Distribution semantics and cyclic relational modeling

Taisuke Sato

Joint work with Yoshitaka Kameya & Ryosuke Kojima

New AI is coming

- Triggered by deep learning, data-centered
  - Big data (sensor data, SNS data, Web, Wikipedia, Knowledge Graph,...)
  - Machine learning (ML) techniques to deal with huge and noisy data
- Steady progress
  - Probabilistic logic learning (PLL) and statistical relational learning (SRL)
    - Probabilistic ILP that lifts propositional graphical models to first order ones
    - New probabilistic modeling languages
  - Relation extraction in IE ➔ millions of propositions from the web
  - Vector representation in NLP ➔ continuous real-valued logic
- What is required?
  - Framework unifying logic and probability (S.Russell 2013 )
  - Declarative interface between modeling and ML tasks
  - Distribution semantics and PRISM are an example

From logic and probability to machine learning

Logic

Completeness theorem 1930

Probability

Foundations of the Theory of Probability 1933

Fenstad's representation theorem 1967

Probability distribution on possible worlds

Logic-based probabilistic modeling

Distribution semantics

PRISM

Tabling(DP)

Proposi-tionalized probability computation

Machine learning

- Structure learning
- Parameter learning
- Viterbi inference
- Bayesian inference

Outline

- Representation theorem
- Distribution semantics
- The PRISM language
- Infinite prob. computation for cyclic relations
- Conclusion

Representation theorem

Representations of Probabilities Defined on First Order Languages, J.E.Fenstad,
Set, Models and Recursion Theory, J.E.Crossley (ed.), North-Holland, pp156-172, 1967

Let

- L: countable 1st order language w.o. "="
- α: (possibly open) formula in L
- $P(\alpha)$: probability satisfying:

$$0 \leq P(\alpha) \leq 1, P(\neg\alpha) = 1 - P(\alpha),$$
$$P(\alpha \vee \beta) + P(\alpha \wedge \beta) = P(\alpha) + P(\beta)$$
$$\begin{cases} P(\alpha) = P(\beta) & \text{if } \vdash \alpha \Leftrightarrow \beta \\ P(\alpha) = 1 & \text{if } \vdash \alpha \end{cases}$$

Then

$$P(\alpha) = \int_S P_\omega(\alpha[\omega])dP(\omega)$$

where

- S: set of all H-interpretations ω on the H-universe D for L'
- L': L augmented with "special constants"
- $P(\cdot)$: σ-additive prob. measure on S
- $P_\omega(\cdot)$: prob. measure on the set of assignments $a \in [I \to D]$
- I: variables are indexed by I (= nat numbers) like $x_1, x_2, \ldots$
- $\alpha[\omega]$: $\{a \mid \omega \models_a \alpha(a(x_1), a(x_2), \ldots)\}$ = set of assignments a satisfying α

$P(\cdot)$ and $P_\omega(\cdot)$

- Two types of prob. measure in $P(\alpha) = \int_S P_\omega(\alpha[\omega])dP(\omega)$

  $P(\exists x\, UFO(x))$ vs $P_\omega(UFO(x))$

- $P(\cdot)$ gives possible worlds semantics:

  For closed α, $P(\alpha)$ is the sum of probabilities of H-models ω of α

  because $P_\omega(\alpha[\omega]) = 1$ if $\omega \models \alpha$, else $= 0$, so $P(\alpha) = \int_{\omega:\omega\models\alpha} \mathbf{1} dP(\omega)$

- $P_\omega(\cdot)$ gives frequency semantics in each ω:

  For open α = Friend(x_1,x_2) with two variables x_1, x_2,

  Friend(x_1,x_2)[ω] is the set of assignments "a" s.t. $\omega \models$ Friend($\underset{\shortparallel}{a(x_1)}^{d_1}$,$a(x_2)$)

  $\Rightarrow$ Friend(x_1,x_2)[ω] = { (d_1,d_2) in DxD | $\omega \models$ Friend(d_1,d_2) }

  $\Rightarrow$ P_ω(Friend(x_1,x_2)[ω]) = P_ω({ (d_1,d_2) in DxD | $\omega \models$ Friend(d_1,d_2) })

  = probability that randomly sampled (d_1,d_2) from ω are friends

Simulating $P(\alpha) = \int_S P_\omega(\alpha[\omega]) dP(\omega)$

S	Fr(a,a)	Fr(a,b)	Fr(b,a)	Fr(b,b)	Prob.
ω_1	f	f	f	t	0.2
ω_2	t	f	t	f	0.3
ω_3	f	t	f	f	0.5

Other worlds have prob. 0

(x_1,x_2)	Prob. $\omega_1, \omega_2, \omega_3$
(a,a)	0.3
(a,b)	0.1
(b,a)	0.2
(b,b)	0.4

- Herbrand universe = D = {a,b} ➔ Hebrand base = {Fr(a,a), Fr(a,b), Fr(b,a), Fr(b,b)}
- 16 possible worlds = $\{\omega_1,\omega_2,\omega_3,\ldots,\omega_{16}\}$ = truth assignments for ground atoms
- Table(left): probability of each possible world

