Advanced Markov Logic Techniques for Scalable Joint Inference in NLP

Deepak Venugopal*, Vibhav Gogate** and Vincent Ng**

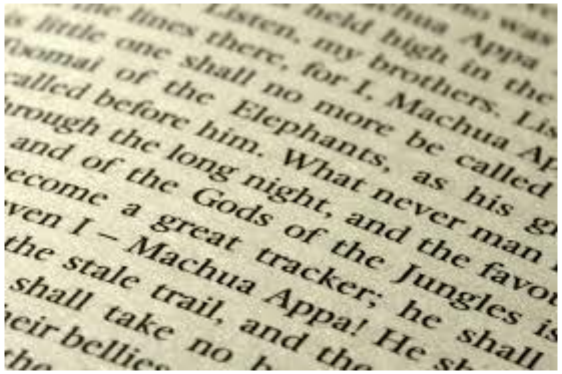
* The University of Memphis
** The University of Texas at Dallas

Outline

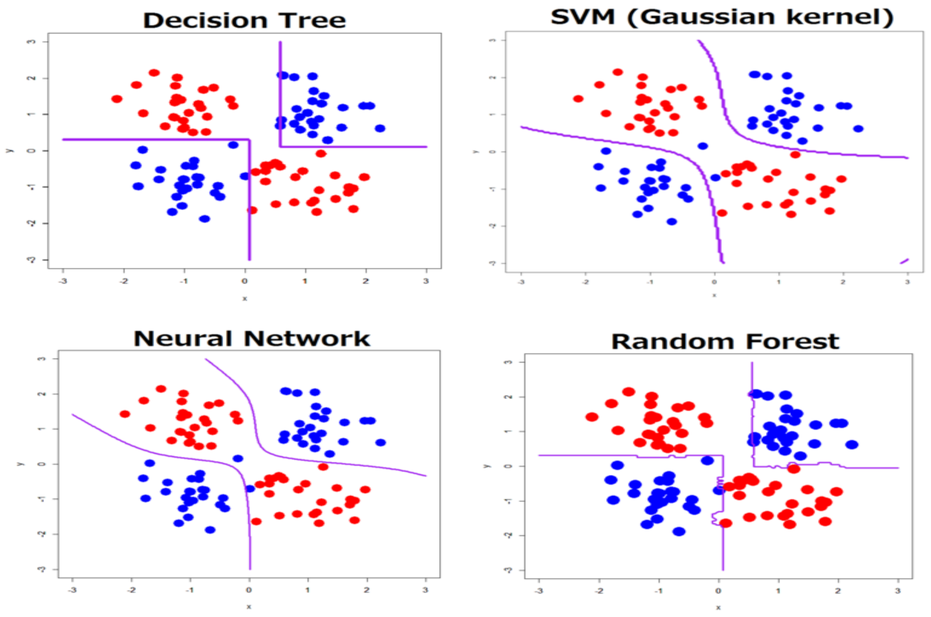
- Motivation
- MLN Basics
- Domain-Lifted Inference
- Approximate Domain-Lifting
- Lifted Generative Weight Learning
- Scalable Weight Learning with Approximate Counting Oracles
- Applications
- Conclusion

Outline

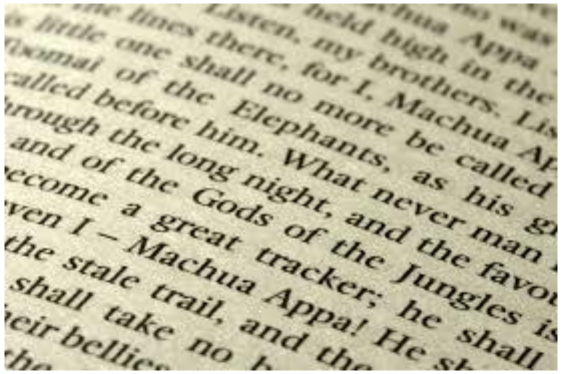
Traditional Machine Learning



Training Data
Instance_1
Instance_2
Instance_3
….



Traditional Machine Learning



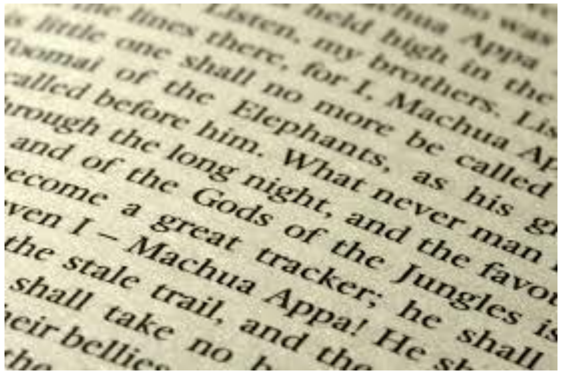
who was
hua Appa
held high in the
Listen, my brothers. Lis
the lines there, for I, Machua Ap
s little one shall no more be called
Toomai of the Elephants, as his g
called before him. What never man
rough the long night, and the favo
and of the Gods of the Jungles is
ecome a great tracker; he shall
ven I – Machua Appa! He sh
the stale trail, and the
shall take no h
eir bellies
the

Training Data
Instance_1
Instance_2
Instance_3
….

- Dependency trees
- Part-subpart relationships,
- Entity-event relationships
- Temporal relations
- Predicate-argument relationship
- …

Hard to translate to relations to feature vectors

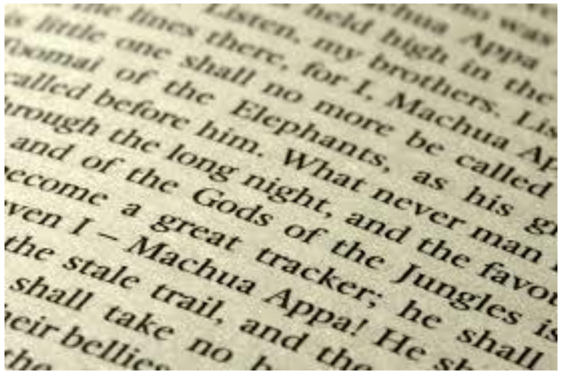
Traditional Machine Learning



Training Data
Instance_1
Instance_2
Instance_3
….

- Forces instances to be i.i.d
  - Instances in language are clearly interdependent.
  - E.g. an event depends upon other described events in the text

Traditional Machine Learning



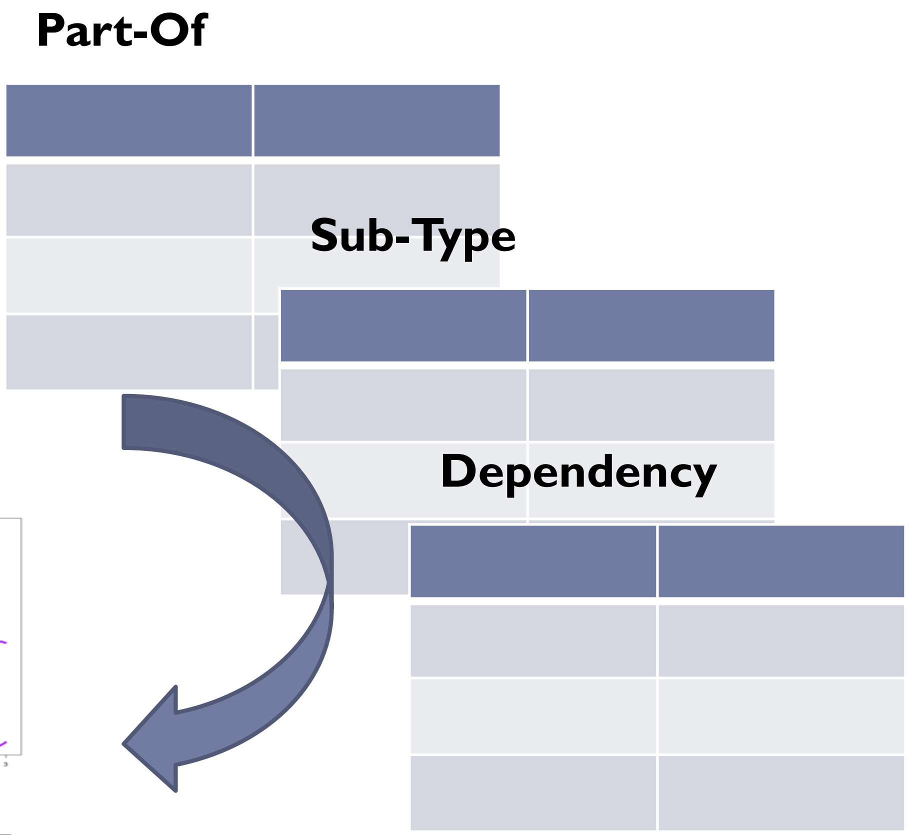
Training Data
Instance_1
Instance_2
Instance_3
….

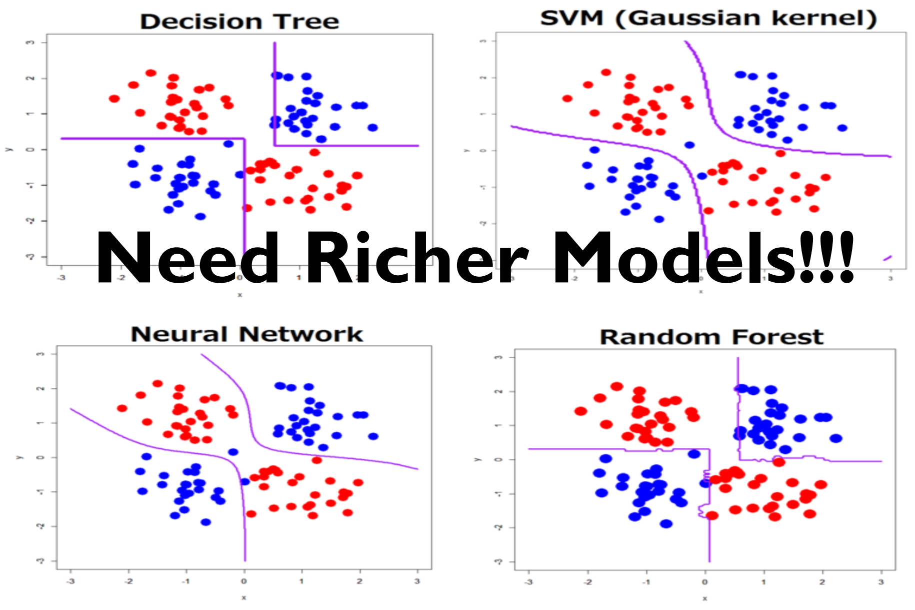
- Language has ambiguity
  - Microsoft Buys Apple
  - Microsoft acquires Apple
  - The merger of Microsoft and Apple

Need to learn robust models that handle noisy data

"New" Machine Learning

- Language data is <u>Relational</u>
- Language data is <u>Noisy</u>
- Language data is <u>Large</u>





Need Richer Models!!!

Statistical Relational Models

Logical/Relatio
nal

Probabilistic

Statistical Relational Models

- Long-standing goal of AI to unify logic & probabilities
- Several early attempts to unify the two (Leiniz 17th Century, Carnap '50, Gaifman '64)
- Over the last decade, the area of Statistical Relational Models has made significant progress towards this goal

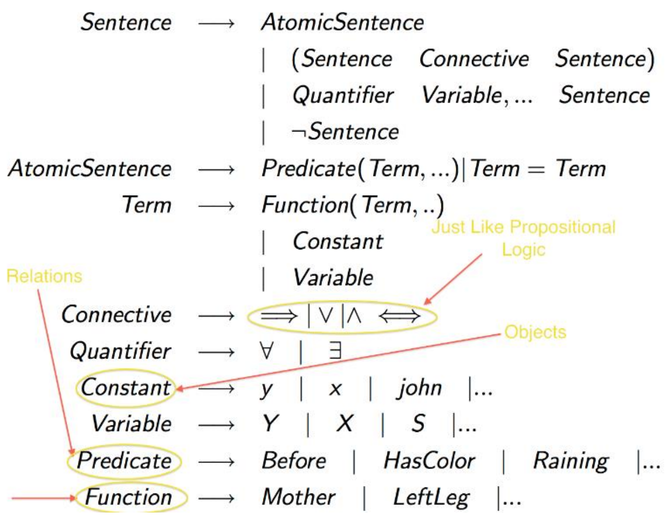
Statistical Relational Models

- Several popular models
  - Markov Logic (Pedro Domingos' group at UW)
  - BLOG (Stuart Russell's group at Berkeley)
  - PSL (Lise Getoor's group at UMD/UCSC)
  - ProbLog (Luc De Raedt's group at KU-Leuven)
- MLNs are arguably one of the most popular Statistical Relational Models
  - Simple interpretation
  - Powerful representation
- This tutorial is mainly focused on MLNs though similar ideas are potentially useful in other Statistical Relational Models as well

Outline

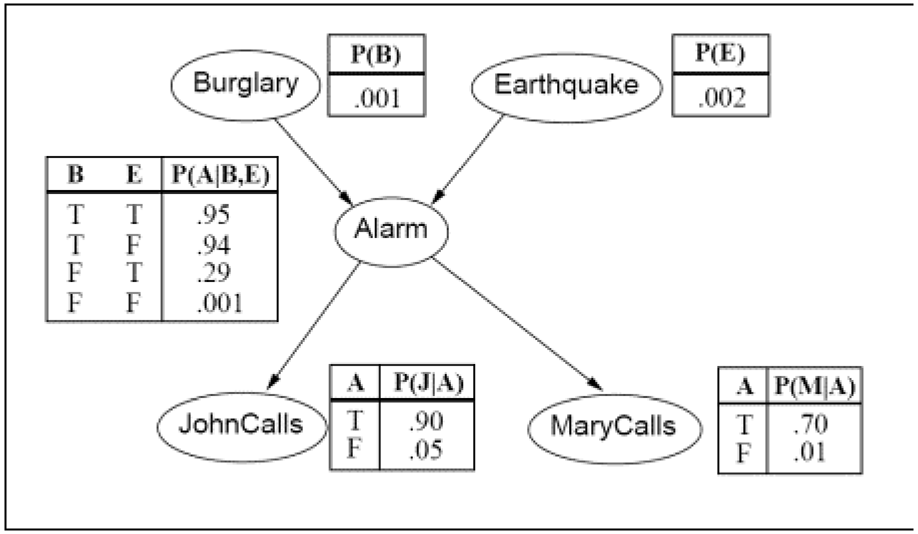
What representation can we learn from noisy, relational big data?

Unify the two (long standing dream of AI)



Just Like Propositional Logic
Relations
Objects

Sentence	$\longrightarrow$	*AtomicSentence*
		| (*Sentence Connective Sentence*)
		| *Quantifier Variable, ... Sentence*
		| $\neg$*Sentence*
AtomicSentence	$\longrightarrow$	*Predicate*(*Term, ...*)|*Term* = *Term*
Term	$\longrightarrow$	*Function*(*Term, ..*)
		| *Constant*
		| *Variable*
Connective	$\longrightarrow$	$\Rightarrow \mid \vee \mid \wedge \quad \Leftrightarrow$
Quantifier	$\longrightarrow$	$\forall \mid \exists$
Constant	$\longrightarrow$	*y* | *x* | *john* |...
Variable	$\longrightarrow$	*Y* | *X* | *S* |...
Predicate	$\longrightarrow$	*Before* | *HasColor* | *Raining* |...
Function	$\longrightarrow$	*Mother* | *LeftLeg* |...



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

First-Order Logic represents complex relational knowledge extremely compactly
E.g. Rules of chess can be written in one page in FOL while it takes 10^{38} pages of finite automata to represent it*!

Probabilistic Graphical Models (Markov Networks, Bayesian Networks) can represent large joint distributions compactly

[*Russell CACM '15]

MLN Representation

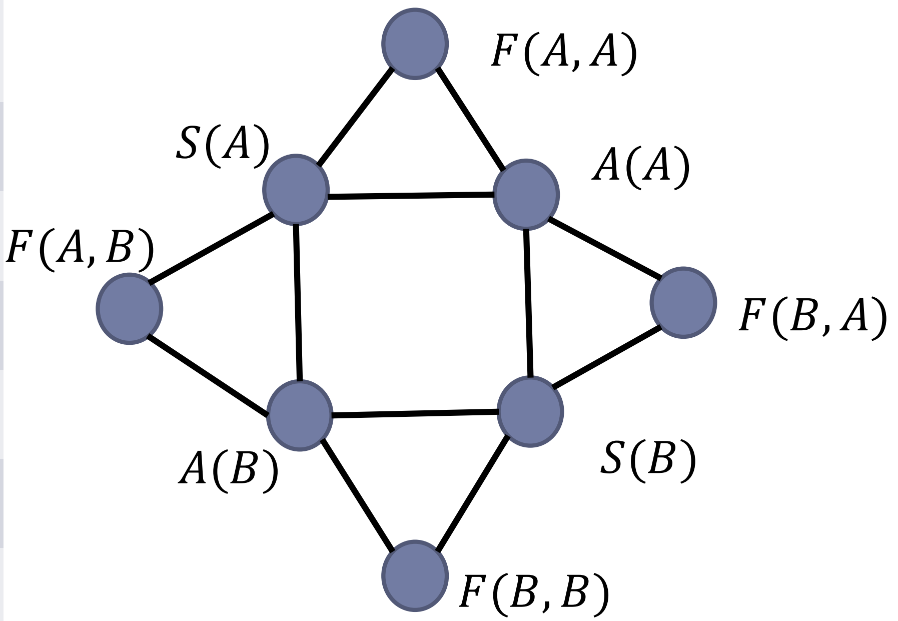
- Weighted first-order formulas
  - $\forall x\, \forall y\, \boldsymbol{Smokes}(\boldsymbol{x}) \wedge \boldsymbol{Friends}(\boldsymbol{x}, \boldsymbol{y}) \Rightarrow \neg \boldsymbol{Asthma}(\boldsymbol{y})$; **1.75**
- Larger the weight, more likely is the formula to be true
- For weight ∞, the formula specifies a hard constraint just as in first-order logic
- Variables in MLN formulas can be instantiated with constants representing real-world objects
- MLNs are a template for generating large Markov networks (undirected Probabilistic Graphical Model)
  - Represent a joint distribution over the possible worlds

Markov Logic Networks (MLNs)

$$\forall x\,\forall y\; \boldsymbol{Smokes}(x) \wedge \boldsymbol{Friends}(x,y) \Rightarrow \neg \boldsymbol{Asthma}(y) \quad 0.75$$

$$\boldsymbol{Smokes}(\boldsymbol{Ana}) \wedge \boldsymbol{Friends}(\boldsymbol{Ana},\boldsymbol{Bob}) \Rightarrow \neg \boldsymbol{Asthma}(\boldsymbol{Bob})\; \mathbf{0.75}$$
$$\boldsymbol{Smokes}(\boldsymbol{Bob}) \wedge \boldsymbol{Friends}(\boldsymbol{Bob},\boldsymbol{Ana}) \Rightarrow \neg \boldsymbol{Asthma}(\boldsymbol{Ana})\; \mathbf{0.75}$$
$$\boldsymbol{Smokes}(\boldsymbol{Bob}) \wedge \boldsymbol{Friends}(\boldsymbol{Bob},\boldsymbol{Bob}) \Rightarrow \neg \boldsymbol{Asthma}(\boldsymbol{Bob})\; \mathbf{0.75}$$
$$\boldsymbol{Smokes}(\boldsymbol{Ana}) \wedge \boldsymbol{Friends}(\boldsymbol{Ana},\boldsymbol{Ana}) \Rightarrow \neg \boldsymbol{Asthma}(\boldsymbol{Ana})\; \mathbf{0.75}$$

Smokes (Ana)	**Friends (Ana,Bob)**	**Asthma (Bob)**	ϕ
0	0	0	$e^{0.75}$
0	0	1	$e^{0.75}$
0	1	0	$e^{0.75}$
…	…	…	…
1	1	0	e^{0}
1	1	1	$e^{0.75}$



Background

	S(A)	S(B)	M(A)	M(B)	F(A,B)	F(B,A)	F(A,A)	F(B,A)
ω_1	0	0	0	0	0	0	0	0
ω_2	0	0	0	0	0	0	0	1
ω_3	0	0	0	0	0	0	1	0
…	…	…	…	…	…	…	…	

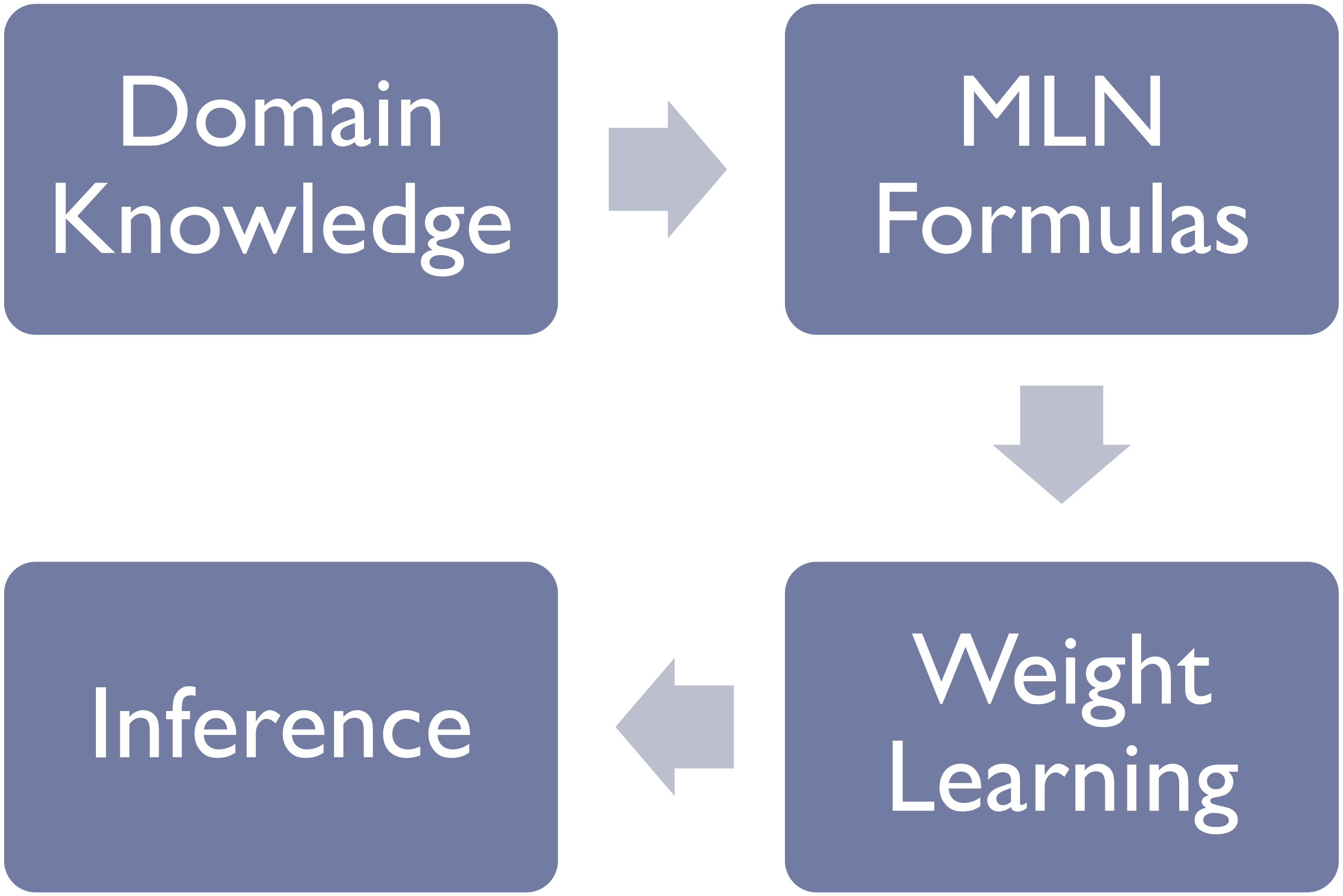
$$P(\omega) = \frac{1}{Z} exp\left(\sum\nolimits_f w_f \#SATGround(f,\omega)\right)$$

$$Z = \sum\nolimits_\omega exp\left(\sum\nolimits_f w_f \#SATGround(f,\omega)\right)$$

MLNs

- Generalize several representations
  - Probabilistic Graphical Models
  - Log-Linear Models
  - Hidden Markov Models
  - Boltzmann Machines
  - Conditional Random Fields
  - Man Entropy Models
  - Logistic Regression
  - Gibbs Distributions

Typical MLN Application Design Process



Inference Problems

- Marginal Inference
  - $P(Smokes(Ana)|Friends(Ana, Bob), Smokes(B) \dots) =?$
- MAP Inference
  - $\max_x P(x|y)$, where y is the evidence and x is the set of remaining variables in the MLN.
- Both the above problems are computationally hard
  - Marginal Inference #P
  - MAP Inference (NP-hard)

Weight Learning

- Can be performed generatively or discriminatively
  - The training data is a relational database
  - Unlike traditional learning algorithms, in typical MLN learning there is just one instance to learn from (the relational DB)
- Max-likelihood weight learning uses inference during each step
  - Typically approximate inference is utilized within weight learning

Joint Inference using MLNs

- Several NLP applications are amenable to joint inference
  - Information extraction, coreference resolution, etc.
  - Joint inference reduces error propagation that is commonly seen in pipeline architectures

- MLNs are a particularly powerful representation for performing joint inference in NLP applications
  - Can specify dependencies between different stages of the pipeline
  - E.g., An extracted trigger and its associated arguments are dependent
  - Using MLNs, we can model dependencies naturally using first-order formulas

Joint Inference using MLNs

- Integer Linear Programming (ILP) has been widely used for joint inference in NLP*
  - Specify joint dependencies as linear constraints
- Powerful solvers can be easily leveraged off-the-shelf for solving the ILP
  - Gurobi
  - IBM Cplex
  - lp_solve
  - SAT4J
  - MiniSAT+
  - ….

*Roth CoNLL '04

Joint Inference using MLNs

- The lifted representation of MLNs makes it possible to specify complex joint constraints compactly
  - In contrast joint inference based on ILPs use a propositional formulation

How can we specify the constraint that event coreferences are transitive?

ILP

$e_1 \wedge e_2 \Rightarrow e_3$
$e_2 \wedge e_3 \Rightarrow e_4$
$e_1 \wedge e_3 \Rightarrow e_2$
$e_2 \wedge e_4 \Rightarrow e_1$
…

$(1 - e_1) + (1 - e_2) + e_3 \geq 1$
$(1 - e_2) + (1 - e_3) + e_4 \geq 1$
…

MLNs

$\forall x, y, z\ EventCoref(x, y) \wedge EventCoref(y, z) \Rightarrow EventCoref(z, x)$

Joint Inference using MLNs

- Weights on MLN formulas make it easier to represent uncertainty of joint constraints

How can we specify the soft constraint that event coreferences are transitive?

ILP

$$(e_1 \wedge e_2 \Rightarrow e_3) \Leftrightarrow v_1$$
$$(e_2 \wedge e_3 \Rightarrow e_4) \Leftrightarrow v_2$$
$$(e_1 \wedge e_3 \Rightarrow e_2) \Leftrightarrow v_3$$
$$(e_2 \wedge e_4 \Rightarrow e_1) \Leftrightarrow v_4$$

…

$$(1 - v_1) + (1 - e_1) + (1 - e_2) + e_3 \geq 1$$
$$e_1 + v_1 \geq 1$$
$$e_2 + v_1 \geq 1$$
$$(1 - e_3) + v_1 \geq 1$$

…

add $\sum_i w\, v_i$ to the objective function

MLNs

$$\forall x, y, z\ EventCoref(x, y) \wedge EventCoref(y, z) \Rightarrow EventCoref(z, x); w$$

Joint Inference using MLNs

ILP

$$(e_1 \land e_2 \Rightarrow e_3) \Leftrightarrow v_1$$
$$(e_2 \land e_3 \Rightarrow e_4) \Leftrightarrow v_2$$
$$(e_1 \land e_3 \Rightarrow e_2) \Leftrightarrow v_3$$
$$(e_2 \land e_4 \Rightarrow e_1) \Leftrightarrow v_4$$

…

add $\sum_i w\, v_i$ to the objective function

Notice that all soft constraints are symmetrical. They have the same structure and same weight.

We can exploit these symmetries for computational efficiency during inference and learning in MLNs.

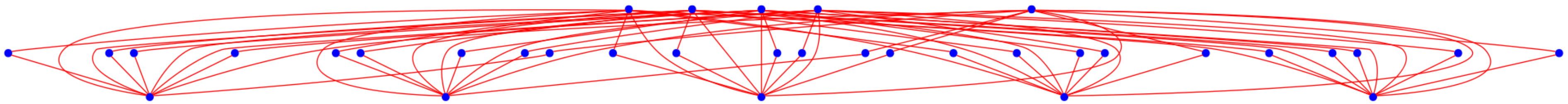
MLNs

$$\forall x, y, z\ EventCoref(x, y) \land EventCoref(y, z) \Rightarrow EventCoref(z, x); w$$

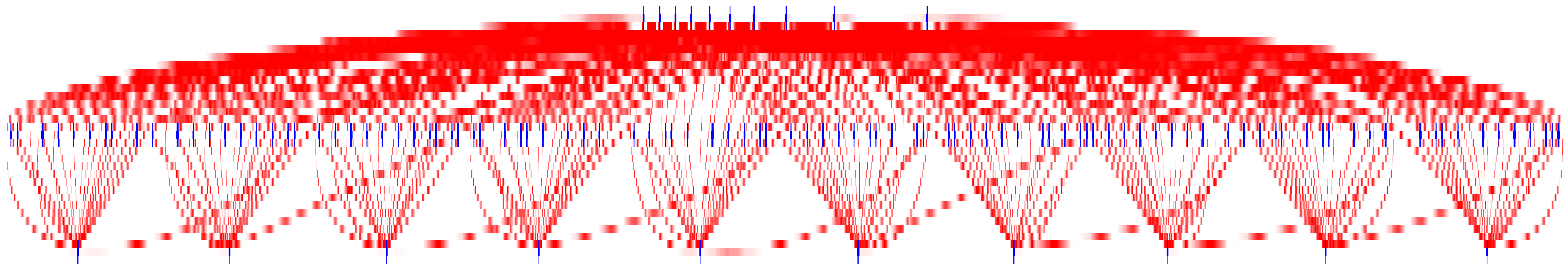
What are the challenges with MLNs?

