Incorporating Logic into Markov Networks with Markov Logic

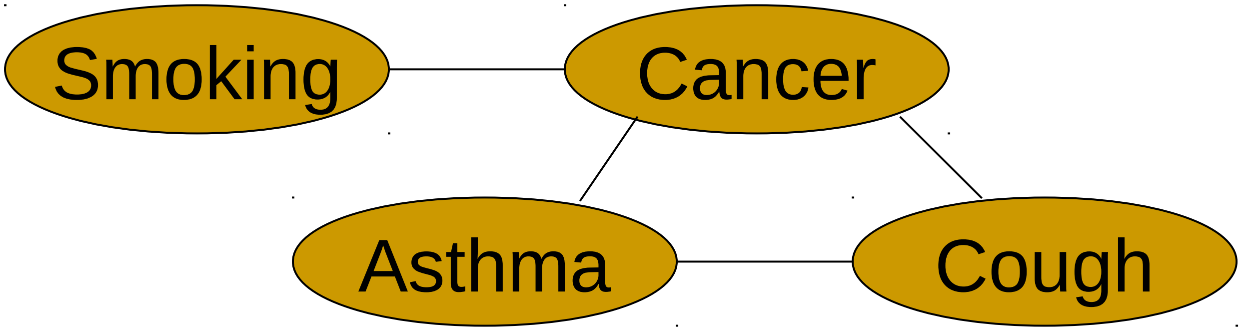
Not my slides (most from Pedro Domingos, some edits)

Overview

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

Markov Networks

- **Undirected** graphical models



- Potential functions defined over cliques

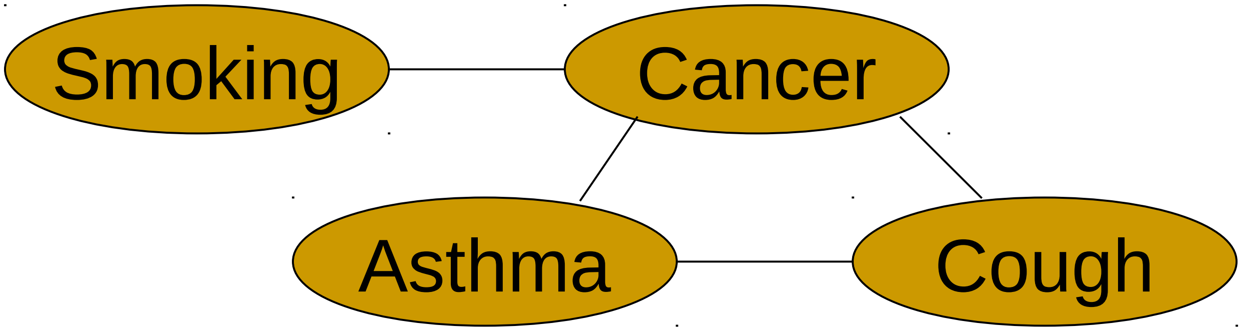
$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$Z = \sum_x \prod_c \Phi_c(x_c)$$

Smoking	Cancer	Φ(S,C)
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

Markov Networks

- **Undirected** graphical models



- Log-linear model:

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

Weight of Feature *i*

Feature *i*

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

$$w_1 = 1.5$$

First-Order Logic

- Constants, variables, functions, predicates
  E.g.: Anna, x, MotherOf(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
- 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)

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

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

$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$
$$\forall x, y \; Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$$

Example: Friends & Smokers

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

Example: Friends & Smokers

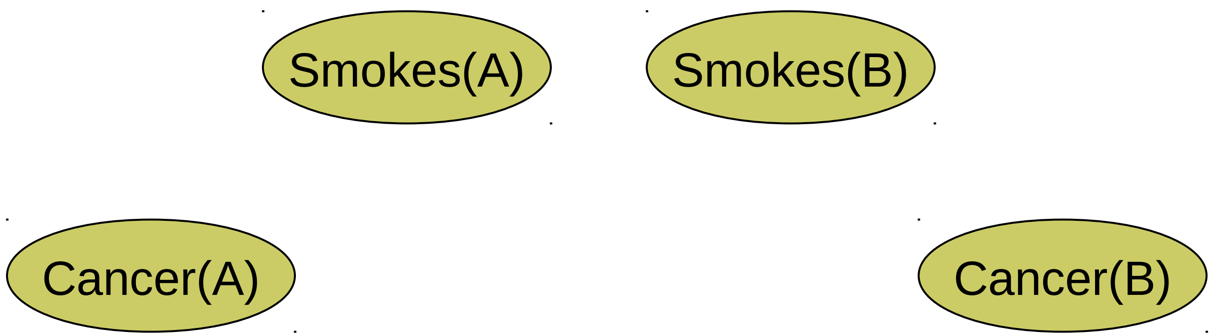
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

