{Anna}, \text{Bob})) = \frac{1}{1 + 0.75} \approx 0.57 \]

\[
\begin{array}{|c|c|}
\hline
\text{Friends}(\text{Anna}, \text{Bob}) & \text{—Happy}(\text{Bob}) & \text{Happy}(\text{Bob}) \\
\hline
0.75 & 1 & \\
\hline
\end{array}
\]
Example

If $P(\neg \text{Friends}(\text{Anna}, \text{Bob}) \lor \text{Happy}(\text{Bob})) = 0.8$

Then $P(\text{Happy}(\text{Bob}) \mid \text{Friends}(\text{Anna}, \text{Bob})) = \frac{1}{1 + 0.75} \approx 0.57$
Example

\[ P(\neg \text{Friends}(\text{Anna}, x) \lor \text{Happy}(x)) = 0.8 \]
Example

\[ P(\neg \text{Friends}(\text{Anna, } x) \lor \text{Happy}(x)) = 0.8 \]

Friends(Anna, Bob)
Inference Problems

LWSAT  PTP = LWMC

TP₁

Lifted

Weighted

MPE = WSAT  PI = WMC

Counting

TP₀ = SAT  MC
Propositional Case

- All conditional probabilities are ratios of partition functions:

\[
P(\text{Query} \mid PKB) = \frac{\sum_{\text{worlds}} 1_{\text{Query}}(\text{world}) \prod_i \Phi_i(\text{world})}{Z(PKB)}
\]

\[
= \frac{Z(PKB \cup \{(\text{Query},0)\})}{Z(PKB)}
\]

- All partition functions can be computed by weighted model counting
Conversion to CNF + Weights

\[ \text{WCNF}(PKB) \]

\[
\text{for all } (F_i, \Phi_i) \in PKB \text{ s.t. } \Phi_i > 0 \text{ do}
\]

\[
PKB \leftarrow PKB \cup \{(F_i \iff A_i, 0)\} \setminus \{(F_i, \Phi_i)\}
\]

\[
CNF \leftarrow \text{CNF}(PKB)
\]

\[
\text{for all } \neg A_i \text{ literals do } W_{\neg A_i} \leftarrow \Phi_i
\]

\[
\text{for all other literals } L \text{ do } w_L \leftarrow 1
\]

\[
\text{return } (CNF, \text{weights})
\]
Probabilistic Theorem Proving

\[ \text{PTP}(PKB, \text{Query}) \]

\[ PKB_Q \leftarrow PKB \cup \{(\text{Query},0)\} \]

return \[ \frac{\text{WMC}(\text{WCNF}(PKB_Q))}{\text{WMC}(\text{WCNF}(PKB))} \]
Probabilistic Theorem Proving

\[ PTP(PKB, Query) \]
\[ PKB_Q \leftarrow PKB U \{(Query,0)\} \]
\[ \text{return } WMC(WCNF(PKB_Q)) / WMC(WCNF(PKB)) \]

Compare:

\[ TP(KB, Query) \]
\[ KB_Q \leftarrow KB U \{\neg Query\} \]
\[ \text{return } \neg \text{SAT}(CNF(KB_Q)) \]
Weighted Model Counting

$\text{WMC}(CNF, weights)$

if all clauses in $CNF$ are satisfied

return $\prod_{A \in A(CNF)} (w_A + w_{\neg A})$

if $CNF$ has empty unsatisfied clause return 0
Weighted Model Counting

\[ \text{WMC}(\text{CNF}, \text{weights}) \]

\begin{align*}
\text{if} & \text{ all clauses in CNF are satisfied} \\
\text{return} & \prod_{A \in A(\text{CNF})} (w_A + w_{\neg A}) \\
\text{if} & \text{ CNF has empty unsatisfied clause return 0} \\
\text{if} & \text{ CNF can be partitioned into CNFs } C_1, \ldots, C_k \\
& \text{sharing no atoms} \\
\text{return} & \prod_{i=1}^{k} \text{WMC}(C_i, \text{weights})
\end{align*}