- Scalability is the primary challenge in utilizing MLNs for large application domains such as NLP
  - Inference and Learning are hard problems in MLNs
  - Need tools for MLNs that can work as a "black-box" for applications
- Can't we just convert MLNs into a Markov Network and apply techniques there?
  - Unfortunately, MLNs typically encode Markov Networks that are orders of magnitude larger than what (propositional) PGM inference algorithms can handle

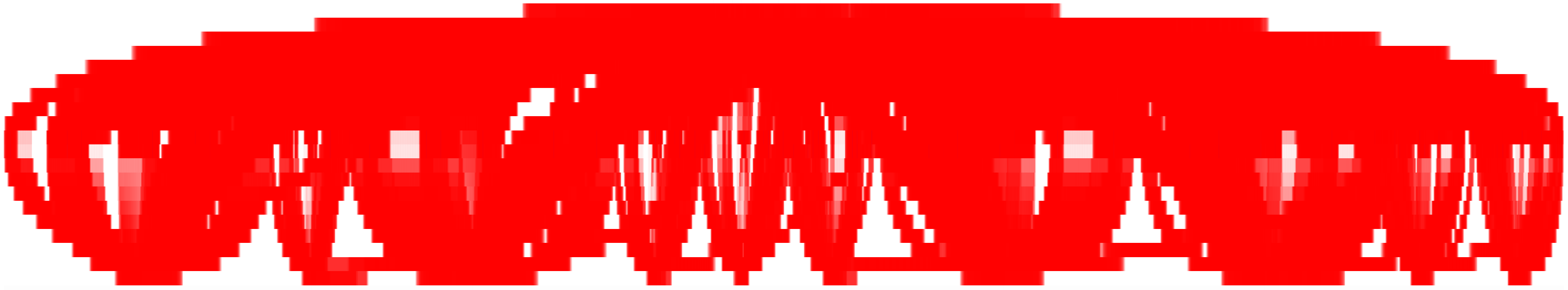
$Smokes(x) \wedge Friends(x,y) \Rightarrow \neg Asthma(y)$



$\#Objects = 5$



$\#Objects = 10$



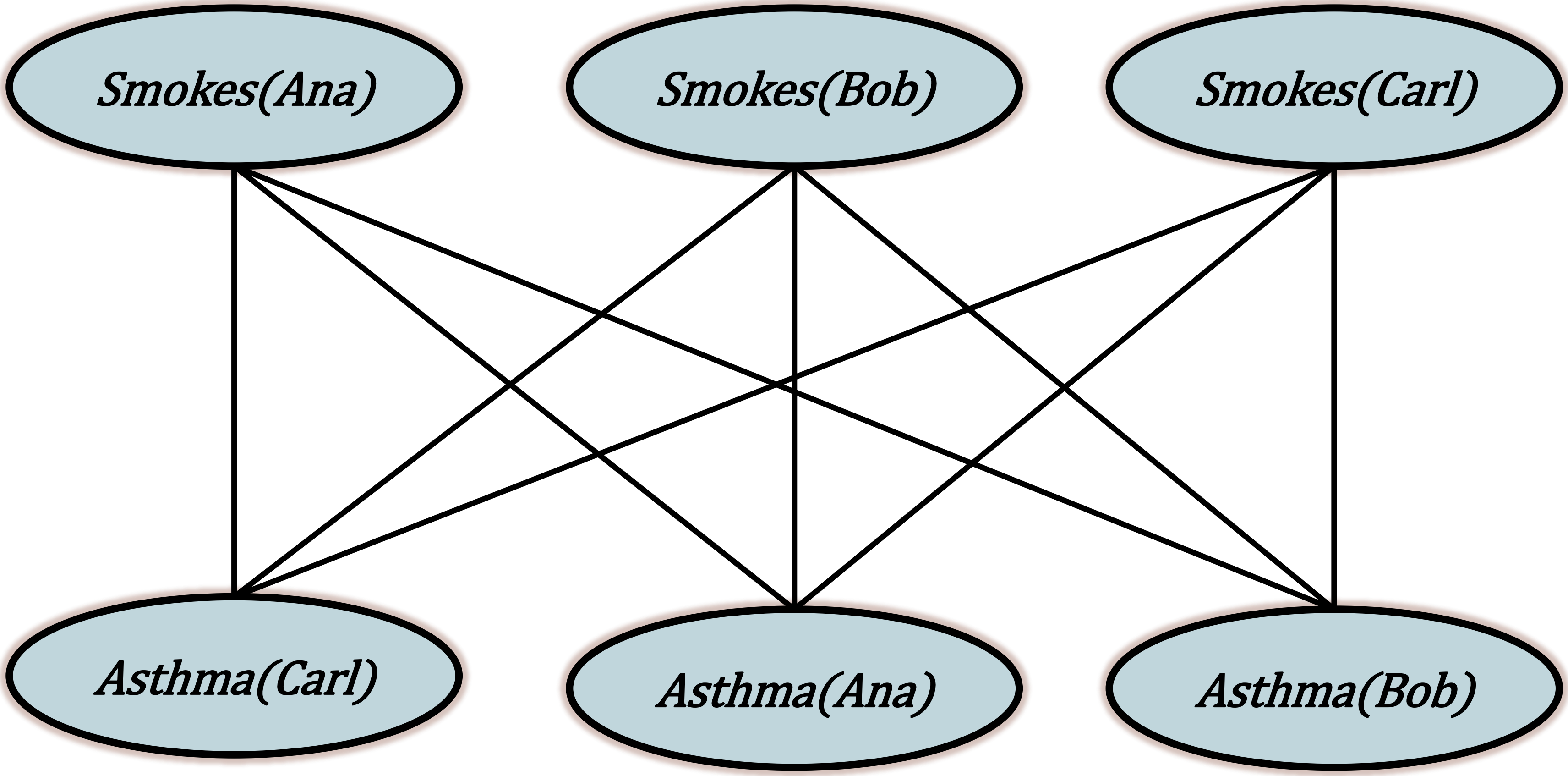
$\#Objects = 50$

Outline

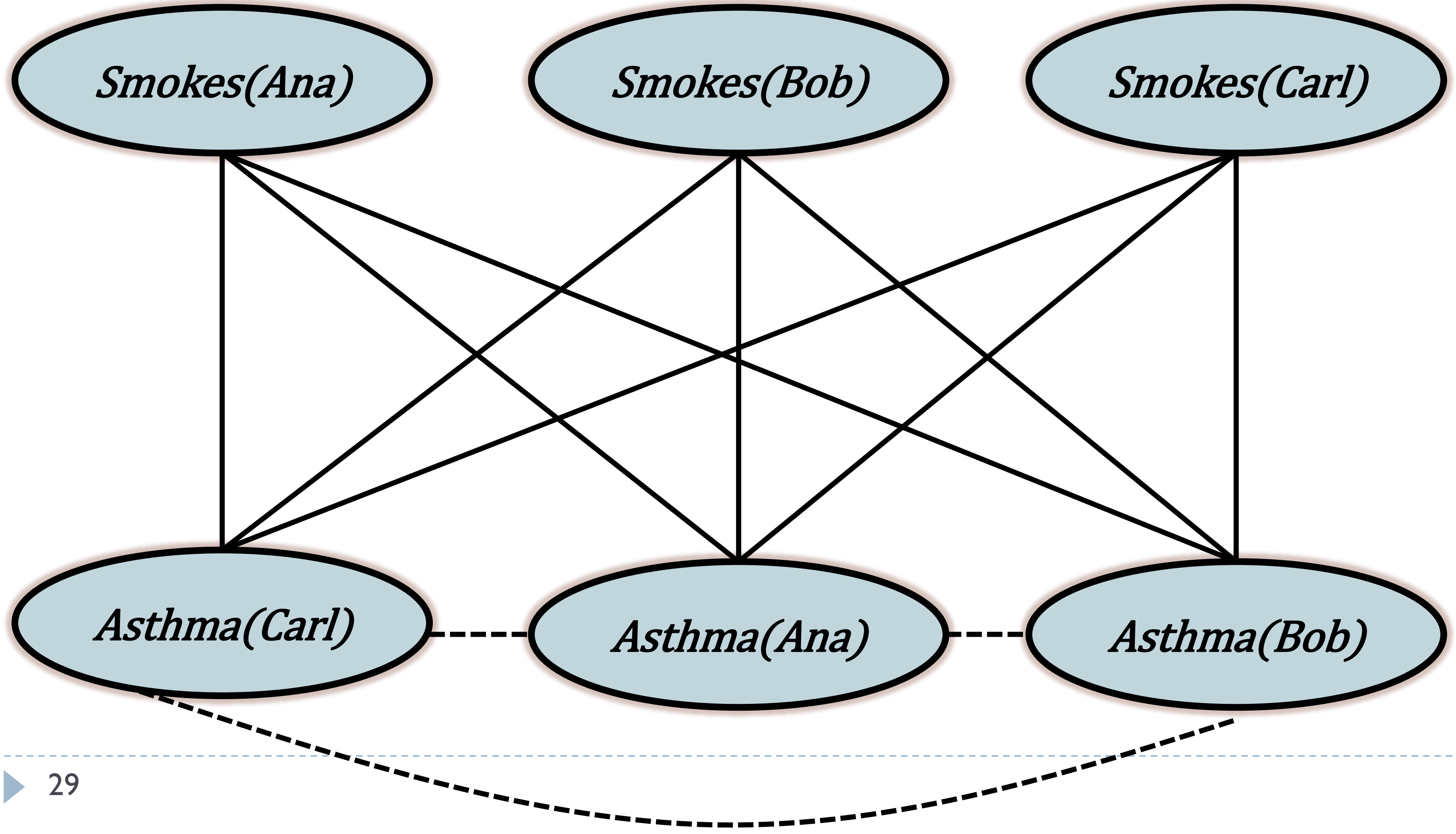
- Motivation
- MLN Basics
- Domain-Lifted Inference
- Approximate Domain-Lifting
- Lifted Generative Weight Learning
- Scalable Weight Learning with Approximate Counting Oracles
- Applications
- Conclusion

Inference

$$\forall x\ \forall y\ Smokes(x) \Rightarrow Asthma(y)\ w$$

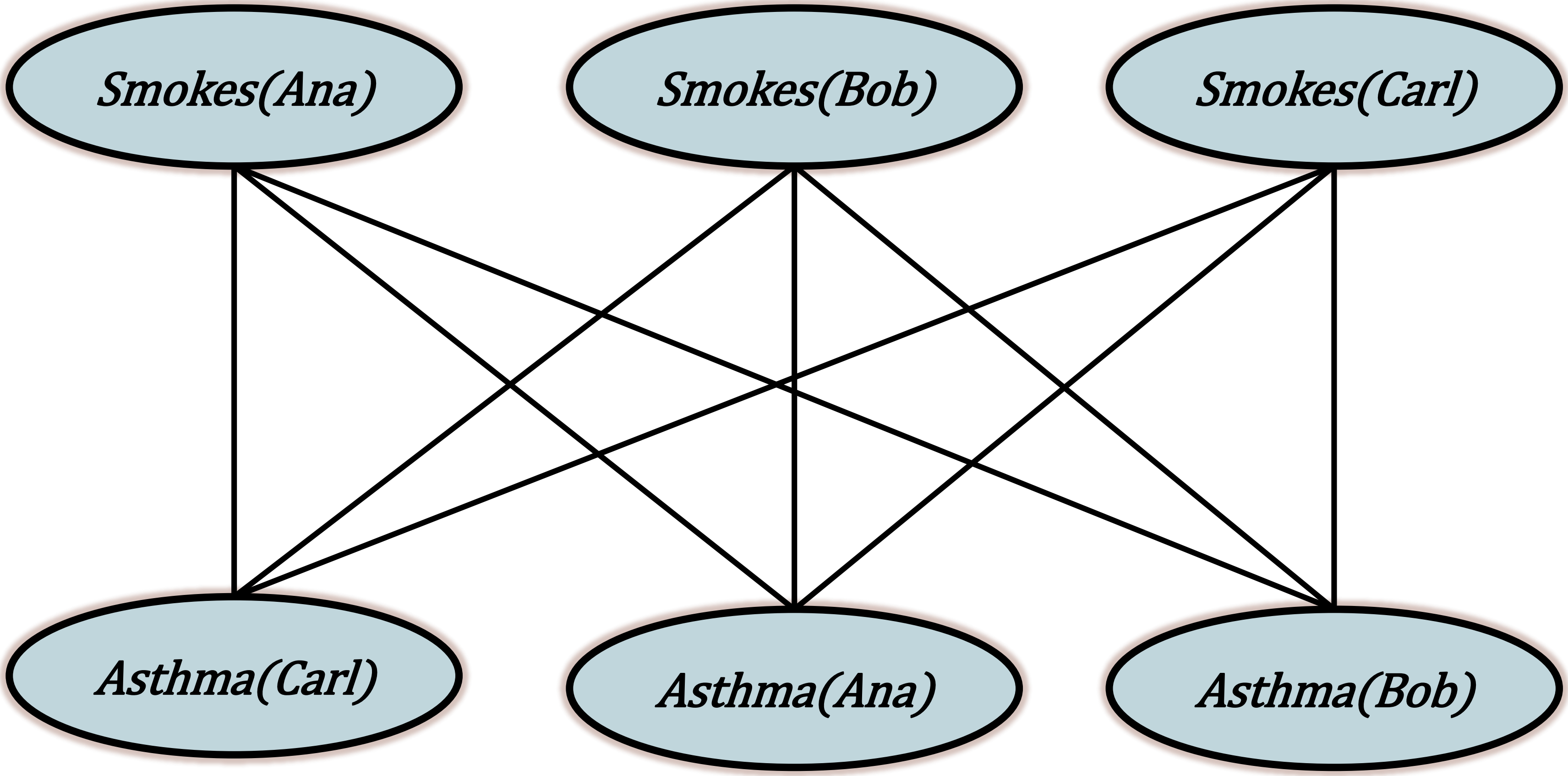


Variable Elimination

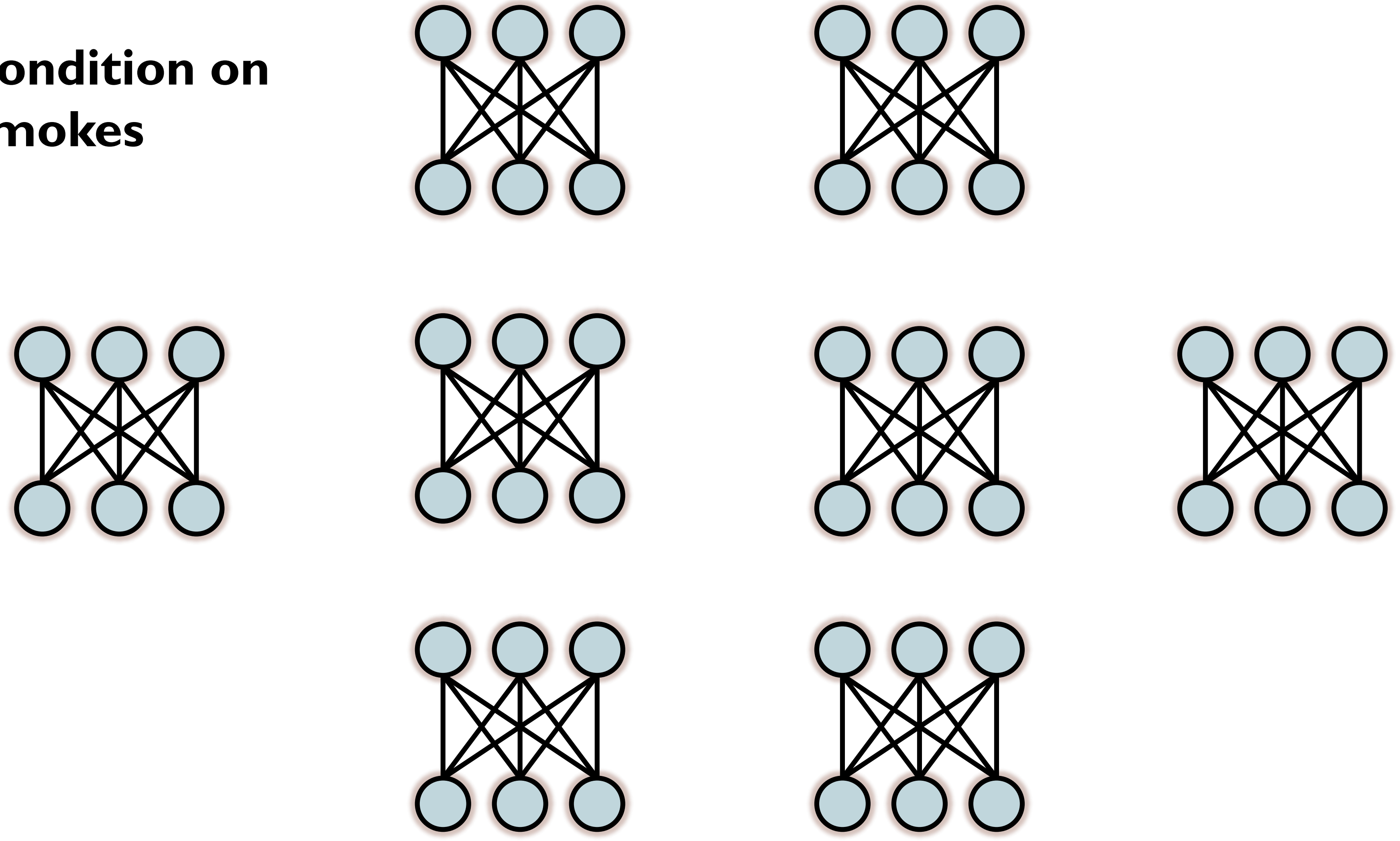


Inference based on conditioning

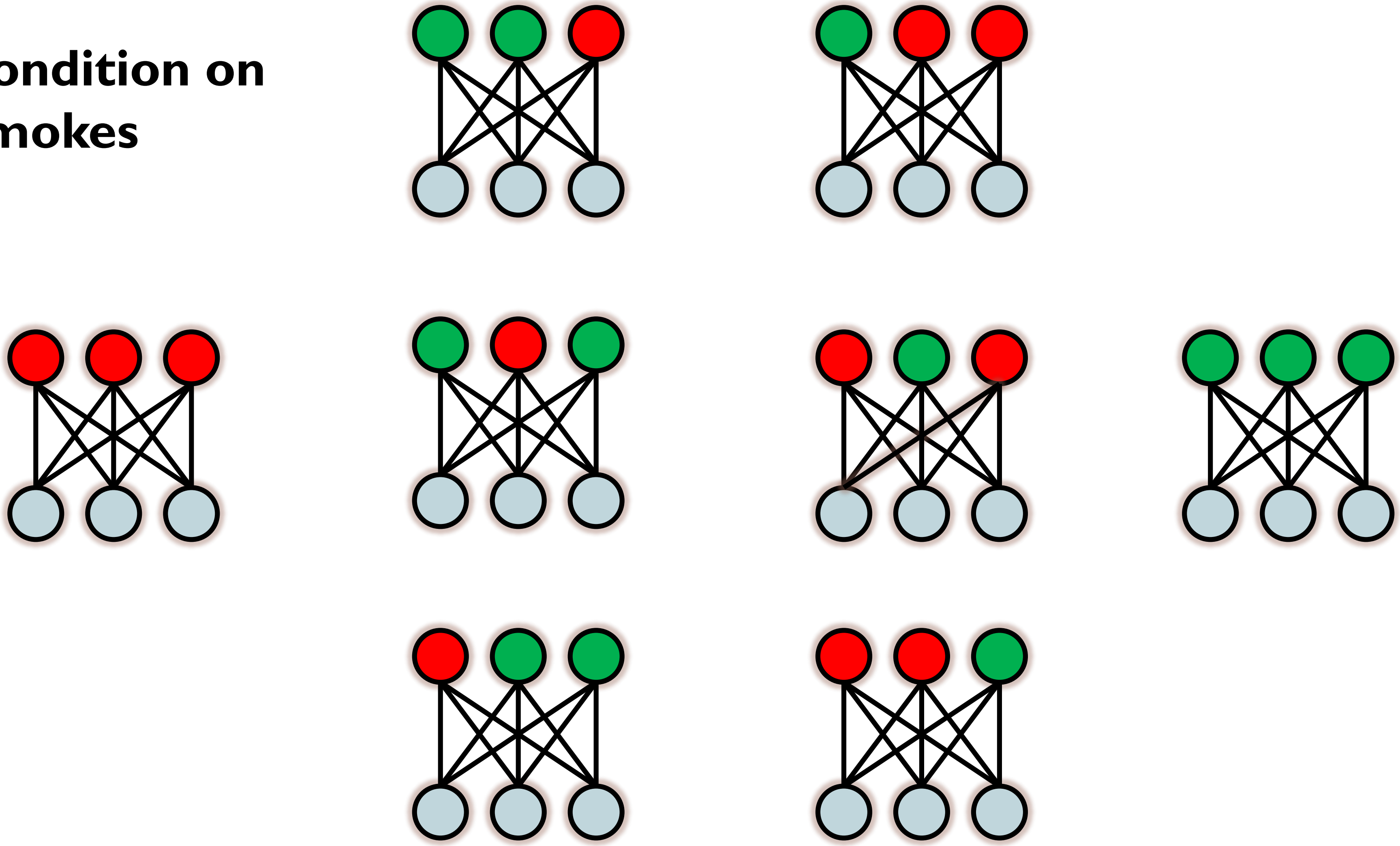
$$\forall x\ \forall y\ Smokes(x) \Rightarrow Asthma(y)\ w$$



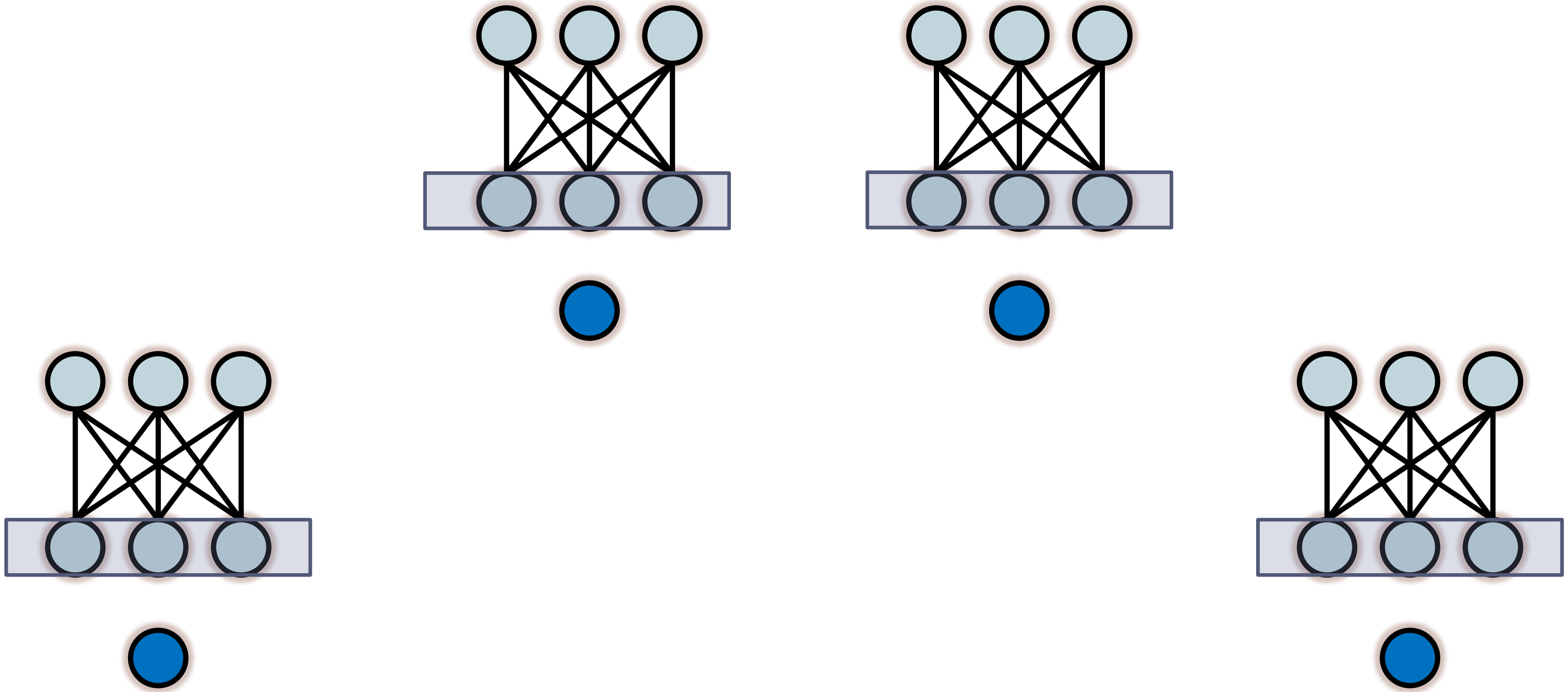
Condition on Smokes



Condition on Smokes



$$O(2^n) \Rightarrow O(n+1)$$



Within each group, conditioning on any of the $Asthma$ atoms yields identical results. Therefore, we can group the three $Asthma$ atoms and condition on a single atom from this group.

Lifted Inference

- Sometime possible to perform tractable inference on large treewidth Markov Networks
  - Need to exploit symmetries within the Markov Network
- Do we need to explicitly build the Markov Network to determine what the symmetries are?
  - In some cases, can identify symmetries directly from first-order structure <u>without</u> grounding the MLN
- Analogous to first-order theorem proving, in lifted inference we work at the compact first-order level

Domain-Lifted Inference

- Informally,
  - Exploit symmetries, reason about groups of objects, reason at first-order level, etc (Poole '03)
- More formally,
  - Inference runs in time polynomial in the number of domain objects (Broeck '11, Jaeger '12)
- Exact lifted inference
  - Lifted Factor Graphs (Poole '03)
  - First-Order Variable Elimination (Braz et al. '07)
  - Weighted First-Order Model Counting (Broeck et al. '10)
  - Probabilistic Theorem Proving (Gogate and Domingos '11)

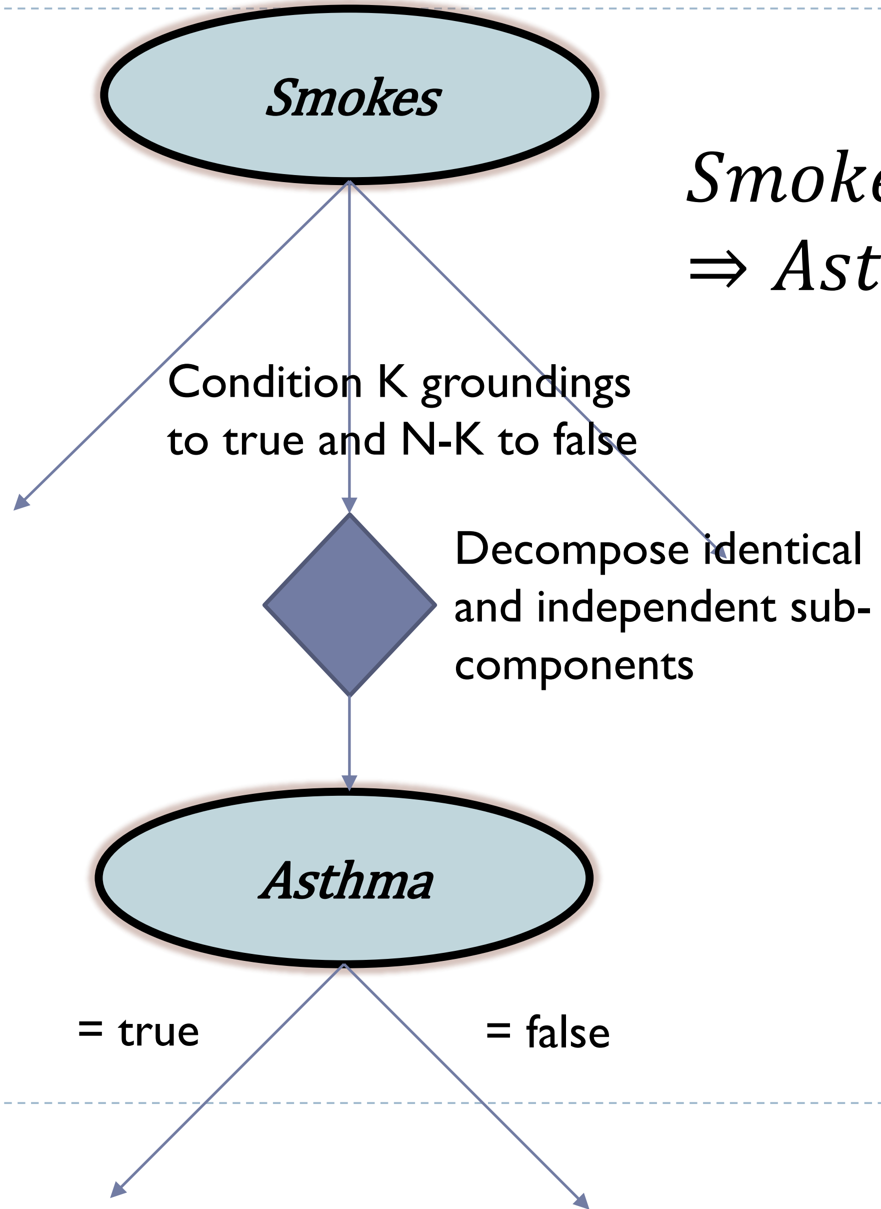
Probabilistic Theorem Proving (PTP)

- DPLL-style algorithm exact inference algorithm that conditions over first-order atoms
- Conditioning rule: Conditions a singleton/unary atom in polynomial time
- Decomposer rule: Reduces the MLN by merging identical and independent sub-components
  - $Stong(x) \Rightarrow Wins(x); w$ contains $|\Delta x|$ independent and identical components in the Markov network
- PTP applies the above two rules recursively until no more applications of the rule is possible. If the remaining MLN is intractable, then the MLN is not liftable through PTP.

PTP

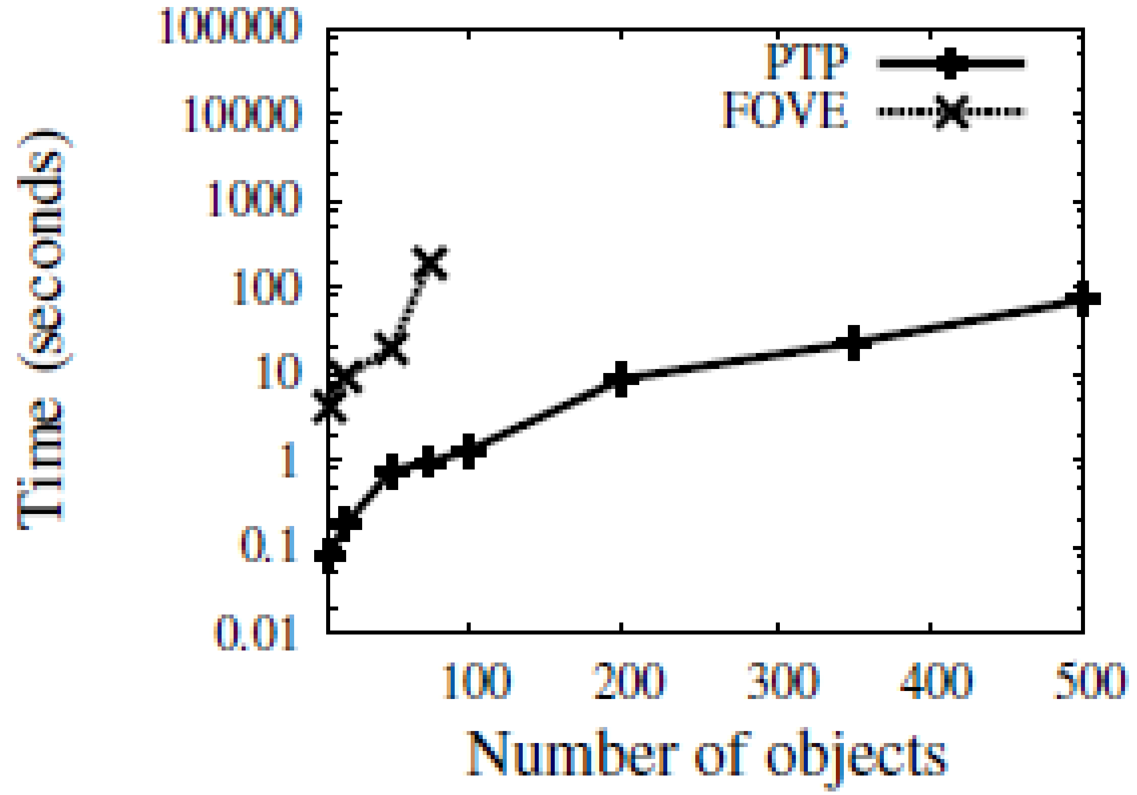
$Smokes(x)$
$\Rightarrow Asthma(y)\ w$

$O(N)$ algorithm for a Markov Network of treewidth N


Smokes
Condition K groundings to true and N-K to false
Decompose identical and independent sub-components
Asthma
= true
= false

Performance: Link Prediction

$$GoodProf(x) \wedge GoodStudent(y) \wedge Advises(x,y) \Rightarrow FutureProf(y)$$
$$CoAuthor(x,y) \Rightarrow Advises(x,y)$$



Approximate Inference

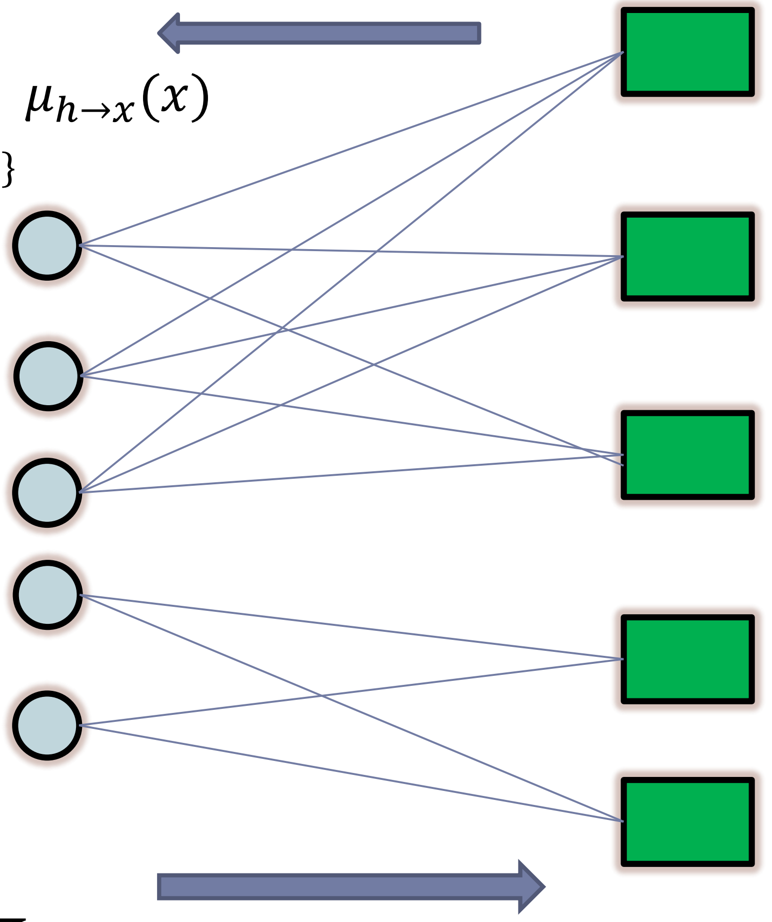
- Similar to inference in Markov Networks, exact inference is infeasible in practical cases. Most practical cases use approximate inference
- Domain-lifted versions of several popular approximate inference algorithms have been developed over past few years
- Marginal Inference
  - Lifted Belief Propagation (Singla and Domingos '07, Kersting et al. '08)
  - Lifted MCMC (Niepert '12, Venugopal and Gogate '12)
  - Lifted Importance Sampling (Gogate et al. '12)
- MAP Inference
  - Lifted MAP (Sarkhel et al. '14, '15, Kersting et al. '14, '15)