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

Example: Friends & Smokers

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

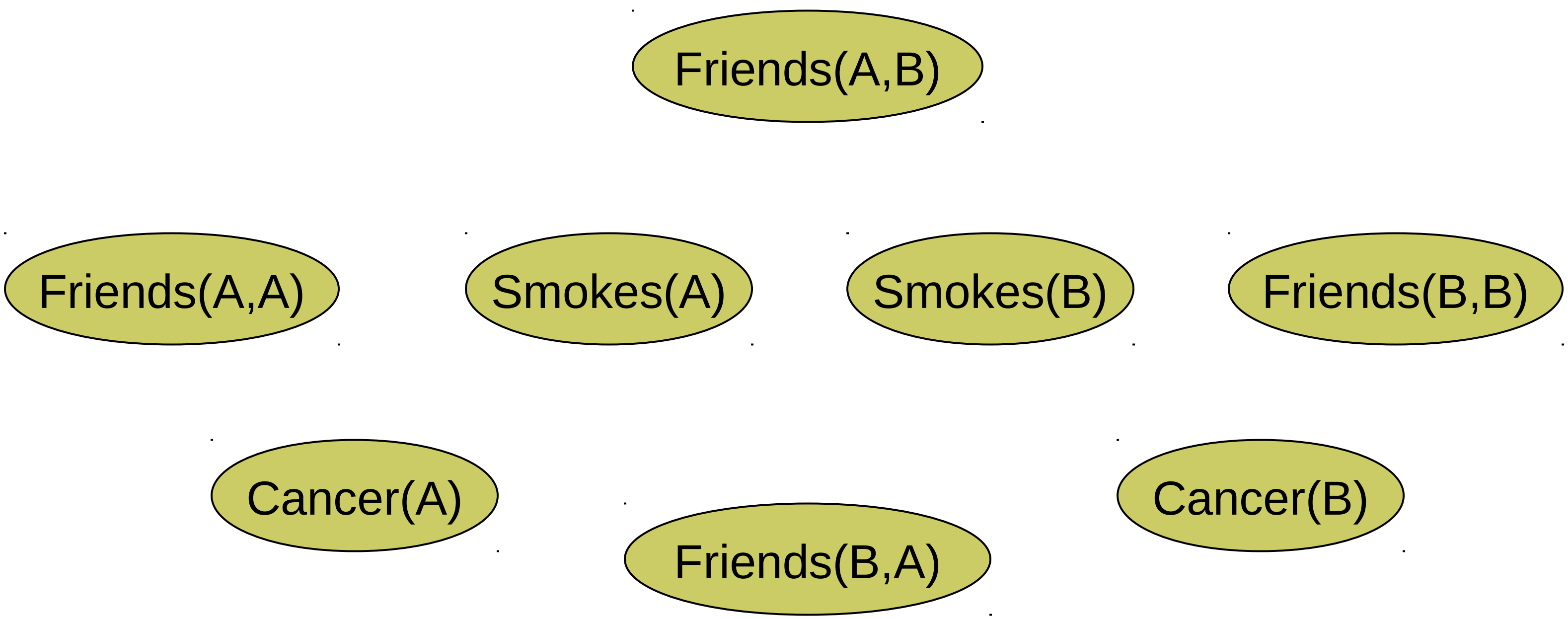
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

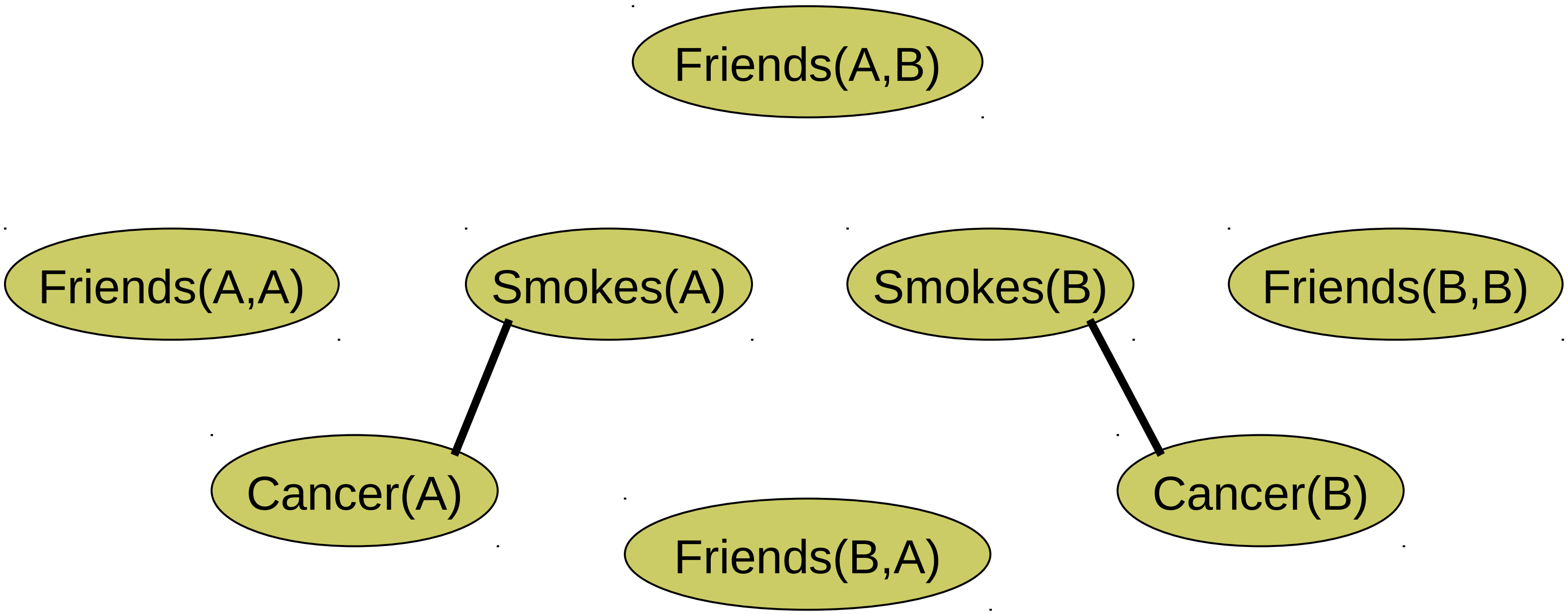
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

