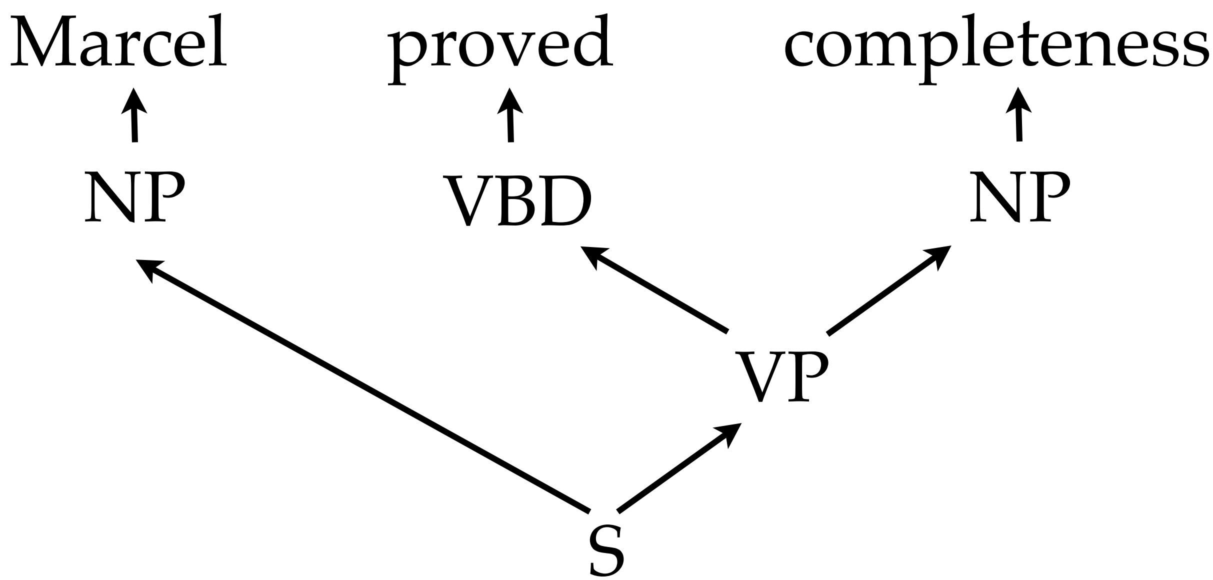
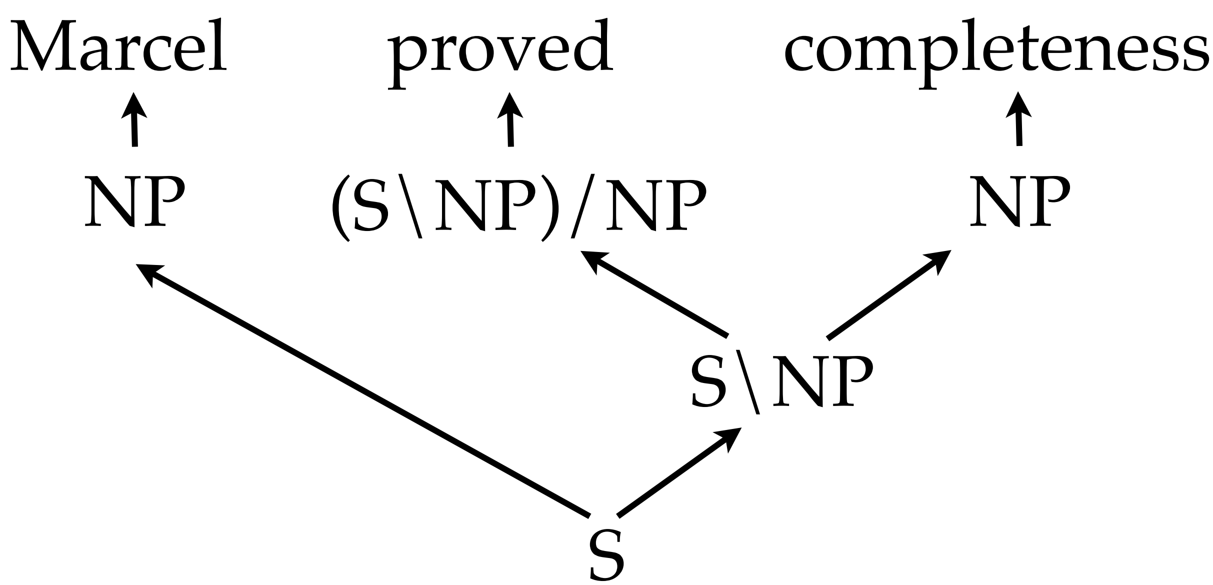
Integrated Supertagging and Parsing