  Table(right): probability distribution on (x_1,x_2) from DxD

$P(\exists x_2 Fr(a,x_2)) = P(\omega_1)*0 + P(\omega_2)*1 + P(\omega_3)*1 = 0.2*0+0.3*1+0.5*1 = 0.8$

$P(\forall x_2 Fr(a,x_2)) = 0$

$P_{\omega 2}(Fr(a,x_2)) = P_{\omega 2}(\{ (a,a), (b,a) \}) = 0.3+0.2 = 0.5$

$P(Fr(a,x_2)) = P(\omega_1)*P_{\omega 1}(R(a,x_2))+ \cdots = 0.2*0+0.3*0.5+0.5*0.5 = 0.4$

From logic and probability to machine learning

Logic

Completeness theorem 1930

Probability

Foundations of the Theory of Probability 1933

Fenstad's representation theorem 1967

possible to simulate in a computer

Logic-based probabilistic modeling

Distribution semantics

PRISM

Tabling(DP)

Proposi-tionalized probability computation

Cyclic relations

Infinite sum of probs.

MPE for cyclic models

EM learning for cyclic models

…

Distribution semantics [Sato'95]

- Fenstad's theorem (possible worlds semantics, not frequency one) adapted to probabilistic logic programs
- Program DB = F U R
  - F : ground atoms representing random choices
  - $P_F(.)$: basic distribution = distribution on H-interpretations for F
  - R : definite clauses
- $P_F(.)$ ➔ least model semantics + Kolmogorov's extension theorem

  ➔ $P_{DB}(.)$ on H-interpretations for DB
  - Least model semantics can be replaced by other semantics : stable model semantics (P-log), largest model semantics
- Semantics for PLP languages（PRISM,ProbLog,LPADs,PITA,...）
  - Infinite models such as probabilistic grammars allowed

Propositional example

- DB = { $\underbrace{\text{a:- b, a:- c,}}^{R}$ $\underbrace{\text{b, c}}^{F}$ }

P_F(b,c) given

P_F(0,0)	P_F(0,1)	P_F(1,0)	P_F(1,1)
0.4	0.3	0.2	0.1

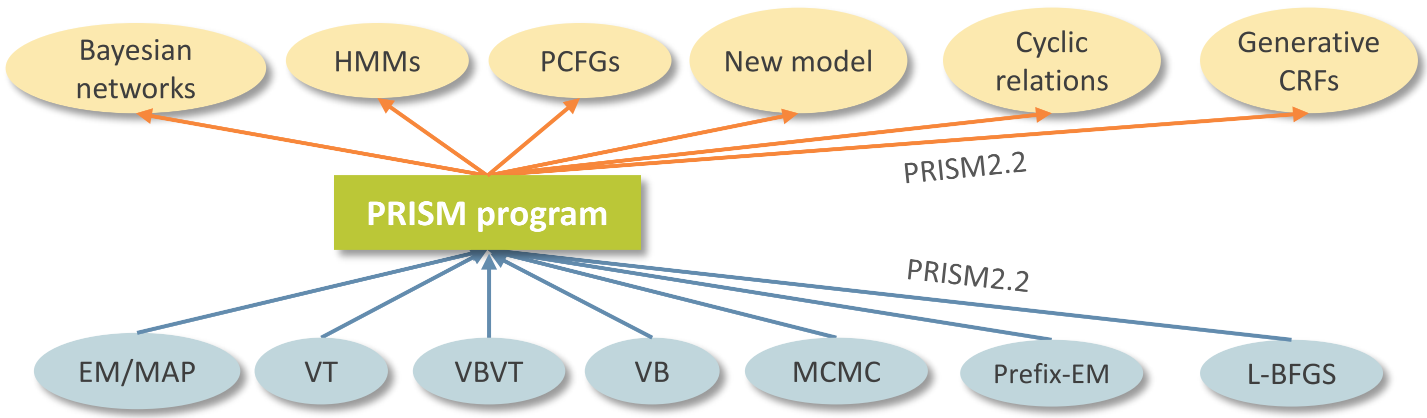
(b,c) ~ $P_F(\cdot\ ,\ \cdot)$ b	 c	Sampled DB'	DB's least H-model	a	P_{DB}(a,b,c)
0 (false)	0	a:-b, a:-c	{}	0	0.4
0	1 (true)	a:-b, a:-c c	{c,a}	1	0.3
1	0	a:-b, a:-c b	{b,a}	1	0.2
1	1	a:-b, a:-c b, c	{b,c,a}	1	0.1
	anything	else			0

PRISM [Sato & Kameya'97]

- An implementation of the distribution semantics
- Prolog + msw/2 (random choice) + parameter learning
- Single data structure (explanation graph) combined with generic algorithms subsumes:
  - BNs: belief propagation algorithm
  - HMMs: the Baum-Welch algorithm
  - PCFGs: the Inside-Outside algorithm
- Rich built-ins (applicable to *all programs*)
  - Exact probability computation using DP on explanation graphs
  - N-Viterbi (MPE) inference
  - Parameter learning (EM/MAP, DAEM, VB, VT, VB-VT)
  - Model scores (BIC, Cheeseman-Stutz, VFE)
  - MCMC for Bayesian inference…

PRISM2.2: new features

- PRISM (http://rjida.meijo-u.ac.jp/prism/)
  - Declarative interface between probabilistic modeling and computation/learning