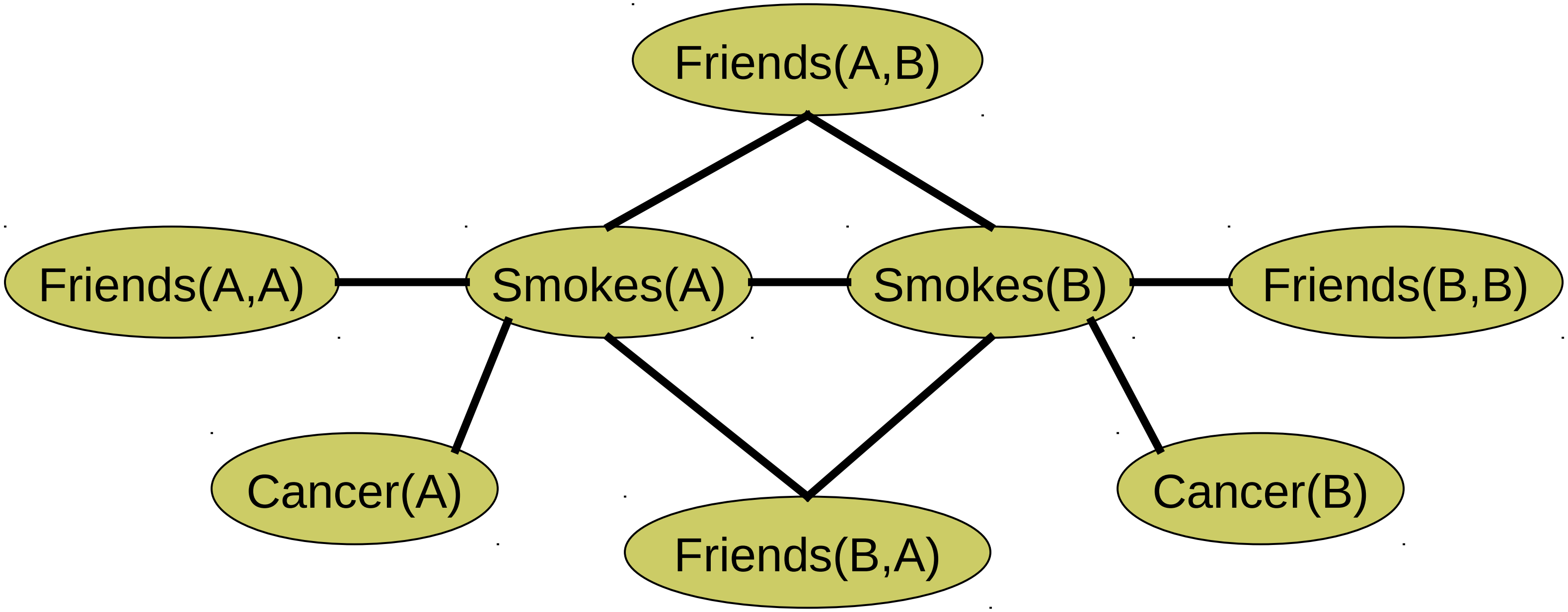
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

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



Markov Logic Networks

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

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

Weight of formula i

No. of true groundings of formula i in x

- Functions, existential quantifiers, etc.
- Infinite and continuous 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.)