Michael Auli
University of Edinburgh

Parsing

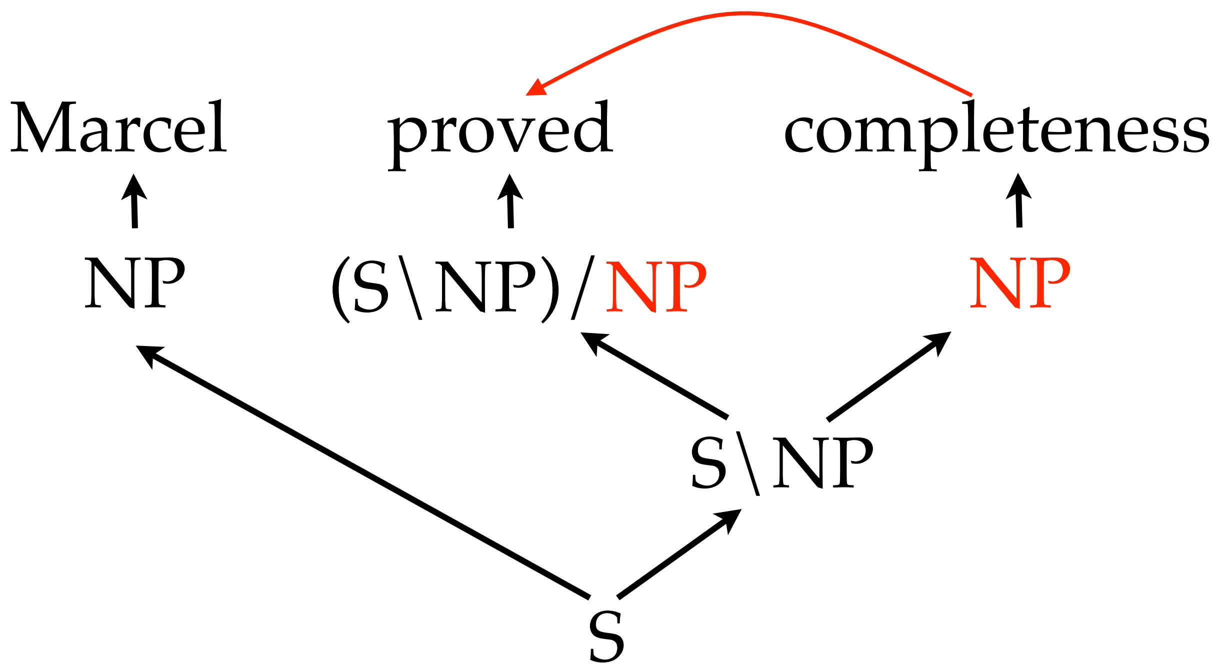


CCG Parsing



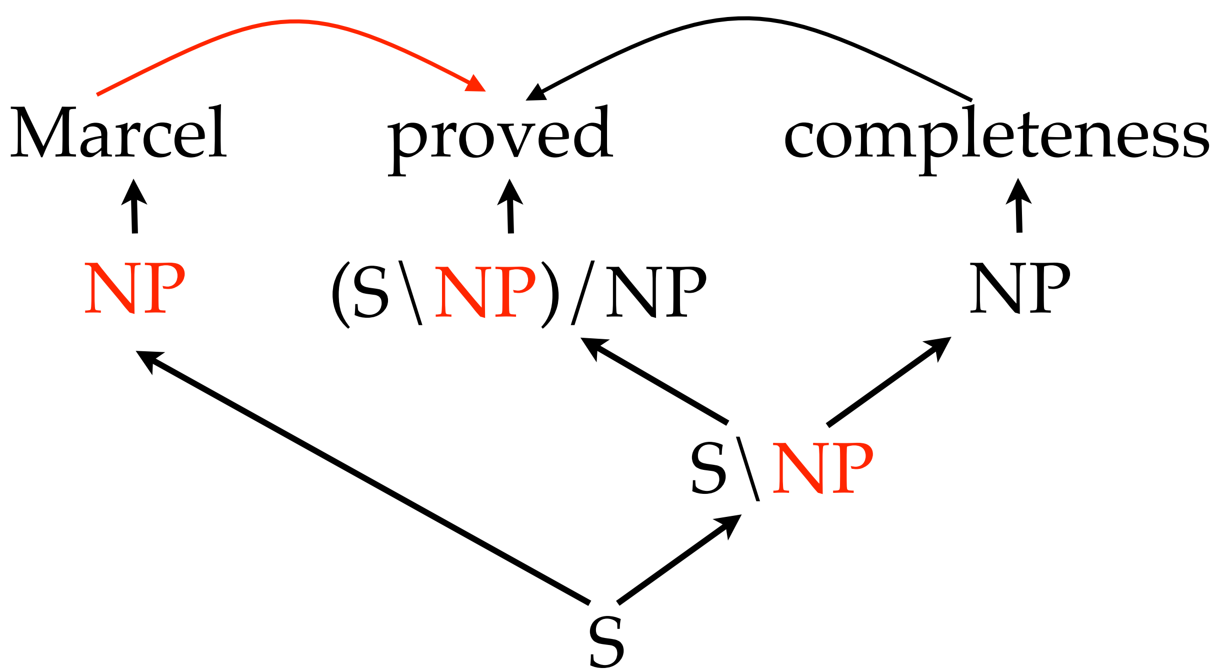
Combinatory Categorial Grammar (CCG; Steedman 2000)

CCG Parsing



<proved, (S\NP)/NP, completeness>

CCG Parsing



<proved, (S\NP)/NP, completeness>

<proved, (S\NP)/NP, Marcel>

Why CCG Parsing?

- MT: Can analyse nearly any span in a sentence
  (Auli ’09; Mehay ‘10; Zhang & Clark 2011; Weese et. al. ’12)
  e.g. “conjectured and proved completeness” $\vdash$S\NP
- Composition of regular and context-free languages -- mirrors situation in syntactic MT (Auli & Lopez, ACL 2011)
- Transparent interface to semantics (Bos et al. 2004)
  e.g. proved $\vdash$ (S\NP)/NP : λx.λy.*proved′ xy*

CCG Parsing is hard!

$$
\begin{array}{ccc}
\text{Marcel} & \text{proved} & \text{completeness} \\
\hline
NP & (S\backslash NP)/NP & NP \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP} \quad > \\
\hline
\multicolumn{3}{c}{S} \quad <
\end{array}
$$

Over 22 tags per word! (Clark & Curran 2004)

Supertagging

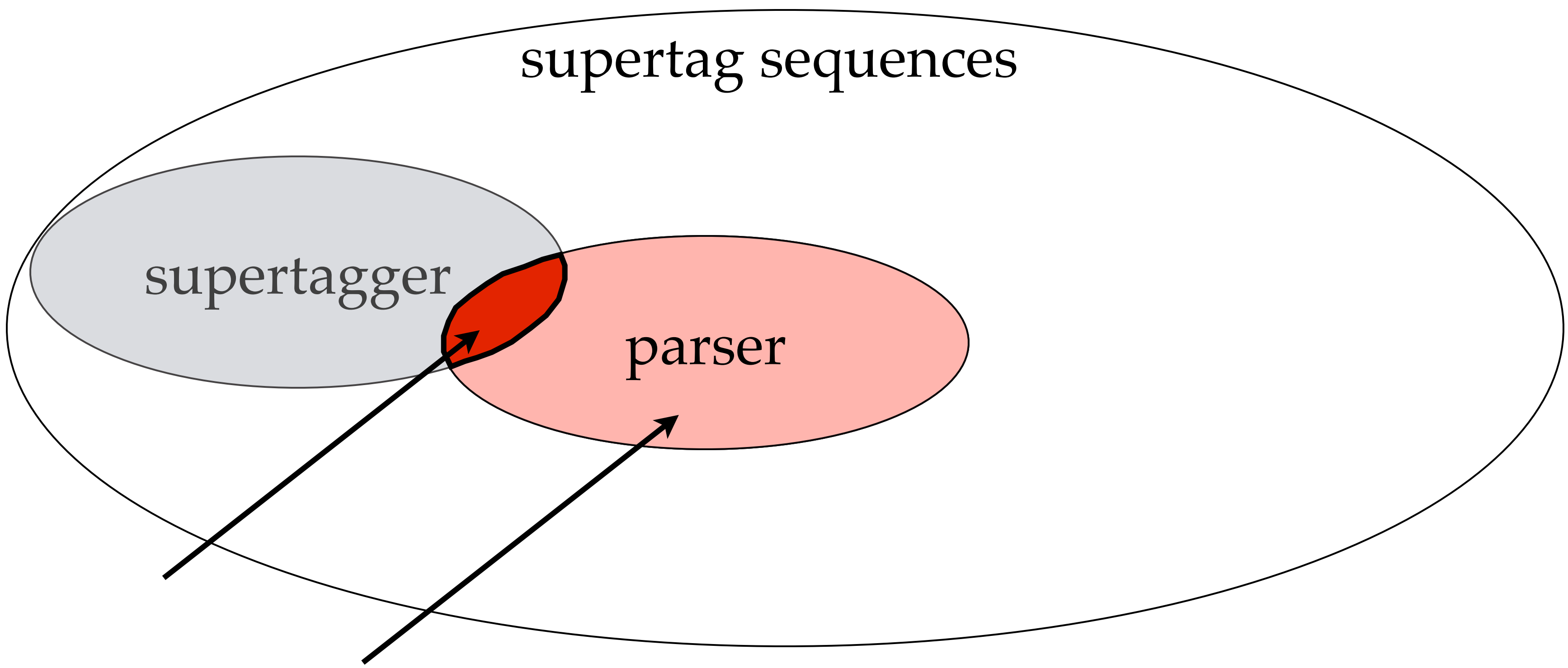
$$
\begin{array}{ccc}
\text{Marcel} & \text{proved} & \text{completeness} \\
\hline
NP & (S\backslash NP)/NP & NP \\
\cline{2-3}
 & \multicolumn{2}{c}{S\backslash NP} \; > \\
\hline
\multicolumn{3}{c}{S} \; <
\end{array}
$$

Supertagging

time	flies	like	an	arrow
NP	$S \backslash NP$	$(S \backslash NP)/NP$	NP/NP	NP

The Problem

- Supertagger has no sense of overall grammaticality.
- But parser restricted by its decisions.
- Supertagger probabilities not used in parser.