Decomp. Step
Weighted Model Counting

\[ \text{WMC}(\text{CNF}, \text{weights}) \]

if all clauses in \( \text{CNF} \) are satisfied

\[ \text{return } \prod_{A \in A(\text{CNF})} (w_A + w_{\neg A}) \]

if \( \text{CNF} \) has empty unsatisfied clause \text{return } 0

if \( \text{CNF} \) can be partitioned into CNFs \( C_1, \ldots, C_k \) sharing no atoms

\[ \text{return } \prod_{i=1}^{k} \text{WMC}(C_i, \text{weights}) \]

choose an atom \( A \)

\[ \text{return } w_A \text{WMC}(\text{CNF} \mid A, \text{weights}) \]
\[ + w_{\neg A} \text{WMC}(\text{CNF} \mid \neg A, \text{weights}) \]

Splitting Step
First-Order Case

- PTP schema remains the same
- Conversion of PKB to hard CNF and weights:
  New atom in $F_i \iff A_i$ is now $\text{Predicate}_i(\text{variables in } F_i, \text{constants in } F_i)$
- New argument in WMC:
  Set of substitution constraints of the form $x = A, x \neq A, x = y, x \neq y$
- Lift each step of WMC
Lifted Weighted Model Counting

\[ \text{LWMC}(\text{CNF}, \text{substs}, \text{weights}) \]

\[
\begin{align*}
\text{if} \quad & \text{all clauses in } \text{CNF} \text{ are satisfied} \\
\text{return} \quad & \prod_{A \in A(\text{CNF})} \left( w_A + w_{\neg A} \right)^{n_A(\text{substs})} \\
\text{if} \quad & \text{CNF has empty unsatisfied clause} \quad \text{return} \quad 0
\end{align*}
\]

Base Case
Lifted Weighted Model Counting

\[ \text{LWMC}(CNF, \text{substs}, \text{weights}) \]

\begin{align*}
& \text{if all clauses in } CNF \text{ are satisfied} \\
& \quad \text{return } \prod_{A \in A(CNF)} \left( w_A + w_{\neg A} \right)^{n_A(\text{substs})} \\
& \text{if } CNF \text{ has empty unsatisfied clause } \text{return } 0 \\
& \text{if there exists a lifted decomposition of } CNF \\
& \quad \text{return } \prod_{i=1}^{k} \left[ \text{LWMC}(CNF_{i,1}, \text{substs}, \text{weights}) \right]^{m_i}
\end{align*}
Lifted Weighted Model Counting

LWMC\((CNF, \text{substs}, \text{weights})\)

if all clauses in CNF are satisfied

return \(\prod_{A \in A(CNF)} (w_A + w_{\neg A})^{n_A(\text{substs})}\)

if CNF has empty unsatisfied clause return 0

if there exists a lifted decomposition of CNF

return \(\prod_{i=1}^{k}[LWMC(CNF_{i,1}, \text{substs}, \text{weights})]^{m_i}\)

choose an atom A

return \(\sum_{i=1}^{l} n_i w_{A}^{t_i} w_{\neg A}^{f_i} \cdot LWMC(CNF | \sigma_j, \text{substs}_j, \text{weights})\)
Extensions

- Unit propagation, etc.
- Caching / Memoization
- Knowledge-based model construction
Approximate Inference

\[
WMC(CNF, \text{weights})
\]

if all clauses in \(CNF\) are satisfied

return \(\prod_{A \in A(CNF)} (w_A + w_{\neg A})\)

if \(CNF\) has empty unsatisfied clause return 0

if \(CNF\) can be partitioned into \(CNFs\) \(C_1, \ldots, C_k\)
sharing no atoms

return \(\prod_{i=1}^k WMC(C_i, \text{weights})\)

choose an atom \(A\)

return \(\frac{w_A}{Q(A \mid CNF, \text{weights})} WMC(CNF \mid A, \text{weights})\)

with probability \(Q(A \mid CNF, \text{weights})\), etc.
MPE Inference

- Replace sums by maxes
- Use branch-and-bound for efficiency
- Do traceback
More on Sunday at Noon

Session on First-Order Inference

- *Probabilistic Theorem Proving*
  V. Gogate and P. Domingos

- *Inference in Probabilistic Logic Programs Using Weighted CNF*
  D. Fierens, G. van den Broeck, I. Thon, B. Gutmann and L. de Raedt
Even More on Monday

IJCAI-11 Tutorial on Lifted Inference in Probabilistic Logical Models

- Eyal Amir
- Pedro Domingos
- Lise Getoor
- Kristian Kersting
- Sriraam Natarajan
- David Poole
- Rodrigo de S. Braz
- Prithviraj Sen
Overview