Lifted Belief Propagation

- Exploits the fact that symmetries in MLNs cause similar messages to be sent and received in BP
- Creates a lifted network for BP that can sometimes be exponentially smaller than the ground network for BP
  - Groups atoms that send and receive same messages into supernodes
  - Groups ground formulas that send and receive the same messages into superfeatures

Loopy Belief Propagation

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

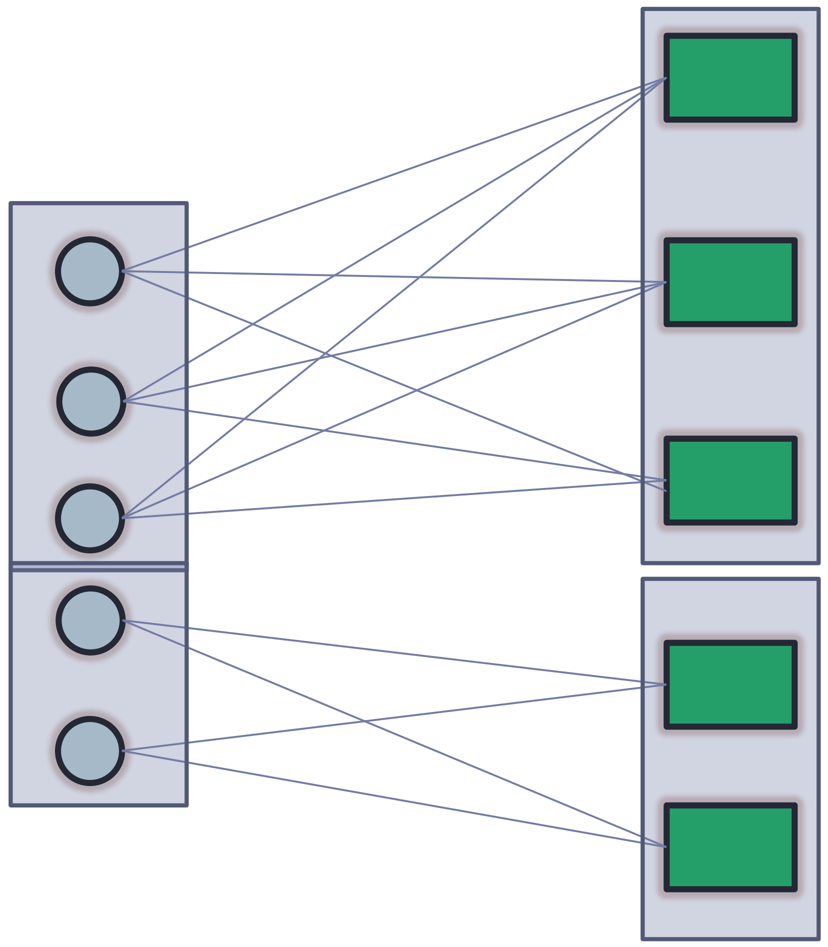


Nodes (ground atoms)

Features (ground formulas)

$$\mu_{f(x)\to x} = \sum_{\sim x} f(x) \prod_{y\in nb(f)\{x\}} \mu_{y\to f}(y)$$

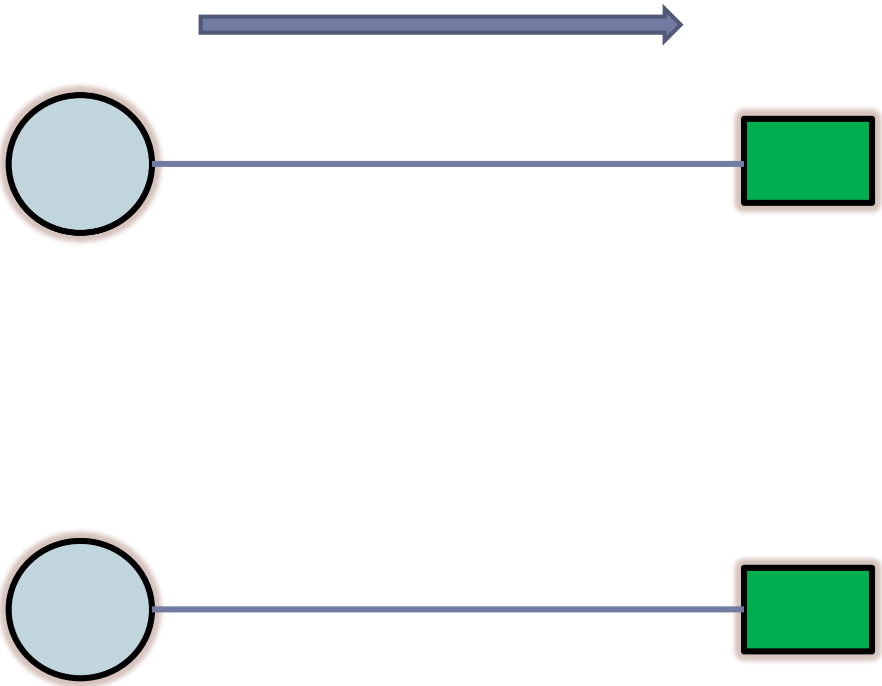
Lifted Belief Propagation



Lifted Belief Propagation

$$\mu_{x\to f}(x) = \mu_{f\to x}^{n(f,x)-1} \prod_{h\in nb(x)\setminus\{f\}} \mu_{h\to x}(x)^{n(h,x)},$$

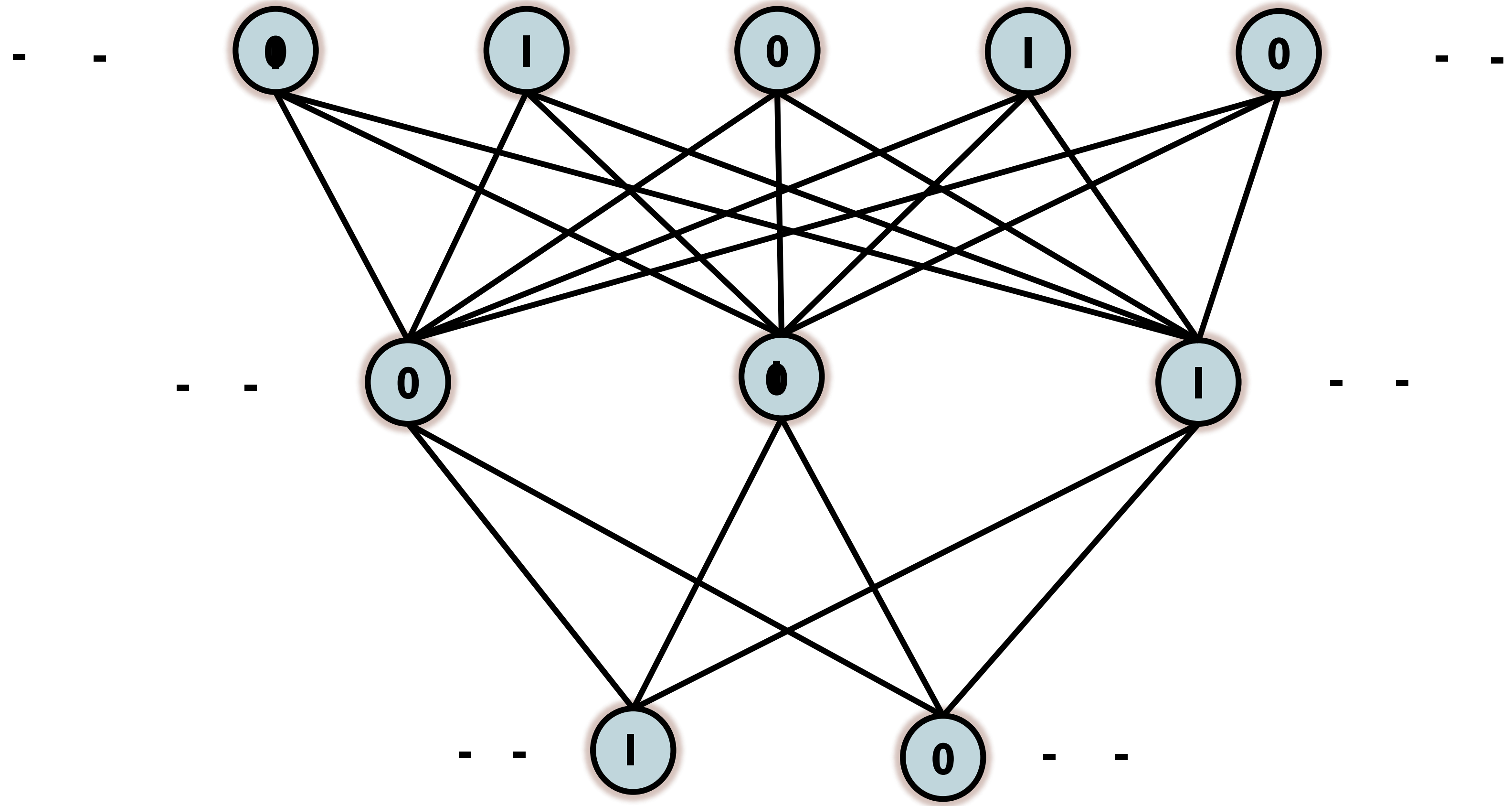
where $n(f,x)$ is the number of identical messages that would be sent from f to x in the ground network



Gibbs Sampling

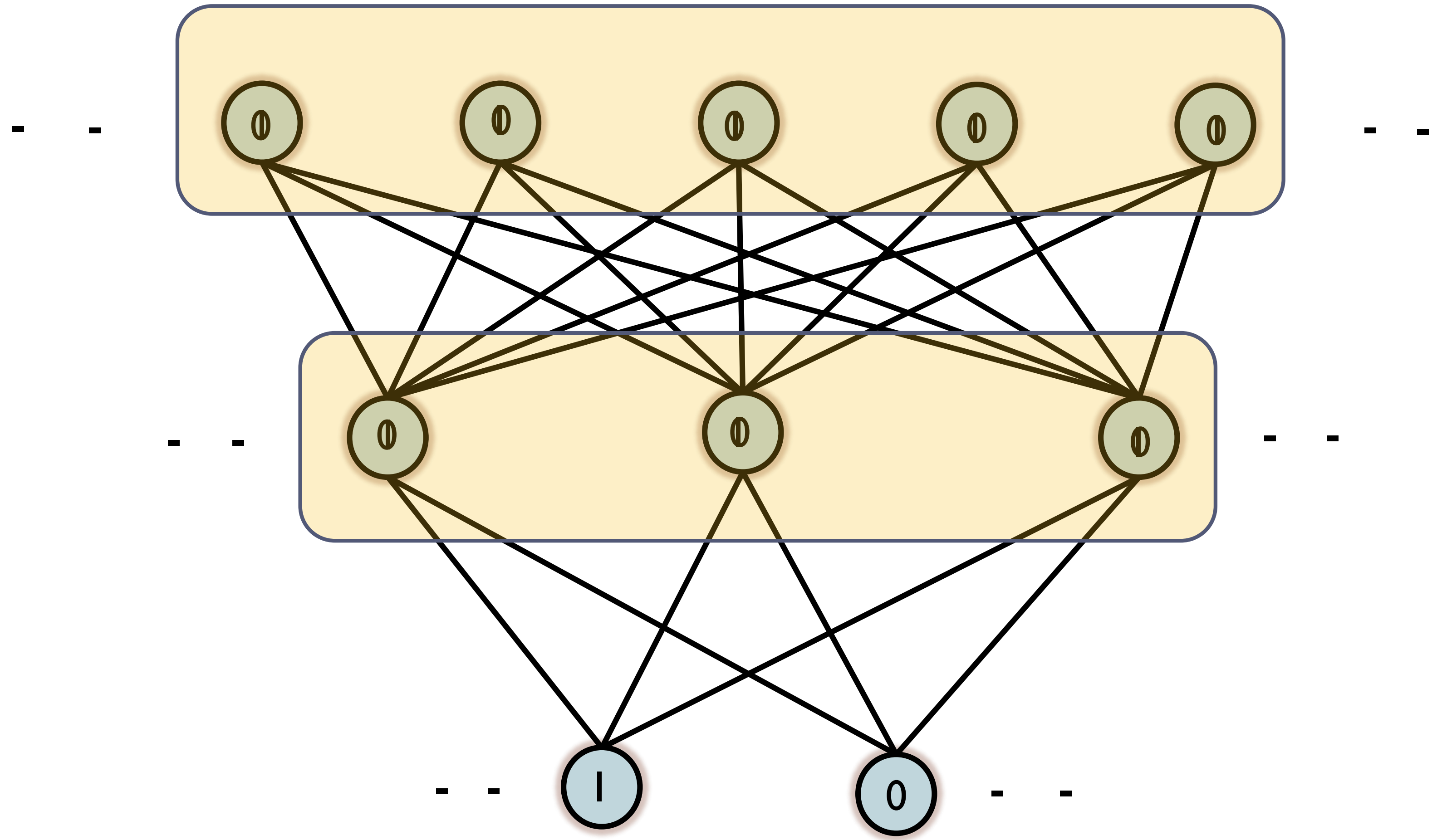
$Word(w, page) \Rightarrow Topic(page, t)\ w_1$

$Topic(page, t) \Rightarrow Class(t, c)\ w_2$



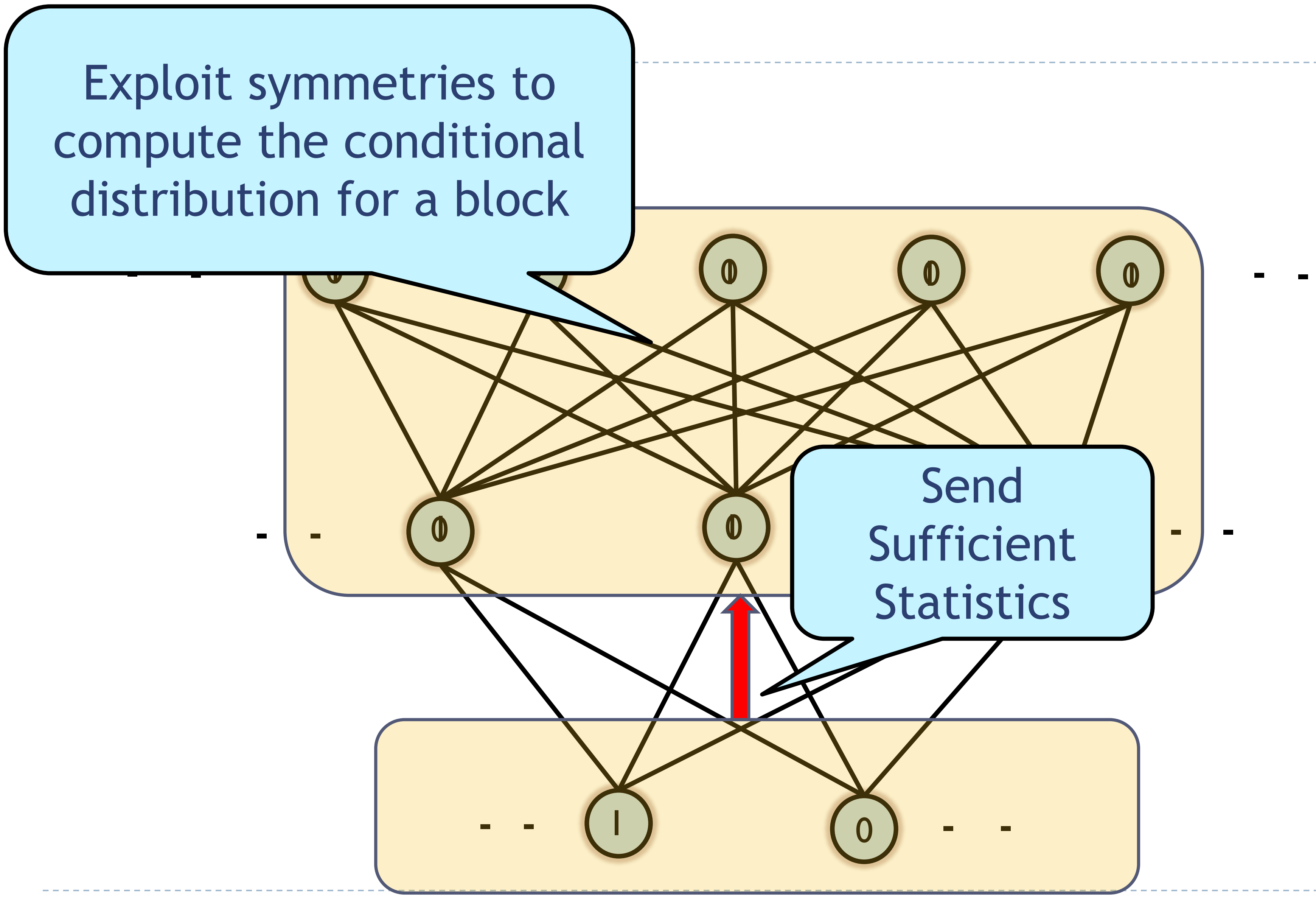
Geman and Geman '84

Blocked Gibbs Sampling (LBG)

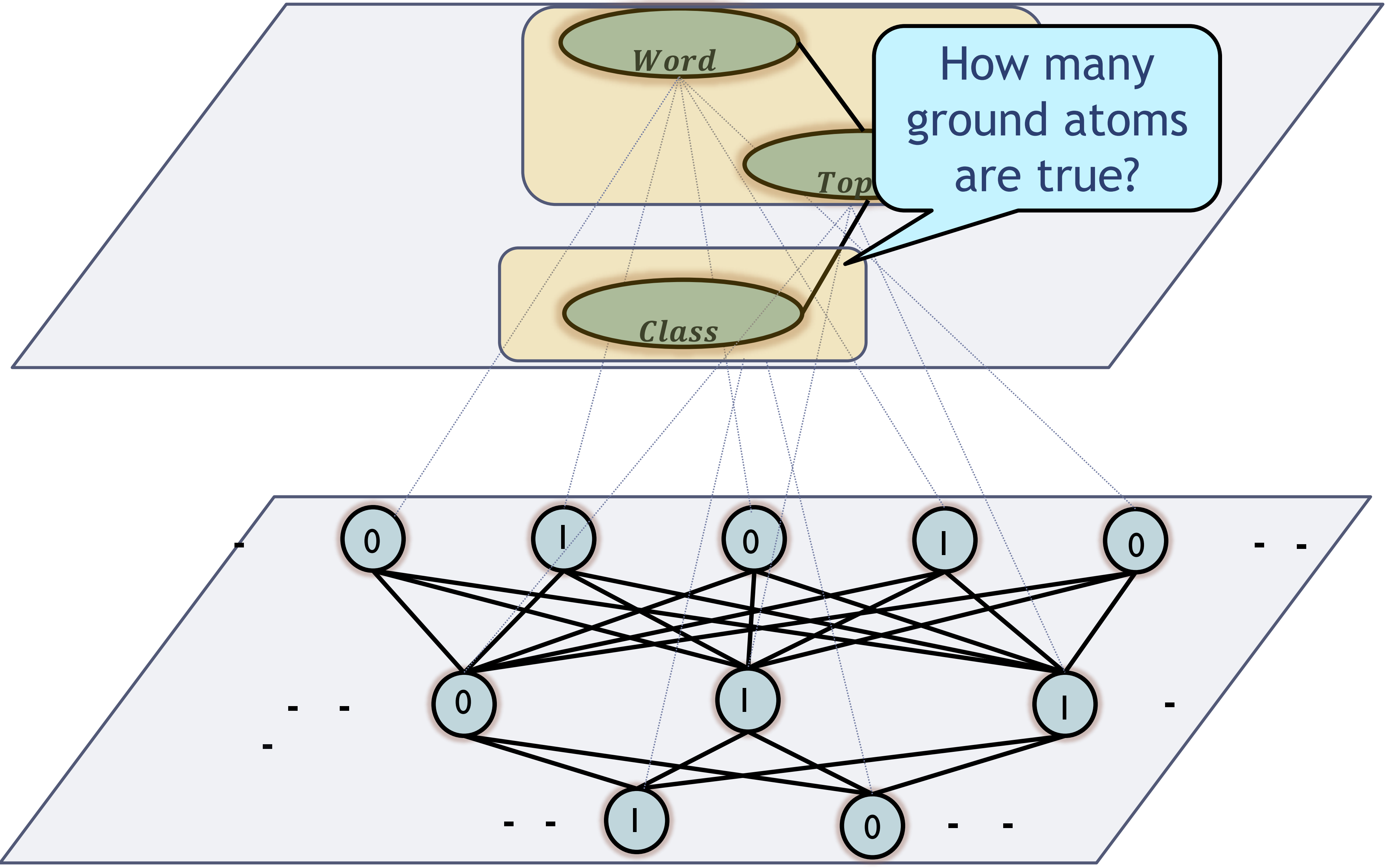


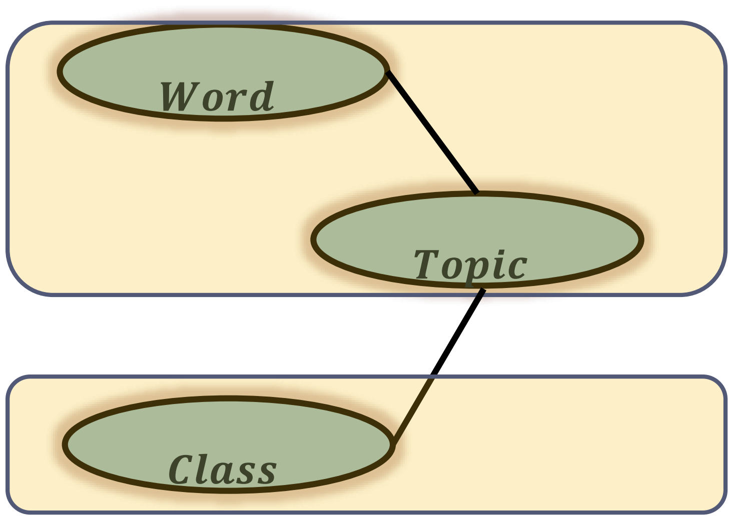
Jensen et al. ’93

Lifted Blocked Gibbs Sampling (LBG)

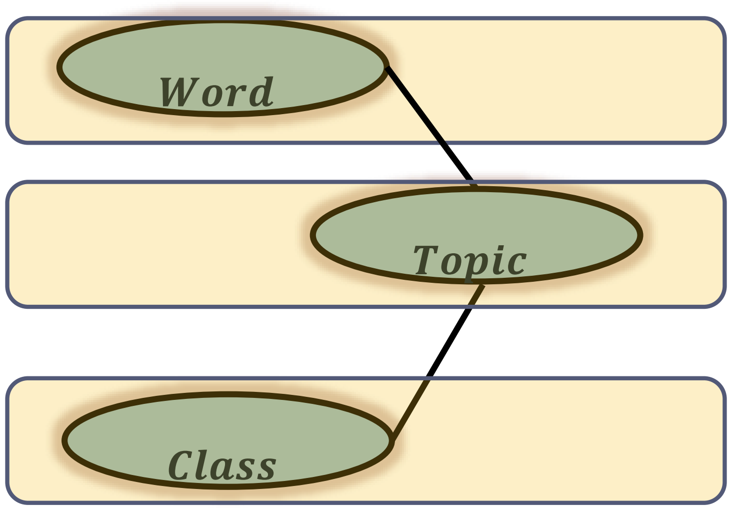

Exploit symmetries to compute the conditional distribution for a block
Send Sufficient Statistics
0
0
0
0
0
1
0

Venugopal and Gogate '12


Word
Top
Class
How many ground atoms are true?
0
1
0
1
0
0
1
1
1
0



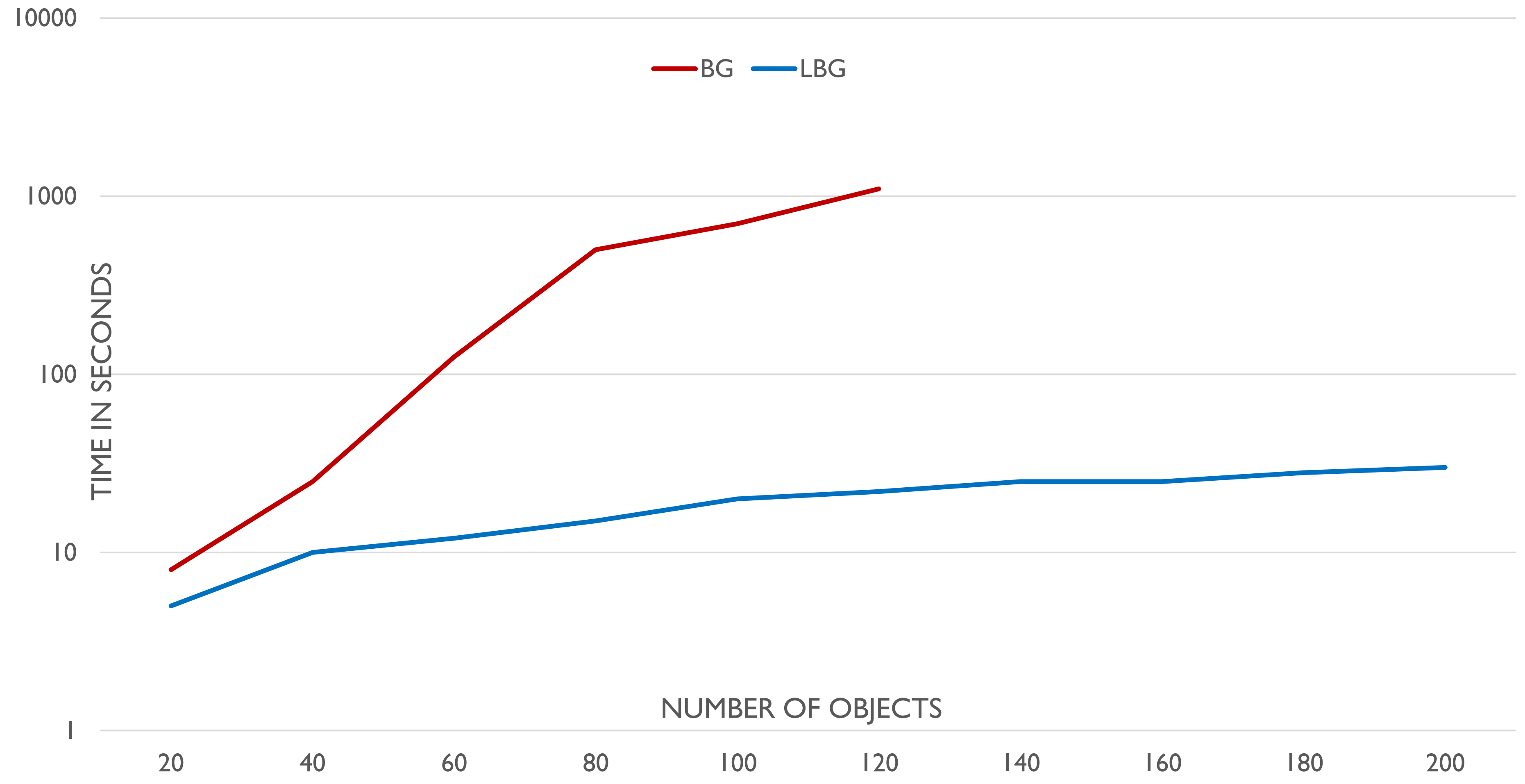
$O(d^2)$



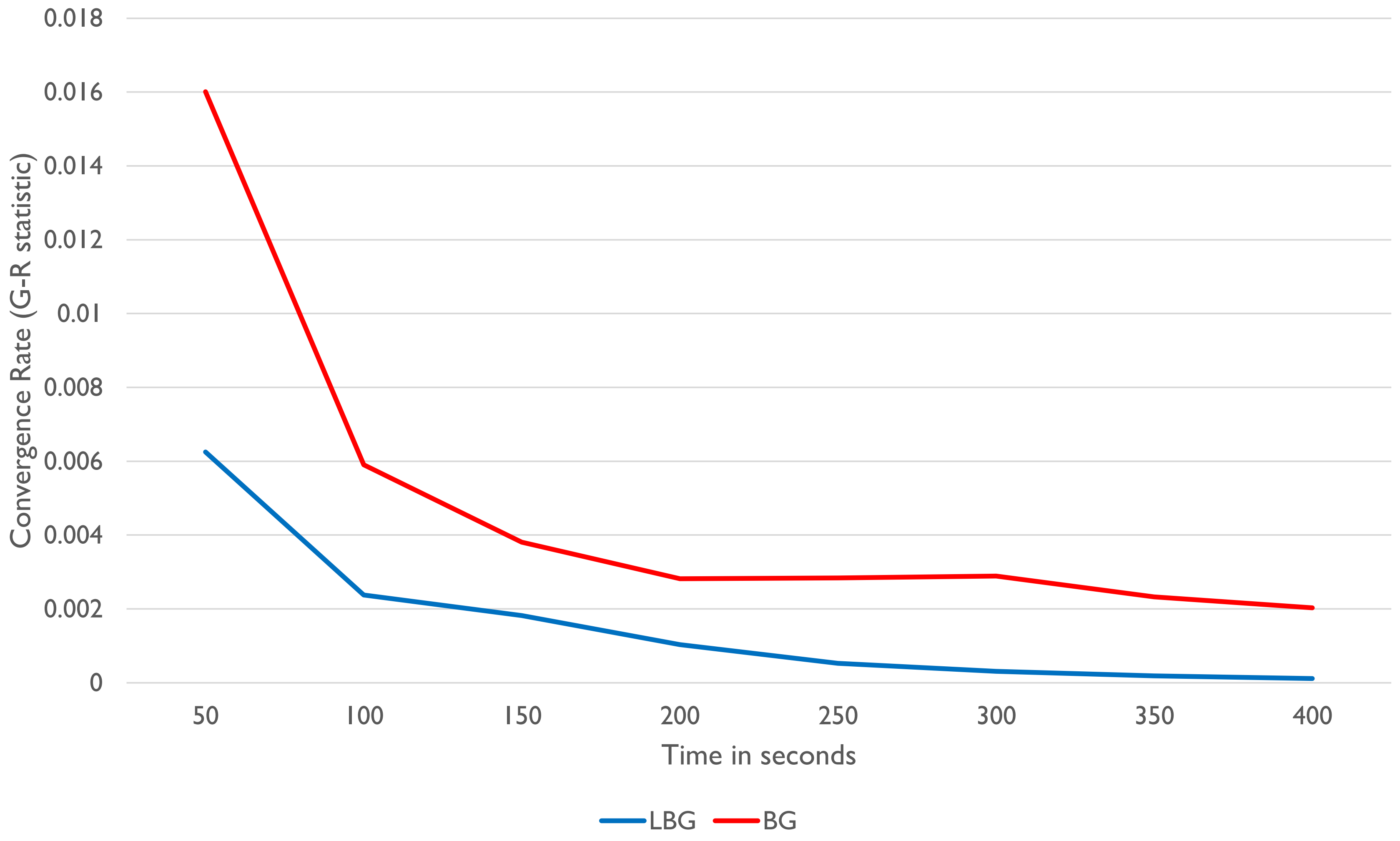
$O(d^3)$

More blocking can yield smaller complexities since more symmetries are retained!!