This talk

- Analysis of state-of-the-art approach

  Trade-off between efficiency and accuracy (ACL 2011a)

- Integrated supertagging and parsing

  with Loopy Belief Propagation and Dual Decomposition (ACL 2011b)

- Training the integrated model

  with Softmax-Margin towards task-specific metrics (EMNLP 2011)

Methods achieve **most accurate** CCG parsing results.

Adaptive Supertagging

time	flies	like	an	arrow
NP	$S\backslash NP$	$(S\backslash NP)/NP$	NP/NP	NP
NP/NP	NP	....	...	...
...	...	$((S\backslash NP)\backslash(S\backslash NP))/NP$		
		....		

Adaptive Supertagging

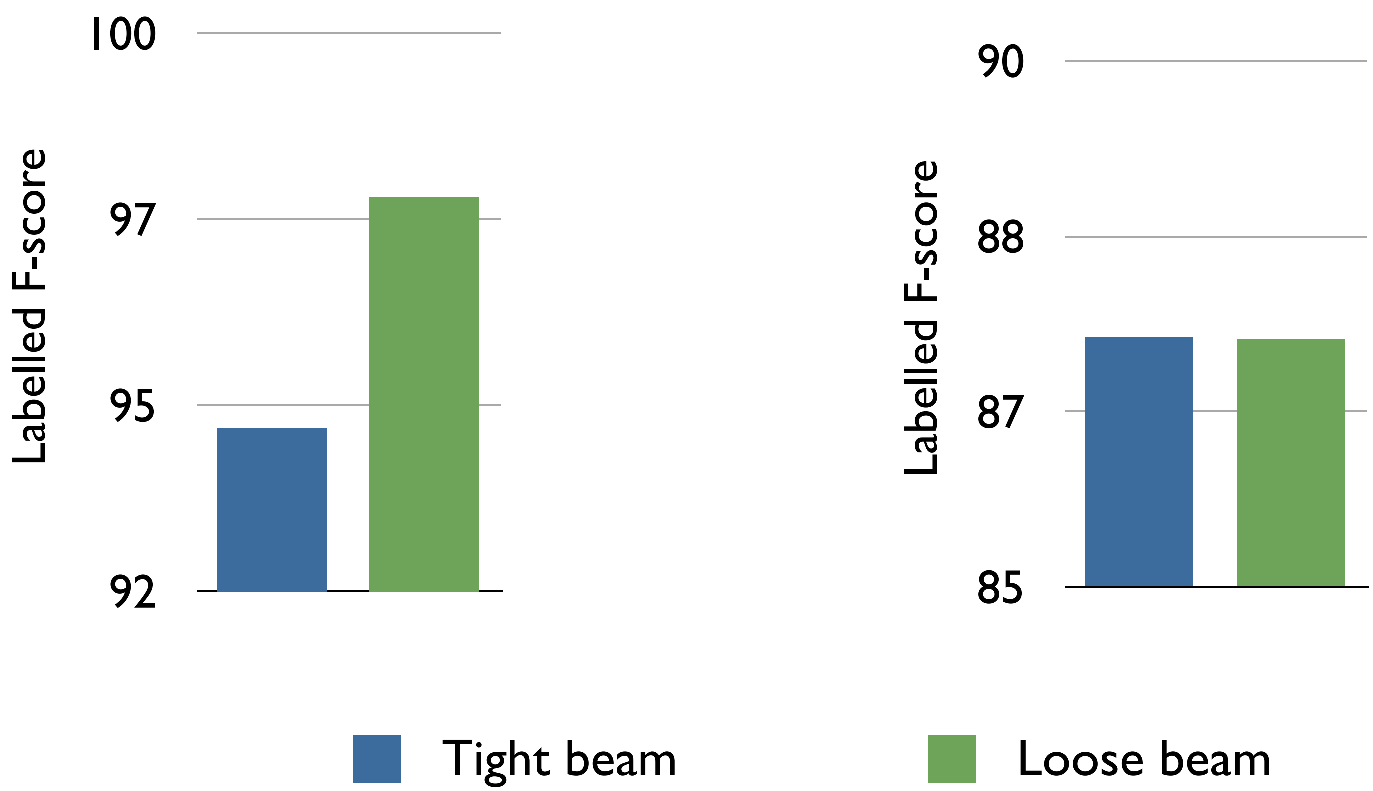
Clark & Curran (2004)

- Algorithm:
  - Run supertagger.
  - Return tags with posterior higher than some alpha.
  - Parse by combining tags (CKY).
  - If parsing succeeds, stop.
  - If parsing fails, lower alpha and repeat.
- **Q: are parses returned in early rounds suboptimal?**

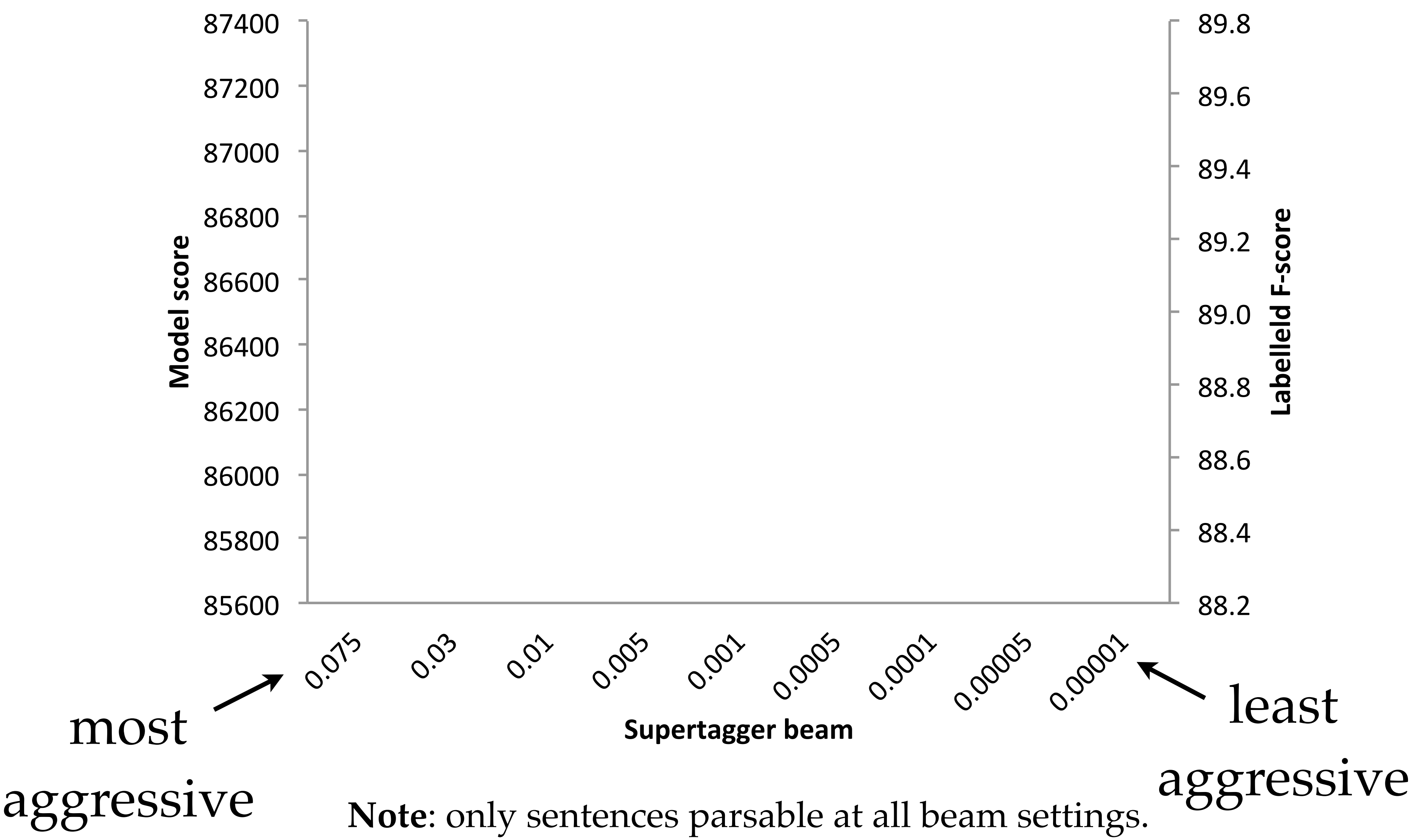
Answer...