Relation to First-Order Logic

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

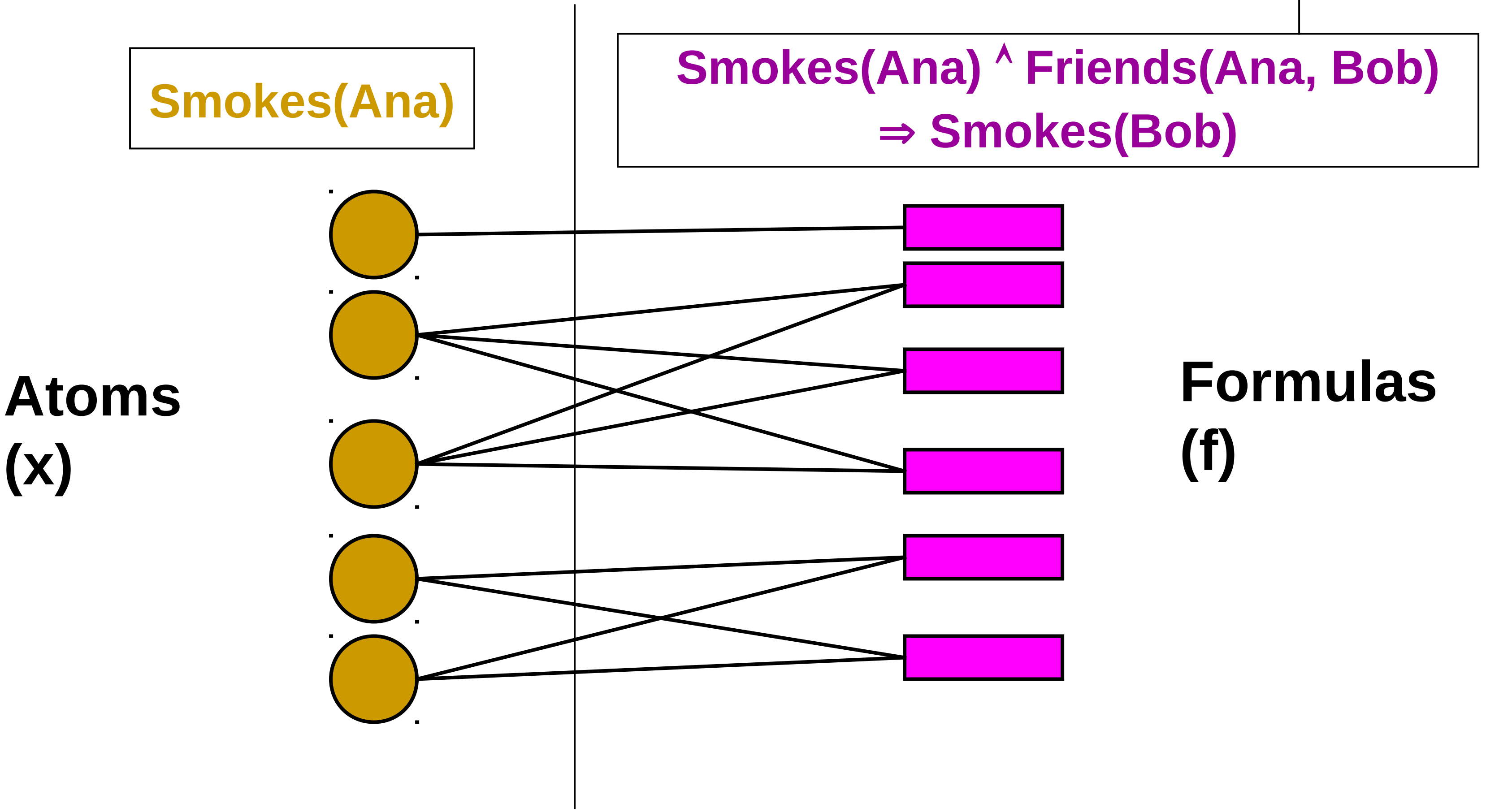
Overview

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

Belief Propagation

- Approximate inference for cyclic graphs
  - **Loopy Belief Propagation**
- Goal: Compute probabilities or MAP state
- Belief propagation: Subsumes Viterbi, etc.
- Bipartite network
  - Variables = Ground atoms
  - Features = Ground formulas
- Repeat until convergence:
  - Nodes send messages to their features
  - Features send messages to their variables
- Messages = Approximate marginals

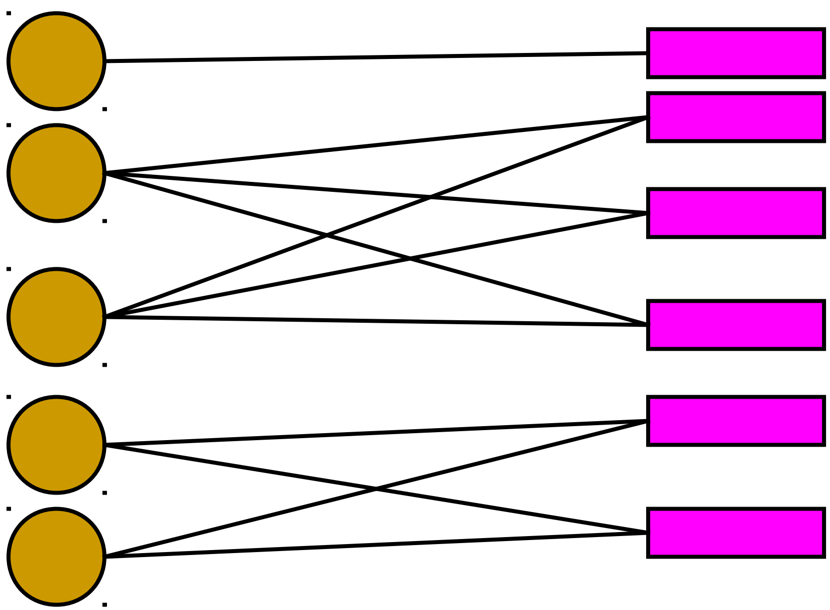
Belief Propagation



Belief Propagation

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x)$$

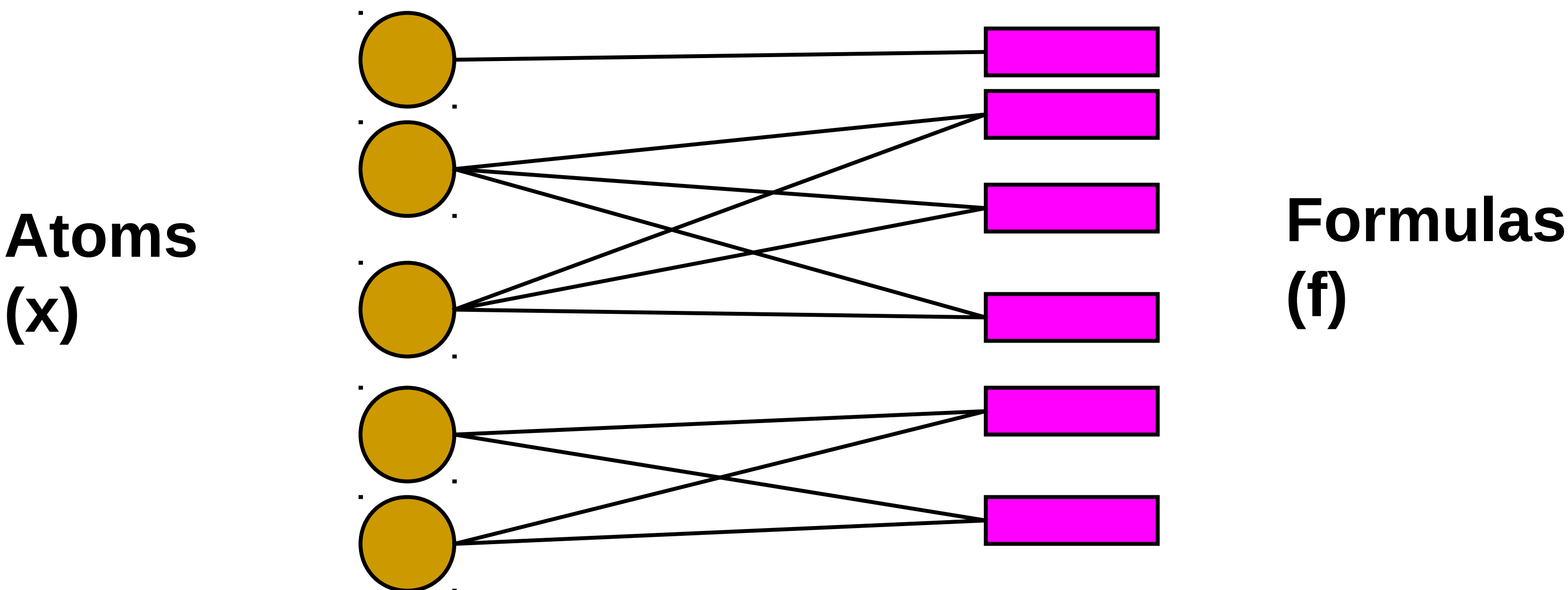
Atoms (x)