Performance: Topics MLN



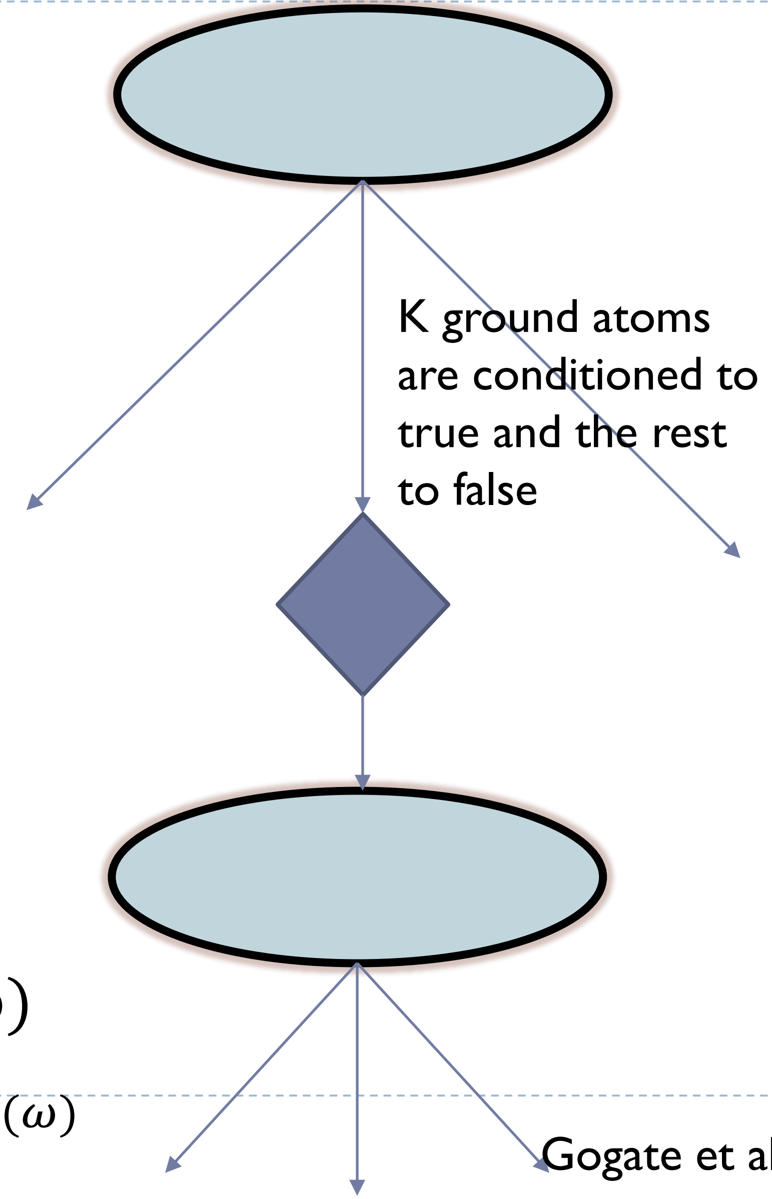
Performance: WebKB



Lifted Importance Sampling

Original MLN is not domain-liftable

Create a lifted proposal distribution by applying rules approximately


K ground atoms are conditioned to true and the rest to false

A Lifted Sample (ω) with weight $P(\omega)/Q(\omega)$

Gogate et al. '12

Challenges to applying Lifted Inference in Practice

- Domain-lifting is applicable to a very small subset of MLN structures
  - MLNs that contain 2 variables in a formula are domain-liftable
  - Lifting MLNs containing 3 variables in a formula is #P-hard
- For formulas commonly seen in NLP, such as $Token(t,i,c) \wedge Infield(i,f,c) \wedge Token(t,i',c') \wedge Infield(i',f,c') \Rightarrow SameField(f,c,c')$, no domain-lifting is possible

Broeck '11 Jaeger '12

Evidence Problem

Even for liftable MLNs, evidence breaks symmetries

$\forall x, \forall y\ Strong(x) \Rightarrow Wins(x,y);\ 1.75$

Atom	Pre-Evidence Marginals
$Wins(A,A)$	0.56
$Wins(A,B)$	0.56
$Wins(A,C)$	0.56
$Wins(B,A)$	0.56
$Wins(B,B)$	0.56
$Wins(B,C)$	0.56
$Wins(C,A)$	0.56
$Wins(C,B)$	0.56
$Wins(C,C)$	0.56

Evidence(True) Atoms
$Strong(C)$
$Wins(A,C)$
$Wins(B,B)$
$Wins(B,C)$
$Wins(C,A)$

Atom	Evidence conditioned Marginals
$Wins(A,A)$	0.6
$Wins(A,B)$	0.6
$Wins(B,A)$	0.63
$Wins(C,B)$	0.85
$Wins(C,C)$	0.85

Inference with evidence only on unary predicates is efficient while inference with evidence on binary predicates is #P-hard (Broeck and Darwiche '13)

Challenges to applying Lifted Inference in Practice

- Domain-lifting only utilizes exact symmetries
  - Advantage: Provable guarantees since lifting does not change the underlying distribution
  - Problem: Computational complexity is often as high as a propositional inference algorithm
- In MLNs that encode complex problems such as those in NLP, domain-lifted inference is insufficient. We need more flexible lifting methods
  - Trade-off guarantees with computational complexity

Outline

- Motivation
- MLN Basics
- Domain-Lifted Inference
- Approximate Domain-Lifting
- Lifted Generative Weight Learning
- Scalable Weight Learning with Approximate Counting Oracles
- Applications
- Conclusion

Evidence Problem

- Conditioning on unary atoms is easy while conditioning on binary atoms is hard
- Represent binary evidence as a combination of unary evidences

Given binary evidence $P(A,A) = P(A,B) = P(B,A) = P(B,B) = P(C,A) = P(D,A) = P(D,D) = 1$, and the rest of the atoms are 0, we can represent evidence as a matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Broeck and Darwiche '13

Boolean Factorization of Evidence

- Decompose the evidence matrix into binary products of vectors

In general, we can factorize the binary evidence as

$$P = QR^T$$

where $QR^T = q_1 r_1^T \vee q_2 r_2^T \vee \cdots q_n r_n^T$. n is the Boolean rank of P

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}^T \vee \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}^T \vee \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T$$

Boolean rank = 3

Boolean Factorization of Evidence

- Inference complexity can be characterized using Boolean rank of the evidence matrix
  - Inference is exponential in Boolean rank

- How can we use this to introduce artificial symmetries?
  - Low rank Boolean matrix factorization
  - Bounding the rank of the factorization will automatically smooth the evidence introducing more symmetries

Boolean Factorization of Evidence

- Low rank approximation

$$\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 1 & 0 & 1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}0\\1\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\\1\end{bmatrix}^T \vee \begin{bmatrix}1\\1\\0\\1\end{bmatrix}\begin{bmatrix}1\\1\\0\\0\end{bmatrix}^T$$

Boolean rank 2 approximation $\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & 1 & 0 & 1\\ 0 & 0 & \mathbf{0} & 0\\ 1 & 0 & 0 & 1\end{bmatrix}$

The new evidence has more symmetries than the original evidence

Boolean Factorization of Evidence

Link(aaai,google)
Link(google,aaai)
Link(google,gmail)
Link(IBM,aaai)

$\Longrightarrow$

Link(aaai,google)
Link(google,aaai)
~~Link(google,gmail)~~
Link(IBM,aaai)
Link(aaai,IBM)

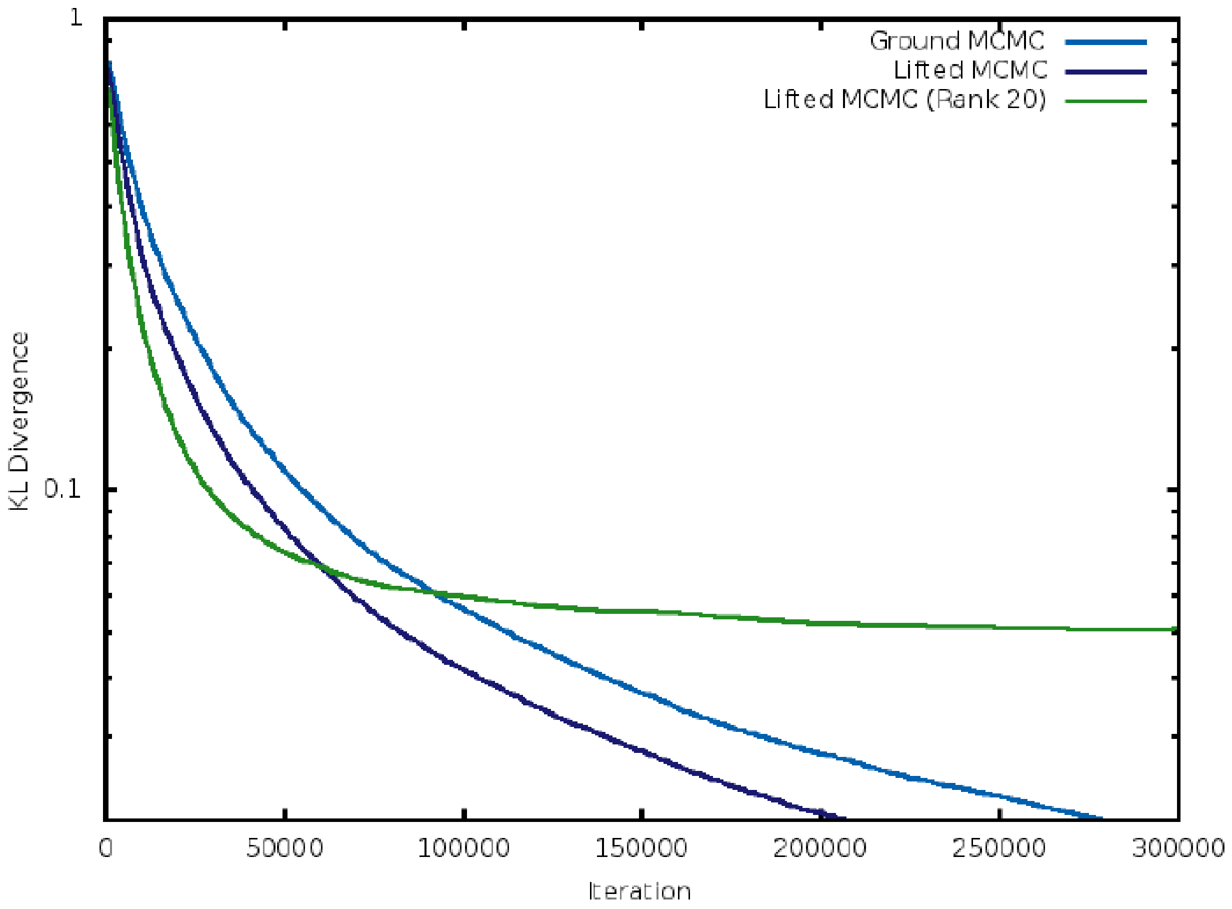
The objects google and IBM become more symmetric

Boolean Factorization of Evidence

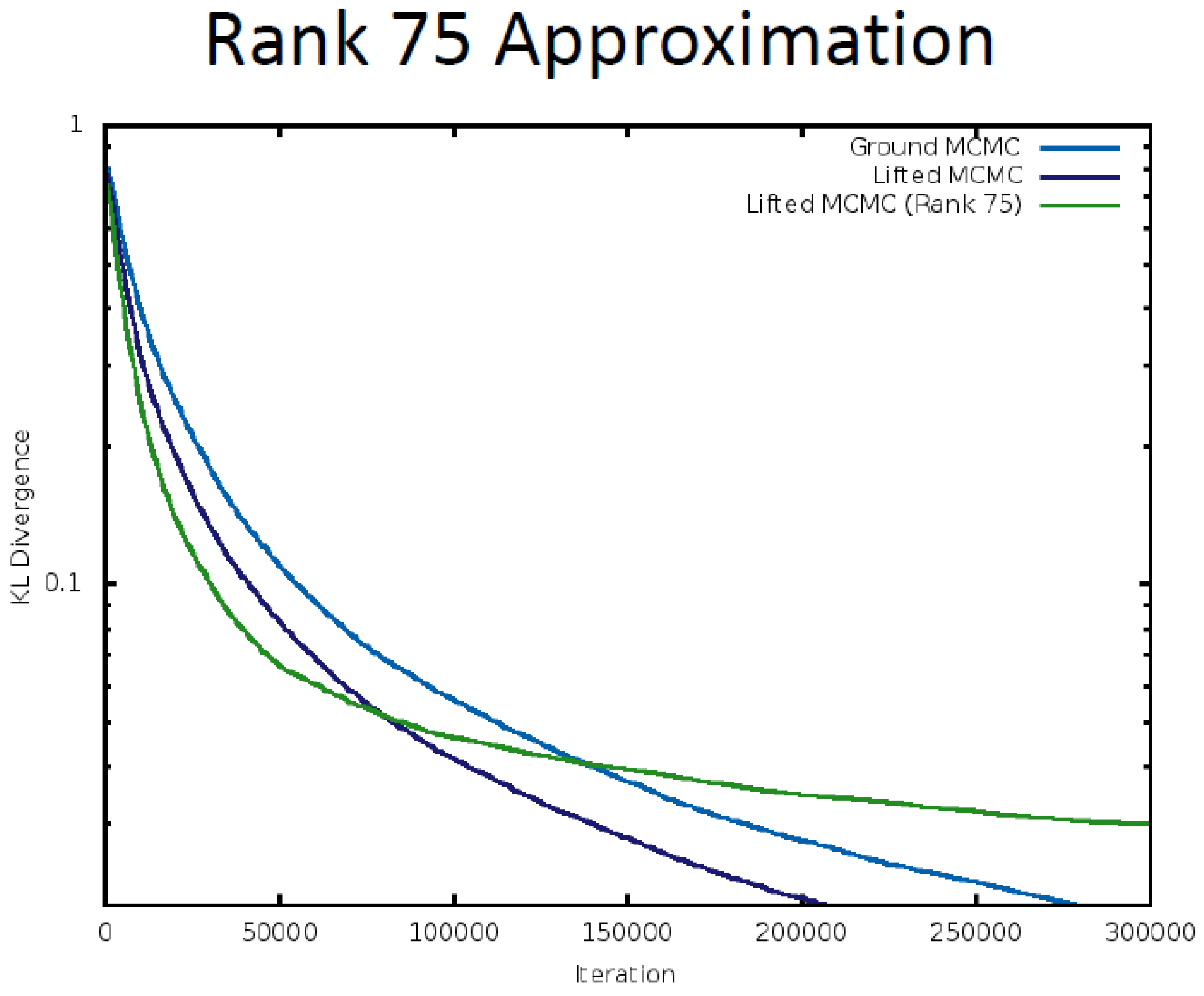
- Lift the MLN by converting binary evidences into their low rank approximations
- Use the low-rank approximation as a proposal distribution in MCMC algorithms (Broeck and Niepert '14)
- Problems: No guarantees on results since we are changing the underlying MLN distribution

Performance on WebKB

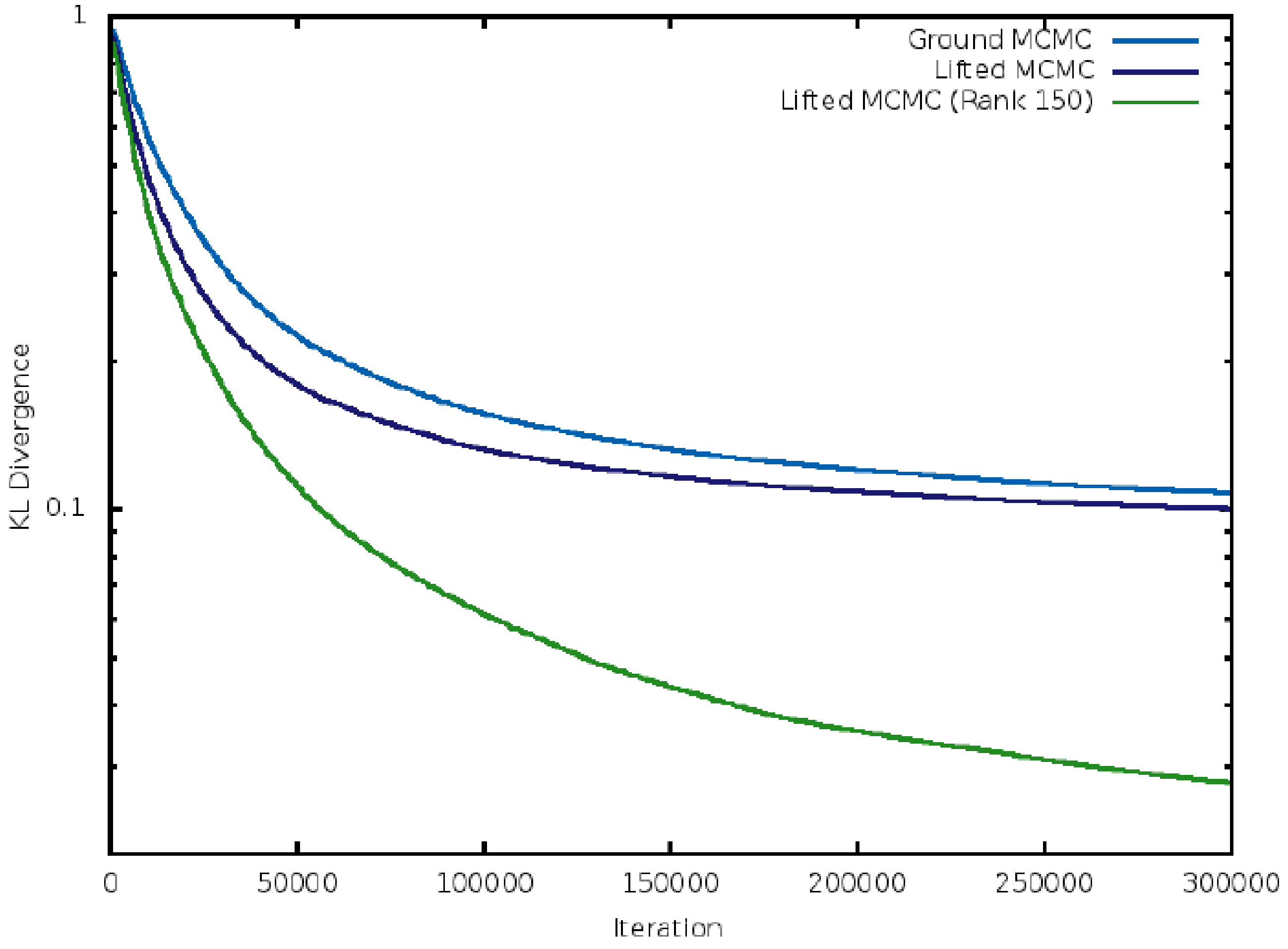
Rank 20 Approximation



Performance on WebKB



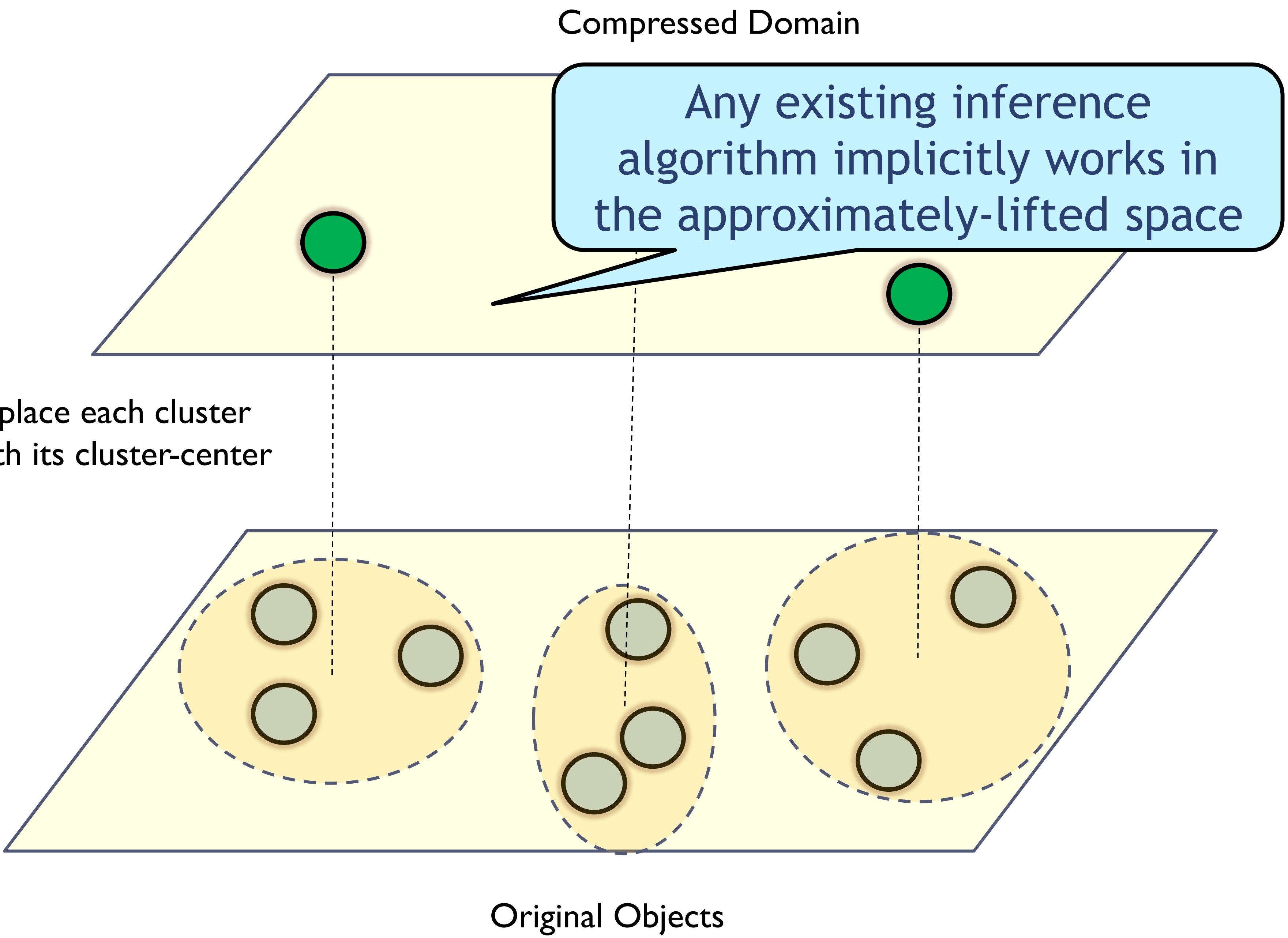
Performance on WebKB



Lifting using Unsupervised Machine Learning

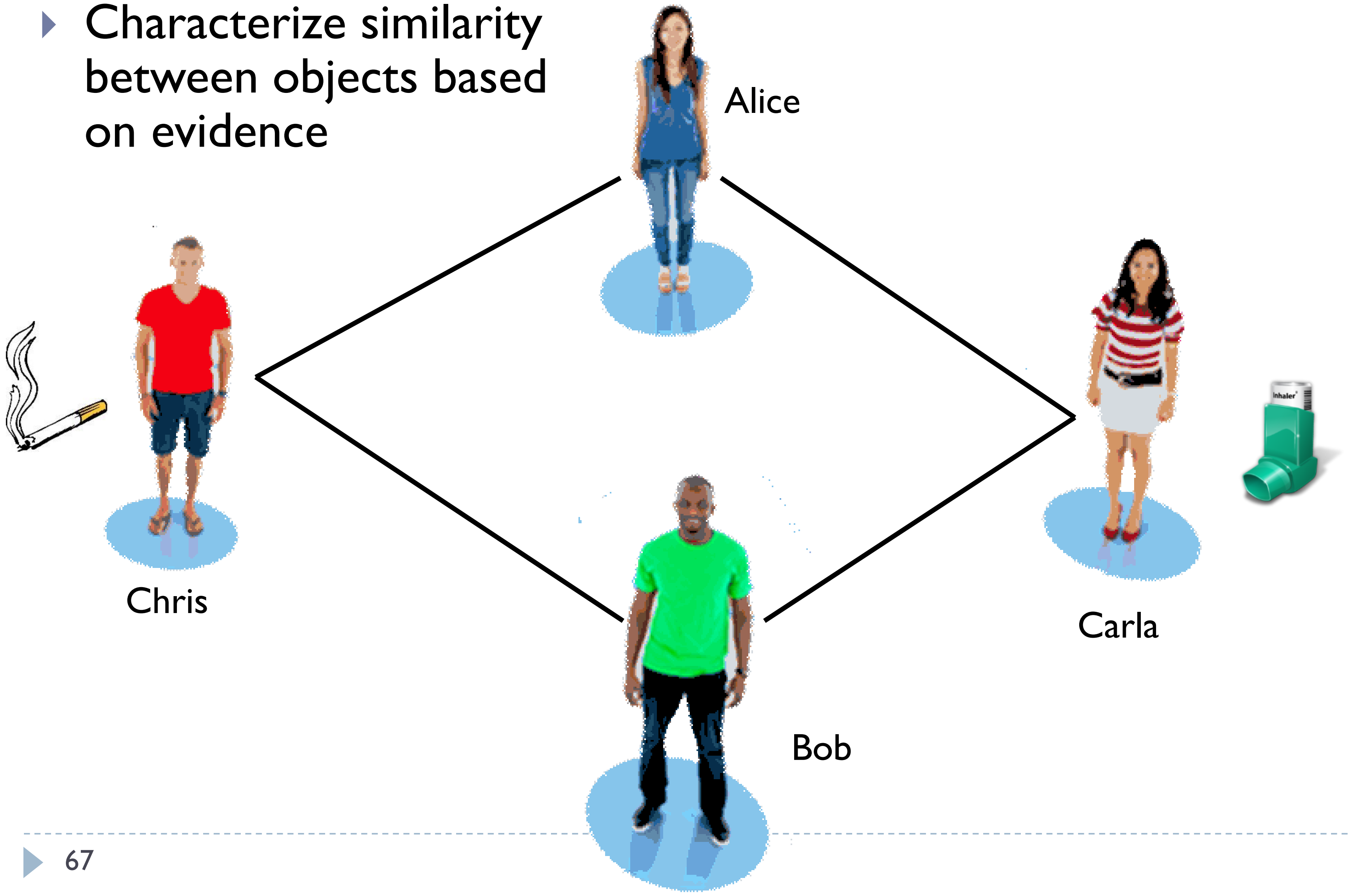
- Formulate lifting as a clustering problem
  - Flexible framework
  - Can use vast off-the-shelf resources to perform clustering
- Example: $word(w,p) \wedge topic(p,t) \wedge linked(p,q) \Longrightarrow topic(q,t)$
  - Is $P\big(topic(*,baseball)\big) \approx P\big(topic(*,basketball)\big)$?
  - Cluster similar topics together to compress the MLN

Venugopal and Gogate '14


Compressed Domain
Any existing inference algorithm implicitly works in the approximately-lifted space
Replace each cluster with its cluster-center
Original Objects

Features for Clustering

- Characterize similarity between objects based on evidence

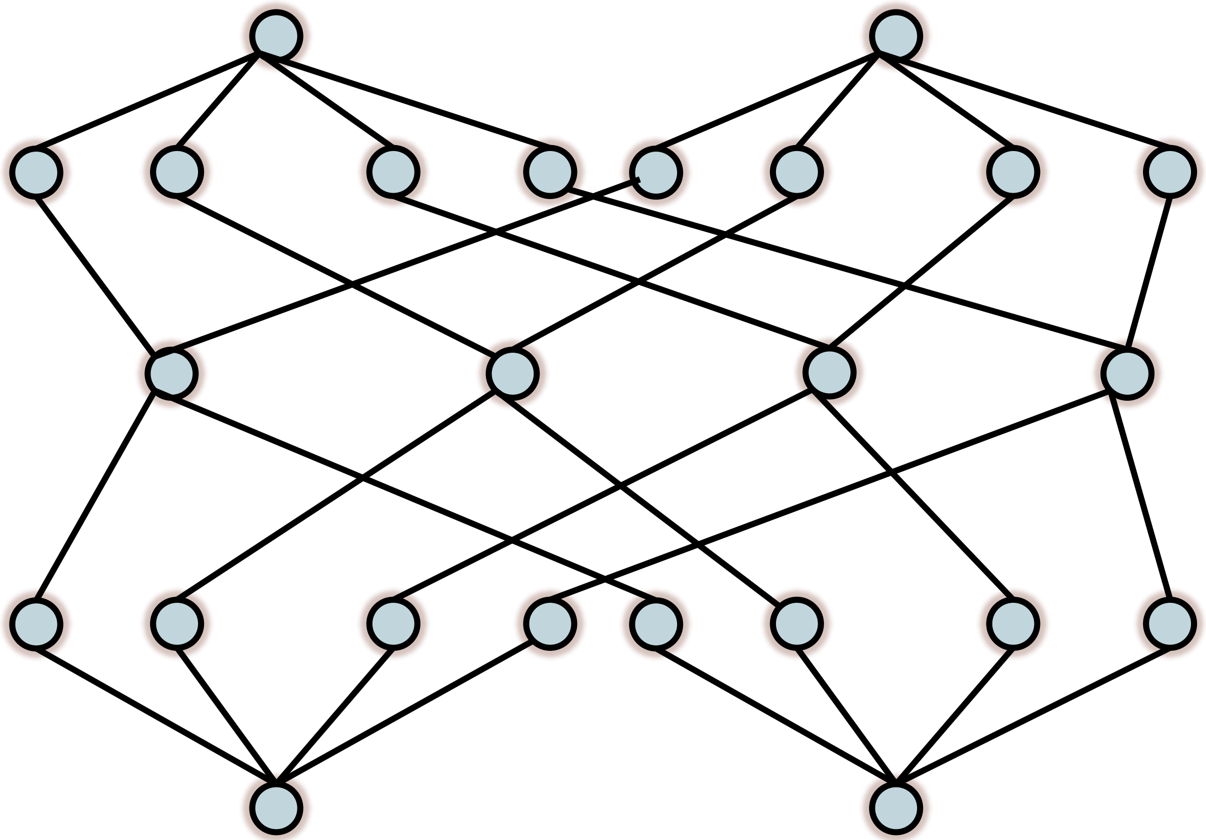


Evidence based Clustering

$Smokes(x) \wedge Friends(x, y) \Rightarrow \neg Asthma(y)$

$x = Ana$

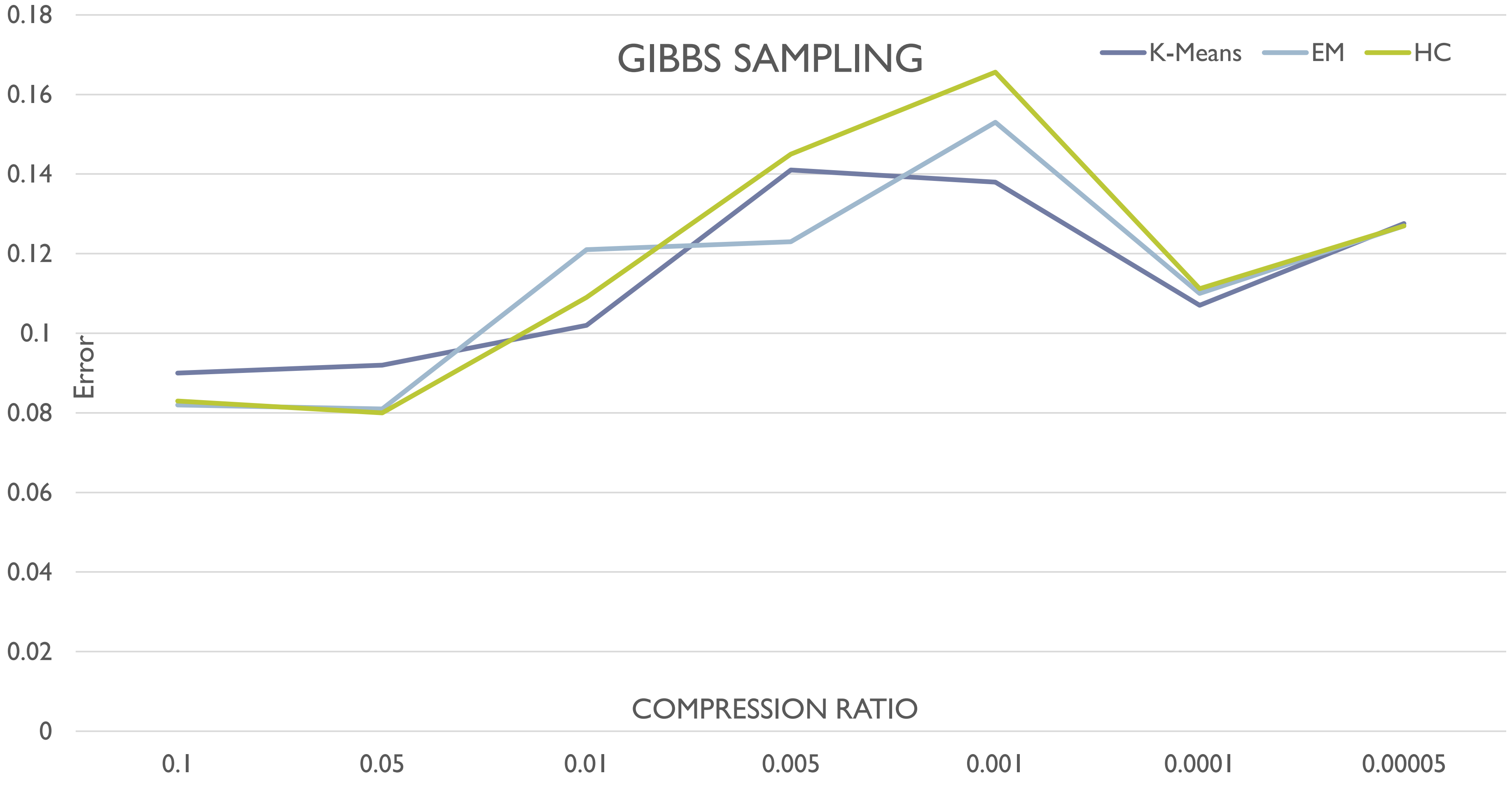
$x = Bob$



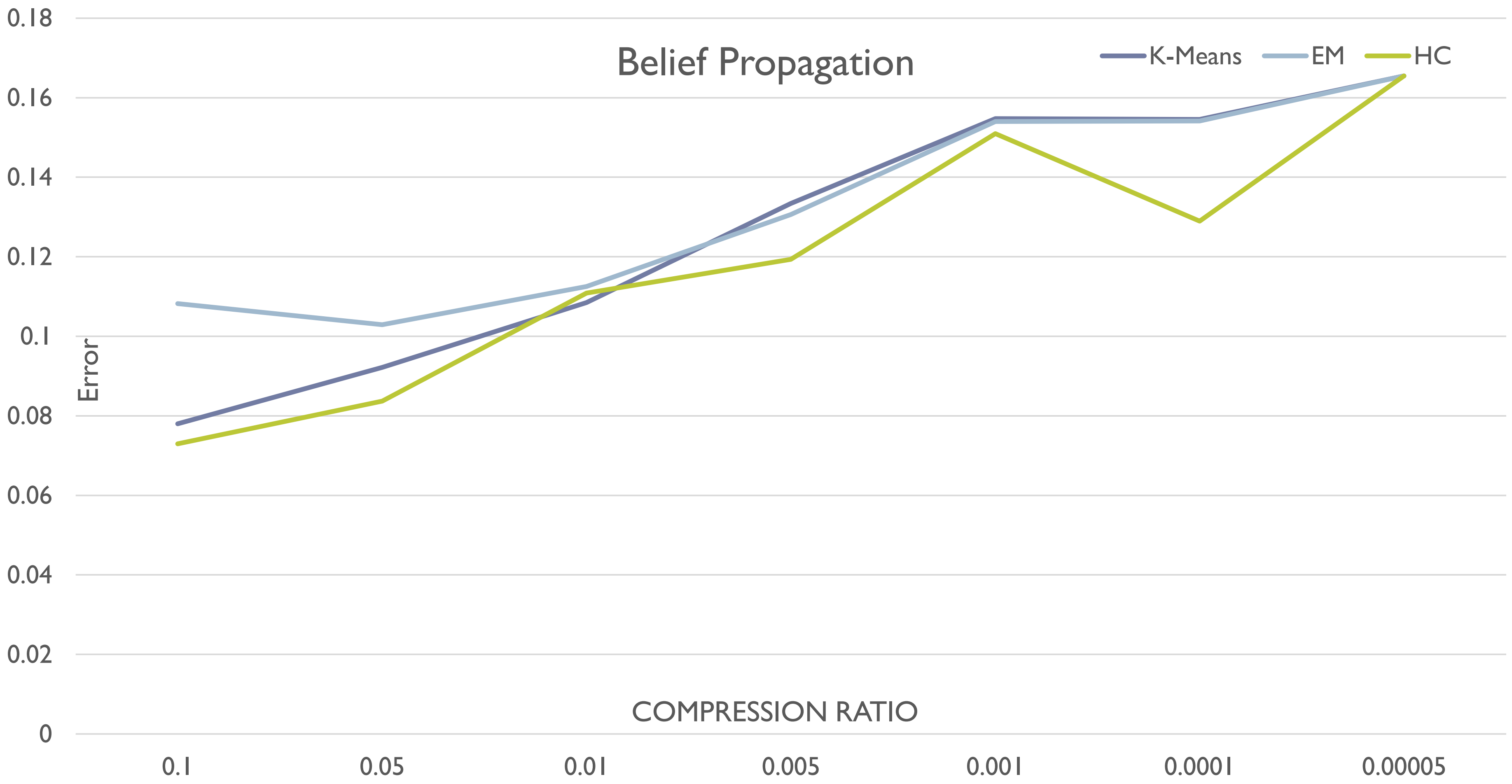
Features for Clustering

- Instantiate the MLN with a single object and count the true groundings of each formula in the evidence
  - Counting the true groundings of a first-order formula is also a hard problem
  - Break the formula into multiple pieces and count independently
- Advantages
  - Clustering adapts itself to changing evidence
  - Can implicitly lift several propositional inference algorithms
- Heuristic method: No guarantees on solution since we are changing the distribution