Oracle parsing
(Huang 2008)

Standard parsing
(Clark and Curran 2007)

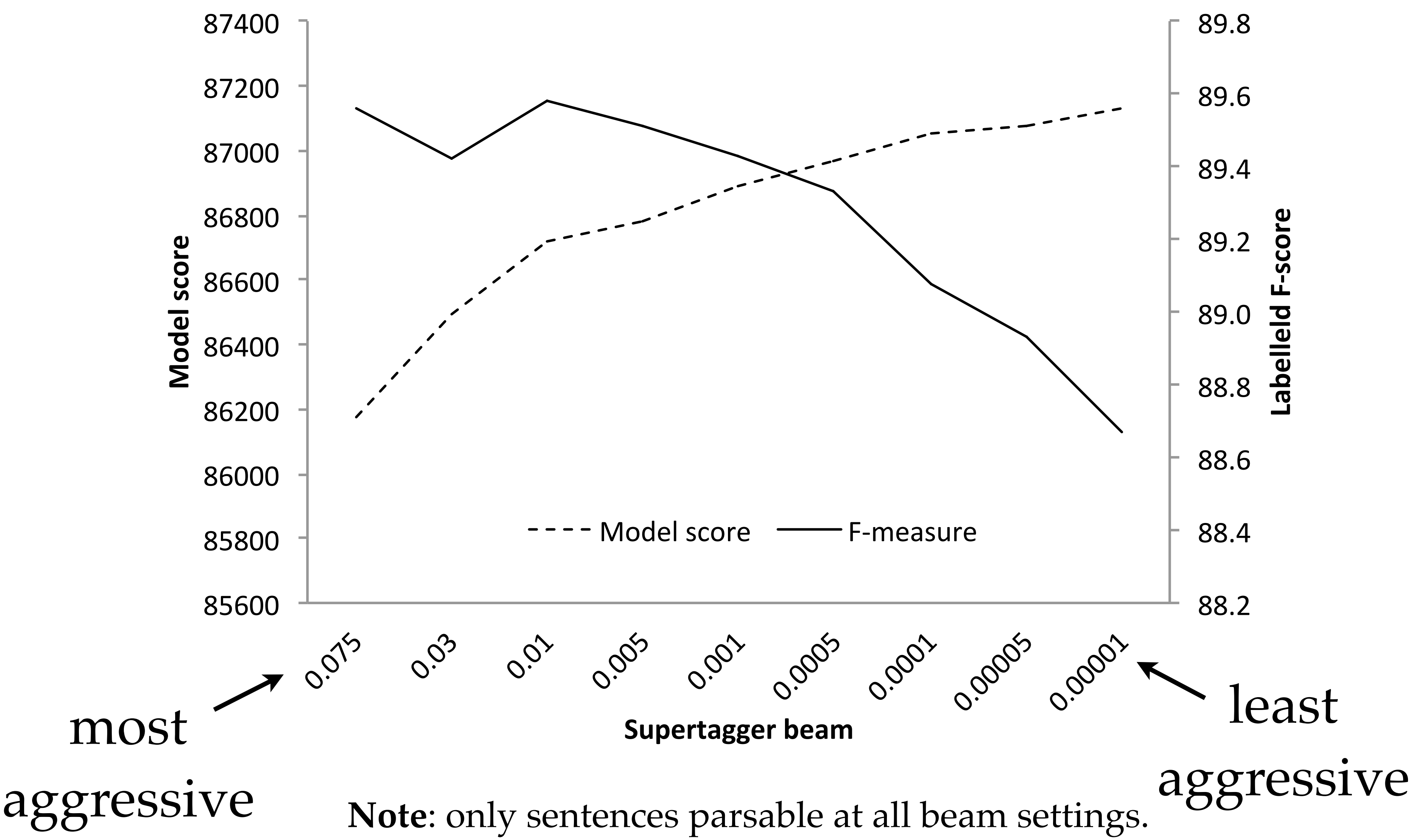


Parsing



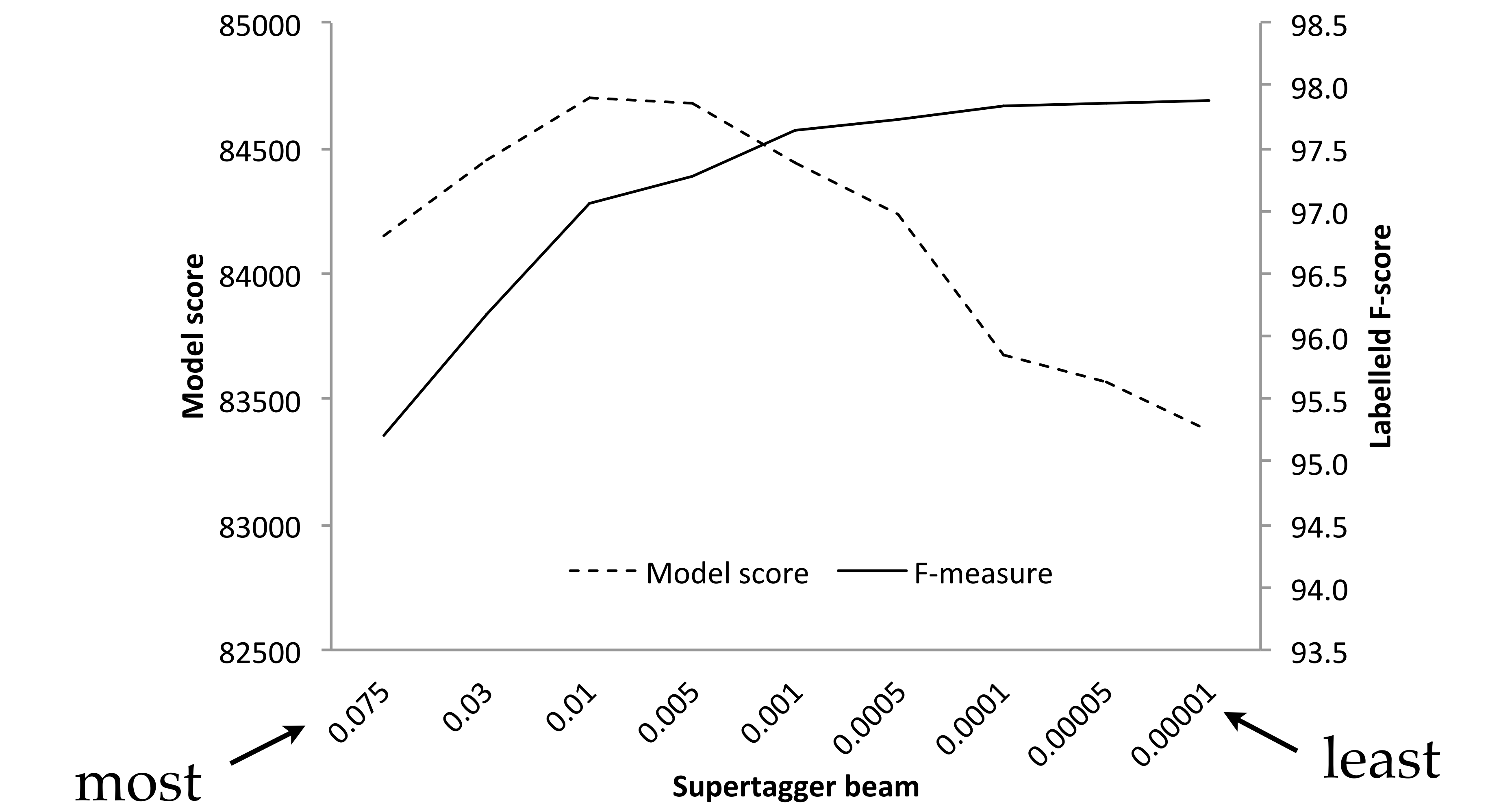
Note: only sentences parsable at all beam settings.

Parsing



Note: only sentences parsable at all beam settings.

Oracle Parsing



Note: only sentences parsable at all beam settings.

What's happening here?

- Supertagger keeps parser from making serious errors.
- But it also occasionally prunes away useful parses.
- Why not combine supertagger and parser into one?

Overview

- Analysis of state-of-the-art approach

  Trade-off between efficiency and accuracy (ACL 2011a)

- Integrated supertagging and parsing

  with Loopy Belief Propagation and Dual Decomposition (ACL 2011b)

- Training the integrated model

  with Softmax-Margin towards task-specific metrics (EMNLP 2011)

Integrated Model

- Supertagger & parser are log-linear models.
- **Idea**: combine their features into one model.