Formulas (f)

Belief Propagation

$$\mu_{x\to f}(x) = \prod_{h\in n(x)\setminus\{f\}} \mu_{h\to x}(x)$$



$$\mu_{f\to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y\in n(f)\setminus\{x\}} \mu_{y\to f}(y) \right)$$

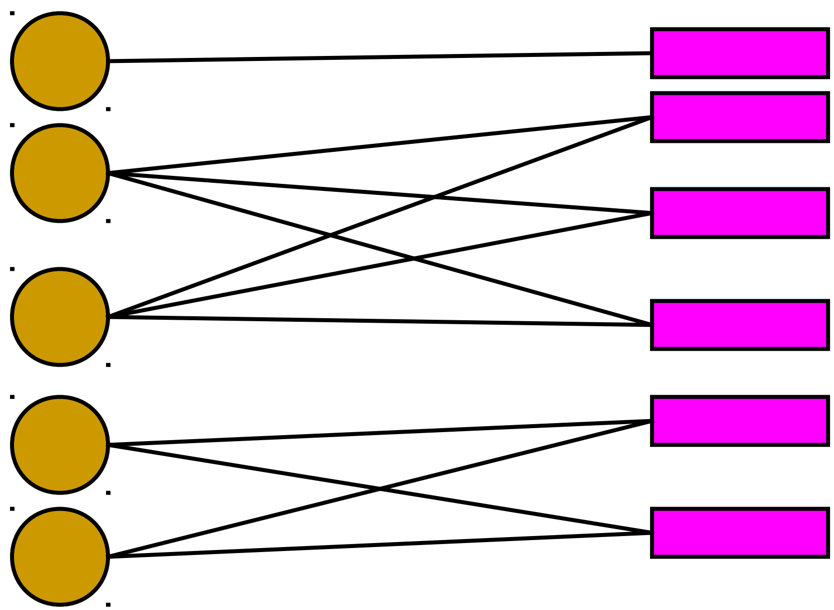
But This Is Too Slow

- One message for each atom/formula pair
- Can easily have **billions of formulas**
  - **Millions of atoms**
- Too many messages!
- Group atoms/formulas which pass same message (as in resolution)
- One message for each pair of clusters
- Greatly reduces the size of the network

Belief Propagation

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x)$$

Atoms (x)



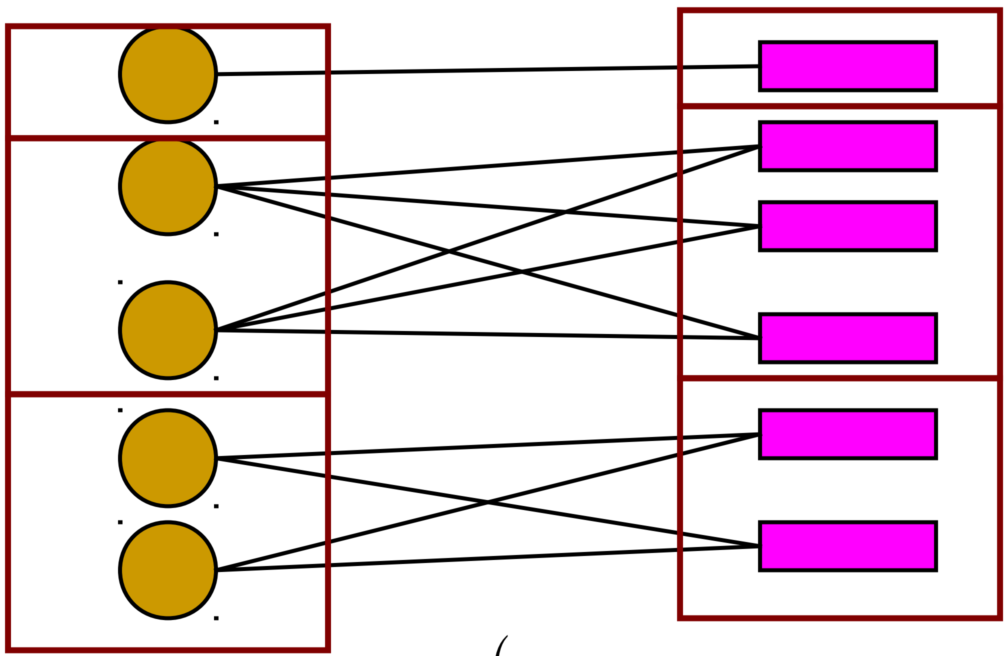
Formulas (f)

$$\mu_{f \to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

Lifted Belief Propagation

$$\mu_{x \to f}(x) = \prod_{h \in n(x) \setminus \{f\}} \mu_{h \to x}(x)$$

Atoms (x)

