Unifying Logical and Statistical AI

Pedro Domingos
Dept. of Computer Science & Eng.
University of Washington

Joint work with Stanley Kok, Hoifung Poon, Matt Richardson and Parag Singla
Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion
AI: The First 100 Years

IQ

1956
2006
2056

Human Intelligence

Artificial Intelligence
AI: The First 100 Years

IQ

1956  2006  2056

Artificial Intelligence

Human Intelligence
Logical and Statistical AI

<table>
<thead>
<tr>
<th>Field</th>
<th>Logical approach</th>
<th>Statistical approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge representation</td>
<td>First-order logic</td>
<td>Graphical models</td>
</tr>
<tr>
<td>Automated reasoning</td>
<td>Satisfiability testing</td>
<td>Markov chain Monte Carlo</td>
</tr>
<tr>
<td>Machine learning</td>
<td>Inductive logic programming</td>
<td>Neural networks</td>
</tr>
<tr>
<td>Planning</td>
<td>Classical planning</td>
<td>Markov decision processes</td>
</tr>
<tr>
<td>Natural language processing</td>
<td>Definite clause grammars</td>
<td>Prob. context-free grammars</td>
</tr>
</tbody>
</table>
We Need to Unify the Two

- The real world is complex and uncertain
- Logic handles complexity
- Probability handles uncertainty
Progress to Date

- Probabilistic logic [Nilsson, 1986]
- Statistics and beliefs [Halpern, 1990]
- Knowledge-based model construction [Wellman et al., 1992]
- Stochastic logic programs [Muggleton, 1996]
- Probabilistic relational models [Friedman et al., 1999]
- Relational Markov networks [Taskar et al., 2002]
- Etc.
- **This talk: Markov logic** [Richardson & Domingos, 2004]
Markov Logic

- **Syntax**: Weighted first-order formulas
- **Semantics**: Templates for Markov nets
- **Inference**: WalkSAT, MCMC, KBMC
- **Learning**: Voted perceptron, pseudo-likelihood, inductive logic programming
- **Software**: Alchemy
- **Applications**: Information extraction, link prediction, etc.
Overview

- Motivation
- **Background**
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion
Markov Networks

- **Undirected** graphical models

- 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

![Graphical Model Diagram](image)

- Log-linear model:

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

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

\[w_1 = 1.5\]
First-Order Logic

- Constants, variables, functions, predicates
  E.g.: Anna, X, mother_of(X), friends(X, Y)
- Grounding: Replace all variables by constants
  E.g.: friends (Anna, Bob)
- **World** (model, interpretation):
  Assignment of truth values to all ground predicates
Overview

- Motivation
- Background
- **Markov logic**
- Inference
- Learning
- Software
- Applications
- Discussion
Markov Logic

- A logical KB is a set of **hard constraints** on the set of possible worlds.
- Let’s make them **soft constraints**: When a world violates a formula, it becomes less probable, not impossible.
- Give each formula a **weight** (Higher weight $\Rightarrow$ Stronger constraint).

$$P(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)$$
Definition

- 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
- Together with a set of constants, it 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: Friends & Smokers

Smoking causes cancer.
Friends have similar smoking habits.
Example: Friends & Smokers

\[ \forall x \text{ Smokes}(x) \implies \text{Cancer}(x) \]
\[ \forall x, y \text{ Friends}(x, y) \implies (\text{Smokes}(x) \iff \text{Smokes}(y)) \]
**Example: Friends & Smokers**

<table>
<thead>
<tr>
<th></th>
<th>∀x Smokes(x) ⇒ Cancer(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>∀x, y Friends(x, y) ⇒ (Smokes(x) ⇔ Smokes(y))</td>
</tr>
</tbody>
</table>
**Example: Friends & Smokers**

<table>
<thead>
<tr>
<th>1.5</th>
<th>$\forall x \text{ Smokes}(x) \implies \text{Cancer}(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$\forall x, y \text{ Friends}(x, y) \implies (\text{Smokes}(x) \iff \text{Smokes}(y))$</td>
</tr>
</tbody>
</table>

Two constants: **Anna** (A) and **Bob** (B)
Example: Friends & Smokers

Two constants: Anna (A) and Bob (B)

<table>
<thead>
<tr>
<th></th>
<th>1.5</th>
<th>( \forall x \ Smokes(x) \Rightarrow Cancer(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>( \forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \leftrightarrow Smokes(y)) )</td>
<td></td>
</tr>
</tbody>
</table>
Example: Friends & Smokers