- PRISM2.2
  - can compute an infinite sum of probabilities for cyclic relations
    - Markov chains, prefix and infix probabilities in PCFGs
  - can learn and compute generative conditional random fields (G-CRFs)
    - logistic regression, linear-chain CRFs, CRF-CFGs

Probability computation in PRISM

- Goal ➔ explanation graph ➔ probability

PCFG$_1$
S ➔ a:0.5 | b:0.3 | S S:0.2

```
values(s,[[a],[b],[s,s]],
     set@[0.5,0.3,0.2]).

pcfg(L):- pcfg([s],L,[]).
pcfg([A|R],L0,L2):-
   ( nonterminal(A) ➔
       msw(A,RHS),
       pcfg(RHS,L0,L1)
   ;  L0=[A|L1] ),
   pcfg(R,L1,L2).
pcfg([],L,L).
```

PCFG program

?- prob(pcfg([s],[a,b],[]),P)

Tabled search

Infinite recursion stopped!

G-IO comp. by DP

```
pcfg([s],[a,b],[])
  ⇔ pcfg([s,s],[a,b],[]) & pcfg([],[],[]) & msw(s,[s,s])
pcfg([s,s],[a,b],[])
  ⇔ pcfg([a],[a,b],[b]) & pcfg([s],[b],[]) & msw(s,[a])
pcfg([s],[b],[])
  ⇔ pcfg([b],[b],[]) & pcfg([],[],[]) & msw(s,[b])
pcfg([b],[b],[]) ⇔ pcfg([],[],[])
pcfg([],[],[])
pcfg([a],[a,b],[b]) ⇔ pcfg([],[b],[b])
pcfg([],[b],[b])
```

0.2

0.5

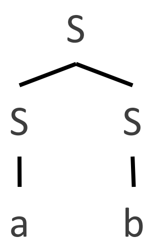
0.3

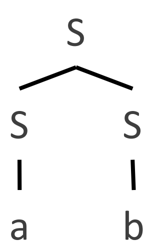
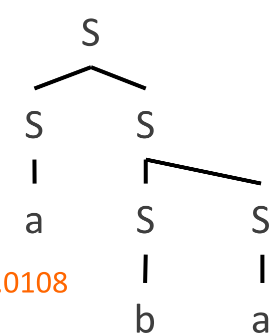
P = 0.5x0.3x0.2 = 0.03

Infinite sum of probabilities

- Prefix u: uw is a sentence for some w
- $P_{refix}(u) = \sum_w P_{cfg}(uw)$, useful for predicting the next word
- $PCFG_1$

  $S \rightarrow a{:}0.5 \mid b{:}0.3 \mid S\,S{:}0.2$

$P_{cfg}([a,b]) =$ P() $= \overset{0.2}{P(S \rightarrow S\,S)}\overset{0.5}{P(S \rightarrow a)}\overset{0.3}{P(S \rightarrow b)}$

$P_{refix}([a,b]) =$ P() + P() + …

0.03

0.0108

generated by cyclic left-corner relation of S in S➔S S

Cyclic explanation graph

- Goal ➔ expl. graph ➔ SCCs ➔ eqs ➔ probs

PCFG$_2$
S ➔ a:0.4 | b:0.3 | S S:0.2 | S:0.1

```
values(s,[[a],[b],[s,s],[s]],
        set@[0.4,0.3,0.2,0.1]).
pre_pcfg(L):-
   pre_pcfg([s],L,[]).
pre_pcfg([A|R],L0,L2):-
  ( nonterminal(A) ➔
      msw(A,RHS),
      pre_pcfg(RHS,L0,L1)
  ;  L0=[A|L1] ),
  ( L1=[] ➔ L2=[]
  ; pre_pcfg(R,L1,L2) ).
pre_pcfg([],L,L).
```

prefix parser program

as prefix

```
?- lin_prob(pre_pcfg([s],[a,b],[]),P)
```

Tabled search

SCC

```
pre_pcfg([s],[a,b],[])
  ⇔ pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s,s],[a,b],[])
   ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([s],[b],[]) &
     msw(s,[a])
   v pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[a,b],[b]) & pre_pcfg([s],[b],[]) &
     msw(s,[s])
   v pre_pcfg([s],[a,b],[]) & msw(s,[s])
 ...
```

Solving prob. equations

P = 0.05

SCCs (strongly connected components)