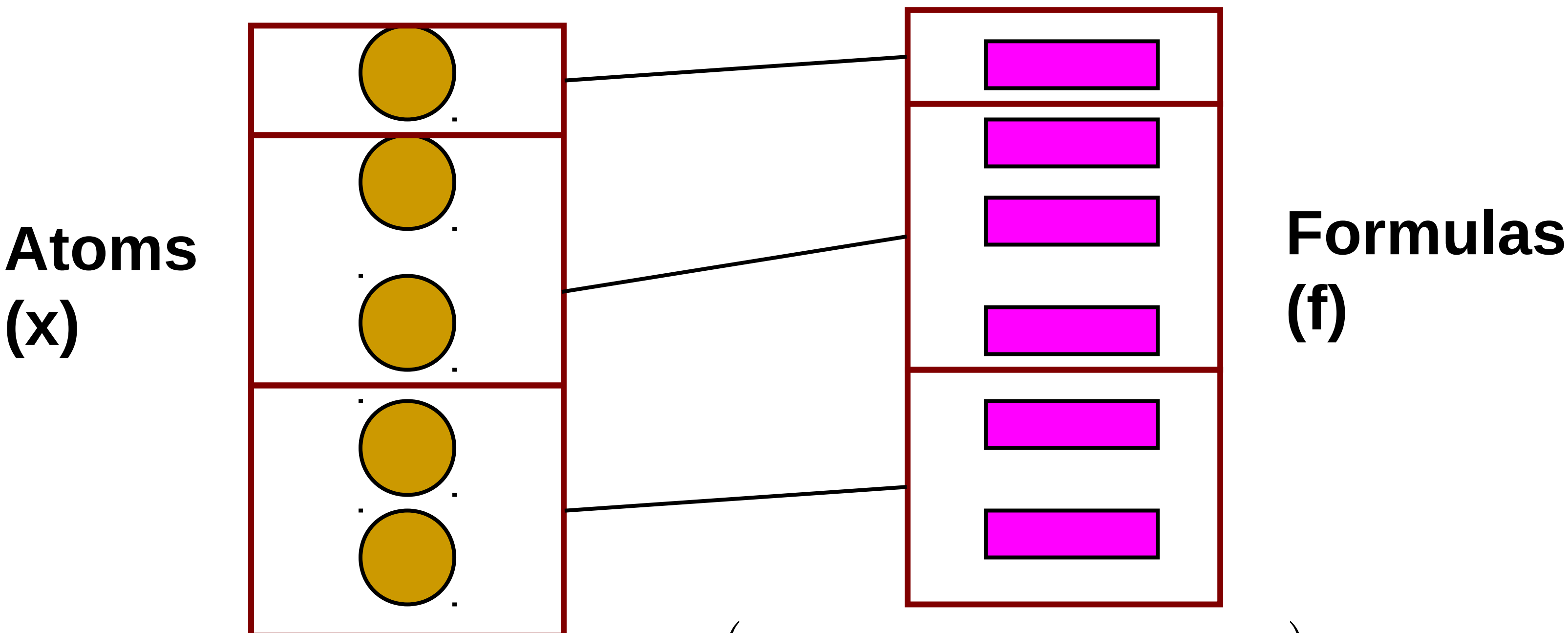


Formulas (f)

$$\mu_{f \to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y \in n(f) \setminus \{x\}} \mu_{y \to f}(y) \right)$$

Lifted Belief Propagation

$$\mu_{x\to f}(x) = \beta \prod_{h\in n(x)\setminus\{f\}} \mu_{h\to x}(x)^{\alpha}$$



$$\mu_{f\to x}(x) = \sum_{\sim\{x\}} \left(e^{wf(x)} \prod_{y\in n(f)\setminus\{x\}} \mu_{y\to f}(y) \right)$$

Why Lifted Belief Propagation?

- Inspired from Lifted resolution in Logic
- Therefore, we need not perform inference on the humongous network (only groups of atoms/nodes with similar properties)
- In logic, we have quantifiers like

$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$

$$\forall x, y \; Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$$

Lifted Belief Propagation

- Form **lifted network**
  - **Supernode:** Set of ground atoms that all send and receive same messages throughout BP
  - **Superfeature:** Set of ground clauses that all send and receive same messages throughout BP
- Run belief propagation on lifted network
- Same results as ground BP
- Time and memory savings can be huge

Forming the Lifted Network

1. **Form initial supernodes**
   One per predicate and truth value
   (true, false, unknown)
2. **Form superfeatures by doing joins of their supernodes**
3. **Form supernodes by projecting superfeatures down to their predicates**
   Supernode = Groundings of a predicate with same number of projections from each superfeature
4. **Repeat until convergence**

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): Voted perceptron
- Learning structure (formulas): Inductive logic programming

Weight Learning

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

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

No. of true groundings of clause *i* in data

Expected no. true groundings according to model

Voted Perceptron

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

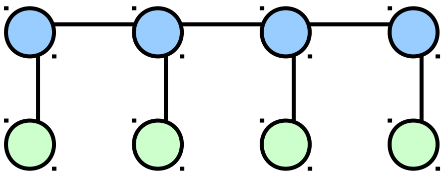
$w_i \leftarrow 0$

for $t \leftarrow 1$ **to** T **do**

  $y_{MAP} \leftarrow \text{Viterbi}(x)$

  $w_i \leftarrow w_i + \eta\,[\text{count}_i(y_{Data}) - \text{count}_i(y_{MAP})]$