- **Problem**: Exact computation of marginal or maximum quantities becomes very expensive because parsing and tagging submodels must agree on the tag sequence.

original parsing problem: $\mathrm{B}\ \mathrm{C} \rightarrow \mathrm{A}$ $\quad O(Gn^3)$

$$\Downarrow$$

new parsing problem: ${}_q\mathrm{B}_s\ {}_s\mathrm{C}_r \rightarrow {}_q\mathrm{A}_r$ $\quad O(G^3n^3)$

Intersection of a regular and context-free language

(Bar-Hillel et al. 1964)

Approximate Algorithms

- Loopy belief propagation: approximate calculation of marginals. (Pearl 1988; Smith & Eisner 2008)
- Dual decomposition: exact (sometimes) calculation of maximum. (Dantzig & Wolfe 1960; Komodakis et al. 2007; Koo et al. 2010)

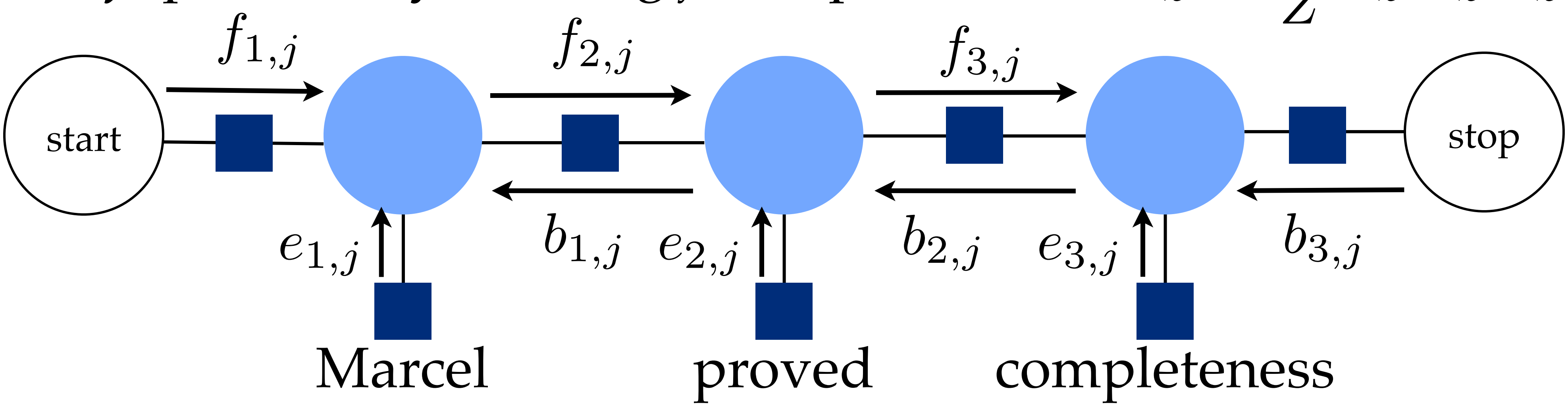
Belief Propagation

emission message: $e_{i,j}$

forward message: $f_{i,j} = \sum_{j'} f_{i-1,j'} e_{i-1,j'} t_{j',j}$

backward message: $b_{i,j} = \sum_{j'} b_{i+1,j'} e_{i+1,j'} t_{j,j'}$

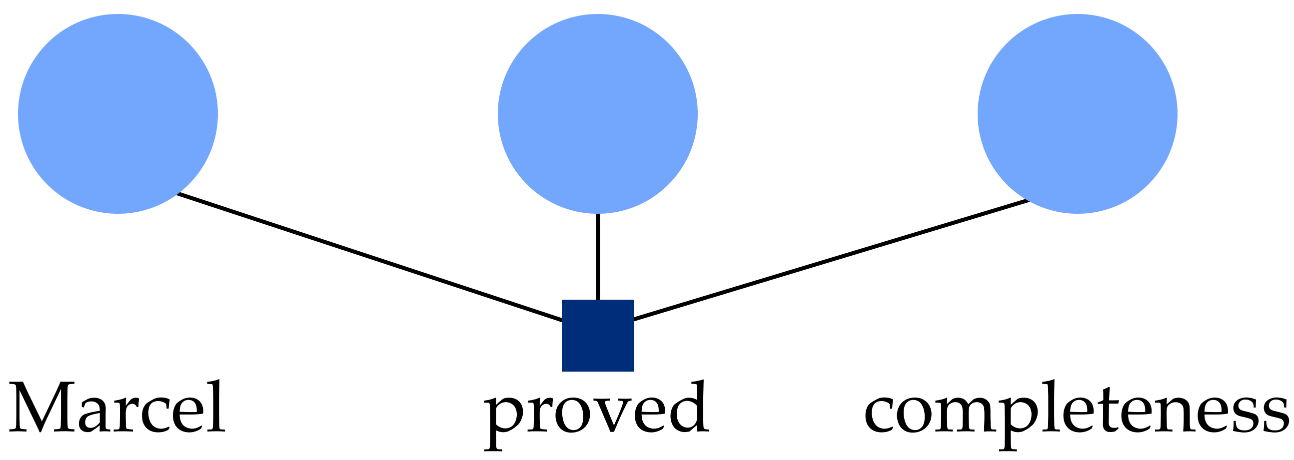
belief (probability) that tag *j* is at position *i*: $p_{i,j} = \frac{1}{Z} f_{i,j} e_{i,j} b_{i,j}$