1.5 $\forall x \ Smokes(x) \Rightarrow Cancer(x)$
1.1 $\forall x, y \ Friends(x, y) \Rightarrow (\ Smokes(x) \Leftrightarrow Smokes(y) \ )$

Two constants: Anna (A) and Bob (B)
Example: Friends & Smokers

1.5 \( \forall x \ Smokes(x) \Rightarrow Cancer(x) \)
1.1 \( \forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y)) \)

Two constants: Anna (A) and Bob (B)
Example: Friends & Smokers

Two constants: **Anna** (A) and **Bob** (B)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>$\forall x \ Smokes(x) \Rightarrow Cancer(x)$</td>
</tr>
<tr>
<td>1.1</td>
<td>$\forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \leftrightarrow Smokes(y))$</td>
</tr>
</tbody>
</table>
Markov Logic Networks

- **MLN is template** for ground Markov nets
- Probability of a world $x$:

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

- **Typed** variables and constants greatly reduce size of ground Markov net
- Functions, existential quantifiers, etc.
- Open question: Infinite domains
Relation to Statistical Models

- Special cases:
  - Markov networks
  - Markov random fields
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Gibbs distributions
  - Boltzmann machines
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields

- Obtained by making all predicates zero-arity

- Markov logic allows objects to be interdependent (non-i.i.d.)

- Discrete distributions
Relation to First-Order Logic

- Infinite weights $\Rightarrow$ First-order logic
- Satisfiable KB, positive weights $\Rightarrow$
  Satisfying assignments = Modes of distribution
- Markov logic allows contradictions between formulas
Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion
MAP/MPE Inference

- **Problem**: Find most likely state of world given evidence

\[
\max_y P(y \mid x)
\]

- Query
- Evidence
MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

\[
\max_y \frac{1}{Z_x} \exp\left( \sum_i w_in_i(x, y) \right)
\]
MAP/MPE Inference

- **Problem**: Find most likely state of world given evidence

\[
\max_y \sum_i w_i n_i(x, y)
\]
MAP/MPE Inference

- **Problem**: Find most likely state of world given evidence
  \[
  \max_y \sum_i w_i n_i(x, y)
  \]
  - This is just the weighted MaxSAT problem
  - Use weighted SAT solver (e.g., MaxWalkSAT [Kautz et al., 1997])
  - Potentially faster than logical inference (!)
The WalkSAT Algorithm

\[
\text{for } i \leftarrow 1 \text{ to } \text{max-tries do} \\
\quad \text{solution} = \text{random truth assignment} \\
\text{for } j \leftarrow 1 \text{ to } \text{max-flips do} \\
\quad \text{if all clauses satisfied then} \\
\quad \quad \text{return solution} \\
\quad c \leftarrow \text{random unsatisfied clause} \\
\quad \text{with probability } p \\
\quad \quad \text{flip a random variable in } c \\
\quad \text{else} \\
\quad \quad \text{flip variable in } c \text{ that maximizes number of satisfied clauses} \\
\text{return failure}
\]
The MaxWalkSAT Algorithm

for $i \leftarrow 1$ to $max-tries$ do
  $solution = \text{random truth assignment}$
  for $j \leftarrow 1$ to $max-flips$ do
    if $\sum \text{weights(sat. clauses)} > \text{threshold}$ then
      return $solution$
    $c \leftarrow \text{random unsatisfied clause}$
    with probability $p$
      flip a random variable in $c$
    else
      flip variable in $c$ that maximizes $\sum \text{weights(sat. clauses)}$
  return failure, best $solution$ found
But ... Memory Explosion

● **Problem:**
  If there are $n$ constants and the highest clause arity is $c$, the ground network requires $O(n^c)$ memory

● **Solution:**
  Exploit sparseness; ground clauses lazily
  $\rightarrow$ LazySAT algorithm [Singla & Domingos, 2006]
Computing Probabilities