return $\sum_t w_i \,/\, T$



Voted Perceptron for MLNs

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

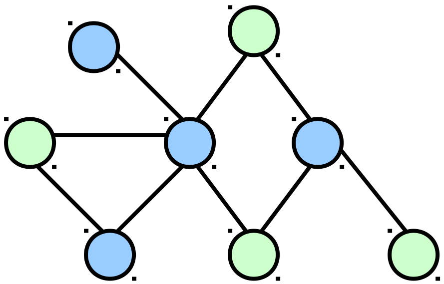
$w_i \leftarrow 0$

for $t \leftarrow 1$ **to** T **do**

$\quad y_{MLN} \leftarrow \text{LiftedBP}(x)$

$\quad w_i \leftarrow w_i + \eta\ [\text{count}_{\text{i}}(y_{Data}) - \text{count}_{\text{i}}(y_{MLN})]$

return $\sum_t w_i \,/\, \mathrm{T}$



Overview

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

Information Extraction

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

Singla, P., & Domingos, P. (2006). Memory-efficent inference in relatonal domains. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 500-505). Boston, MA: AAAI Press.

H. Poon & P. Domingos, Sound and Efficient Inference with Probabilistic and Deterministic Dependencies”, in Proc. AAAI-06, Boston, MA, 2006.

P. Hoifung (2006). Efficent inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.

Segmentation

Author

Title

Venue

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

Singla, P., & Domingos, P. (2006). Memory-efficent inference in relatonal domains. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 500-505). Boston, MA: AAAI Press.

H. Poon & P. Domingos, Sound and Efficient Inference with Probabilistic and Deterministic Dependencies", in Proc. AAAI-06, Boston, MA, 2006.

P. Hoifung (2006). Efficent inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.

Entity Resolution

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

Singla, P., & Domingos, P. (2006). Memory-efficent inference in relational domains. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 500-505). Boston, MA: AAAI Press.

H. Poon & P. Domingos, Sound and Efficient Inference with Probabilistic and Deterministic Dependencies”, in Proc. AAAI-06, Boston, MA, 2006.

P. Hoifung (2006). Efficent inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.

Entity Resolution

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

Singla, P., & Domingos, P. (2006). Memory-efficent inference in relatonal domains. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (pp. 500-505). Boston, MA: AAAI Press.

H. Poon & P. Domingos, Sound and Efficient Inference with Probabilistic and Deterministic Dependencies”, in Proc. AAAI-06, Boston, MA, 2006.

P. Hoifung (2006). Efficent inference. In Proceedings of the Twenty-First National Conference on Artificial Intelligence.

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
- Markov logic implementation: Seven formulas

Types and Predicates

```
token = {Parag, Singla, and, Pedro, ...}
field = {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, ...}
```

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, ...}

Token(token, position, citation)  ← Evidence
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, ...}

Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
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″)
```

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

```
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”)
```

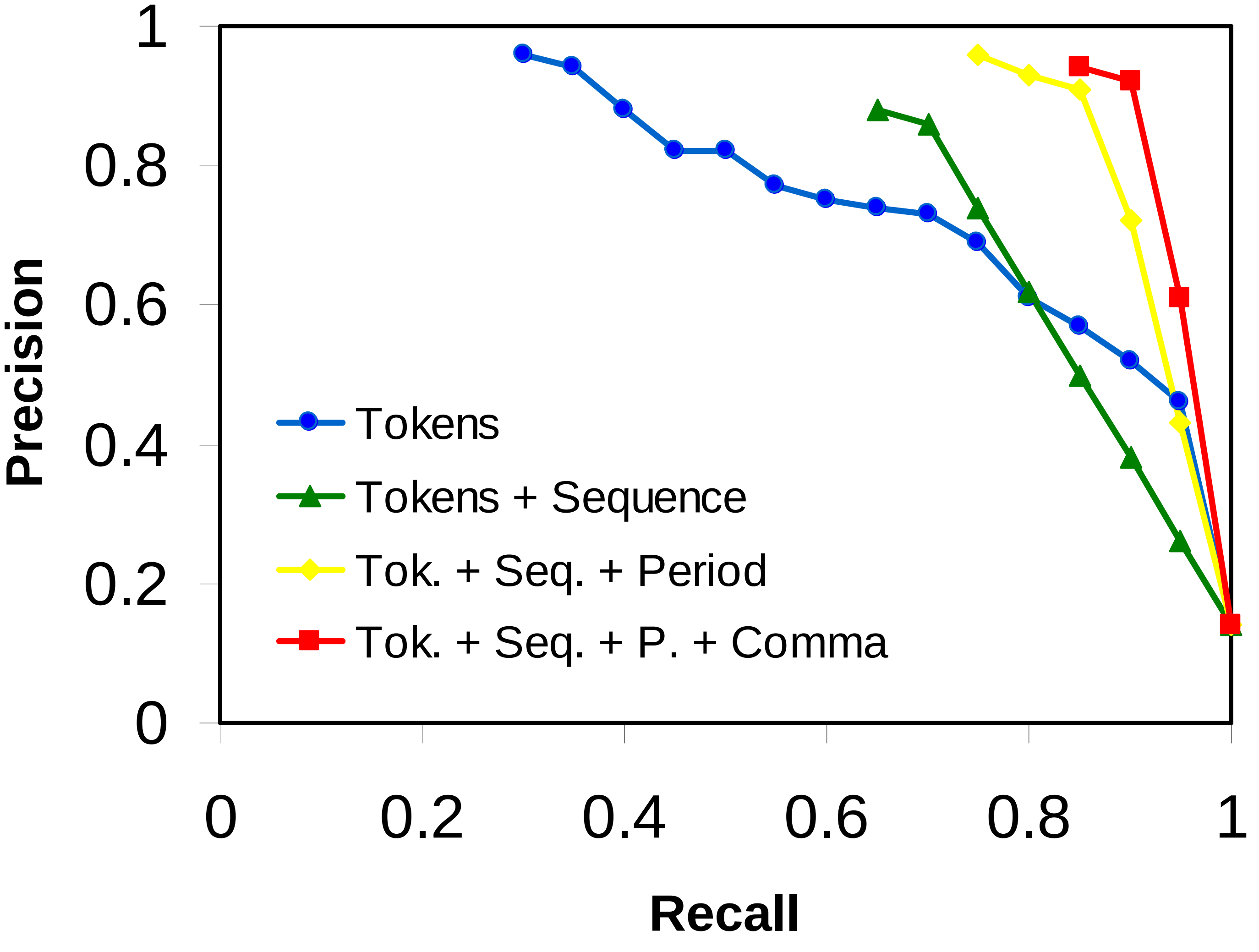
Formulas