Forward-backward is belief propagation (Smyth et al. 1997)

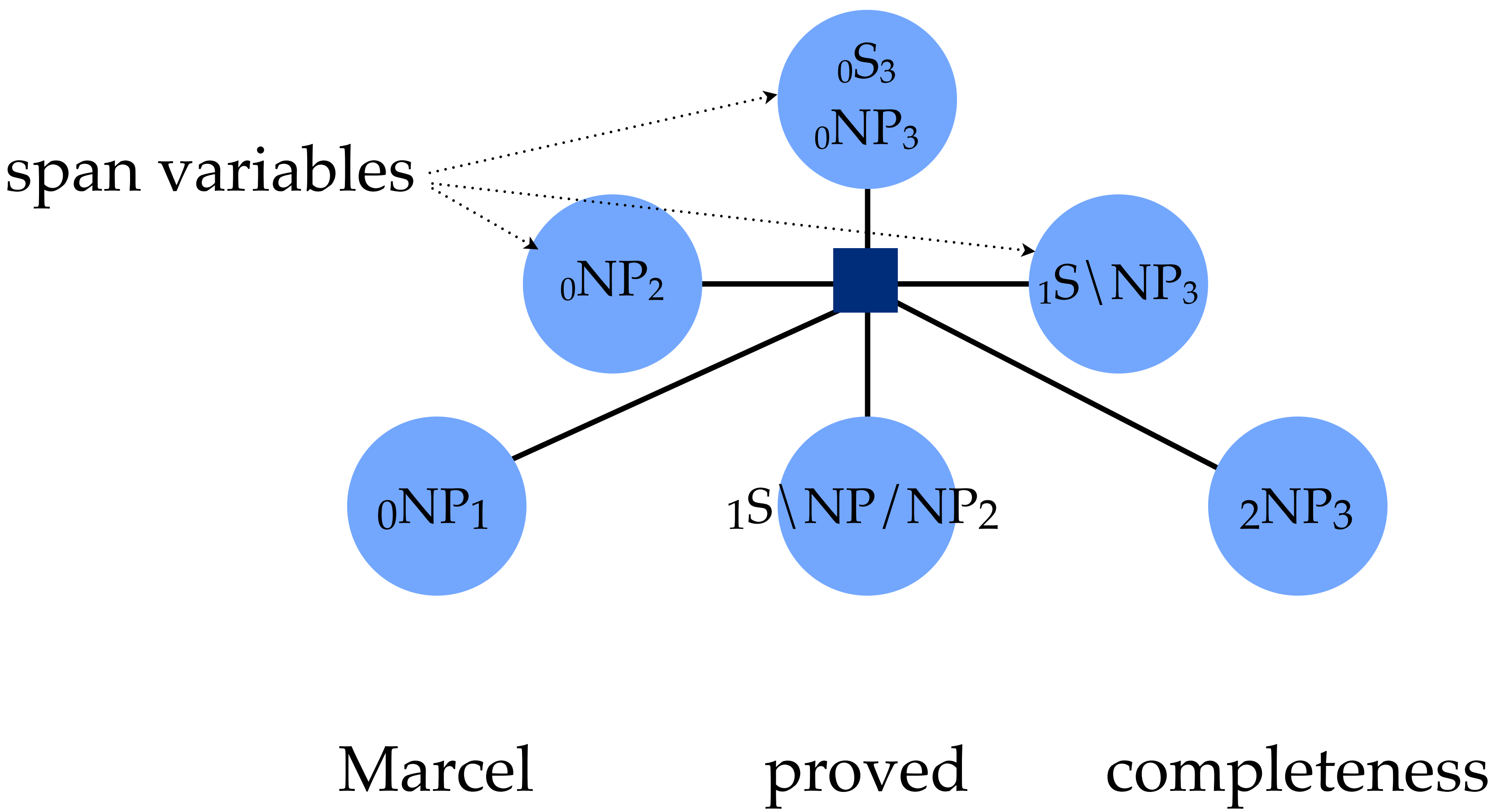
Belief Propagation

Notational convenience: one factor describes whole distribution over supertag sequence...



Belief Propagation

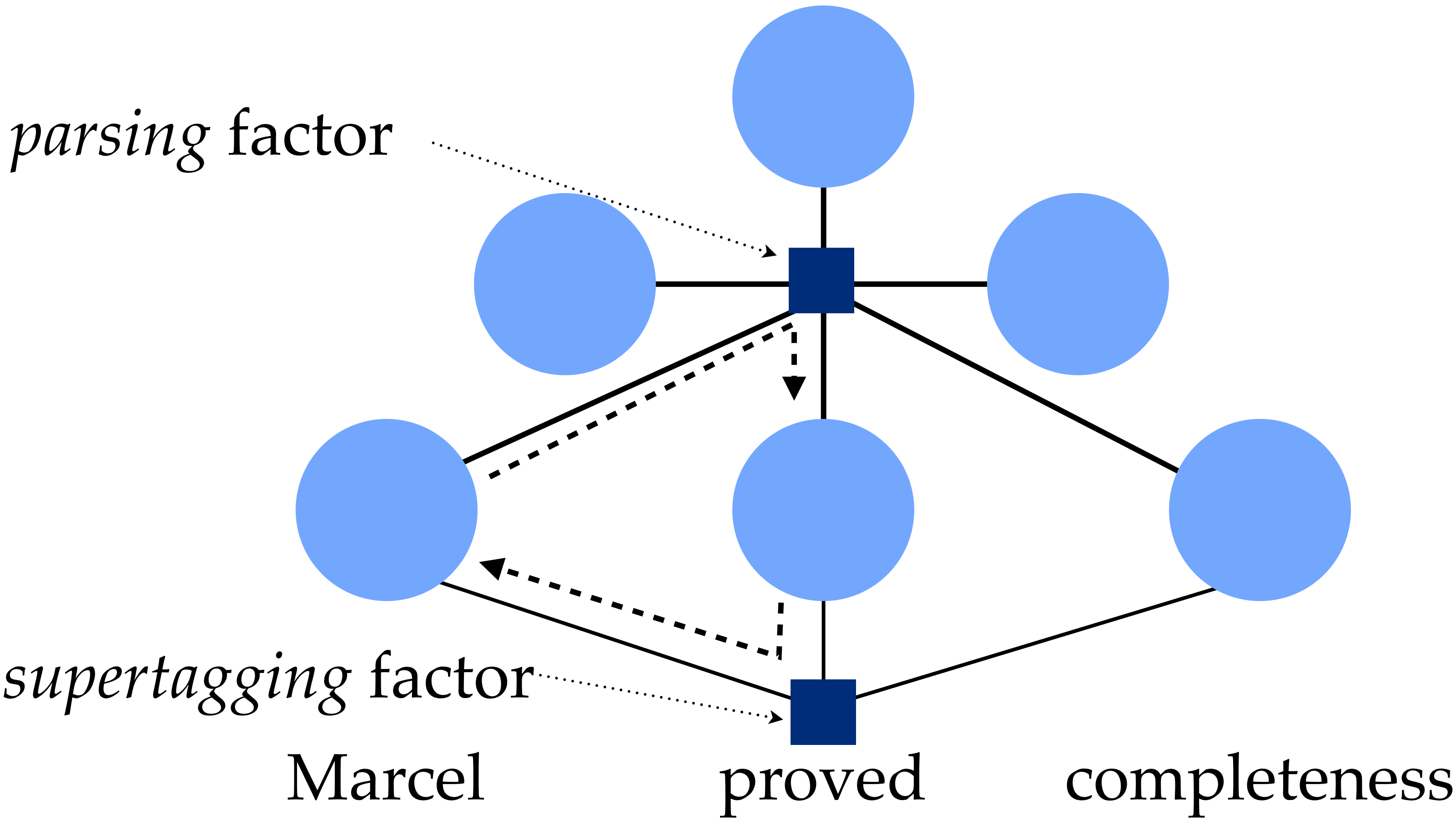
We can also do the same for the distribution over parse trees