- $P(\text{Formula}|\text{MLN,C}) = ?$
- MCMC: Sample worlds, check formula holds
- $P(\text{Formula}_1|\text{Formula}_2,\text{MLN,C}) = ?$
- If $\text{Formula}_2 = \text{Conjunction of ground atoms}$
  - First construct min subset of network necessary to answer query (generalization of KBMC)
  - Then apply MCMC (or other)
- Can also do lifted inference [Braz et al, 2005]
Ground Network Construction

\[ \text{network} \leftarrow \emptyset \]
\[ \text{queue} \leftarrow \text{query nodes} \]
\[ \text{repeat} \]
\[ \ \quad \text{node} \leftarrow \text{front(queue)} \]
\[ \ \quad \text{remove node from queue} \]
\[ \ \quad \text{add node to network} \]
\[ \ \quad \text{if node not in evidence then} \]
\[ \ \quad \quad \text{add neighbors(node) to queue} \]
\[ \text{until} \ \text{queue} = \emptyset \]
MCMC: Gibbs Sampling

\[\text{state} \leftarrow \text{random truth assignment}\]
\[\text{for } i \leftarrow 1 \text{ to num-samples do}\]
  \[\text{for each variable } x\]
  \[\text{sample } x \text{ according to } P(x|\text{neighbors}(x))\]
  \[\text{state} \leftarrow \text{state with new value of } x\]
\[P(F) \leftarrow \text{fraction of states in which } F \text{ is true}\]
But ... Insufficient for Logic

- **Problem:**
  Deterministic dependencies break MCMC
  Near-deterministic ones make it very slow

- **Solution:**
  Combine MCMC and WalkSAT
  → MC-SAT algorithm  [Poon & Domingos, 2006]
Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- 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

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

- 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 \cite{Besag, 1975}
- Does not require inference at each step
- 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

- Approximate expected counts by counts in MAP state of \( y \) given \( x \)
Voted Perceptron

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

\[

d_{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 } \frac{\sum_{t} w_{i}}{T}
\]
Voted Perceptron for MLNs

- HMMs are special case of MLNs
- Replace Viterbi by MaxWalkSAT
- Network can now be arbitrary graph

\[
\begin{align*}
  w_i &\leftarrow 0 \\
  \text{for } t &\leftarrow 1 \text{ to } T \text{ do} \\
  y_{\text{MAP}} &\leftarrow \text{MaxWalkSAT}(x) \\
  w_i &\leftarrow w_i + \eta \left[ \text{count}_i(y_{\text{Data}}) - \text{count}_i(y_{\text{MAP}}) \right] \\
  \text{return } \sum_t w_i / T
\end{align*}
\]
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 search, shortest-first search
Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- **Software**
- Applications
- Discussion
Alchemy

Open-source software including:

- Full first-order logic syntax
- Generative & discriminative weight learning
- Structure learning
- Weighted satisfiability and MCMC
- Programming language features

www.cs.washington.edu/ai/alchemy
<table>
<thead>
<tr>
<th></th>
<th>Alchemy</th>
<th>Prolog</th>
<th>BUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation</strong></td>
<td>F.O. Logic + Markov nets</td>
<td>Horn clauses</td>
<td>Bayes nets</td>
</tr>
<tr>
<td><strong>Inference</strong></td>
<td>Model checking, MCMC</td>
<td>Theorem proving</td>
<td>MCMC</td>
</tr>
<tr>
<td><strong>Learning</strong></td>
<td>Parameters &amp; structure</td>
<td>No</td>
<td>Params.</td>
</tr>
<tr>
<td><strong>Uncertainty</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Relational</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion
Applications

- Information extraction*
- Entity resolution
- Link prediction
- Collective classification
- Web mining
- Natural language processing

- Computational biology
- Social network analysis
- Robot mapping
- Activity recognition
- Online games
- Probabilistic Cyc
- Etc.

* Markov logic approach won LLL-2005 information extraction competition [Riedel & Klein, 2005]
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).


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 = \{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)
Types and Predicates

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

Optional

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, ...}

\[\text{Evidence}\]
\[
\text{Token}(\text{token}, \text{position}, \text{citation})
\]
\[
\text{InField}(\text{position}, \text{field}, \text{citation})
\]
\[
\text{SameField}(\text{field}, \text{citation}, \text{citation})
\]
\[
\text{SameCit}(\text{citation}, \text{citation})
\]
Types and Predicates

\[
token = \{\text{Parag, Singla, and, Pedro, ...}\}
\]

\[
field = \{\text{Author, Title, Venue}\}
\]

\[
citation = \{C1, C2, ...\}
\]

\[
position = \{0, 1, 2, ...\}
\]

\[
\text{Token}(\text{token, position, citation})
\]

\[
\text{InField}(\text{position, field, citation})
\]

\[
\text{SameField}(\text{field, citation, citation})
\]

\[
\text{SameCit}(\text{citation, citation})
\]