- SCCs are partially ordered ➔ DP possible

SCC

SCC

SCC

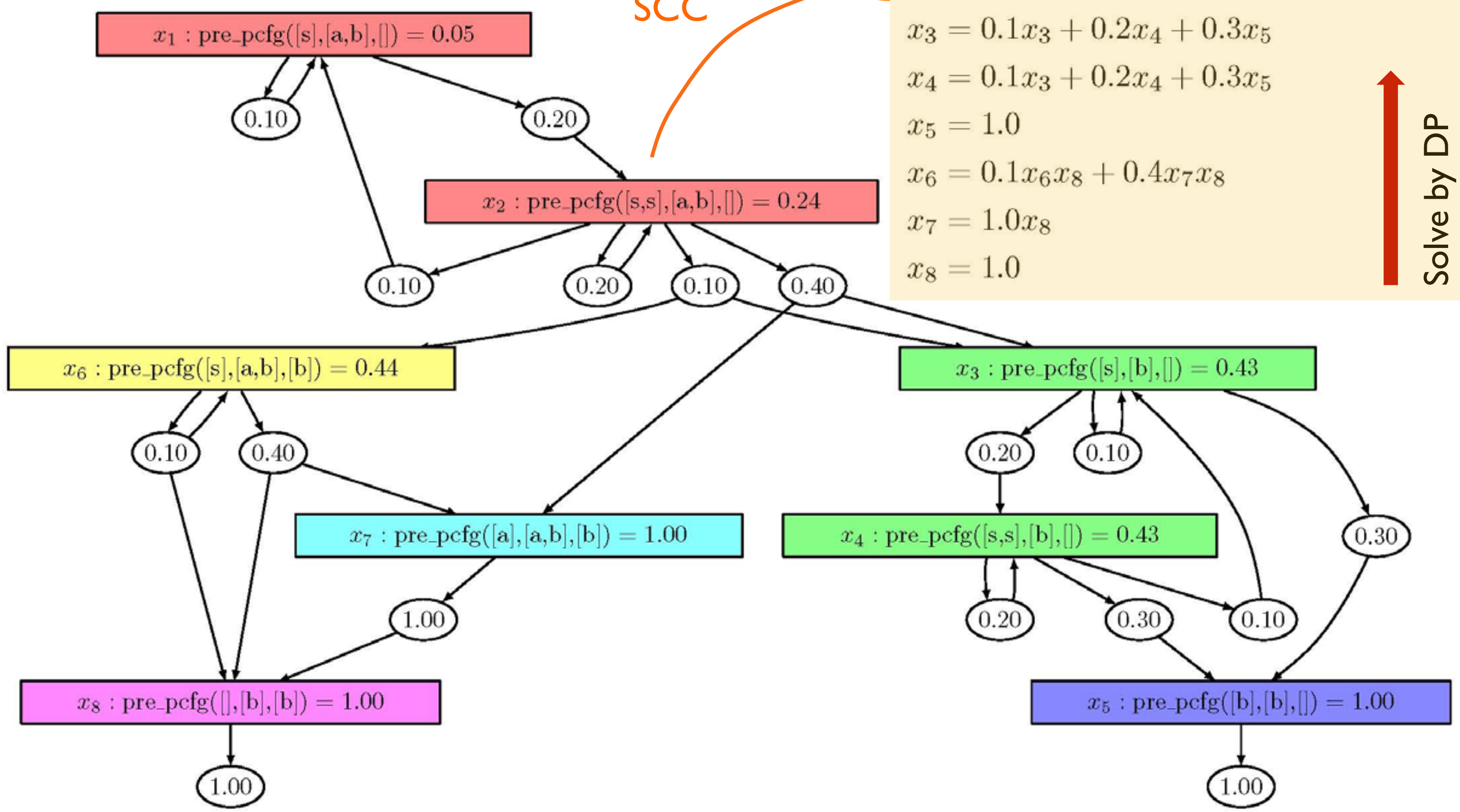
pre_pcfg([s],[a,b],[])
  ⇔ pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s]) v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s,s],[a,b],[])
  ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([s],[b],[]) & msw(s,[a])
   v pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[a,b],[b]) & pre_pcfg([s],[b],[]) & msw(s,[s])
   v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s],[b],[])
  ⇔ pre_pcfg([b],[b],[]) & msw(s,[b]) v pre_pcfg([s,s],[b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[b],[]) & msw(s,[s])
pre_pcfg([s,s],[b],[])
  ⇔ pre_pcfg([b],[b],[]) & msw(s,[b]) v pre_pcfg([s,s],[b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[b],[]) & msw(s,[s])
pre_pcfg([b],[b],[])
pre_pcfg([s],[a,b],[b])
  ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([],[b],[b]) & msw(s,[a])
   v pre_pcfg([s],[a,b],[b]) & pre_pcfg([],[b],[b]) & msw(s,[s])
pre_pcfg([a],[a,b],[b]) ⇔ pre_pcfg([],[b],[b])
pre_pcfg([],[b],[b])

A set of probability equations

$$
\begin{aligned}
x_1 &= 0.1x_1 + 0.2x_2 \\
x_2 &= 0.1x_1 + 0.2x_2 + 0.1x_3x_6 + 0.4x_3x_7 \\
x_3 &= 0.1x_3 + 0.2x_4 + 0.3x_5 \\
x_4 &= 0.1x_3 + 0.2x_4 + 0.3x_5 \\
x_5 &= 1.0 \\
x_6 &= 0.1x_6x_8 + 0.4x_7x_8 \\
x_7 &= 1.0x_8 \\
x_8 &= 1.0
\end{aligned}
$$