- Representation
- Inference
- **Learning**
- Applications
- Discussion
Learning

- Data is a relational database
- Closed world assumption (if not: EM)
- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (formulas)
Generative Weight Learning

- Maximize likelihood
- Use gradient ascent or L-BFGS
- No local maxima
  \[ \frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)] \]
  - No. of true groundings of clause \( i \) in data
  - Expected no. true groundings according to model
- Requires inference at each step (slow!)
Pseudo-Likelihood

\[ PL(x) \equiv \prod_{i} P(x_i \mid \text{neighbors}(x_i)) \]

- Likelihood of each variable given its neighbors in the data [Besag, 1975]
- Does not require inference at each step
- Consistent estimator
- Widely used in vision, spatial statistics, etc.
- But PL parameters may not work well for long inference chains
Discriminative Weight Learning

- Maximize conditional likelihood of query \( (y) \) given evidence \( (x) \)

\[
\frac{\partial}{\partial w_i} \log P_w(y \mid x) = n_i(x, y) - E_w[n_i(x, y)]
\]

- No. of true groundings of clause \( i \) in data
- Expected no. true groundings according to model

- Expected counts can be approximated by counts in MAP state of \( y \) given \( x \)
Voted Perceptron

- Originally proposed for training HMMs discriminatively [Collins, 2002]
- Assumes network is linear chain

\[
\begin{align*}
& w_i \leftarrow 0 \\
& \text{for } t \leftarrow 1 \text{ to } T \text{ do} \\
& \quad y_{MAP} \leftarrow \text{Viterbi}(x) \\
& \quad w_i \leftarrow w_i + \eta \left[ \text{count}_i(y_{Data}) - \text{count}_i(y_{MAP}) \right] \\
& \text{return } \sum_t w_i / T
\end{align*}
\]
HMMs are special case of MLNs
Replace Viterbi by prob. theorem proving
Network can now be arbitrary graph

\[ w_i \leftarrow 0 \]
\[ \text{for } t \leftarrow 1 \text{ to } T \text{ do} \]
\[ y_{MAP} \leftarrow \text{PTP}(\text{MLN } U \{x\}, y) \]
\[ w_i \leftarrow w_i + \eta \left[ \text{count}_i(y_{Data}) - \text{count}_i(y_{MAP}) \right] \]
\[ \text{return } \sum_t w_i / T \]
Structure Learning

- Generalizes feature induction in Markov nets
- Any inductive logic programming approach can be used, but . . .
- Goal is to induce any clauses, not just Horn
- Evaluation function should be likelihood
- Requires learning weights for each candidate
- Turns out not to be bottleneck
- Bottleneck is counting clause groundings
- Solution: Subsampling
Structure Learning

- **Initial state:** Unit clauses or hand-coded KB
- **Operators:** Add/remove literal, flip sign
- **Evaluation function:** Pseudo-likelihood + Structure prior
- **Search:**
  - Beam, shortest-first [Kok & Domingos, 2005]
  - Bottom-up [Mihalkova & Mooney, 2007]
  - Relational pathfinding [Kok & Domingos, 2009, 2010]
Alchemy

Open-source software including:
- Full first-order logic syntax
- MAP and marginal/conditional inference
- Generative & discriminative weight learning
- Structure learning
- Programming language features

alchemy.cs.washington.edu
<table>
<thead>
<tr>
<th></th>
<th>Alchemy</th>
<th>Prolog</th>
<th>BUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation</td>
<td>F.O. Logic + Markov nets</td>
<td>Horn clauses</td>
<td>Bayes nets</td>
</tr>
<tr>
<td>Inference</td>
<td>Probabilistic thm. proving</td>
<td>Theorem proving</td>
<td>Gibbs sampling</td>
</tr>
<tr>
<td>Learning</td>
<td>Parameters &amp; structure</td>
<td>No</td>
<td>Params.</td>
</tr>
<tr>
<td>Uncertainty</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Relational</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Overview

● Representation
● Inference
● Learning
● Applications
● Discussion
Applications to Date

- Natural language processing
- Information extraction
- Entity resolution
- Link prediction
- Collective classification
- Social network analysis
- Robot mapping
- Activity recognition
- Scene analysis
- Computational biology
- Probabilistic Cyc
- Personal assistants
- Etc.
Information Extraction