```
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")
```

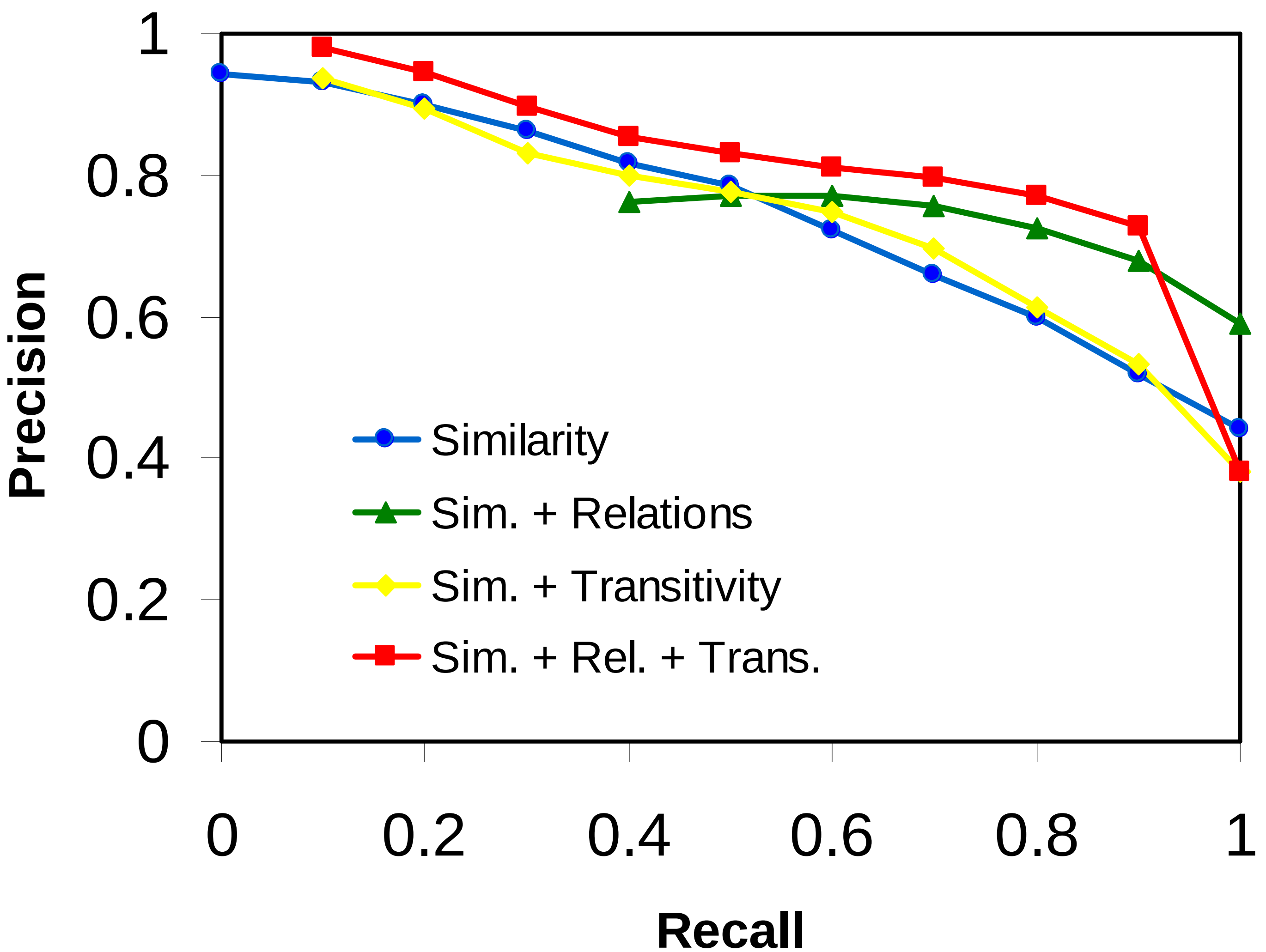
Formulas

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) ^ !Token(“.”,i,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”)
```

Results: Segmentation on Cora



Results: Matching Venues on Cora



Other Examples

- **Citation matching**
  Best results to date on CiteSeer and Cora
  [Poon & Domingos, AAAI-07]
- **Unsupervised coreference resolution**
  Best results to date on MUC and ACE
  (better than supervised)
  [under review]
- **Ontology induction**
  From TextRunner output (2 million tuples)
  [Kok & Domingos, ECML-08]

Overview

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

Next Steps

- Induce knowledge over ontology using structure learning
- Apply Markov logic to other NLP tasks
- Connect the pieces
- Close the loop

Conclusion

- Language and knowledge: Chicken and egg
- Solution: Bootstrap
- Markov logic provides language & algorithms
  - Weighted first-order formulas → Markov network
  - Lifted belief propagation
  - Voted perceptron
- Several successes to date
- Open-source software: Alchemy

alchemy.cs.washington.edu