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

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

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

\[
\text{SameField}(+f,c,c') \iff \text{SameCit}(c,c') \\
\text{SameField}(f,c,c') \land \text{SameField}(f,c',c'') \Rightarrow \text{SameField}(f,c,c'') \\
\text{SameCit}(c,c') \land \text{SameCit}(c',c'') \Rightarrow \text{SameCit}(c,c'')
\]
Formulas

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

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

Token(+t,i,c) => 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) \Rightarrow\text{ InField}(i,+f,c)
\text{InField}(i,+f,c) \iff \text{InField}(i+1,+f,c)
f \neq f' \Rightarrow (!\text{InField}(i,+f,c) \lor !\text{InField}(i,+f',c))

Token(+t,i,c) \land \text{InField}(i,+f,c) \land Token(+t,i',c')
\land \text{ InField}(i',+f,c') \Rightarrow \text{SameField}(+f,c,c')

SameField(+f,c,c') \iff \text{SameCit}(c,c')
SameField(f,c,c') \land 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) => 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

$\text{Token}(+t,i,c) \Rightarrow \text{InField}(i,+f,c)$

$\text{InField}(i,+f,c) \iff \text{InField}(i+1,+f,c)$

$f \neq f' \Rightarrow (\neg \text{InField}(i,+f,c) \lor \neg \text{InField}(i,+f',c))$

$\text{Token}(+t,i,c) \land \text{InField}(i,+f,c) \land \text{Token}(+t,i',c')$

$\land \text{InField}(i',+f,c') \Rightarrow \text{SameField}(+f,c,c')$

$\text{SameField}(+f,c,c') \iff \text{SameCit}(c,c')$

$\text{SameField}(f,c,c') \land \text{SameField}(f,c',c'')$

$\Rightarrow \text{SameField}(f,c,c'')$

$\text{SameCit}(c,c') \land \text{SameCit}(c',c'') \Rightarrow \text{SameCit}(c,c'')$
Formulas

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

\[ \text{Token}(+t,i,c) \land \text{InField}(i,+f,c) \land \text{Token}(+t,i',c') \land \text{InField}(i',+f,c') \Rightarrow \text{SameField}(+f,c,c') \]
\[ \text{SameField}(+f,c,c') \Leftrightarrow \text{SameCit}(c,c') \]
\[ \text{SameField}(f,c,c') \land \text{SameField}(f,c',c'') \Rightarrow \text{SameField}(f,c,c'') \]
\[ \text{SameCit}(c,c') \land \text{SameCit}(c',c'') \Rightarrow \text{SameCit}(c,c'') \]
Results: Segmentation on Cora

![Graph showing precision and recall for different segmentation methods.]

- **Tokens**
- **Tokens + Sequence**
- **Tok. + Seq. + Period**
- **Tok. + Seq. + P. + Comma**
Results:
Matching Venues on Cora

![Graph showing precision vs recall for different similarity measures. The x-axis represents recall ranging from 0 to 1, and the y-axis represents precision ranging from 0 to 1. The graph compares four measures:
- **Similarity** (blue line)
- **Sim. + Relations** (green line)
- **Sim. + Transitivity** (yellow line)
- **Sim. + Rel. + Trans.** (red line)

The graph illustrates how each measure performs in terms of precision at various recall levels, with **Sim. + Rel. + Trans.** generally showing the best performance.
Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion
The Interface Layer

Applications

Interface Layer

Infrastructure
Networking

Applications

WWW

Email

Interface Layer

Internet

Protocols

Infrastructure

Routers
Programming Systems

Applications

Interface Layer

High-Level Languages

Programming

Compilers

Infrastructure

Code Optimizers
Artificial Intelligence

Robotics

Planning

Applications

NLP

Multi-Agent Systems

Vision

Interface Layer

Representation

Infrastructure

Inference

Learning
Artificial Intelligence

Robotics

Applications

Planning

NLP

Multi-Agent

Systems

Vision

Interface Layer

First-Order Logic?

Representation

Infrastructure

Inference

Learning
Artificial Intelligence

Robotics

Applications

Planning

NLP

Multi-Agent Systems

Vision

Interface Layer

Graphical Models?

Representation

Infrastructure

Inference

Learning
Artificial Intelligence

Robotics

Planning

NLP

Multi-Agent Systems

Vision

Alchemy: www.cs.washington.edu/ai/alchemy

Applications

Representation

Infrastructure

Inference

Learning