Inside-outside is belief propagation (Sato 2007)

Belief Propagation

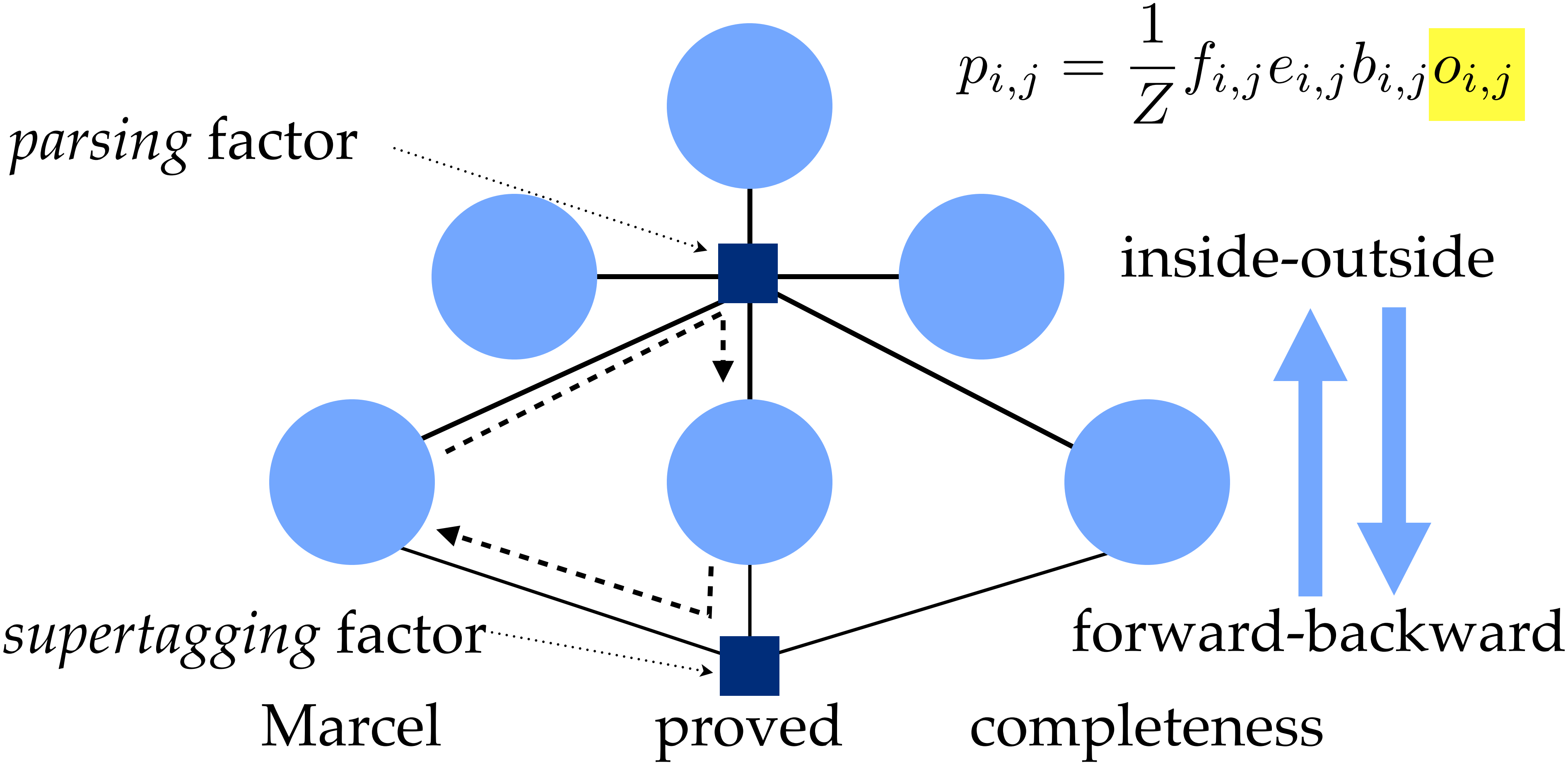
Graph is not a tree!



Loopy Belief Propagation

Graph is not a tree!