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Segmentation

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Entity Resolution

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Entity Resolution

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


State of the Art

- Segmentation
  - HMM (or CRF) to assign each token to a field
- Entity resolution
  - Logistic regression to predict same field/citation
  - Transitive closure
- Alchemy implementation: Seven formulas
Types and Predicates

token = \{\text{Parag, Singla, and, Pedro, ...} \}
field = \{\text{Author, Title, Venue} \}
citation = \{C1, C2, ... \}
position = \{0, 1, 2, ... \}

Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
SameCit(citation, citation)
Types and Predicates

token = \{\text{Parag, Singla, and, Pedro, ...}\}
field = \{\text{Author, Title, Venue, ...}\}
citation = \{C1, C2, ...\}
position = \{0, 1, 2, ...\}

Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
SameCit(citation, citation)
Types and Predicates

token = {Parag, Singla, and, Pedro, ...}
field = {Author, Title, Venue}
citation = {C1, C2, ...}
position = {0, 1, 2, ...}

证据

**Token(token, position, citation)**
**InField(position, field, citation)**
**SameField(field, citation, citation)**
**SameCit(citation, citation)**
Types and Predicates

token = \{Parag, Singla, and, Pedro, \ldots\}
field = \{Author, Title, Venue\}
citation = \{C1, C2, \ldots\}
position = \{0, 1, 2, \ldots\}

\text{Token}(token, position, citation)
\text{InField}(position, field, citation)
\text{SameField}(field, citation, citation)
\text{SameCit}(citation, citation)

Query
Formulas

Token(+t, i, c) => InField(i, +f, c)
InField(i, +f, c) <=> InField(i+1, +f, c)
f != f' => (!InField(i, +f, c) v !InField(i, +f', c))

Token(+t, i, c) ^ InField(i, +f, c) ^ Token(+t, i', c')
  ^ InField(i', +f, c') => SameField(+f, c, c')
SameField(+f, c, c') <=> SameCit(c, c')
SameField(f, c, c') ^ SameField(f, c', c'')
  => SameField(f, c, c'')
SameCit(c, c') ^ SameCit(c', c'') => SameCit(c, c'')
Formulas

Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,c’)
^ InField(i’,+f,c’) => SameField(+f,c,c’)
SameField(+f,c,c’) <=> SameCit(c,c’)
SameField(f,c,c’) ^ SameField(f,c’,c”) => SameField(f,c,c’’)
SameCit(c,c’) ^ SameCit(c’,c”) => SameCit(c,c”)

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications
Formulas

\[ \text{Token}(+t,i,c) \Rightarrow \text{InField}(i,+f,c) \]
\[ \text{InField}(i,+f,c) \Leftrightarrow \text{InField}(i+1,+f,c) \]
\[ f \neq f' \Rightarrow (\neg \text{InField}(i,+f,c) \lor \neg \text{InField}(i,+f',c)) \]

\[ \text{Token}(+t,i,c) \land \text{InField}(i,+f,c) \land \text{Token}(+t,i',c') \land \text{InField}(i',+f,c') \Rightarrow \text{SameField}(+f,c,c') \]
\[ \text{SameField}(+f,c,c') \Leftrightarrow \text{SameCit}(c,c') \]
\[ \text{SameField}(f,c,c') \land \text{SameField}(f,c',c'') \Rightarrow \text{SameField}(f,c,c'') \]
\[ \text{SameCit}(c,c') \land \text{SameCit}(c',c'') \Rightarrow \text{SameCit}(c,c'') \]
Formulas

Token(+t,i,c) \implies \text{InField}(i,+f,c)  
\text{InField}(i,+f,c) \iff \text{InField}(i+1,+f,c)  
f \neq f' \implies (\neg \text{InField}(i,+f,c) \lor \neg \text{InField}(i,+f',c))  

\text{Token}(+t,i,c) \land \text{InField}(i,+f,c) \land \text{Token}(+t,i',c')  
\land \text{InField}(i',+f,c') \implies \text{SameField}(+f,c,c')  
\text{SameField}(+f,c,c') \iff \text{SameCit}(c,c')  
\text{SameField}(f,c,c') \land \text{SameField}(f,c',c'') \implies \text{SameField}(f,c,c'')  
\text{SameCit}(c,c') \land \text{SameCit}(c',c'') \implies \text{SameCit}(c,c'')
Formulas