Solve by DP

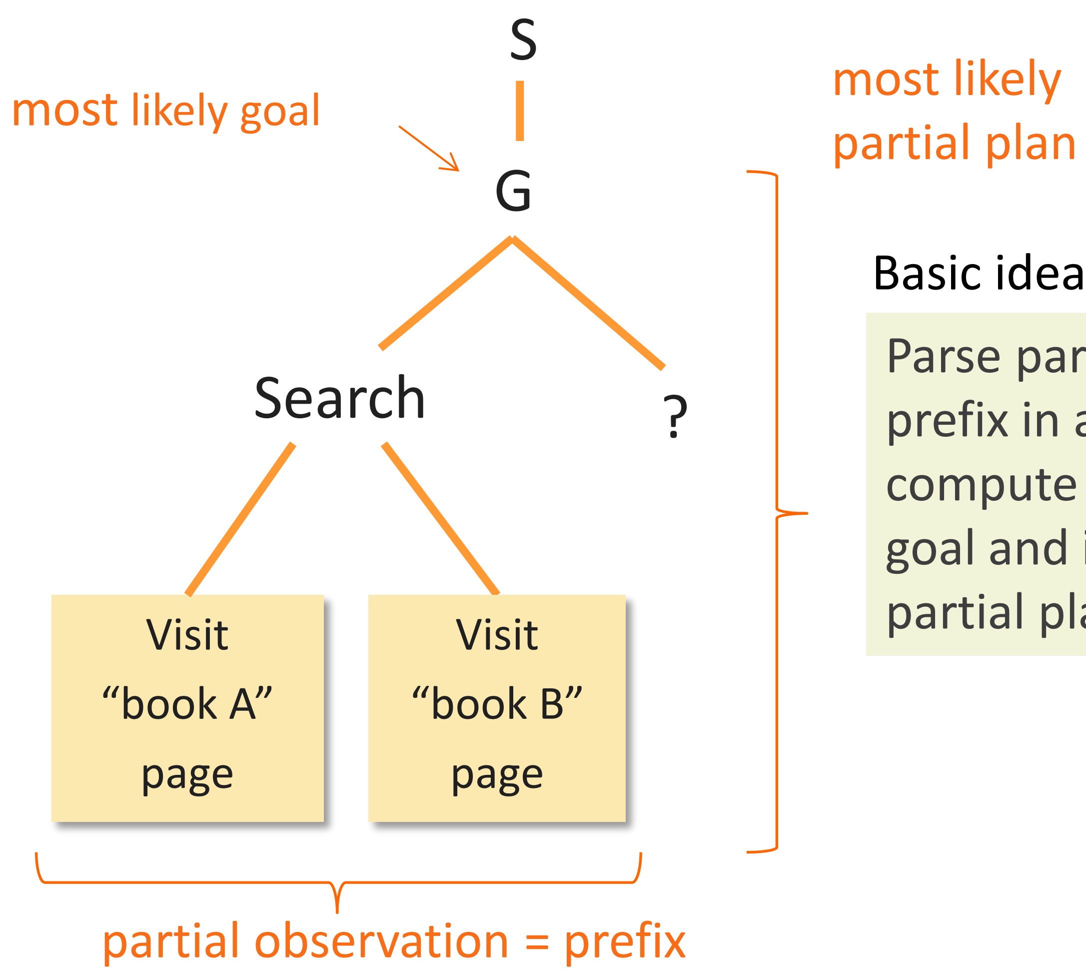
SCC



Plan recognition by parsing
[Vilain 90,Kautz+ 91]

Plan recognition	CFG parsing
plan	parse tree
(sub)goal	NT (non-terminal)
action	word
successful action seq.	sentence

Online plan recognition



Basic idea:

Parse partial obs. as a prefix in a PCFG and compute the most likely goal and its most likely partial plan

Classifying websurfers

Actions on a web site directory

up	: climb up
down	: down
sibling	: visit sibling page
revisit	: same page
move	: others

We capture intentions (survey, news,…) and plans of visitors online from their partial action sequences as a prefix

Web site

down up down up

prefix
(online situation)

complete log data

Experiment

- Three data sets and five goals (survey,news,…)
  - Clustering web log data in the Internet Traffic Archive (NASA, ClarkNet, U of S) using a mixture of PCFGs that share the same CFG but with a different set of parameters yields five clusters
- We write 102 CFG rules (32 NTs)
  - S → Survey, S → News,…, UpDown →Up Down

    UpDown →Up SameLayer Down, Up →Up up,…
- Parameters are learned from each dataset by MLE
  - $PCFG_{NASA}$, $PCFG_{CNet}$, $PCFG_{UofS}$ obtained
- Complete dataset
  - Access log data paired with their most likely goal/plan inferred by the Viterbi algorithm

Two tasks

- Parameter learning from partial observations as prefixes
  - $\sum_u P_{refix}(u) > 1$, so prefixes cannot have a distribution and no MLE possible
    But parameter learning by maximizing $\prod_u P_{refix}(u|\theta)$ is still possible
    using the EM-like algorithm ➔ Prefix-EM algorithm
  - Viterbi training from prefixes is possible ➔ Prefix-VT algorithm derived
  - We compare three learning methods (Prefix-EM,Prefix-VT,PCFG-EM) by classifying prefixes of NASA dataset into five goals (survey,news,…)
- Online goal and plan recognition from partial obs. as prefixes
  - To classify prefixes of Web log data into five goals (survey,news,…) and infer their most likely partial plan
  - We compare four recognition methods (Prefix, HMM, LR, SVM)
  - An example of online goal recognition

The Prefix EM algorithm

Step 1: For each t $(1 \le t \le T)$, compute an explanation graph $expl(G_t)$ and topologically sort the SCCs extracted from $expl(G_t)$ in descending order in terms of the $\succ$ relation. Let $SCC_t^1 \succ \cdots \succ SCC_t^{N_t}$ be the sorted SCCs. SCC_t^1 contains the top-goal G_t such that $P(G_t)$=$P(1)$.