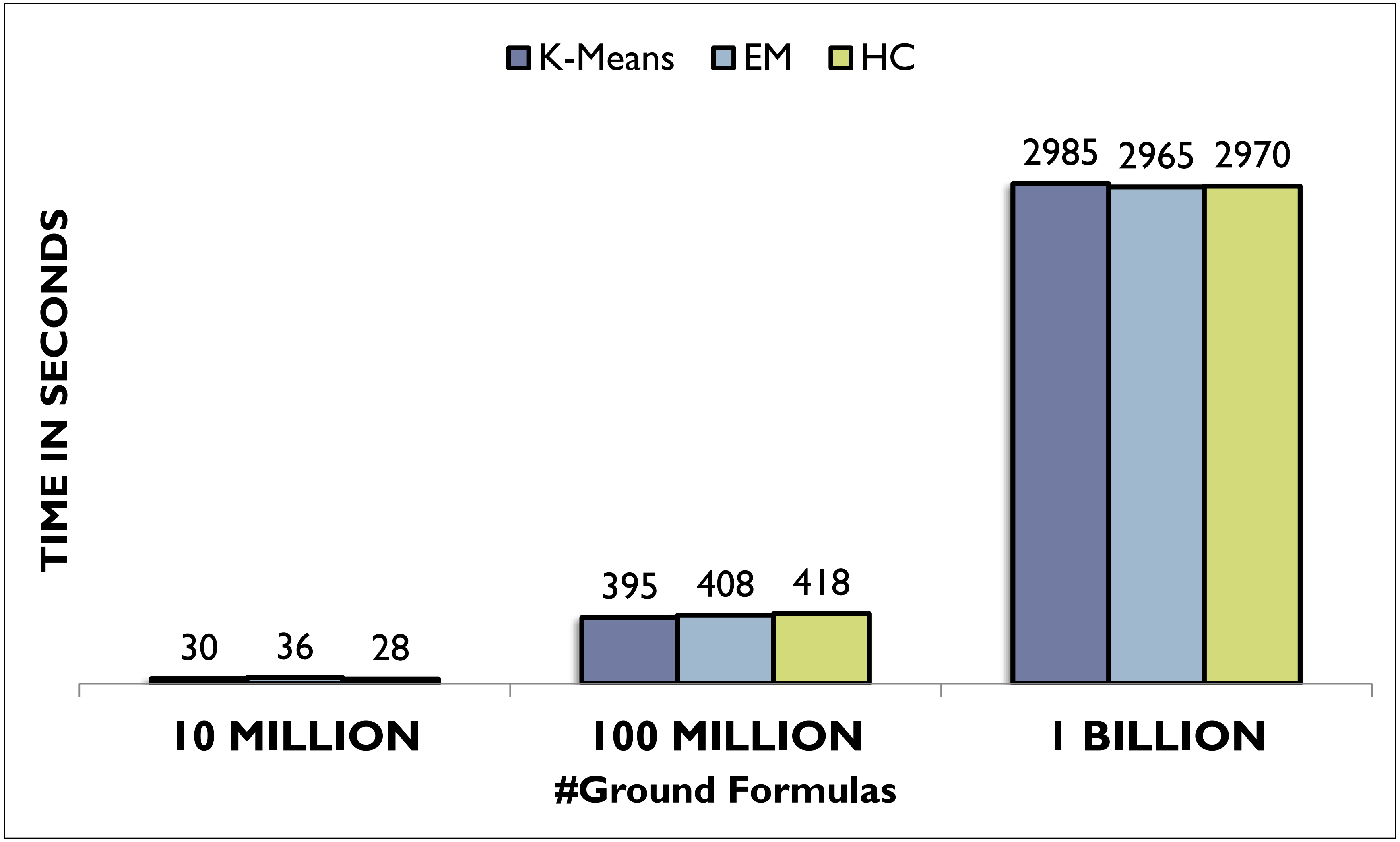
Performance: Citation Segmentation



Performance: Citation Segmentation



Experiments: Citation Segmentation



Approximate lifting of BP

- Problem with lifted BP: The lifted network turns out to be quite similar to the propositional network in the absence of symmetries
- Stop the construction of the lifted network early
  - Start with a coarse network and refine it by keeping track identical BP messages
  - Stop refinement when the number of supernodes or superfeatures reaches a pre-specified threshold
  - Lifted network is approximate but is much smaller than the propositional network

Singla et al. '14

Performance

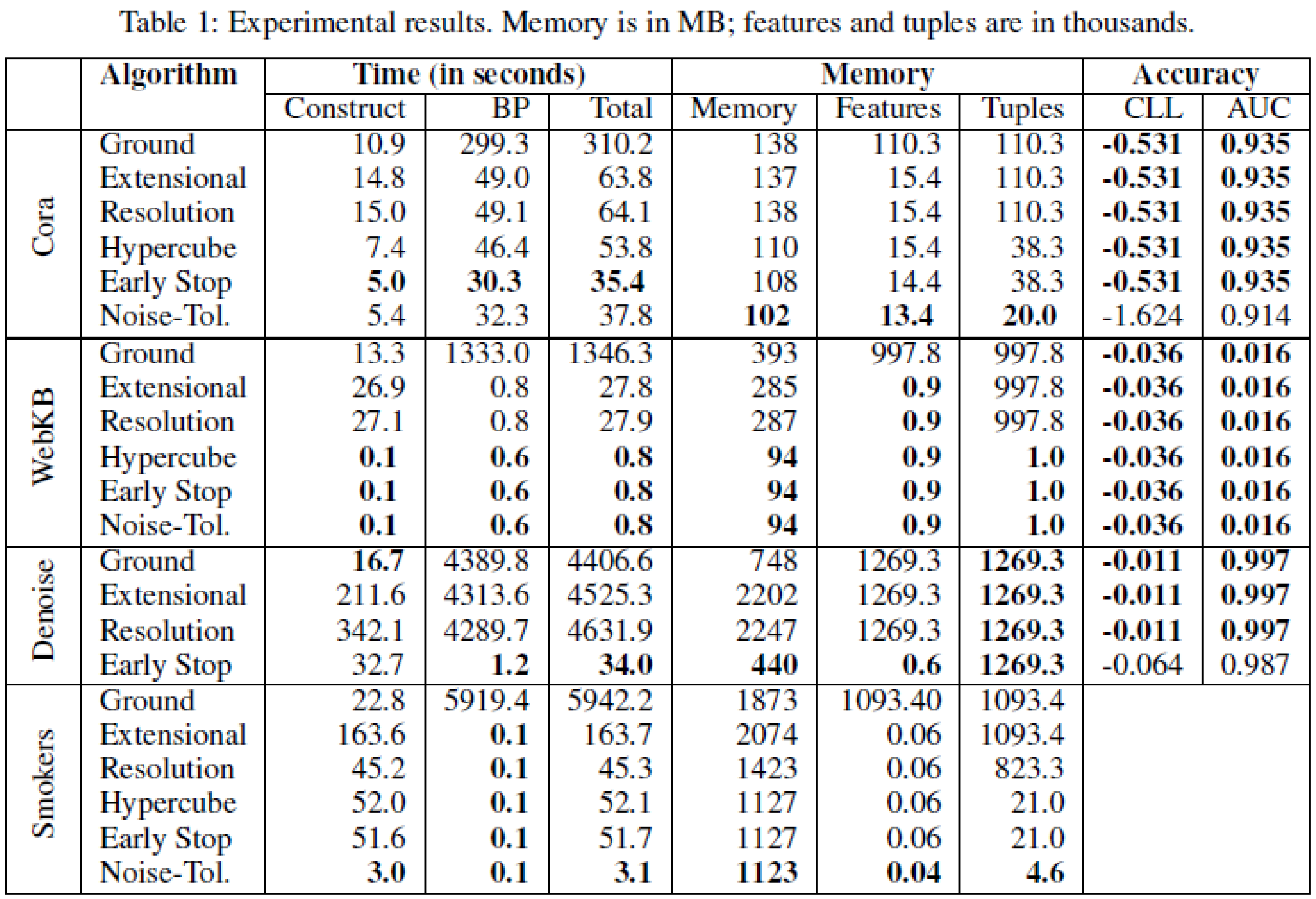


Table 1: Experimental results. Memory is in MB; features and tuples are in thousands.

	Algorithm	Time (in seconds)			Memory			Accuracy	
		Construct	BP	Total	Memory	Features	Tuples	CLL	AUC
Cora	Ground	10.9	299.3	310.2	138	110.3	110.3	**-0.531**	**0.935**
	Extensional	14.8	49.0	63.8	137	15.4	110.3	**-0.531**	**0.935**
	Resolution	15.0	49.1	64.1	138	15.4	110.3	**-0.531**	**0.935**
	Hypercube	7.4	46.4	53.8	110	15.4	38.3	**-0.531**	**0.935**
	Early Stop	**5.0**	**30.3**	**35.4**	108	14.4	38.3	**-0.531**	**0.935**
	Noise-Tol.	5.4	32.3	37.8	**102**	**13.4**	**20.0**	-1.624	0.914
WebKB	Ground	13.3	1333.0	1346.3	393	997.8	997.8	**-0.036**	**0.016**
	Extensional	26.9	0.8	27.8	285	**0.9**	997.8	**-0.036**	**0.016**
	Resolution	27.1	0.8	27.9	287	**0.9**	997.8	**-0.036**	**0.016**
	Hypercube	**0.1**	**0.6**	**0.8**	**94**	**0.9**	**1.0**	**-0.036**	**0.016**
	Early Stop	**0.1**	**0.6**	**0.8**	**94**	**0.9**	**1.0**	**-0.036**	**0.016**
	Noise-Tol.	**0.1**	**0.6**	**0.8**	**94**	**0.9**	**1.0**	**-0.036**	**0.016**
Denoise	Ground	**16.7**	4389.8	4406.6	748	1269.3	**1269.3**	**-0.011**	**0.997**
	Extensional	211.6	4313.6	4525.3	2202	1269.3	**1269.3**	**-0.011**	**0.997**
	Resolution	342.1	4289.7	4631.9	2247	1269.3	**1269.3**	**-0.011**	**0.997**
	Early Stop	32.7	**1.2**	**34.0**	**440**	**0.6**	**1269.3**	-0.064	0.987
Smokers	Ground	22.8	5919.4	5942.2	1873	1093.40	1093.4		
	Extensional	163.6	**0.1**	163.7	2074	0.06	1093.4		
	Resolution	45.2	**0.1**	45.3	1423	0.06	823.3		
	Hypercube	52.0	**0.1**	52.1	1127	0.06	21.0		
	Early Stop	51.6	**0.1**	51.7	1127	0.06	21.0		
	Noise-Tol.	**3.0**	**0.1**	**3.1**	**1123**	**0.04**	**4.6**		

Approximate Lifting for MAP

- MAP inference in MLNs can be solved using off-the-shelf linear programming solvers
  - LP implementations work in a propositional representation
- Lift the MLN as a pre-processing step before giving it to the LP solver
- Exact lifting rule
  - Non-shared variables can be reduced to a single object
  - In $Word(w, p) \Rightarrow Topic(p, t)$, the words (w) and topics (t) are non-shared variables
  - For non-shared variables the MAP solution is symmetrical for all instantiations. $Word(W_1, p), Word(W_2, p)$, … have the same assignment in the MAP solution

Sarkhel et al. '14, '15

Approximate Lifting for MAP

- To lift approximately, add equality constraints to the linear program to reduce the set of feasible solutions

$$EventCoref(x,y) \wedge EventCoref(y,z) \wedge EventCoref(z,x)$$

Add constraints of the form

$$EventCoref(E_1,E_2) \Leftrightarrow EventCoref(E_3,E_4)$$
$$EventCoref(E_2,E_3) \Leftrightarrow EventCoref(E_3,E_4)$$
$$\ldots$$

Note that the constraints above are still propositional. We can define this in a more compact manner.

Approximate Lifting for MAP

- Partition the domain into subsets
  - Let original domain of events be $\{E_1, E_2, E_3\}$
  - Partition $\pi = \{\{E_1, E_2\}, \{E_3\}\} = \{v_1, v_2\}$
- Ground the original MLN using the partition
  - $EventCoref(v_1, v_1) \wedge EventCoref(v_1, v_1) \wedge EventCoref(v_1, v_1)$
  - $EventCoref(v_1, v_2) \wedge EventCoref(v_2, v_1) \wedge EventCoref(v_1, v_2)$
  - …
- Implicitly adds equality constraints
  - $EventCoref(E_1, E_1) \Leftrightarrow EventCoref(E_1, E_2)$
  - $EventCoref(E_1, E_2) \Leftrightarrow EventCoref(E_2, E_1)$
  - …

Approximate Lifting for MAP

- How to partition the domain?
  - Exponential number of possible partitionings
  - Much smaller number of effective partitionings since some partitions are exchangeable
- Can represent the effective partitions as a lattice
- Grounding according to the partitions specified as we go up the lattice gives us provably more accurate MAP solutions

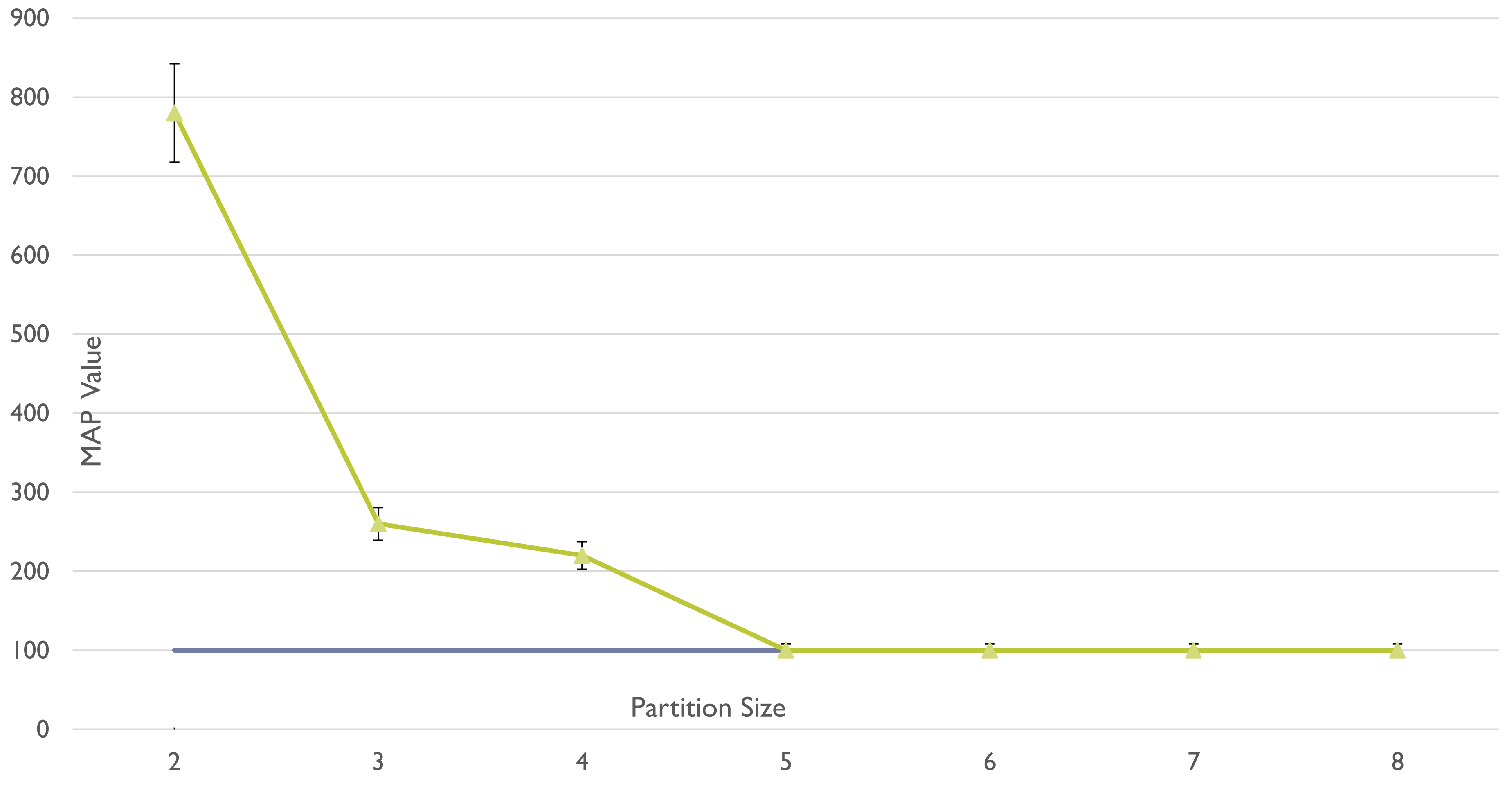
Approximate Lifting for MAP

Increased complexity and better accuracy

$$\begin{tikzcd}
 & \{\{E_1\},\{E_2\},\{E_3\},\{E_4\}\} & \\
 & \{\{E_1\},\{E_2\},\{E_3,E_4\}\} \arrow[u] & \\
\{\{E_1\},\{E_2,E_3,E_4\}\} \arrow[ur] & & \{\{E_1,E_2\},\{E_3,E_4\}\} \arrow[ul] \\
 & \{E_1,E_2,E_3,E_4\} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Partitions at the same level are exchangeable

Comparing Optimal MAP value to Approximation for Citation Segmentation



Matrix-Factorization based smoothing, Approximately Lifted BP, MAP, Clustering-based lifting,…

Lifted/Counting Belief Propagation, Lifted MCMC, Lifted Importance Sampling, Lifted MAP,…

Lifted factor graphs, FOVE, WFOMC, PTP,…

MCSAT, SampleSearch, GiSS, And-OR, AC's

Importance Sampling, Gibbs Sampling, Belief Propagation,…

Exploit logical structure and symmetries

Outline

Weight Learning Algorithms

- Task: Given the MLN formulas and training data, learn the parameters/weights for each formula
- In MLN weight learning, typically there is just a single training instance to learn from (a relational database)
- Similar to inference, grounding the MLN is problematic in weight learning since it creates a huge Markov network
- Domain specific-heuristics try reduce the size of the Markov network
  - E.g., In Entity Resolution (Singla&Domingos '06) use the canopy approach (Macallum et al. '00) to estimate atoms that are likely to be false
- Problem: Need domain knowledge to make MLNs work for new applications
  - Application designers cannot use MLNs off-the-shelf

Generative Learning

- Task: Given the formulas and complete relational database (a full world ω), estimate parameters θ
- $\ell(\theta;\omega) = log P_\theta(\omega) = \sum_i \theta_i N_i(\omega) - log\, Z_\theta$
- Use gradient ascent to maximize the log-likelihood
- The gradient for the i-th formula is equal to
  - $\frac{\partial}{\partial\theta_i} = N_i(\omega) - \mathbb{E}_\theta[N_i]$
  - $N_i(\omega)$ is the number of groundings of the i-th formula that is true in the data
  - $\mathbb{E}_\theta[N_i]$ is the expected number of groundings of the i-th formula that is true in the data using the weights θ

Lifted Generative Learning

- Lifted generative learning uses a lifted inference oracle to compute $\mathbb{E}_\theta[N_i(\omega)]$
  - Note that there is no conditioning on data when computing the expectation
- Consider an MLN with a single formula F with N groundings $f_1, f_2,\ldots f_N$
- $\mathbb{E}_\theta[N_i(\omega)] = \mathrm{P}(f_1) + \mathrm{P}(f_2) + \cdots \mathrm{P}(f_N)$
- We require to compute marginal probabilities for each grounding of the formula which is still a hard problem even when we o not condition on evidence

Haaren et al. '15

Lifted Generative Learning

- Two key benefits
  - Can compute gradient from a small number of marginal
  - Each marginal can be computed in a domain-lifted manner
- Group the groundings into equiprobable sets and compute the marginal over groups
  - $\mathbb{E}_\theta[N_i(\omega)] = |E_1|\mathrm{P}\left(f_{E_1}\right) + |E_2|\mathrm{P}\left(f_{E_2}\right) + \cdots |E_N|\mathrm{P}\left(f_{E_N}\right)$
- Compute the probability of each equiprobable set using a lifted inference oracle (WFOMC)
  - The structure of the MLN should allow for lifted inference

Outline

Problems with Lifted Weight Learning

- What if the MLN structure does not allow lifted inference
  - E.g., $EventCoref(x, y) \lor EventCoref(y, z) \Rightarrow EventCoref(z, x)$
- The same approach will simply not work for discriminative learning which typically converges faster in real-world applications
  - We need to condition on certain query variables
  - Evidence problem
- New approach: Scale up weight learning using approximate counting oracles
  - Scales up a family of weight learning algorithms including voted perceptron, contrastive divergence and pseudo-likelihood learning

Scalable Weight Learning using Approximate Counting

- Task: Compute $N_i(\omega) - \mathbb{E}_\theta[N_i(\omega)]$ efficiently
  - Computing $N_i(\omega)$ is a #P-hard problem (Domingos and Lowd '09) in the general case
  - Computing $\mathbb{E}_\theta[N_i(\omega)]$ is an even harder problem
- To solve the counting problem efficiently, encode each formula as a graphical model (Venugopal et al. '15)
  - Key property: Given a world ω, counting the number of satisfied groundings in ω ($N_i(\omega)$) is equivalent to computing the partition function of the encoded graphical model

Scalable Weight Learning using Approximate Counting

$$f = \forall x \, \forall y \, \forall z \, \neg R(x,y) \vee S(y,z)$$
$$\Delta_x = \Delta_y = \Delta_z = \{A, B\}$$

$R(A,A)$	1
$R(A,B)$	0
$R(B,A)$	0
$R(B,B)$	1

Identity Function $\longrightarrow$

x	y	$\emptyset_1$
A	A	1
A	B	0
B	A	0
B	B	1

$S(A,A)$	0
$S(A,B)$	1
$S(B,A)$	0
$S(B,B)$	1

Inverse Function $\longrightarrow$

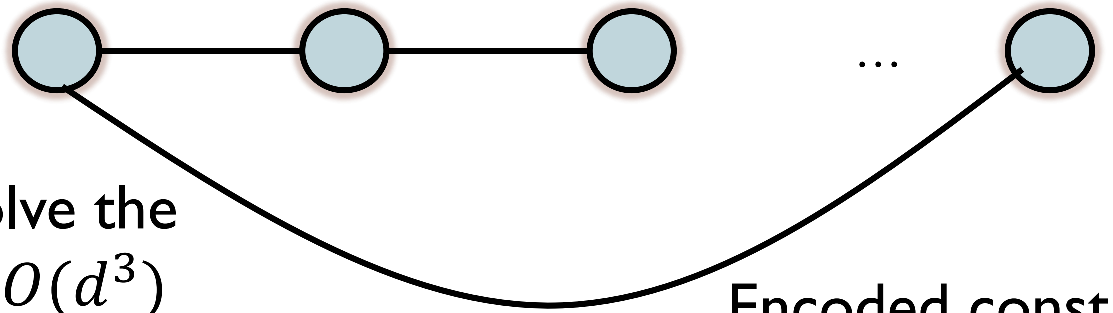
y	z	$\emptyset_2$
A	A	1
A	B	0
B	A	1
B	B	0

Scalable Weight Learning using Approximate Counting

- Any exact or approximate counting oracle for the graphical model can be used to compute $N_i(\omega)$

d^n possible groundings

$R_1(x_1, x_2) \vee R_2(x_2, x_3) \ldots R_n(x_n, x_1)$



Junction Trees can solve the counting problem in $O(d^3)$ time/space

Encoded constraint network has tree-width 2

Scalable Weight Learning with Approximate Counting Oracles

- How about the expectation $\mathbb{E}_\theta[N_i(\omega)]$?
  - Use approximate inference methods to estimate the expectation
- Within each step of typical approximate inference methods (Gibbs sampling, MaxWalkSAT,, etc.), we need to re-solve the counting problem multiple times
  - Use approximate counting oracles here as well
  - The direction of the gradient is more important than obtaining the correct expectation

Contrastive Divergence

- Approximate the expectation term in the gradient using Gibbs samples (Hinton '02)
  - A few Gibbs samples are sufficient to "show" the correct direction
  - Do not need to simulate the Markov chain until it reaches convergence
  - The approximate gradient $\frac{\partial}{\partial \theta_i} = M_i(\omega) - \overline{\mathbb{E}_\theta}\,[M_i(\omega)]$, where $\overline{\mathbb{E}_\theta}\,[M_i(\omega)]$ is estimated from k Gibbs samples generated using an approximate counting oracle

Voted Perceptron

- Approximate the expectation term in the gradient using the mode of the distribution (Collins '02)
  - If the probability mass is mostly distributed around the mode, then this gives a good approximation to the gradient
  - Problem: In multi-modal distributions we will end up in local optima
  - The approximate gradient $\frac{\partial}{\partial \theta_i} = M_i(\omega) - \overline{\mathbb{E}_\theta}\ [M_i(\omega)]$, where $\overline{\mathbb{E}_\theta}\ [M_i(\omega)]$ is estimated from the MAP assignment for the current θ using an approximate counting oracle

Pseudo Likelihood Learning

- Alternate objective that does not use gradient ascent
  - Assumes a relaxed, tractable form of the MLL optimization problem
  - However, need to pre-compute several conditional probabilities
  - Computing each probability requires the computation of $N_i(\omega)$ which we approximate with the approximate counting oracle

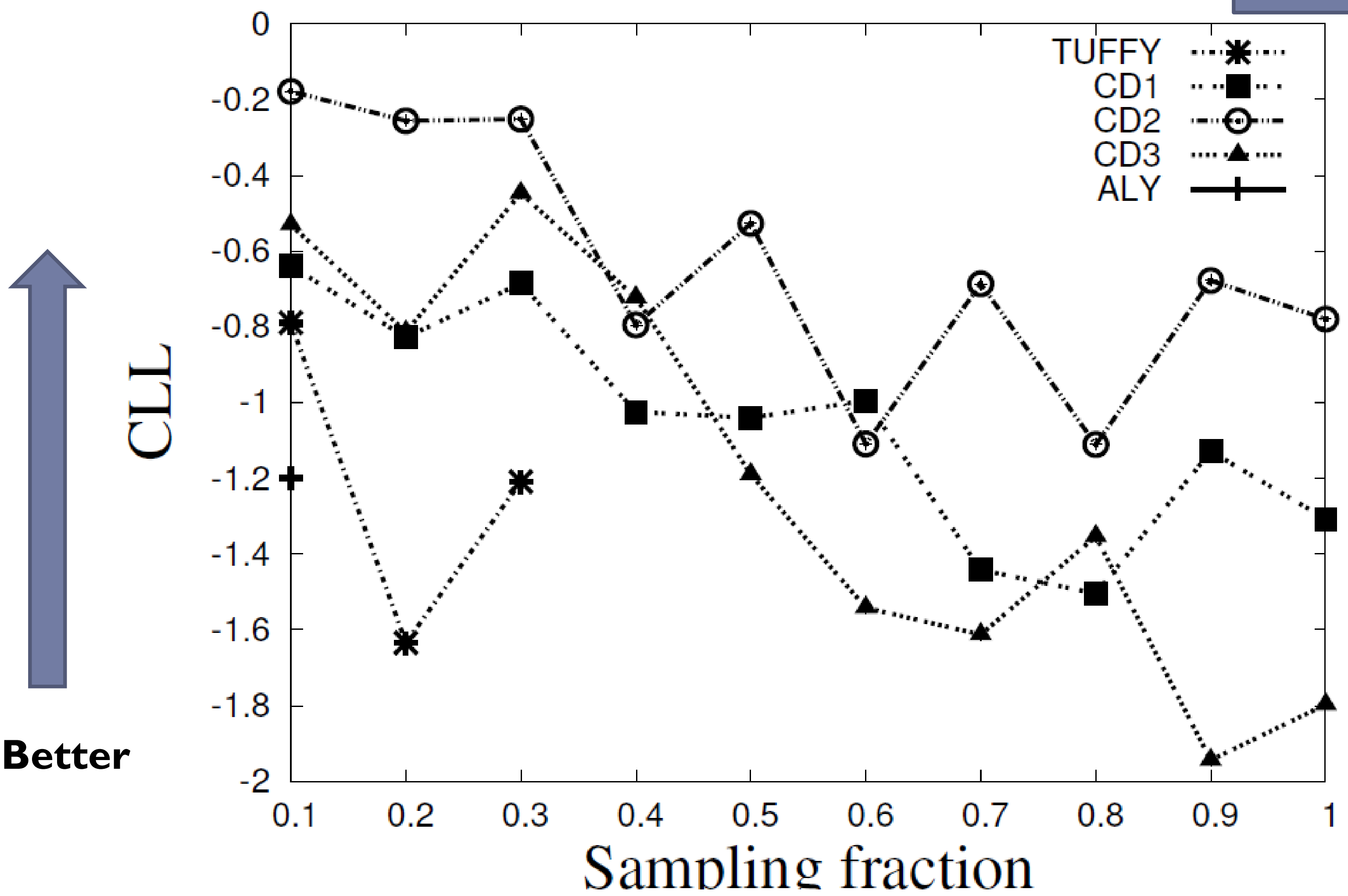
Performance

System	WebKB	Protein	ER	Smoker
Alchemy	✗	✗	✗	1.21 (0.03)
Tuffy	-0.89 (0.05)	✗	✗	-0.69 (0.04)
CD	-0.64 (0.004)	-0.779 (0.0002)	-0.694 (0.001)	-0.7 (0.004)
VP	-0.91 (0.001)	-0.78 (0.001)	-0.693 (0.001)	-1.16 (0.1)
PLL	-0.72 (0.001)	-0.74 (0.0004)	-0.72 (0.001)	-1.61 (0.08)