Token(+t,i,c) \implies \text{InField}(i,+f,c)
\text{InField}(i,+f,c) \iff \text{InField}(i+1,+f,c)
f \neq f' \implies (\neg \text{InField}(i,+f,c) \lor \neg \text{InField}(i,+f',c))

\begin{align*}
\text{Token}(+t,i,c) \land \text{InField}(i,+f,c) \land \text{Token}(+t,i',c') \\
\land \text{InField}(i',+f,c') & \implies \text{SameField}(+f,c,c') \\
\text{SameField}(+f,c,c') & \iff \text{SameCit}(c,c')
\end{align*}

\begin{align*}
\text{SameField}(f,c,c') \land \text{SameField}(f,c',c'') & \implies \text{SameField}(f,c,c'') \\
\text{SameCit}(c,c') \land \text{SameCit}(c',c'') & \implies \text{SameCit}(c,c'')
\end{align*}
Formulas

Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
^ InField(i',+f,c') => SameField(+f,c,c')

SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c'')
=> SameField(f,c,c'')
SameCit(c,c') ^ SameCit(c',c'') => SameCit(c,c'')
Formulas

Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,c’)
  ^ InField(i’,+f,c’) => SameField(+f,c,c’)
SameField(+f,c,c’) <=> SameCit(c,c’)

SameField(f,c,c’) ^ SameField(f,c’,c”)
  => SameField(f,c,c”)
SameCit(c,c’) ^ SameCit(c’,c”) => SameCit(c,c”)
Formulas

\[ \text{Token}(+t,i,c) \implies \text{InField}(i,+f,c) \]
\[ \text{InField}(i,+f,c) \land \lnot \text{Token}(\"\.",i,c) \iff \text{InField}(i+1,+f,c) \]
\[ f \neq f' \implies (\lnot \text{InField}(i,+f,c) \lor \lnot \text{InField}(i,+f',c)) \]
\[ \text{Token}(+t,i,c) \land \text{InField}(i,+f,c) \land \text{Token}(+t,i',c') \land \text{InField}(i',+f,c') \implies \text{SameField}(+f,c,c') \]
\[ \text{SameField}(+f,c,c') \iff \text{SameCit}(c,c') \]
\[ \text{SameField}(f,c,c') \land \text{SameField}(f,c',c'') \implies \text{SameField}(f,c,c'') \]
\[ \text{SameCit}(c,c') \land \text{SameCit}(c',c'') \implies \text{SameCit}(c,c'') \]
Results: Segmentation on Cora

![Graph showing the results of segmentation on Cora dataset. The x-axis represents Recall, and the y-axis represents Precision. Different lines represent various segmentation methods: Tokens, Tokens + Sequence, Tok. + Seq. + Period, and Tok. + Seq. + P. + Comma.]
Results:
Matching Venues on Cora

![Graph showing precision and recall for matching venues on Cora. The graph compares different methods: Similarity, Sim. + Relations, Sim. + Transitivity, and Sim. + Rel. + Trans. Each method is represented by a different line on the graph. The x-axis represents recall, ranging from 0 to 1, and the y-axis represents precision, also ranging from 0 to 1. The graph illustrates how each method performs across different recall values, with Sim. + Rel. + Trans. generally performing better than the others.]
Overview

- Representation
- Inference
- Learning
- Applications
- Discussion
Foundations for Probabilistic Models

- Graphs are not enough
- We need logic
### Logical Models vs. Graphical Models (I)

<table>
<thead>
<tr>
<th></th>
<th>Graphical models</th>
<th>Logical models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required by probability theory</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Representable distributions</td>
<td>All (BNs)</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Positive (MNs)</td>
<td></td>
</tr>
<tr>
<td>Context-free independences</td>
<td>Some</td>
<td>All</td>
</tr>
<tr>
<td>Context-specific independences</td>
<td>None</td>
<td>All</td>
</tr>
<tr>
<td>Normalization constraints</td>
<td>Some</td>
<td>All</td>
</tr>
</tbody>
</table>
### Logical Models vs. Graphical Models (II)

<table>
<thead>
<tr>
<th></th>
<th>Graphical models</th>
<th>Logical models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>$\text{Exp(treewidth)}$</td>
<td>Circuit complexity</td>
</tr>
<tr>
<td>Visual aid</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Densely connected distrs.</td>
<td>Unreadable</td>
<td>Readable</td>
</tr>
<tr>
<td>First-order</td>
<td>Plates</td>
<td>All</td>
</tr>
<tr>
<td>Lifted inference</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Available technology</td>
<td>Lots, used</td>
<td>Lots, unused</td>
</tr>
</tbody>
</table>