Repeat Step 2 until $\mathcal{L} = \prod_{t=1}^{T} P(G_t)$ converges.

Step 2-1(computing inside probabilities):
For each t $(1 \le t \le T)$ and for each k from N_t to $k = 1$, solve a set of probability equations $X = \boldsymbol{M}X + Y$ for SCC_t^k and obtain the inside probabilities X of goals in SCC_t^k. Store the matrix $\boldsymbol{M}$ for later use at **Step2-2**.

Step 2-2(computing outside probabilities):
Put $E[\sigma(i_d, v) \mid G_t] = 0$ for every $\mathtt{msw}(i_d, v)$ and every t $(1 \le t \le T)$.
For each t $(1 \le t \le T)$, do the following.
For every j, $P(1)_j = 0$. Goals in $expl(G_t)$ are numbered so that 1= G_t. $P(i)$ is the prob. of goal i and $P(i)_j = \frac{\partial P(i)}{\partial P(j)}$.
For every goal $P(n)$ in SCC_t^1, set $P(n)_n = 1$.
For each k from 1 to N_t, do the following.
Let $X = \boldsymbol{M}X + Y$ be a set of probability equations for SCC_t^k.
For every P-variable $P(j)$ appearing in $\boldsymbol{M}$ or Y, solve

$$\frac{\partial X}{\partial P(j)} = \boldsymbol{M}\frac{\partial X}{\partial P(j)} + \frac{\partial \boldsymbol{M}}{\partial P(j)}X + \frac{\partial Y}{\partial P(j)}$$

and store all the $P(i)_j$'s in $\dfrac{\partial X}{\partial P(j)}$.
matrix whose member is either 0 or 1 in the second case.
For every $P(i)$ in X, increment the outside probability $P(1)_j$ by

$$P(1)_j \Leftarrow P(1)_i \cdot P(i)_j + P(1)_j.$$

For each $\mathtt{m} = \mathtt{msw}(i_d, v)$ in $\boldsymbol{M}$ or Y, increment $E[\sigma(i_d, v) \mid G_t]$ by

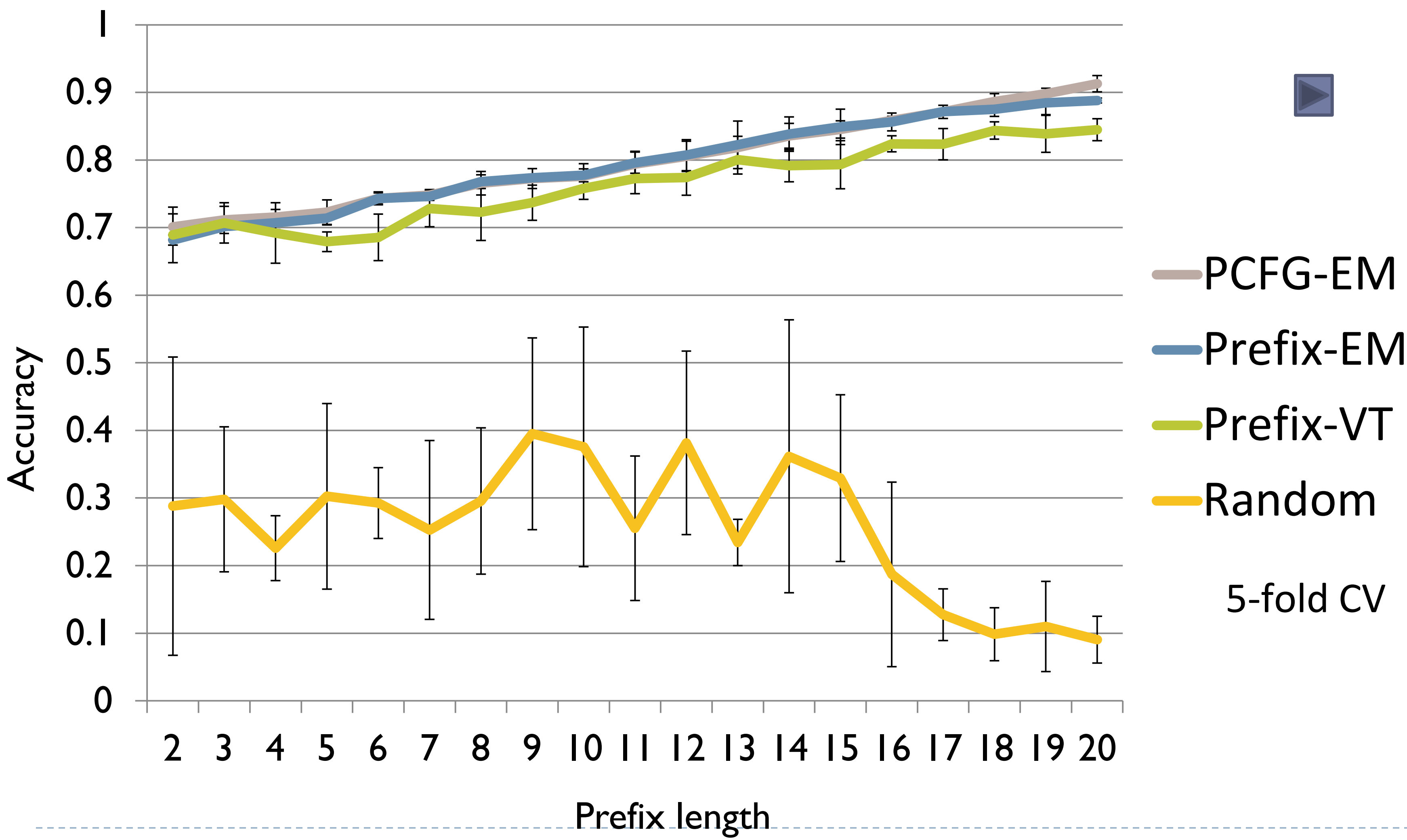
$$E[\sigma(i_d, v) \mid G_t] \Leftarrow \frac{P(\mathtt{m}) \cdot P(1)_{\mathtt{m}}}{P(G_t)} + E[\sigma(i_d, v) \mid G_t]$$