Conditional Log Likelihood Scores along with standard deviation for 5-fold cross validation

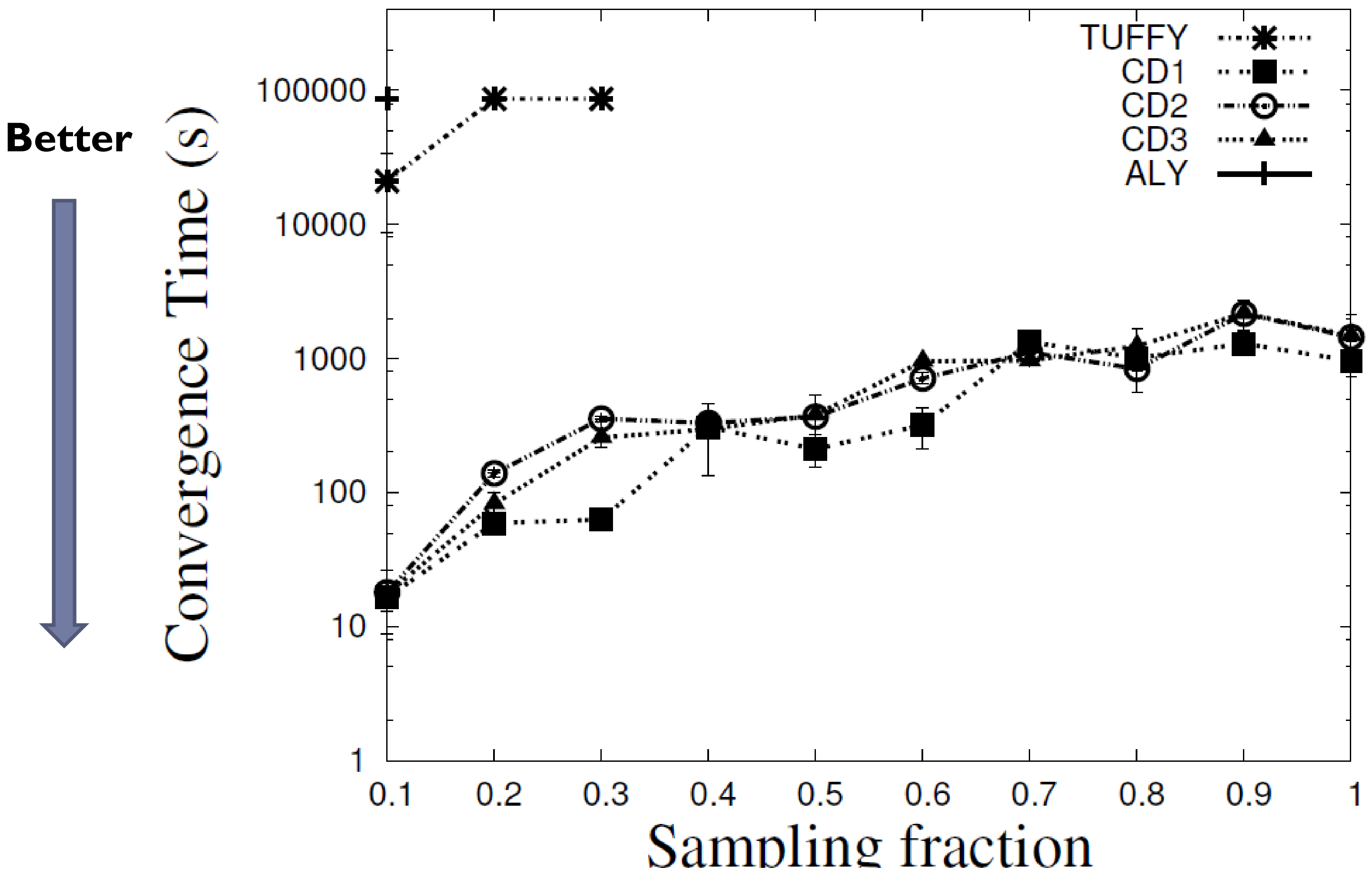
Performance

CDk : ibound = k



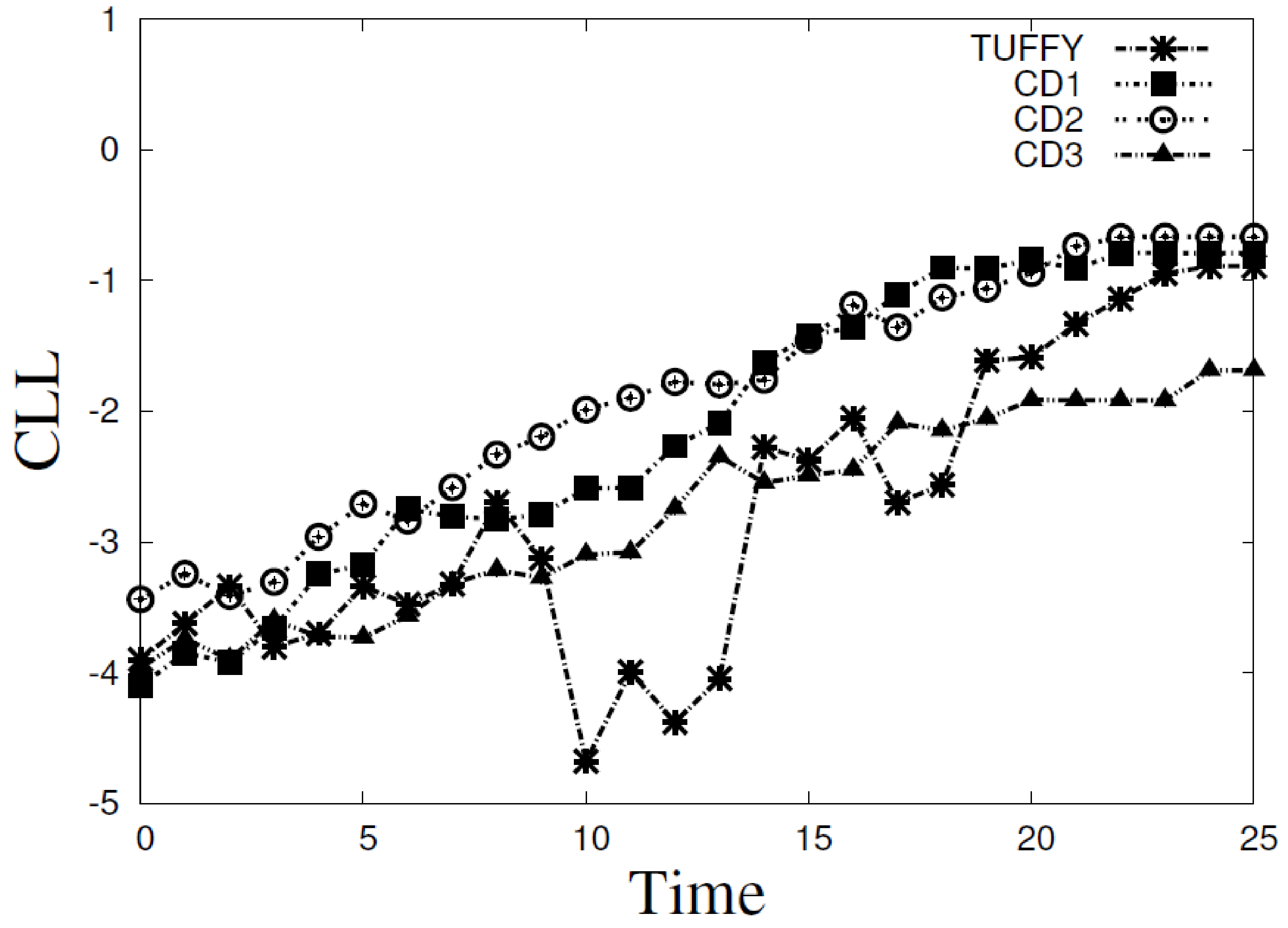
Protein Interaction MLN

Performance



Protein Interaction MLN

Performance



Better

Collective Classification MLN (WebKB)

Outline

An Overview of MLN Software along with Demonstrations

- Alchemy 1.0
  - Earliest implementation of MLN inference and learning algorithms
  - MCSAT, Gibbs Sampling, Belief Propagation, MaxWalkSAT,…
  - Contrastive Divergence, Voted Perceptron, …
- Pros
  - Several inference and learning algorithms along with various tunable options
- Cons
  - Lack of lifted inference algorithms (except Lifted BP)
  - Uses a pre-grounding approach and therefore the grounding process takes extremely long and/or runs out memory often

An Overview/Demo of MLN Software

- Alchemy 2.0
  - A suite of lifted inference algorithms built on top of Alchemy
  - PTP, Lifted Gibbs, Lifted Importance Sampling
- Pros
  - When the right symmetries are present, lifted inference algorithms are highly scalable
- Cons
  - Only uses exact symmetries. Therefore, not scalable for general MLNs
  - Often fails when arbitrary evidence is presented to the MLN since it breaks symmetries

An Overview/Demo of MLN Software

- Tuffy/Hazy
  - Built on top of MySQL database
  - MCSAT, MaxWalkSAT
- Pros
  - Efficient grounding technique
  - Supports weighted evidence
- Cons
  - Not too many algorithm options
  - Propositional algorithms, does not exploit symmetries or use lifted inference methods

An Overview/Demo of MLN Software

- Markov the Beast
  - MAP inference
- Pros
  - Integrates cutting plane algorithms into MLN inference
- Cons
  - Propositional algorithm, does not exploit symmetries or use lifted inference methods

An Overview/Demo of MLN Software

- Rockit
  - MAP Inference
- Pros
  - Integrates ILP solvers into MLN inference
  - Parallel implementation
- Cons
  - Propositional algorithms does not exploit symmetries or use lifted inference methods

An Overview/Demo of MLN Software

- Magician (beta version)
  - Approximate Inference and learning
  - Gibbs sampling and Contrastive Divergence
- Pros
  - Does not pre-ground the MLN
  - Uses an approximate counting oracle (Iterative Join Graph Propagation) to perform scalable inference and weight learning
- Cons
  - Not too many algorithm options
  - Under development

Text Classification

Task: Label the topic of a web-page

For almost three quarters it looked like this one was going to be different. Newton got a head start on a second straight NFL MVP award, a new touchdown…	Sports
The Oscars were a big success. Several celebrities gathered…	Entertainment

Assume that each page has exactly one label

Text Classification

- Predicate Definitions
  - $Topic(page, topic!)$:The ! mark signifies mutual exclusion
  - $HasWord(word, page)$
  - $Linked(page, page)$: Model interdependency between pages using
- Formulas
  - $HasWord(w, p) \Rightarrow Topic(p, t)$
  - However, the above formula is problematic because it learns a single weight for all possible words and topics.
  - We can specify that we need a different weight for each possible word-topic pair by augmenting variables with "+"
  - $HasWord(+w, p) \Rightarrow Topic(p, +t)$

Possible Approximate Symmetries

- Words may be exchangeable.
  - Happy; Elated; Glad
- Related topics may also exhibit approximate symmetries
  - Technology; Science
- Clustering can help us exploit some of these approximate symmetries

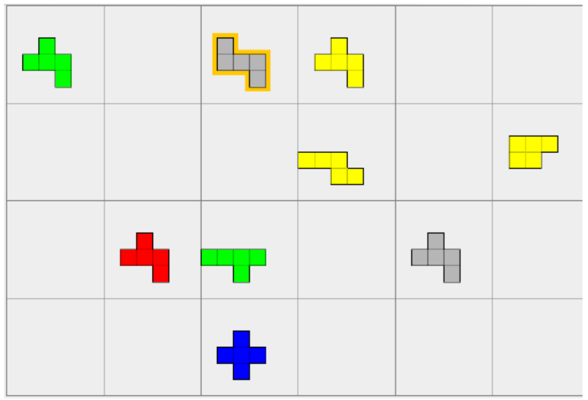
Situated Incremental Natural Language Understanding

- Understand language in a situated domain, i.e., regarding visually present entities, in an incremental manner.
  - Applications in dialogue systems for robotics
- Jointly use discourse context, language in the dialogue utterance and visual cues for the dialogue

- Dataset: User instructs computer to pick up, drop, etc. pieces of a puzzle. For each dialogue utterance, the corpus records game state, system action and intended interpretation of the utterance
- Task: Given the current state of the puzzle board, the previous system action and the utterance, predict the interpretation of the utterance as a frame in an incremental, word-by-word manner

Kennington et al. '12

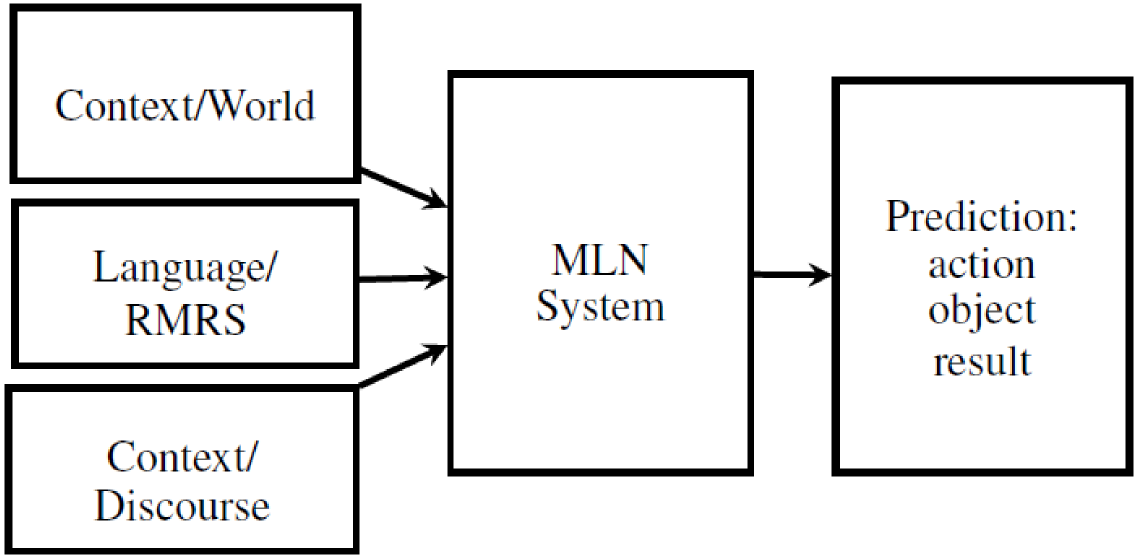
Task



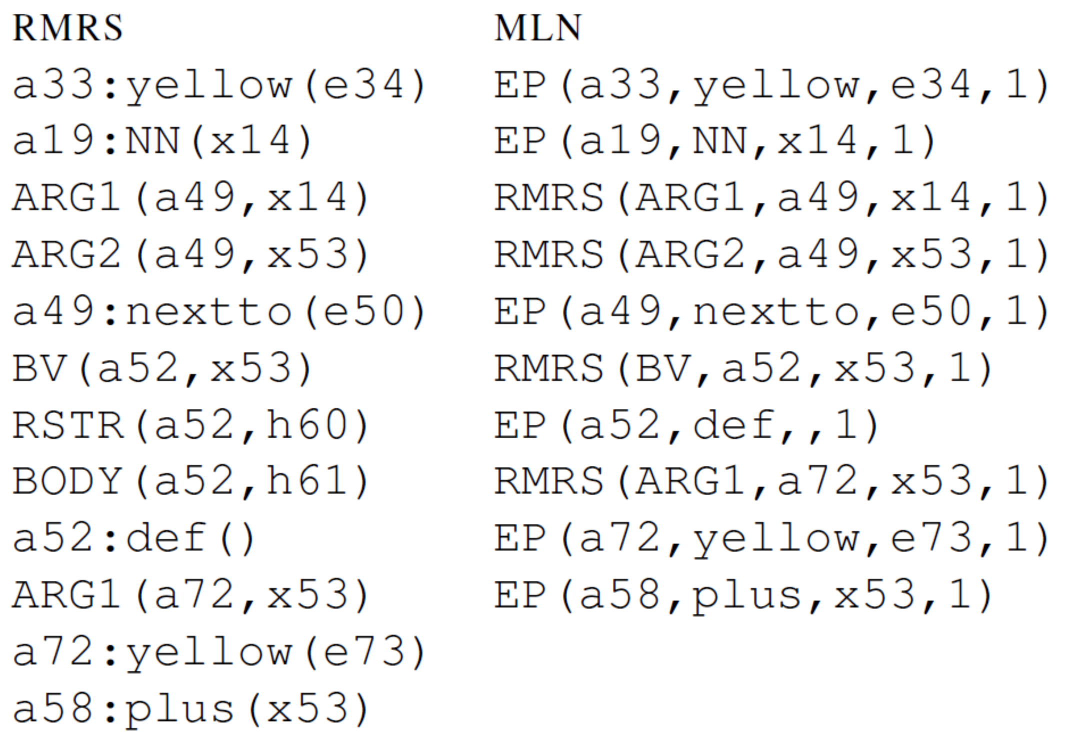
n	word	interpretation
1	*rotate*	action:rotate
2	*the*	...
3	*yellow*	argument:*yellow objects*
4	*piece*	argument:*yellow pieces*
5	*next*	...
6	*to*	...
7	*the*	...
8	*yellow*	argument:*yellow pieces by yellow objects*
9	*plus*	argument:*yellow piece next to unique yellow plus*
10	*clockwise*	option:clockwise

Incremental interpretation word-by-word for a 10 word utterance

Model


Context/World
Language/ RMRS
Context/ Discourse
MLN System
Prediction: action object result

RMRS to MLN Predicates



RMRS	MLN
`a33:yellow(e34)`	`EP(a33,yellow,e34,1)`
`a19:NN(x14)`	`EP(a19,NN,x14,1)`
`ARG1(a49,x14)`	`RMRS(ARG1,a49,x14,1)`
`ARG2(a49,x53)`	`RMRS(ARG2,a49,x53,1)`
`a49:nextto(e50)`	`EP(a49,nextto,e50,1)`
`BV(a52,x53)`	`RMRS(BV,a52,x53,1)`
`RSTR(a52,h60)`	`EP(a52,def,,1)`
`BODY(a52,h61)`	`RMRS(ARG1,a72,x53,1)`
`a52:def()`	`EP(a72,yellow,e73,1)`
`ARG1(a72,x53)`	`EP(a58,plus,x53,1)`
`a72:yellow(e73)`	
`a58:plus(x53)`	

Figure 4: RMRS and MLN for *yellow piece next to the yellow plus*

MLN

Contextual Information with linguistic information specified in RMRS

1 $EP(a1, a2, +w, a3, b) \Rightarrow Action(+a, b)$

2 $PrevAction(+a, b) \Rightarrow Action(+a, b)$

3 $EP(a1, a2, +w, a3, b)) \Rightarrow Option(+o, b)$

4 $PrevOption(+o, b) \Rightarrow Option(+o, b)$

5 $EP(a1, a2, +w, a3, b)) \wedge Property(p, +pr, b)$
$\Rightarrow Argument(p, b)$

6 $EP(a1, a2, w1, a3, b) \wedge RMRS(+t, a4, a3, b)$
$\wedge RMRS(+t, a4, a5, b) \wedge EP(a5, a6, w2, a5,)$
$\wedge Property(p, +pr, b) \Rightarrow Argument(p, b)$

Connecting linguistic information specified in RMRS to Visual scene properties

Performance

- Query predicates: *Action, Argument and Option*

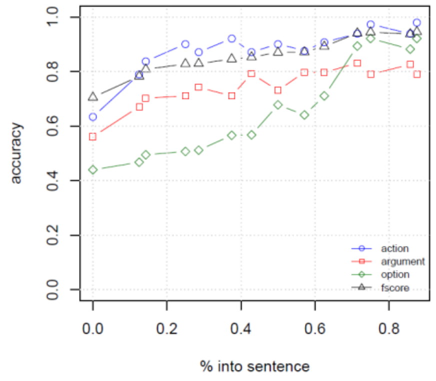


Figure 7: incremental accuracies

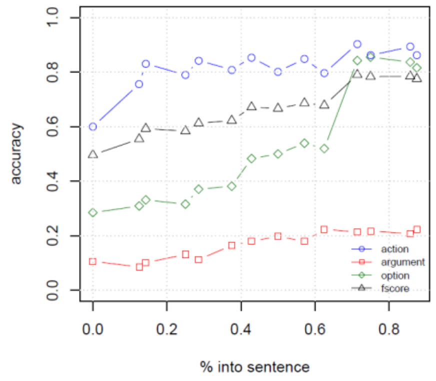
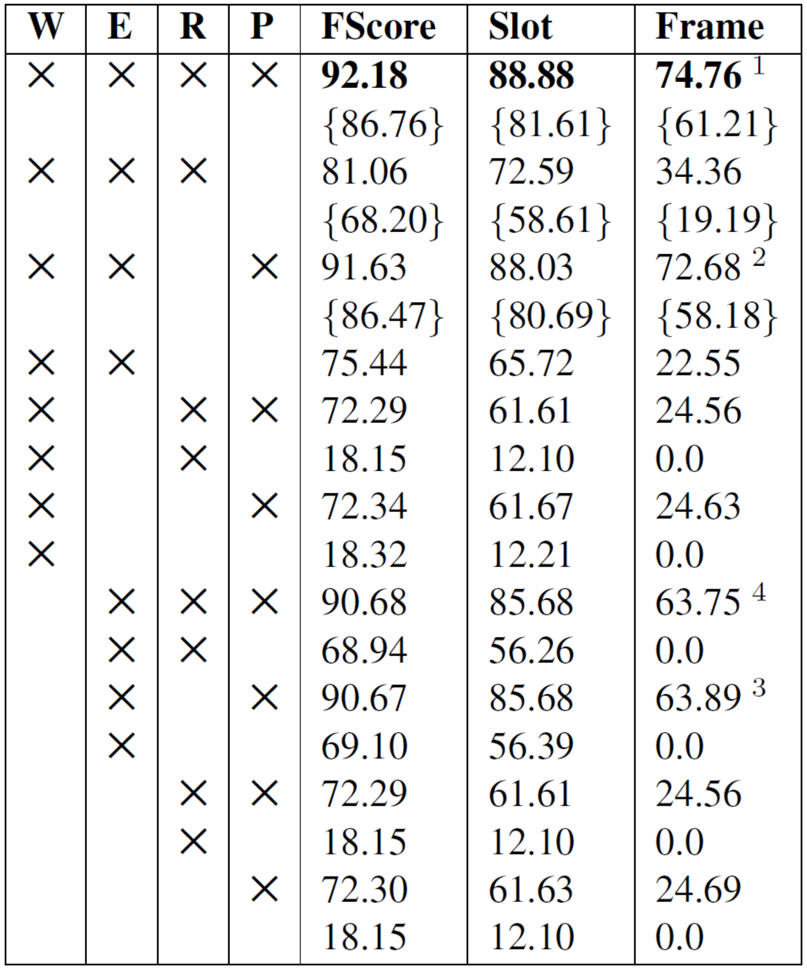


Figure 8: incremental accuracies, no discourse context

Performance

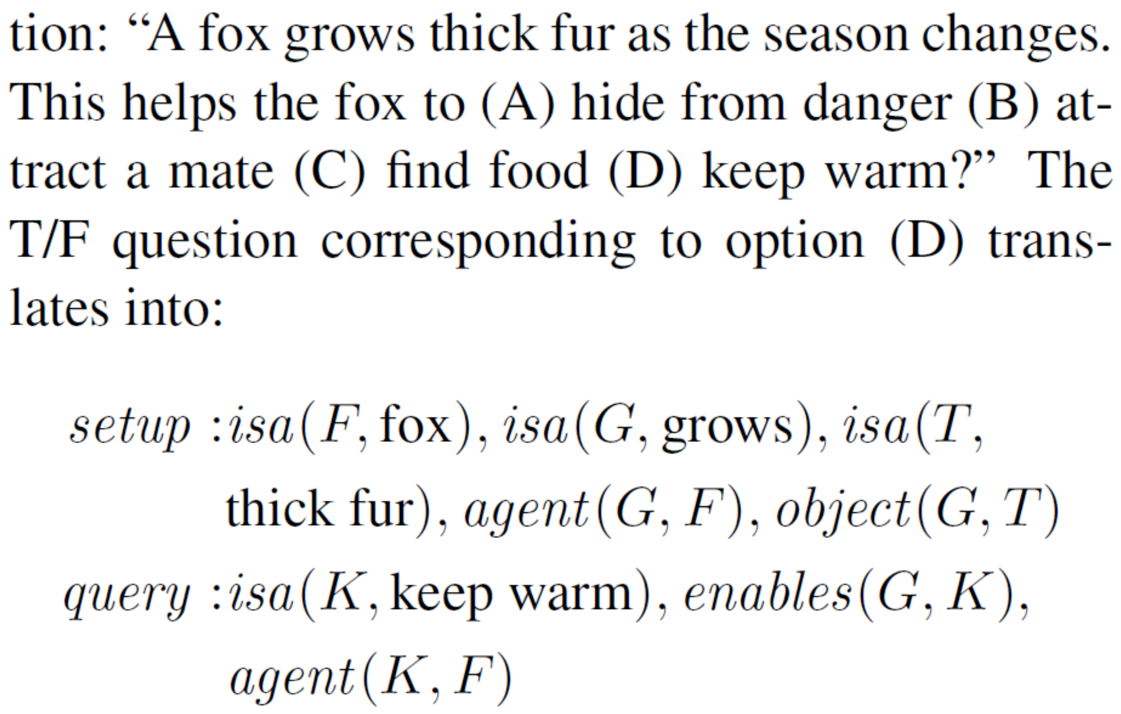


W	E	R	P	FScore	Slot	Frame
×	×	×	×	**92.18**	**88.88**	**74.76** [1]
				{86.76}	{81.61}	{61.21}
×	×	×		81.06	72.59	34.36
				{68.20}	{58.61}	{19.19}
×	×		×	91.63	88.03	72.68 [2]
				{86.47}	{80.69}	{58.18}
×	×			75.44	65.72	22.55
×		×	×	72.29	61.61	24.56
×		×		18.15	12.10	0.0
×			×	72.34	61.67	24.63
×				18.32	12.21	0.0
	×	×	×	90.68	85.68	63.75 [4]
	×	×		68.94	56.26	0.0
	×		×	90.67	85.68	63.89 [3]
	×			69.10	56.39	0.0
		×	×	72.29	61.61	24.56
		×		18.15	12.10	0.0
			×	72.30	61.63	24.69
				18.15	12.10	0.0

Table 2: Comparison of combinations using **W**orld, **E**Ps (words), **R**MRS and **P**revious context. Number in brackets are for tests on automatically transcribed speech.

Question Answering

- Task: Answer basic questions with knowledge extracted from 4-th grade text books and web text
  - Represent extracted knowledge as MLN formulas



tion: "A fox grows thick fur as the season changes. This helps the fox to (A) hide from danger (B) attract a mate (C) find food (D) keep warm?" The T/F question corresponding to option (D) translates into:

$$\begin{aligned} setup &: isa(F, \text{fox}), isa(G, \text{grows}), isa(T, \\ &\quad \text{thick fur}), agent(G, F), object(G, T) \\ query &: isa(K, \text{keep warm}), enables(G, K), \\ &\quad agent(K, F) \end{aligned}$$

Khot et al. '15

Question Answering

- Three different MLN formulations
- First-Order MLN: Directly encode extracted KB rules as MLN formulas
- Entity Resolution MLN: Encode the relationship between constants in the KB rules using concepts of entity resolution
- PRALINE MLN: Encode the MLN to perform probabilistic alignment in a graphs that encode the question and the KB

First-Order MLN

- Add if-then rules extracted from the text directly as MLN formulas

$$\begin{aligned}
&isa(g, \text{grow}), isa(a, \text{some animals}),\\
&isa(f, \text{thicker fur}), isa(w, \text{the winter}),\\
&agent(g, a), object(g, f), in(g, w)\\
&\ \Rightarrow \exists s, r : isa(s, \text{stays}), isa(r, \text{warm}),\\
&\qquad enables(g, s), agent(s, a), object(s, r)
\end{aligned}$$

- Soft Evidences: $entails(fox, some\ animals)$

First-Order MLN

- Semantic rules capture meaning of predicates
  - Every event has a unique agent, cause-effect relations etc.

- $\bigwedge_{q \in Q} q \Rightarrow Result$, compute $P(Result)$

- Inference: Reduce the ground network size by incrementally processing the MLN

Entity Resolution based MLN

- Rules over entity/event constants

$$agent(Grow, Animals), object(Grow, Fur) \Rightarrow$$
$$enables(Grow, StayWarm)$$

- Two events/entities are considered similar when tied to lexically similar strings
- Similar entities/events participate in similar relations w.r.t other entities/events

ER Based MLN

$$isa(x, s), entails(s, s') \rightarrow isa(x, s').$$

$$isa(x, s), isa(y, s) \rightarrow sameAs(x, y).$$

$$w: \ isa(x, s), !isa(y, s) \rightarrow !sameAs(x, y)$$

$$r(x, y), sameAs(y, z) \rightarrow r(x, z).$$

Any other predicate

Defines soft clusters or equivalence classes of entities/events

ER Based MLN

- Partial matching rules
  - For every extracted rule of the form $\bigwedge_i L_i \Rightarrow R$, rules of the form $L\_i \Rightarrow R$ helps in more flexible partial matching of the antecedent

- Pros
  - ER based MLN does not generate a massive ground network
  - Much more scalable than FO-MLN
- Cons
  - Fails on questions where same string is bound to distinct entities. E.g. “A student puts two plants, one in a sunny room and the other in a dark room”. Here the two plants are distinct entities but ER MLN treats it as the same.

Probabilistic Alignment and Inference

- Introduce a new unary predicate *holds* to capture probability of a string constant being true given that the setup for the query is true and the KB rules hold

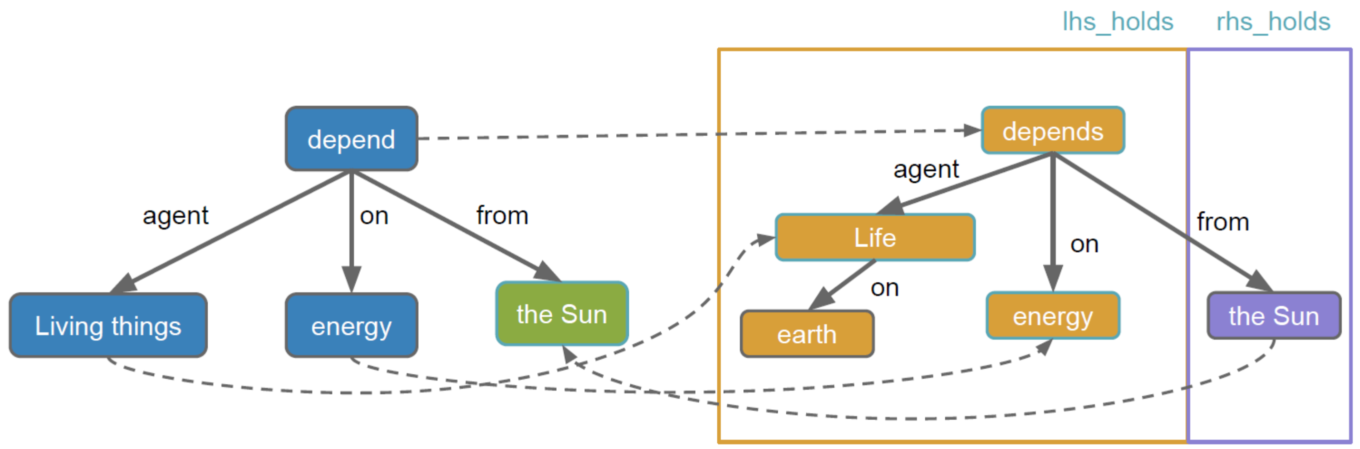


Figure 1: KB rule and question as a graph where blue:*setup*; green:*query*; orange:*antecedent*; purple:*consequent*; dotted lines: alignments. *lhsHolds* combines individual probabilities of *antecedent* nodes and *rhsHolds* captures the probability of the *consequent*.

Probabilistic Alignment and Inference

- Evidence
  - $node(nodeid)$
  - $edge(nodeid, nodeid, label)$
  - $setup(nodeid)$
  - $query(nodeid)$
  - $inLHS(nodeid)$ (antecedent of KB rules)
  - $inRHS(nodeid)$ (consequent of KB rules)

- Graph alignment rules
  - $aligns(x, y), edge(x, u, r), edge(y, v, s) \Rightarrow aligns(u, v)$

Probabilistic Alignment and Inference

- Inference Rules

$$holds(x), aligns(x,y) \Rightarrow holds(y)$$

$$w : holds(x), inLhs(x,r) \Rightarrow lhsHolds(r)$$

$$w : !holds(x), inLhs(x,r) \Rightarrow !lhsHolds(r)$$

$$lhsHolds(r) \Rightarrow rhsHolds(r).$$

$$rhsHolds(r), inRhs(r,x) \Rightarrow holds(x).$$

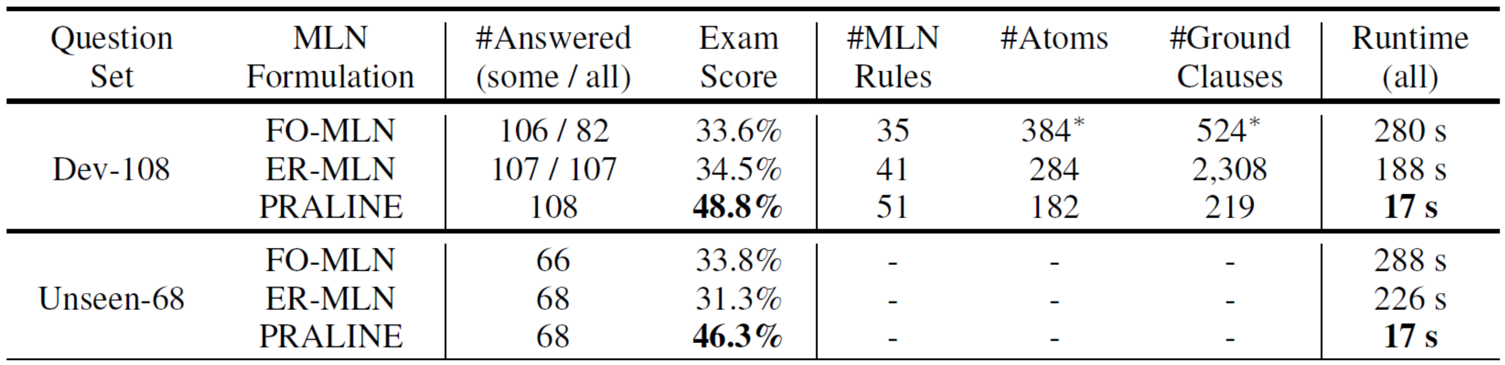
Probabilistic Alignment and Inference

- Acyclic Inference

$$w_p : proves(x, y), holds(x) \Rightarrow holds(y)$$
$$w_a : aligns(x, y) \Rightarrow proves(x, y)$$

$$proves(x, y), inrhs(x, r1), inlhs(y, r2) \\ \Rightarrow ruleProves(r1, r2).$$

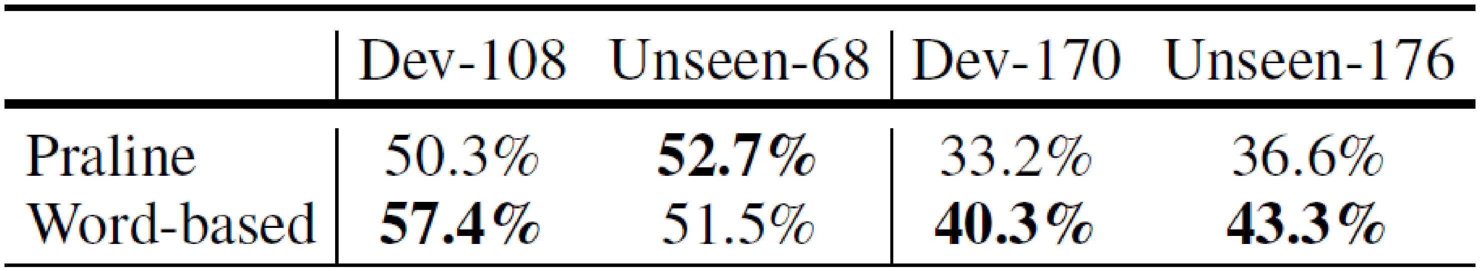
Performance (4th grade question answering)



Question Set	MLN Formulation	#Answered (some / all)	Exam Score	#MLN Rules	#Atoms	#Ground Clauses	Runtime (all)
Dev-108	FO-MLN	106 / 82	33.6%	35	384*	524*	280 s
	ER-MLN	107 / 107	34.5%	41	284	2,308	188 s
	PRALINE	108	**48.8%**	51	182	219	**17 s**
Unseen-68	FO-MLN	66	33.8%	-	-	-	288 s
	ER-MLN	68	31.3%	-	-	-	226 s
	PRALINE	68	**46.3%**	-	-	-	**17 s**

Performance (comparison with a simpler word based approach)

- Word based model
  - Calculate entailment score for words in KB rule and words in question



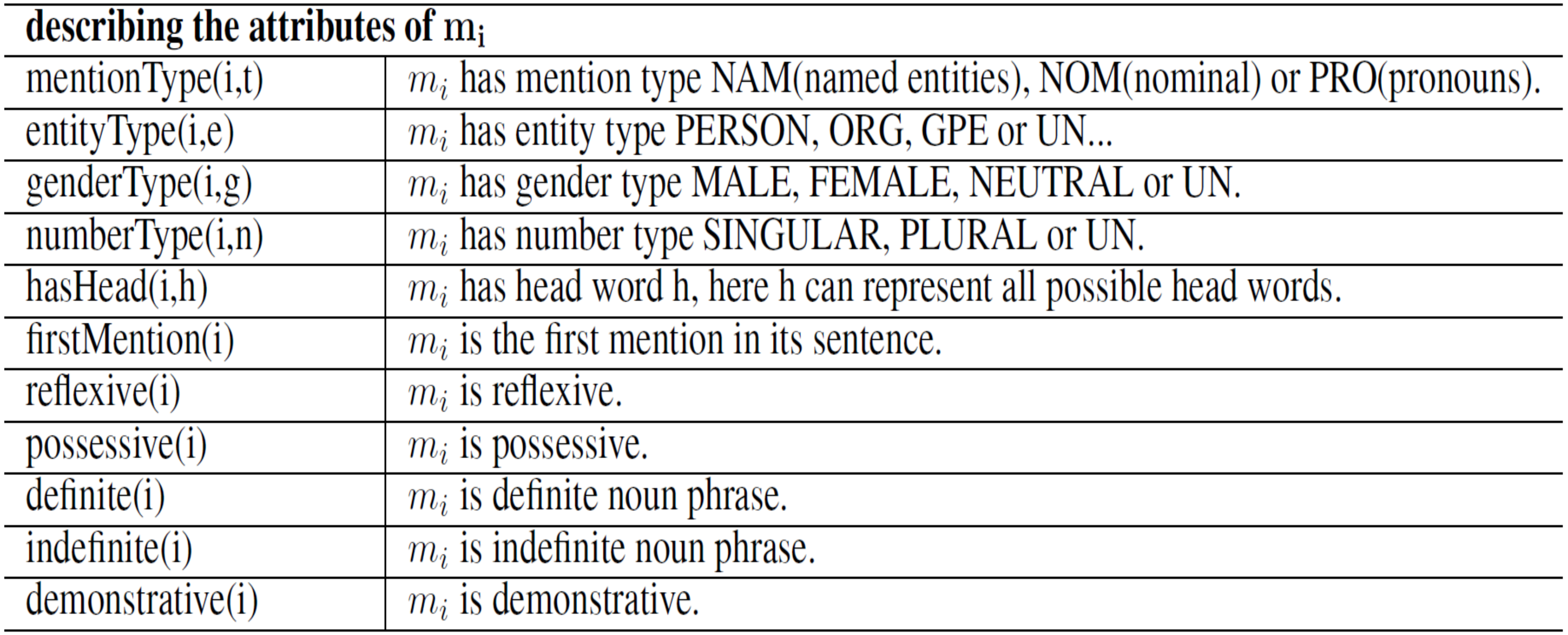
	Dev-108	Unseen-68	Dev-170	Unseen-176
Praline	50.3%	**52.7%**	33.2%	36.6%
Word-based	**57.4%**	51.5%	**40.3%**	**43.3%**

Joint Inference for Coreference Resolution

- Task: Cluster together mentions that refer to the same entity

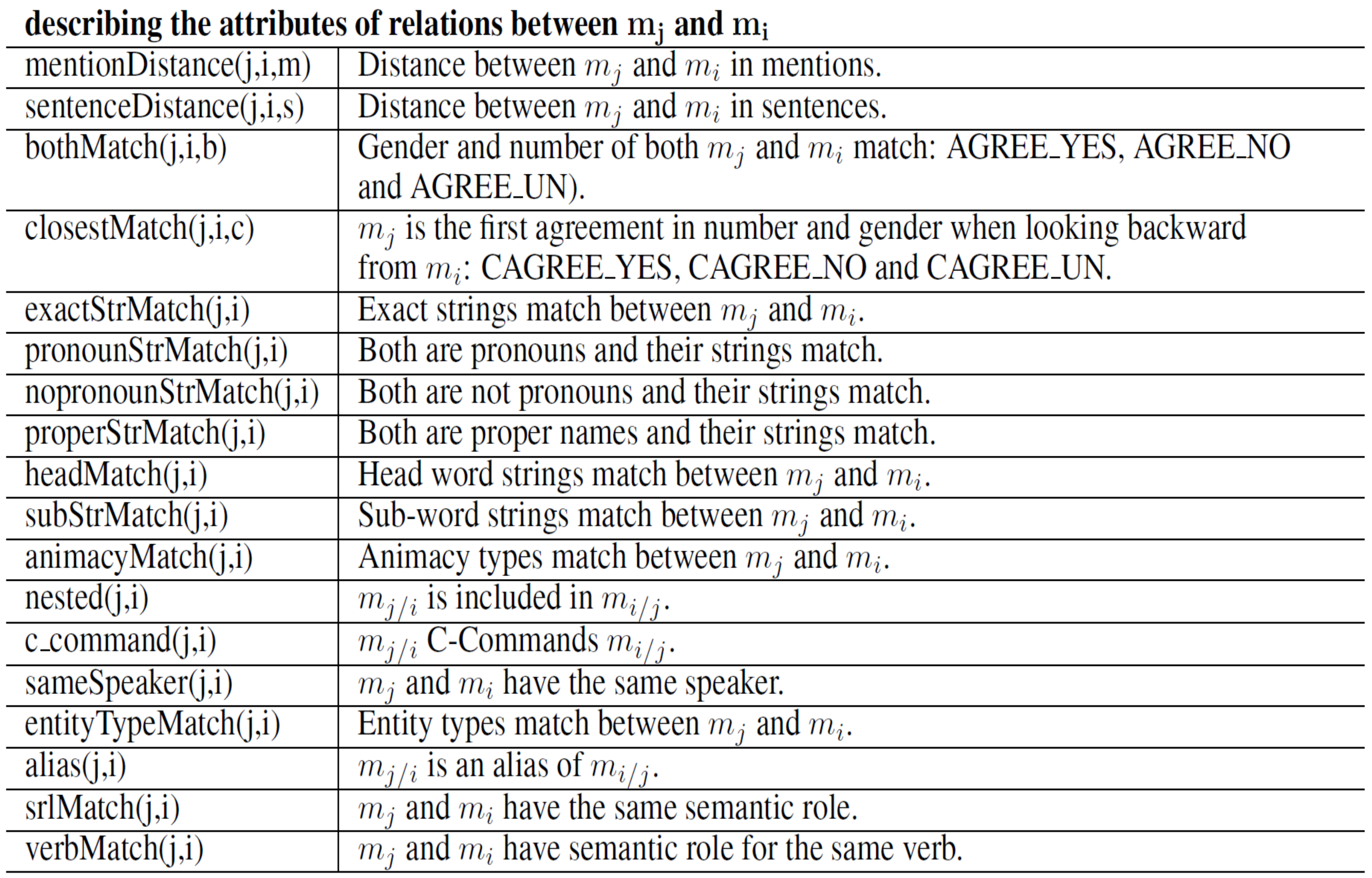
- Three main tasks in the pipeline
  - Mention Detection of anaphoric noun phrases
  - Pairwise classification of whether detected mentions refer to the same entity
  - Mention clustering resolves conflicts and generates the mention clusters referring to a common entity

MLN observed predicates



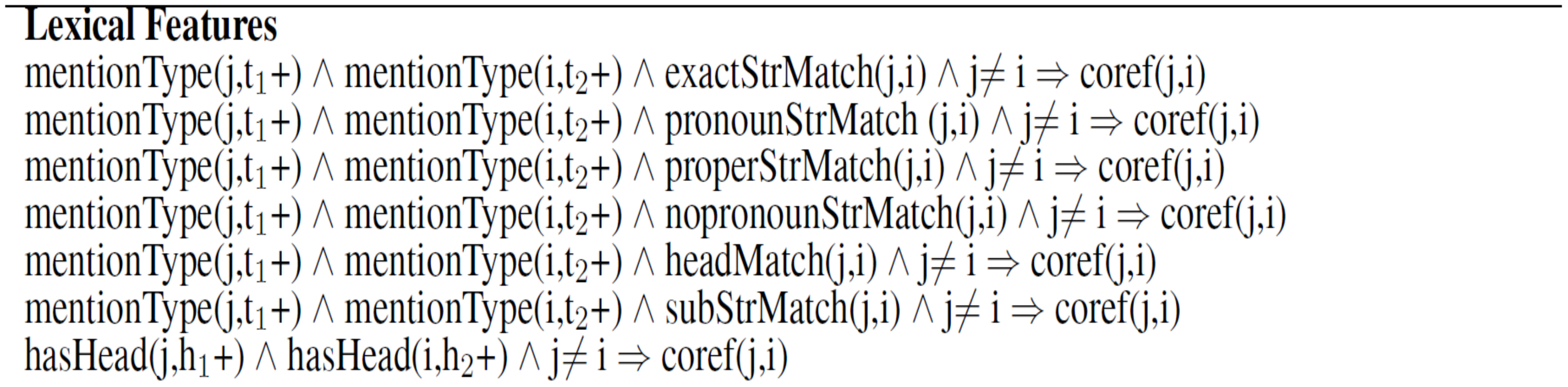
describing the attributes of m_i	
mentionType(i,t)	m_i has mention type NAM(named entities), NOM(nominal) or PRO(pronouns).
entityType(i,e)	m_i has entity type PERSON, ORG, GPE or UN...
genderType(i,g)	m_i has gender type MALE, FEMALE, NEUTRAL or UN.
numberType(i,n)	m_i has number type SINGULAR, PLURAL or UN.
hasHead(i,h)	m_i has head word h, here h can represent all possible head words.
firstMention(i)	m_i is the first mention in its sentence.
reflexive(i)	m_i is reflexive.
possessive(i)	m_i is possessive.
definite(i)	m_i is definite noun phrase.
indefinite(i)	m_i is indefinite noun phrase.
demonstrative(i)	m_i is demonstrative.

MLN observed predicates



describing the attributes of relations between $\mathrm{m_j}$ and $\mathrm{m_i}$	
mentionDistance(j,i,m)	Distance between m_j and m_i in mentions.
sentenceDistance(j,i,s)	Distance between m_j and m_i in sentences.
bothMatch(j,i,b)	Gender and number of both m_j and m_i match: AGREE_YES, AGREE_NO and AGREE_UN).
closestMatch(j,i,c)	m_j is the first agreement in number and gender when looking backward from m_i: CAGREE_YES, CAGREE_NO and CAGREE_UN.
exactStrMatch(j,i)	Exact strings match between m_j and m_i.
pronounStrMatch(j,i)	Both are pronouns and their strings match.
nopronounStrMatch(j,i)	Both are not pronouns and their strings match.
properStrMatch(j,i)	Both are proper names and their strings match.
headMatch(j,i)	Head word strings match between m_j and m_i.
subStrMatch(j,i)	Sub-word strings match between m_j and m_i.
animacyMatch(j,i)	Animacy types match between m_j and m_i.
nested(j,i)	$m_{j/i}$ is included in $m_{i/j}$.
c_command(j,i)	$m_{j/i}$ C-Commands $m_{i/j}$.
sameSpeaker(j,i)	m_j and m_i have the same speaker.
entityTypeMatch(j,i)	Entity types match between m_j and m_i.
alias(j,i)	$m_{j/i}$ is an alias of $m_{i/j}$.
srlMatch(j,i)	m_j and m_i have the same semantic role.
verbMatch(j,i)	m_j and m_i have semantic role for the same verb.

Local Formulas



Lexical Features

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{exactStrMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{pronounStrMatch}\ (j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

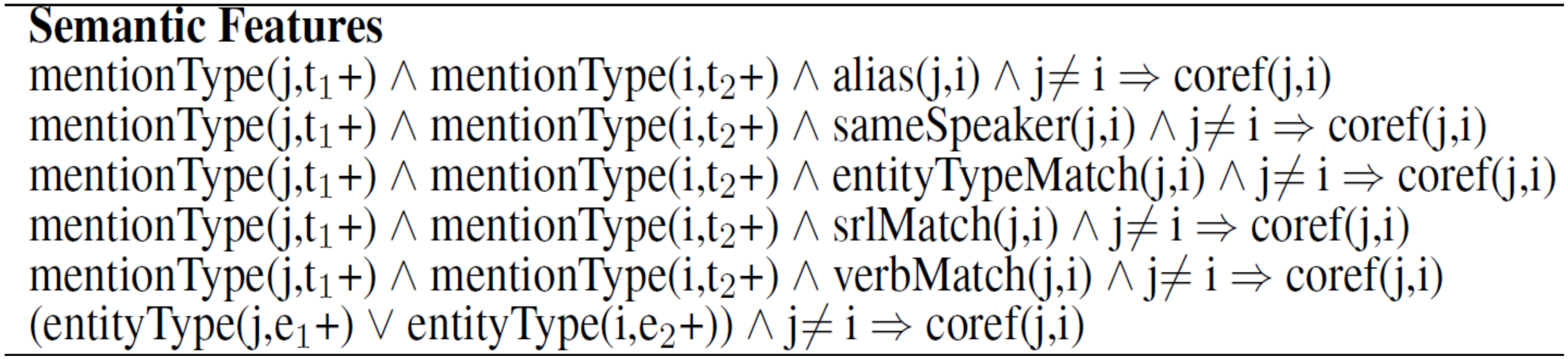
$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{properStrMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{nopronounStrMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{headMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{subStrMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{hasHead}(j,h_1+) \wedge \text{hasHead}(i,h_2+) \wedge j \neq i \Rightarrow \text{coref}(j,i)$



Semantic Features

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{alias}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{sameSpeaker}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{entityTypeMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{srlMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{verbMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{entityType}(j,e_1+) \vee \text{entityType}(i,e_2+)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

Local Formulas

Grammatical Features

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{genderType}(j,g_1+) \wedge \text{genderType}(i,g_2+) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{numberType}(j,n_1+) \wedge \text{numberType}(i,n_2+) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{bothMatch}(j,i,b+) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{closestMatch}(j,i,c+) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{animacyMatch}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{nested}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{c\_command}(j,i) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{mentionType}(j,t_1+) \vee \text{mentionType}(i,t_2+)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{reflexive}(j) \vee \text{reflexive}(i)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{possessive}(j) \vee \text{possessive}(i)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{definite}(j) \vee \text{definite}(i)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{indefinite}(j) \vee \text{indefinite}(i)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{demonstrative}(j) \vee \text{demonstrative}(i)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

Distance and position Features

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{sentenceDistance}(j,i,s+) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$\text{mentionType}(j,t_1+) \wedge \text{mentionType}(i,t_2+) \wedge \text{mentionDistance}\ (j,i,m+) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

$(\text{firstMention}(j) \vee \text{firstMention}(i)) \wedge j \neq i \Rightarrow \text{coref}(j,i)$

Global Features

Best-First constraint:

$$coref(j,i) \Rightarrow \neg coref(k,i) \quad \forall j,k < i(k \neq j)$$

Transitivity constraint:

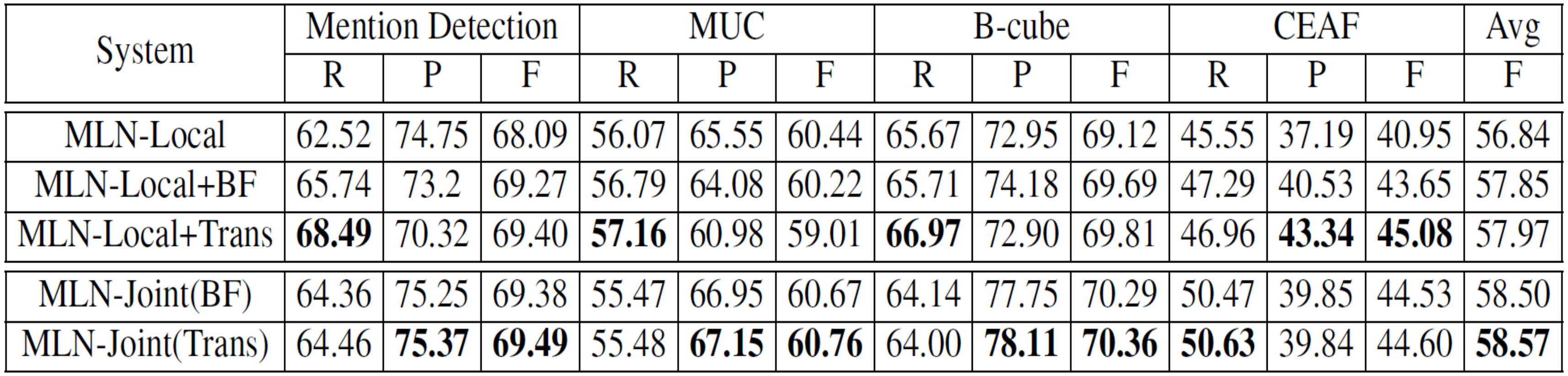
$$coref(j,k) \wedge coref(k,i) \wedge j < k < i \Rightarrow coref(j,i) \quad (3)$$

$$coref(j,k) \wedge coref(j,i) \wedge j < k < i \Rightarrow coref(k,i) \quad (4)$$

$$coref(j,i) \wedge coref(k,i) \wedge j < k < i \Rightarrow coref(j,k) \quad (5)$$

Performance on CoNLL-11 shard task

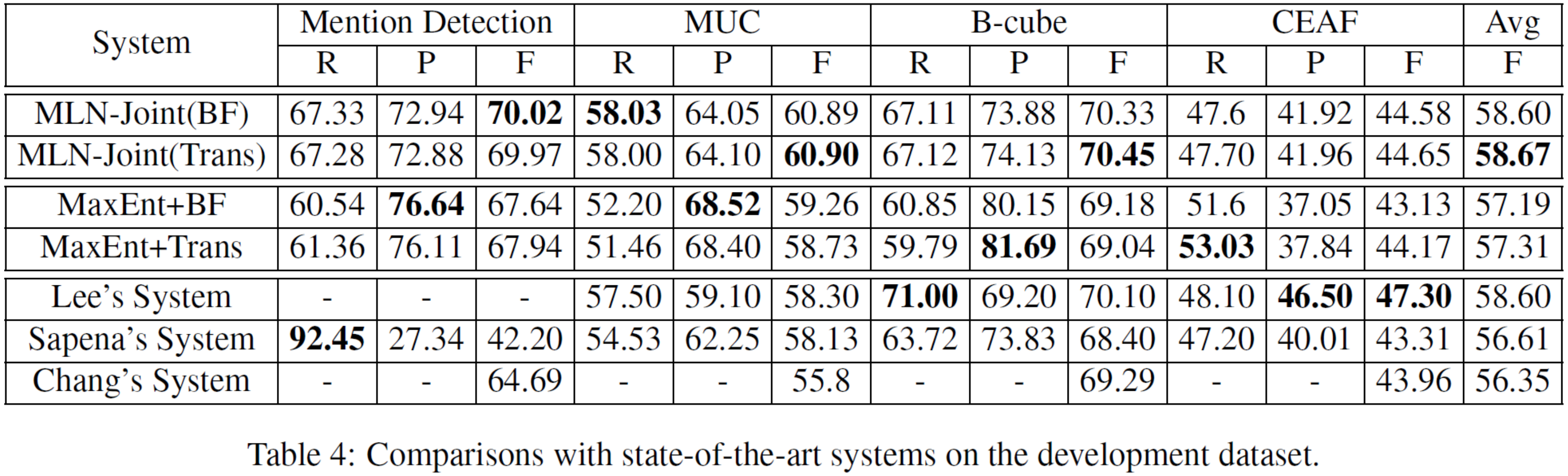
- Online parameter learning using MIRA
- Inference using ILPs



System	Mention Detection			MUC			B-cube			CEAF			Avg
	R	P	F	R	P	F	R	P	F	R	P	F	F
MLN-Local	62.52	74.75	68.09	56.07	65.55	60.44	65.67	72.95	69.12	45.55	37.19	40.95	56.84
MLN-Local+BF	65.74	73.2	69.27	56.79	64.08	60.22	65.71	74.18	69.69	47.29	40.53	43.65	57.85
MLN-Local+Trans	**68.49**	70.32	69.40	**57.16**	60.98	59.01	**66.97**	72.90	69.81	46.96	**43.34**	**45.08**	57.97
MLN-Joint(BF)	64.36	75.25	69.38	55.47	66.95	60.67	64.14	77.75	70.29	50.47	39.85	44.53	58.50
MLN-Joint(Trans)	64.46	**75.37**	**69.49**	55.48	**67.15**	**60.76**	64.00	**78.11**	**70.36**	**50.63**	39.84	44.60	**58.57**

Table 3: Comparison between different MLN-based systems, using 10-fold cross validation on the training dataset.

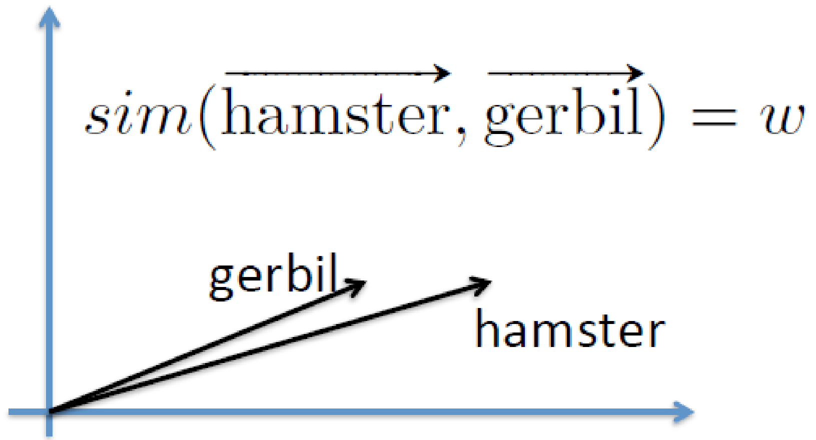
Performance on CoNLL-11 shard task



System	Mention Detection			MUC			B-cube			CEAF			Avg
	R	P	F	R	P	F	R	P	F	R	P	F	F
MLN-Joint(BF)	67.33	72.94	**70.02**	**58.03**	64.05	60.89	67.11	73.88	70.33	47.6	41.92	44.58	58.60
MLN-Joint(Trans)	67.28	72.88	69.97	58.00	64.10	**60.90**	67.12	74.13	**70.45**	47.70	41.96	44.65	**58.67**
MaxEnt+BF	60.54	**76.64**	67.64	52.20	**68.52**	59.26	60.85	80.15	69.18	51.6	37.05	43.13	57.19
MaxEnt+Trans	61.36	76.11	67.94	51.46	68.40	58.73	59.79	**81.69**	69.04	**53.03**	37.84	44.17	57.31
Lee's System	-	-	-	57.50	59.10	58.30	**71.00**	69.20	70.10	48.10	**46.50**	**47.30**	58.60
Sapena's System	**92.45**	27.34	42.20	54.53	62.25	58.13	63.72	73.83	68.40	47.20	40.01	43.31	56.61
Chang's System	-	-	64.69	-	-	55.8	-	-	69.29	-	-	43.96	56.35

Table 4: Comparisons with state-of-the-art systems on the development dataset.

Semantic Similarity

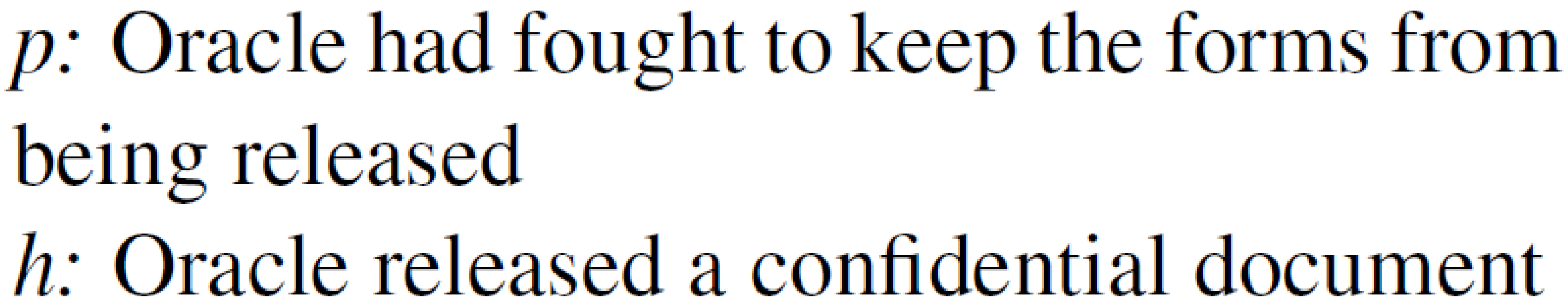

$$sim(\overrightarrow{\text{hamster}}, \overrightarrow{\text{gerbil}}) = w$$

$$x\left(\text{hamster}(x) \rightarrow \text{gerbil}(x)\right) \mid f(w)$$

Belthagyet al. '13

Tasks

Recognizing Textual Entailment (RTE)



p: Oracle had fought to keep the forms from being released
h: Oracle released a confidential document

Does *h* entail *p*?

Tasks

Semantic Textual Similarity (STS)

S_1: A man is slicing a cucumber.
S_2: A man is slicing a zucchini.

S_1: A boy is riding a bicycle.
S_2: A little boy is riding a bike.

Score similarity between 0 and 5

MLNs for RTE and STS

- Both RTE and STS can be modeled as probabilistic entailment in Markov logic
  - Given sentences $S_1 and\ S_2$, RTE is the probability of $S_1 \vDash S_2$
  - Given sentences $S_1 and\ S_2$, STS is the average probability of $S_1 \vDash S_2$ and $S_2 \vDash S_1$

MLNs

Uses Boxer to derive logic form of sentences

S_1: A man is slicing a cucumber.
S_2: A man is slicing a zucchini.

$$\exists x_0, e_1, x_2 \big(man(x_0) \wedge slice(e_1) \wedge Agent(e_1, x_0) \wedge cucumber(x_2) \wedge Patient(e_1, x_2)\big)$$

$$man(x) \wedge slice(e) \wedge Agent(e, x) \wedge zucchini(y) \wedge Patient(e, y) \rightarrow result()$$

$$P(S_1 \vDash S_2) = P(Result)$$

Word based Inference Rules

- Create weighted rules for all pairs of words in the two sentences

$$cucumber(x) \rightarrow zucchini(x) \mid wt(\text{cuc.}, \text{zuc.})$$

$$wt(a,b) = log(\frac{\cos(\vec{a}, \vec{b})}{1 - \cos(\vec{a}, \vec{b})}) - prior$$

Phrase based Inference Rules

S_1: A boy is riding a bicycle.
S_2: A little boy is riding a bike.