Step 2-3: For each $\mathtt{msw}(i_d, v)$ in a program, update $\theta_{i_d,v} = P(\mathtt{msw}(i_d, v))$ by

$$\theta_{i_d,v} = \frac{\sum_{t=1}^{T} E[\sigma(i_d, v) \mid G_t] + \alpha_{i_d,v}}{\sum_v \sum_{t=1}^{T} E[\sigma(i_d, v) \mid G_t] + \alpha_{i_d,v}}$$

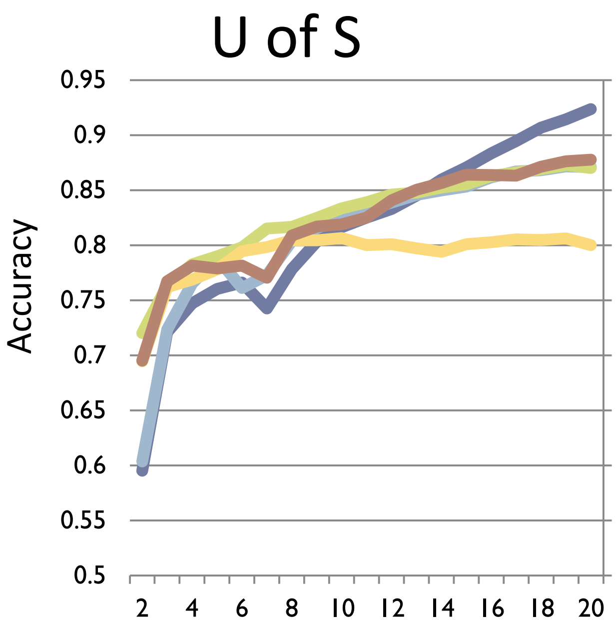
where $\alpha_{i_d,v}$ is a pseudo count for $\mathtt{msw}(i_d, v)$.

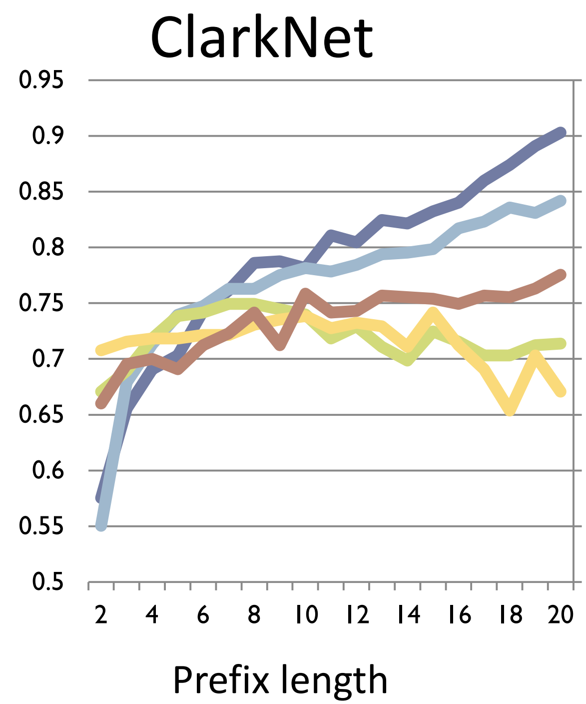
Parameter learning (NASA)

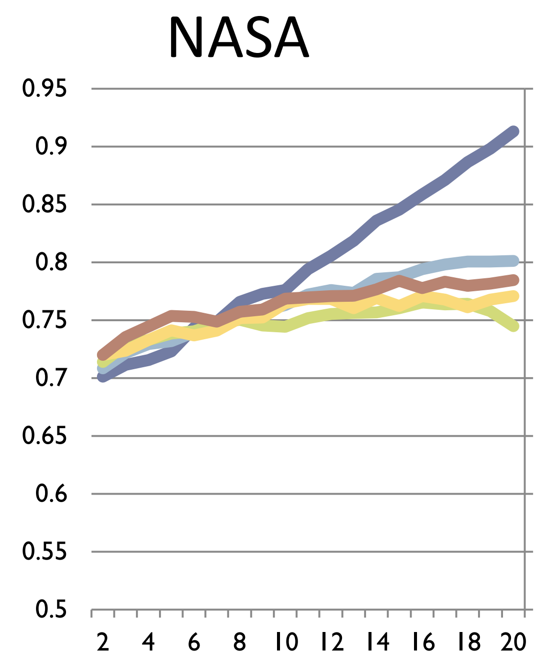


Online goal recognition

Infer the visitor's intention (survey, news, survey-specific-areas, news-specific-areas, others)







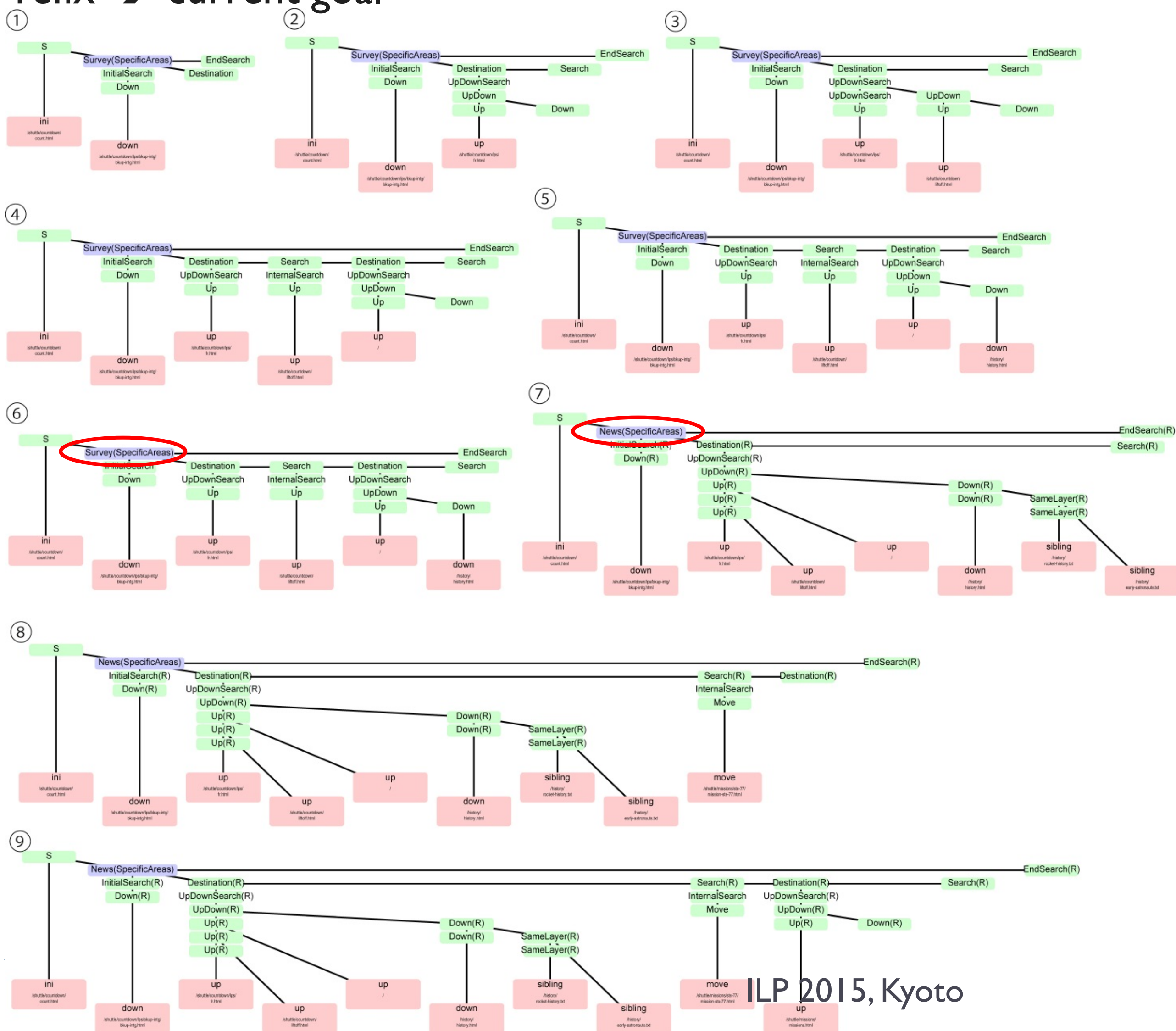
5.12×10^4 2.77×10^5 3.14×10^6

(PCFG's entropy $= -\sum_t p(t) \log p(t)$ of PCFG[Chi+99])

5-fold CV

Online plan recognition

- Prefix ➔ current goal



Conclusion

- Emerging technologies
  - will combine big data, propositions, logic, probability and KB
  - need of a solid framework unifying them
- Distribution semantics
  - computational adaptation of Fenstad's representation theorem to LP
  - framework unifying logic and probability for probabilistic ILP
  - semantic basis for PRISM,ProbLog,LPADs,PITA,...
- PRISM
  - an implementation of the distribution semantics
  - aims to provide one-stop service for logic-based probabilistic modeling
  - PRISM2.2 beta can compute and learn cyclic relational models that have been relatively unexplored
  - soon available from http://rjida.meijo-u.ac.jp/prism/