Boy and "little boy" need to be linked in the inference rule

- Vectorize a phrase using either vector addition or multiplication treating phrase words as vector components. The weight of the inference rule is then compute similar to word based rules

Missing Phrases

S_1: A man is driving.
S_2: A man is driving a car.

$$S_1: \ \exists x_0, e_1 \big(man(x_0) \land agent(x_0, e_1) \land drive(e_1)\big)$$

$$S_2: \ \exists x_0, e_1, x_2 \big(man(x_0) \land agent(x_0, e_1) \land drive(e_1) \land patient(e_1, x_2) \land car(x_2)\big)$$

$P(S_1 \vDash S_2) = 0$ due to missing phrase.
For RTE this is fine, but for STS, we need to replace the deterministic and with something more flexible

Average Evidence Combiner

Divide MLN formula into smaller mini-clauses and average the probabilities (Natarajan et al. '10)

$$man(x_0) \wedge agent(x_0, e_1) \rightarrow result(x_0, e_1, x_2) \mid w$$

$$drive(e_1) \wedge agent(x_0, e_1) \rightarrow result(x_0, e_1, x_2) \mid w$$

$$drive(e_1) \wedge patient(e_1, x_2) \rightarrow result(x_0, e_1, x_2) \mid w$$

$$car(x_2) \wedge patient(e_1, x_2) \rightarrow result(x_0, e_1, x_2) \mid w$$

$$w = \frac{1}{n} \times (log(\frac{p}{1-p}) - prior)$$

$P(Result) = p$ if the antecedents of all mini-clauses are satisfied

Average Evidence Combiner

$$man(x) \wedge agent(x, e) \rightarrow result() \mid w$$

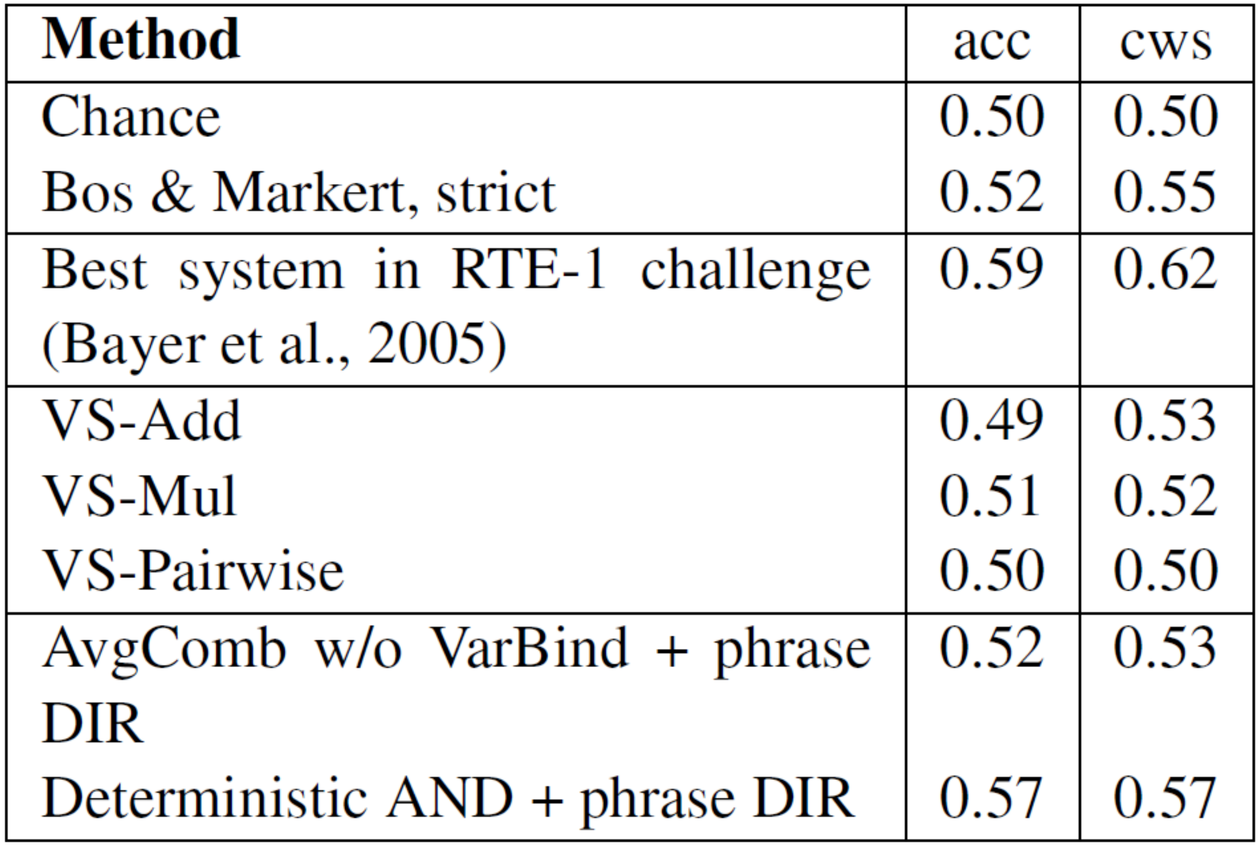
$$drive(e) \wedge agent(x, e) \rightarrow result() \mid w$$

$$drive(e) \wedge patient(e, x) \rightarrow result() \mid w$$

$$car(x) \wedge patient(e, x) \rightarrow result() \mid w$$

- Removing variable binding improves inference efficiency.
- Less powerful: Does not differentiate between "A man is walking and a woman is running" and "A man is running and a woman is walking"
- Works well in practice for STS

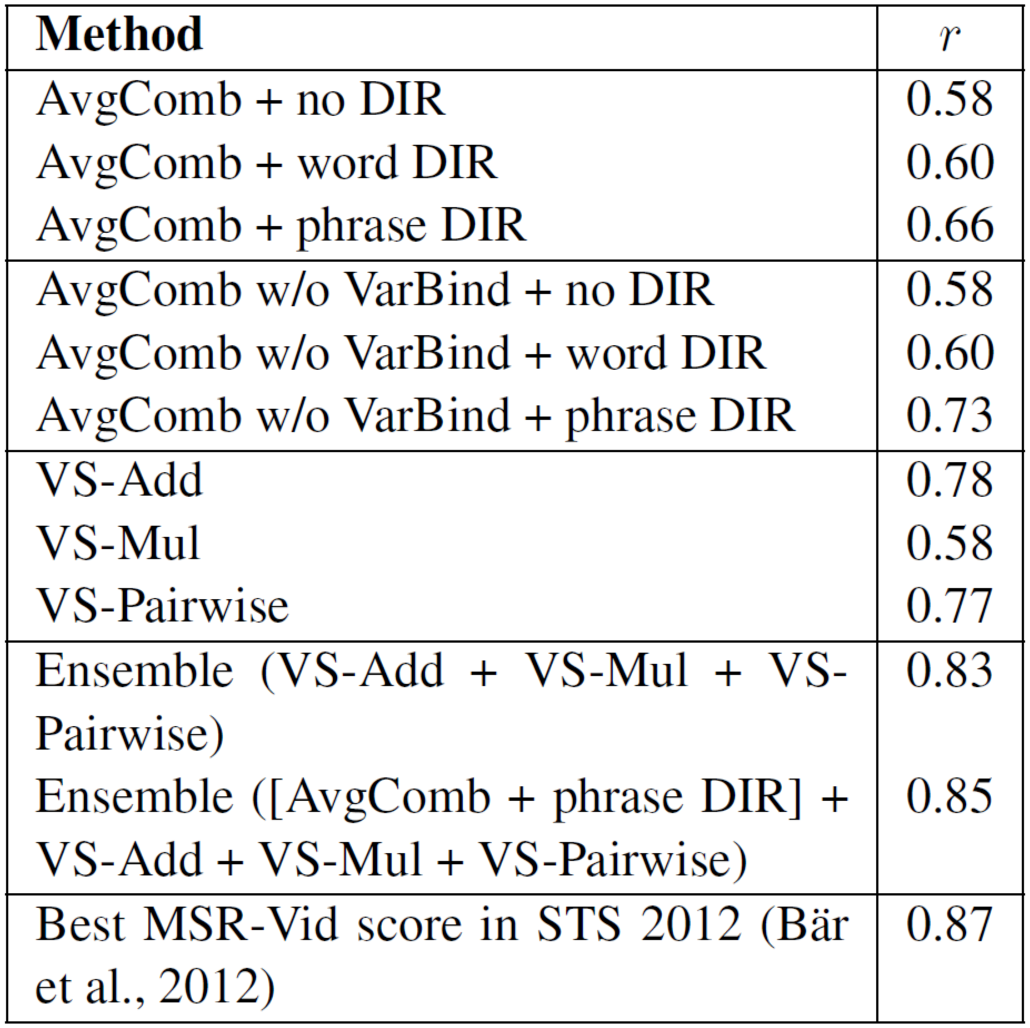
Performance



Method	acc	cws
Chance	0.50	0.50
Bos & Markert, strict	0.52	0.55
Best system in RTE-1 challenge (Bayer et al., 2005)	0.59	0.62
VS-Add	0.49	0.53
VS-Mul	0.51	0.52
VS-Pairwise	0.50	0.50
AvgComb w/o VarBind + phrase DIR	0.52	0.53
Deterministic AND + phrase DIR	0.57	0.57

Table 1: Results on the RTE-1 Test Set.

Performance



Method	r
AvgComb + no DIR	0.58
AvgComb + word DIR	0.60
AvgComb + phrase DIR	0.66
AvgComb w/o VarBind + no DIR	0.58
AvgComb w/o VarBind + word DIR	0.60
AvgComb w/o VarBind + phrase DIR	0.73
VS-Add	0.78
VS-Mul	0.58
VS-Pairwise	0.77
Ensemble (VS-Add + VS-Mul + VS-Pairwise)	0.83
Ensemble ([AvgComb + phrase DIR] + VS-Add + VS-Mul + VS-Pairwise)	0.85
Best MSR-Vid score in STS 2012 (Bär et al., 2012)	0.87

Table 2: Results on the STS video dataset.

Fine-Grained Sentiment Analysis

- A single polarity of reviews alone is sometimes insufficient
  - "Despite the pretty design I would never recommend it, because the sound quality is unacceptable"
  - Has both positive and negative polarity statements
  - Need Fine grained sentiment analysis

s1 = Despite the pretty design,
s2 = I would never recommend it
s3 = because the sound quality is unacceptable,
CONCESSION(S1,S2)
CAUSE-EXPLANATION-EVIDENCE(s2,s3)

Zirnet al. '11

Fine-Grained Sentiment Analysis

The set of constants correspond to discourse segments

$$
\begin{aligned}
\forall x : \neg positive(x) &\quad\Rightarrow\quad negative(x) \\
\forall x : negative(x) &\quad\Rightarrow\quad \neg positive(x)
\end{aligned}
$$

$$
\begin{aligned}
\forall x : positive\_source_{\ell}(x) &\quad\Leftrightarrow\quad positive(x) \\
\forall x : negative\_source_{\ell}(x) &\quad\Leftrightarrow\quad negative(x)
\end{aligned}
$$

Prior Features

Neighborhood Relations

Precedence relationships with previous segment

$$\forall x, y : pre(x, y) \wedge positive(x) \Rightarrow positive(y)$$

$$\forall x, y : pre(x, y) \wedge negative(x) \Rightarrow negative(y)$$

Discourse Relations

Work considers two types of relations across segments: CONTRAST and NO-CONTRAST

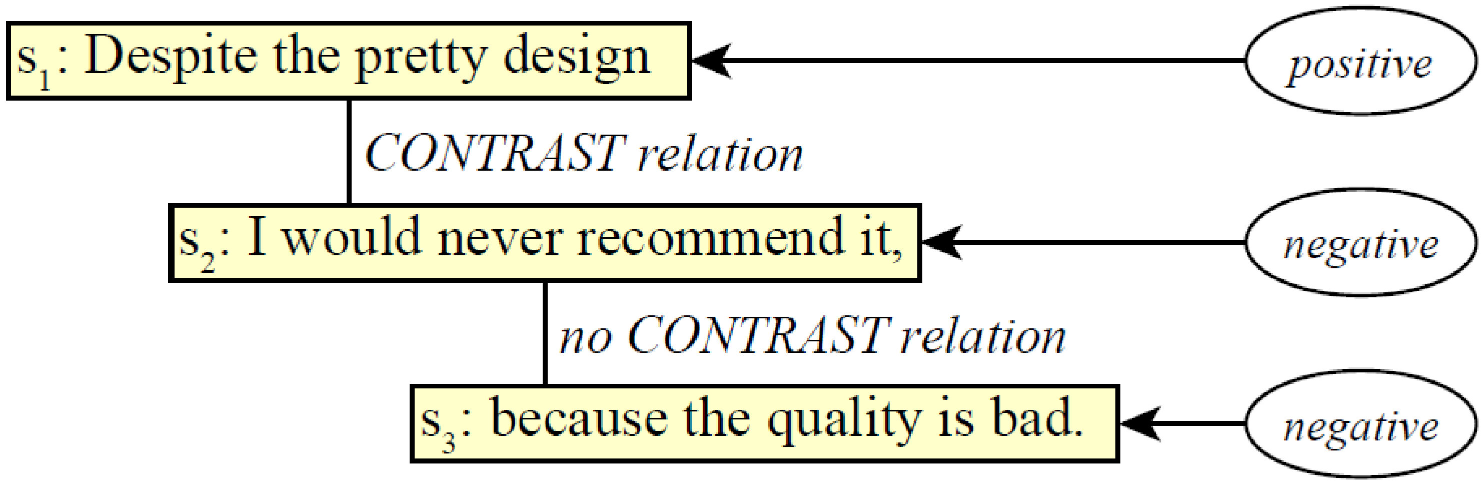

s_1: Despite the pretty design
CONTRAST relation
s_2: I would never recommend it,
no CONTRAST relation
s_3: because the quality is bad.
positive
negative
negative

Figure 1: Sentiment polarities of and discourse relations between the segments of a sentence.

Discourse Relations

Connect polarities with discourse relations

$$
\begin{aligned}
\forall x, y : contrast(x, y) \;\wedge\; positive(x) &\quad \Rightarrow negative(y) \\
\forall x, y : contrast(x, y) \;\wedge\; negative(x) &\quad \Rightarrow positive(y) \\
\forall x, y : ncontrast(x, y) \wedge positive(x) &\quad \Rightarrow positive(y) \\
\forall x, y : ncontrast(x, y) \wedge negative(x) &\quad \Rightarrow negative(y)
\end{aligned}
$$

Information Extraction (BioNLP)

HOIL-1L interacting protein (HOIP) a ubiquitin ligase that can catalyze the assembly of linear polyubiquitin chains is recruited to DC40 in a TRAF2 dependent manner.

Event	Type	Trigger Word	Arg-1	Arg-1 Role	Arg-2	Arg-2 Role
E-1	Binding	recruited	HOIP	Theme	DC40	Theme
E-2	Regulation	dependent	E-1	Theme	TRAF2	Cause

Venugopal et al. '14

Biomedical Event Extraction

Query	Hidden	Evidence
$TriggerType(sid, tid, ttype!)$	$Simple(sid, tid)$	$Word(sid, tid, word)$
$ArgumentRole(sid, aid, tid, arole!)$	$Regulation(sid, tid)$	$Dependency(sid, aid, tid, dtype)$

$$\neg TriggerType(i, j, None) \Rightarrow \exists k\ ArgumentRole(i, k, j, Theme)$$

$$Simple(i, j) \Rightarrow \nexists k\ ArgumentRole(i, k, j, Cause)$$

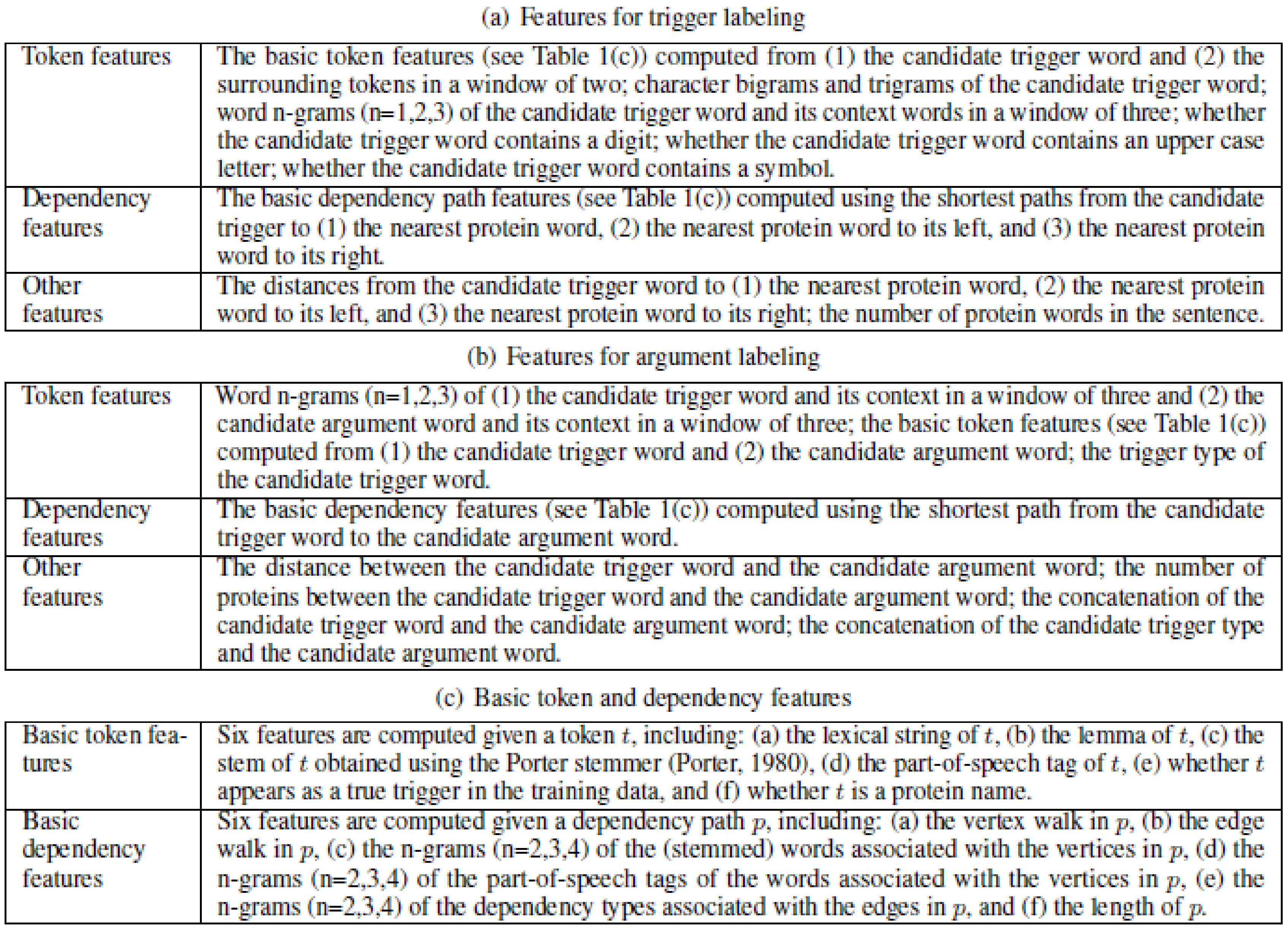
$$TriggerType(i, j, None) \Leftrightarrow ArgumentRole(i, k, j, None)$$

$$\neg ArgumentRole(i, k, j, None) \wedge TriggerType(i, k, None) \Rightarrow Regulation(i, j)$$

$$Simple(i, j) \Leftrightarrow TriggerType(i, j, GeneExpression) \vee TriggerType(i, j, Phosphorylation) \ldots$$

$$Regulation(i, j) \Leftrightarrow TriggerType(i, j, Postive) \vee TriggerType(i, j, Negative) \ldots$$

$$Word(i, j, +w) \wedge TriggerType(i, j, +t) \wedge Depdency(i, k, j, +d) \wedge ArgumentRole(i, k, j, +a)$$



(a) Features for trigger labeling

Token features	The basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the surrounding tokens in a window of two; character bigrams and trigrams of the candidate trigger word; word n-grams (n=1,2,3) of the candidate trigger word and its context words in a window of three; whether the candidate trigger word contains a digit; whether the candidate trigger word contains an upper case letter; whether the candidate trigger word contains a symbol.
Dependency features	The basic dependency path features (see Table 1(c)) computed using the shortest paths from the candidate trigger to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right.
Other features	The distances from the candidate trigger word to (1) the nearest protein word, (2) the nearest protein word to its left, and (3) the nearest protein word to its right; the number of protein words in the sentence.

(b) Features for argument labeling

Token features	Word n-grams (n=1,2,3) of (1) the candidate trigger word and its context in a window of three and (2) the candidate argument word and its context in a window of three; the basic token features (see Table 1(c)) computed from (1) the candidate trigger word and (2) the candidate argument word; the trigger type of the candidate trigger word.
Dependency features	The basic dependency features (see Table 1(c)) computed using the shortest path from the candidate trigger word to the candidate argument word.
Other features	The distance between the candidate trigger word and the candidate argument word; the number of proteins between the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger word and the candidate argument word; the concatenation of the candidate trigger type and the candidate argument word.

(c) Basic token and dependency features

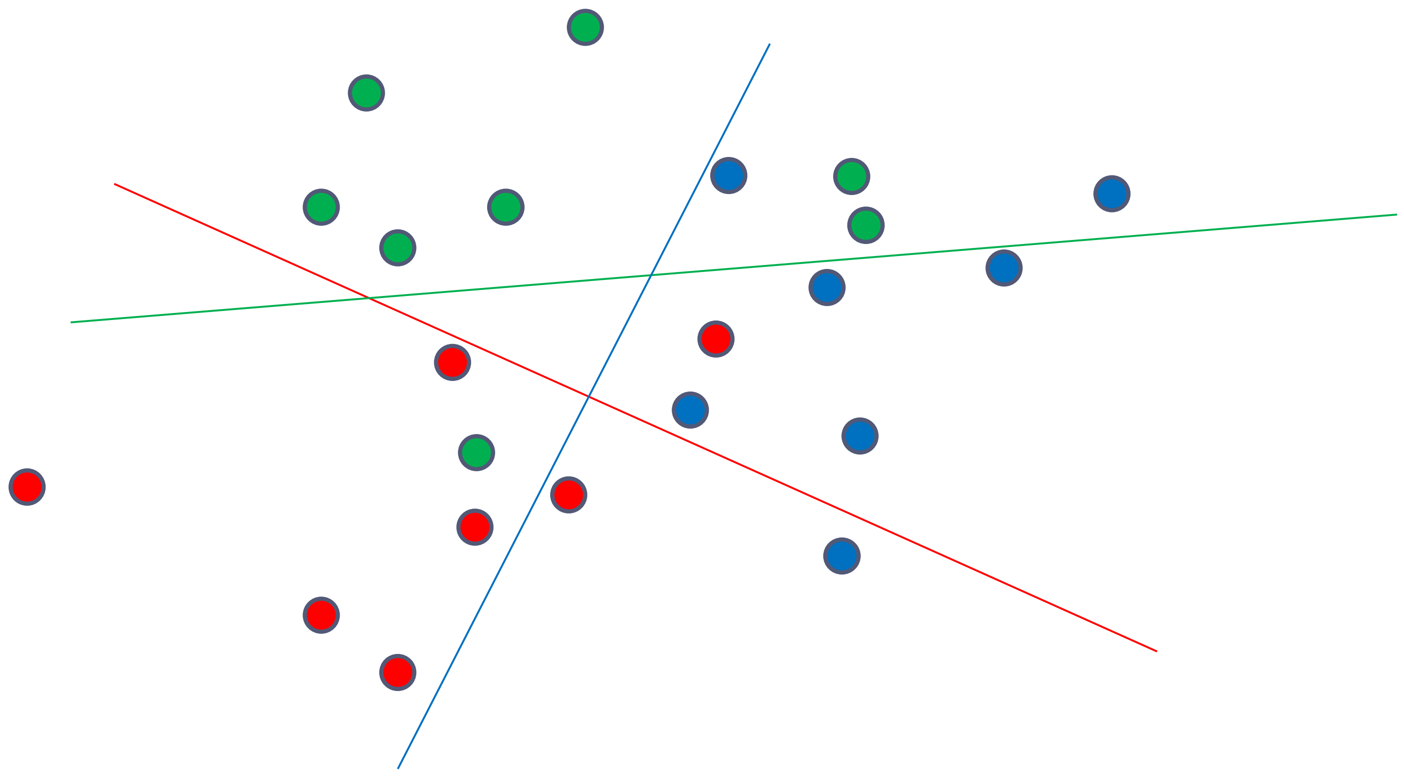
Basic token features	Six features are computed given a token t, including: (a) the lexical string of t, (b) the lemma of t, (c) the stem of t obtained using the Porter stemmer (Porter, 1980), (d) the part-of-speech tag of t, (e) whether t appears as a true trigger in the training data, and (f) whether t is a protein name.
Basic dependency features	Six features are computed given a dependency path p, including: (a) the vertex walk in p, (b) the edge walk in p, (c) the n-grams (n=2,3,4) of the (stemmed) words associated with the vertices in p, (d) the n-grams (n=2,3,4) of the part-of-speech tags of the words associated with the vertices in p, (e) the n-grams (n=2,3,4) of the dependency types associated with the edges in p, and (f) the length of p.

Table 1: Features for trigger labeling and argument labeling.

High Dimensional Features

- Difficult to include high-dimensional features directly into the MLN
- Trigram: $word(i-1,+w) \wedge word(\mathrm{i},+\mathrm{w}_1) \wedge word(i+$

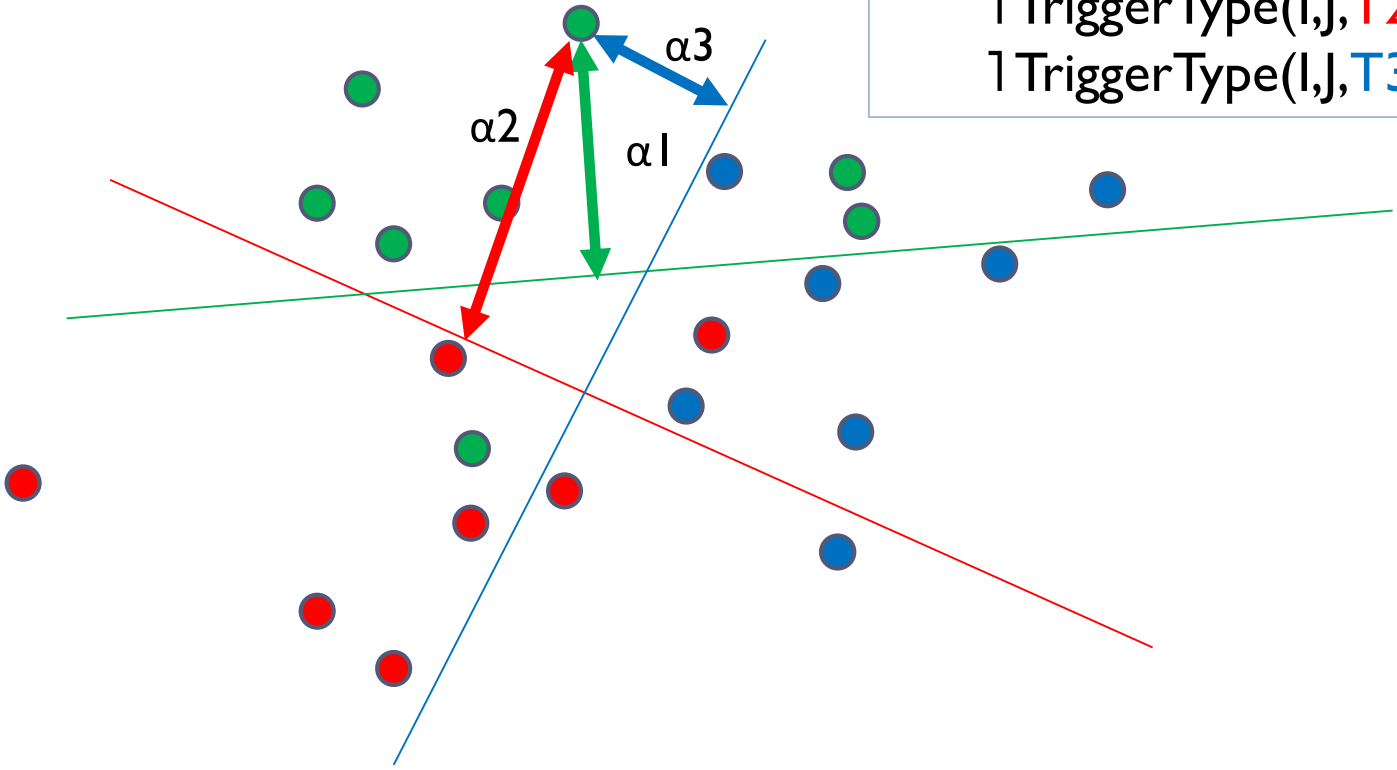
Reduce High-Dimensional Features to Low-Dimensional Unit Clauses



SVM-Multiclass Classifier
for trigger labeling

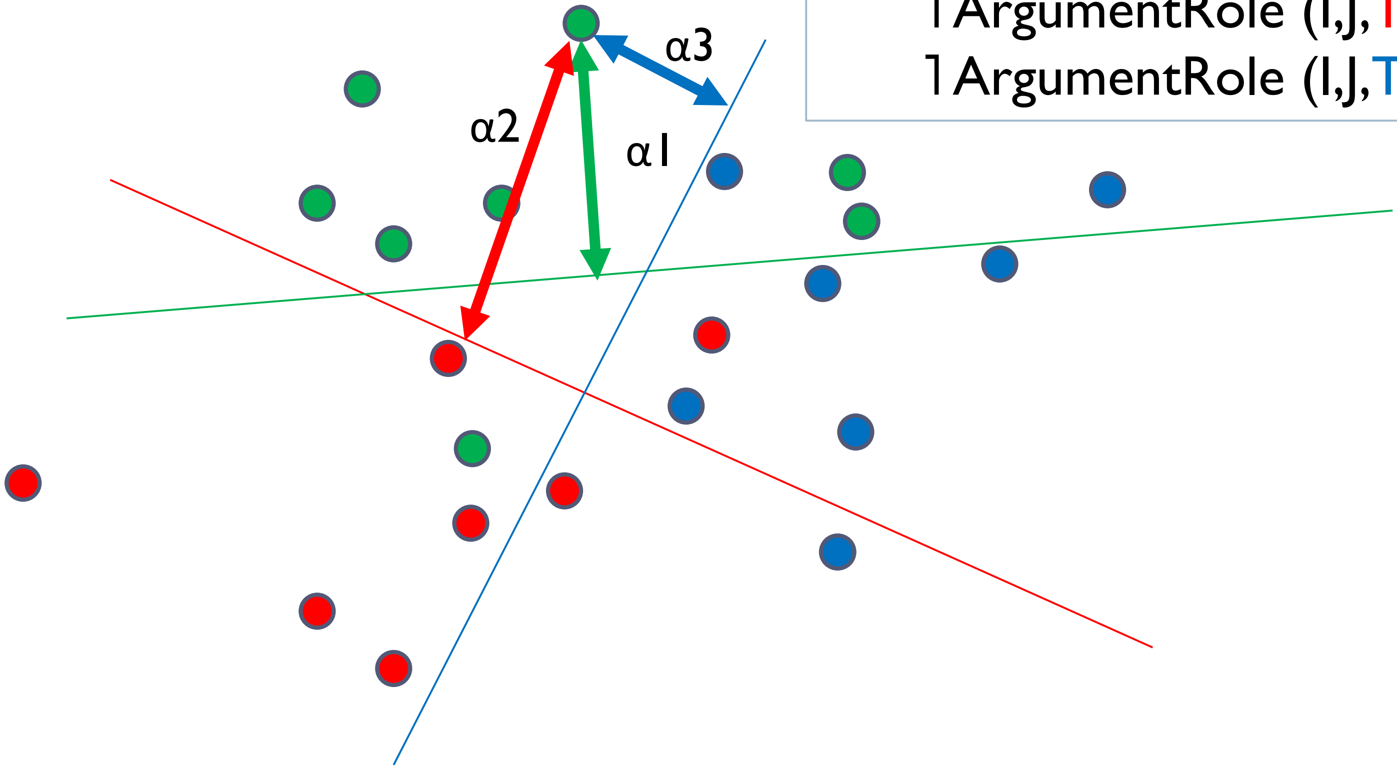
Reduce High-Dimensional Features to Low-Dimensional Unit Clauses

TriggerType(I,J,T1); α1
¬TriggerType(I,J,T2); α2
¬TriggerType(I,J,T3); α3



Reduce High-Dimensional Features to Low-Dimensional Unit Clauses

ArgumentRole(I,J,T1); α1
˥ArgumentRole (I,J,T2); α2
˥ArgumentRole (I,J,T3); α3



Performance

BioNLP '13

System	P	R	F1
UTD-MLN	48.95	59.24	53.61
EVEX	45.44	58.03	50.97
TEES	46.17	56.32	50.74
BIOSEM	42.47	62.83	50.68
NCBI	40.53	61.72	48.93
DLUTNLP	40.81	57.00	47.56

Performance

BioNLP '11

System	P	R	F1
UTD-MLN	53.42	63.61	58.07
Miwa-12	53.35	63.48	57.98
Riedel-11	-	-	56
UTurku	49.56	57.65	53.30
MSR	48.64	54.71	51.50

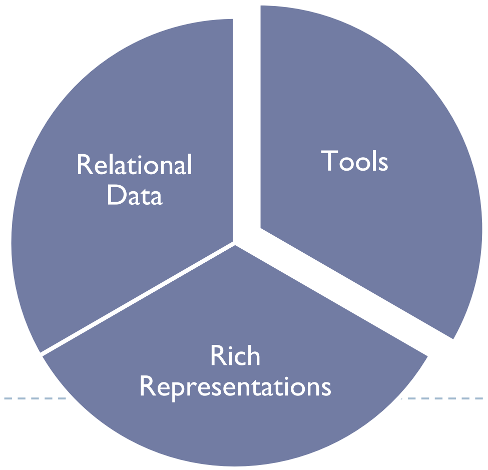
Performance

BioNLP '09

System	P	R	F1
Miwa12	52.67	65.19	58.27
UTD-MLN	53.96	63.08	58.16
Riedel-11	-	-	57.4
Miwa-10	50.13	64.16	56.28
Bjorne-09	46.73	58.48	51.95
Poon-MLN	43.7	58.6	50.0
Riedel-MLN	36.9	55.6	44.4

Conclusion

- MLNs show a lot of potential for NLP applications
  - Compact representation for complex domain knowledge and large relational data
  - Ability to handle uncertainty
- Bottleneck: Lack of scalable inference and learning algorithms
  - To scale up to NLP problems, we need approximations that trade-off scalability with strong guarantees
  - Approximate lifting and learning seem to be the most promising directions to achieve desired scalability
  - Easy-to-use software and tools will make MLNs more applicable to a wider NLP research community


Relational Data
Tools
Rich Representations