$$p_{i,j} = \frac{1}{Z} f_{i,j} e_{i,j} b_{i,j} o_{i,j}$$

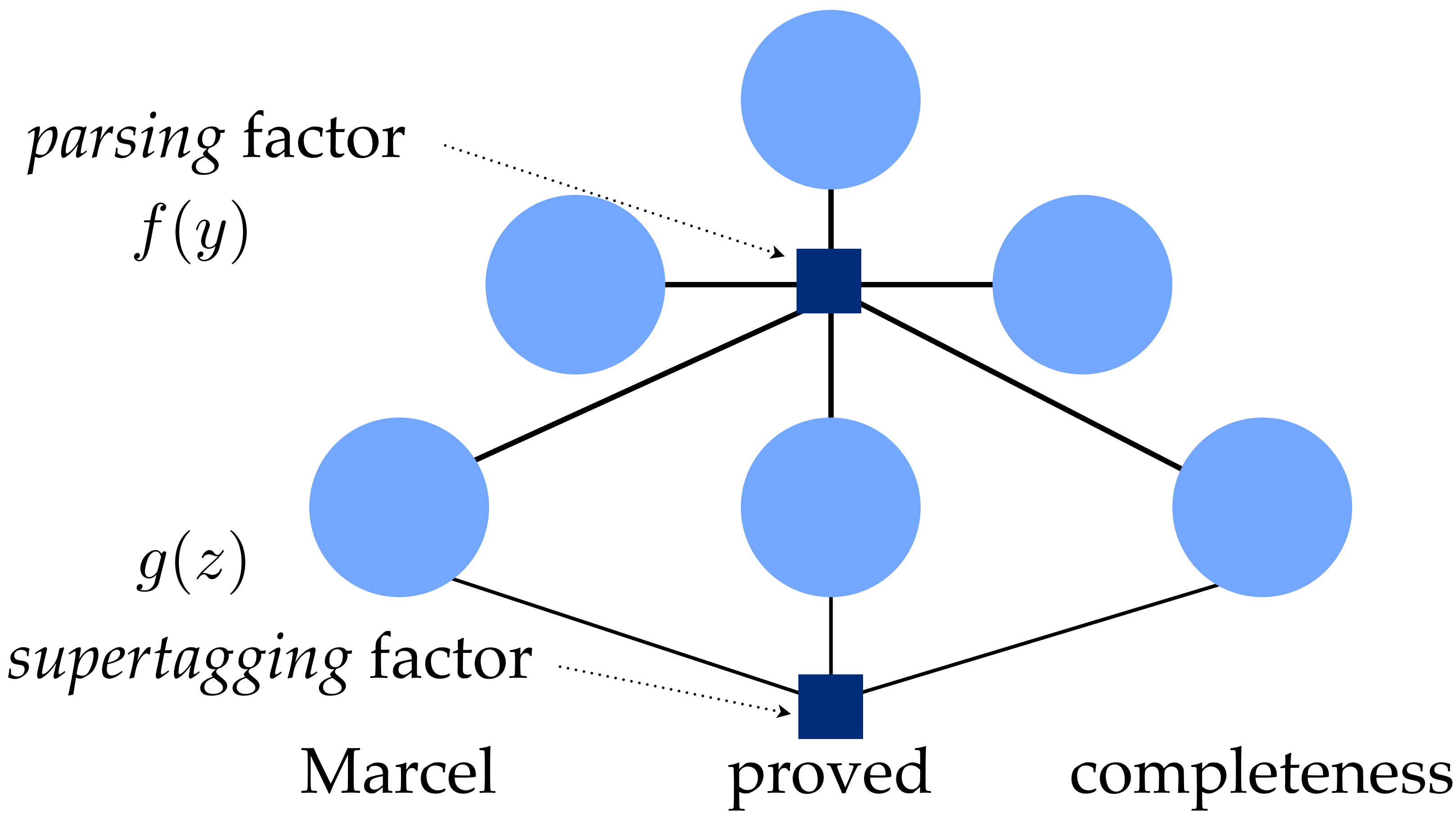


Loopy Belief Propagation

- Computes *approximate* marginals, no guarantees.
- Complexity is additive: $O(Gn^3 + Gn)$
- Used to compute minimum-risk parse (Goodman 1996).

Dual Decomposition

$\arg\max_{y,z} f(y) + g(z)$ s.t. $y(i,t) = z(i,t)$ for all *i*, *t*



Dual Decomposition

$$\arg\max_{y,z} f(y) + g(z) \qquad \text{s.t. } y(i,t) = z(i,t) \text{ for all } i, t$$

$$L(u) = \max_y f(y) + \sum_{i,t} u(i,t) \cdot y(i,t)$$

relaxed
modified
original
subproblem
problem

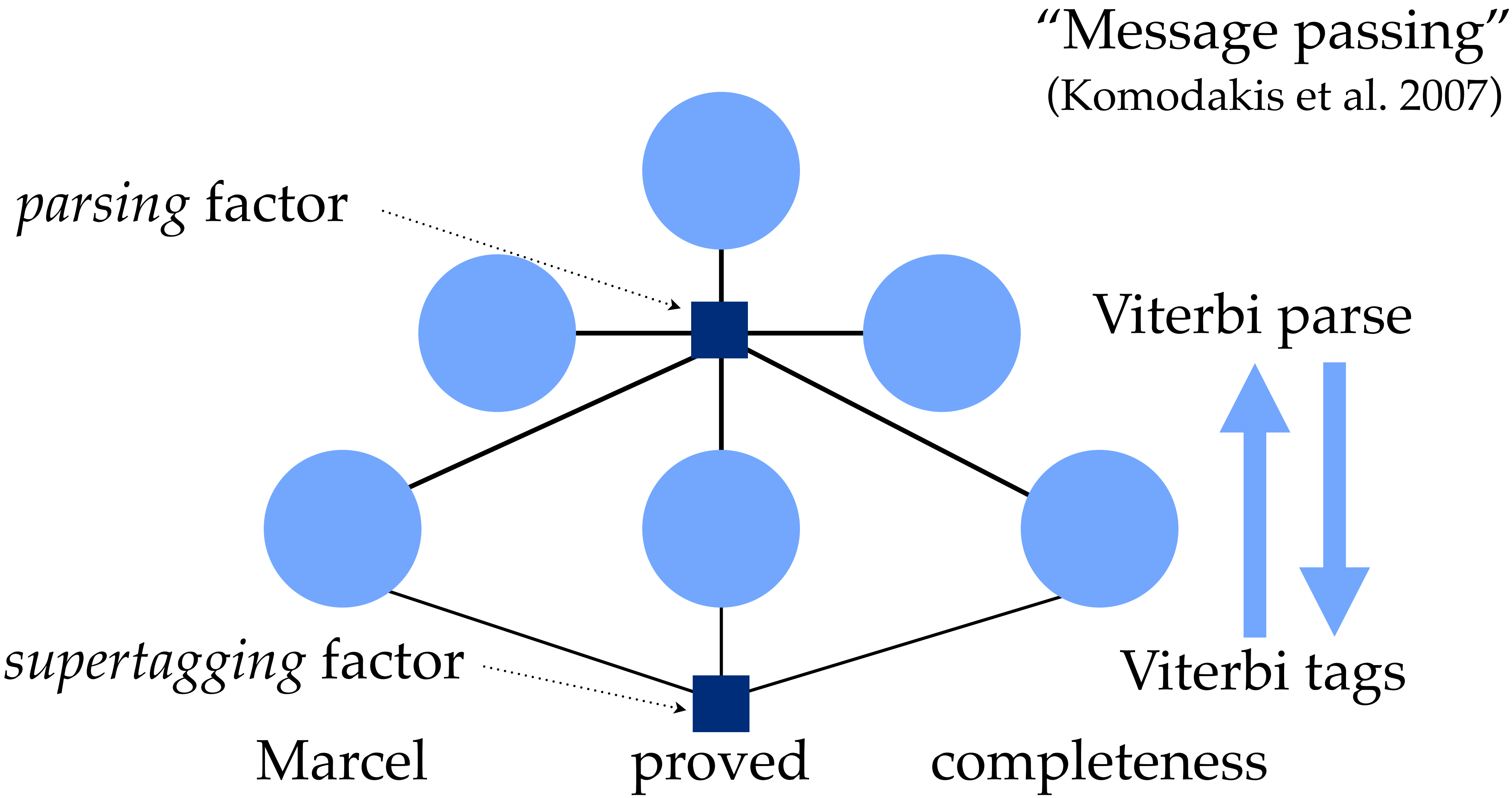
$$+ \max_z g(z) - \sum_{i,t} u(i,t) \cdot z(i,t)$$

Dual objective: find assignment of $u(i,t)$ that minimises $L(u)$

$$u(i,t) = u(i,t) + \alpha \cdot [y(i,t) - z(i,t)]$$ (Rush et al. 2010)

Solution provably solves original problem.

Dual Decomposition



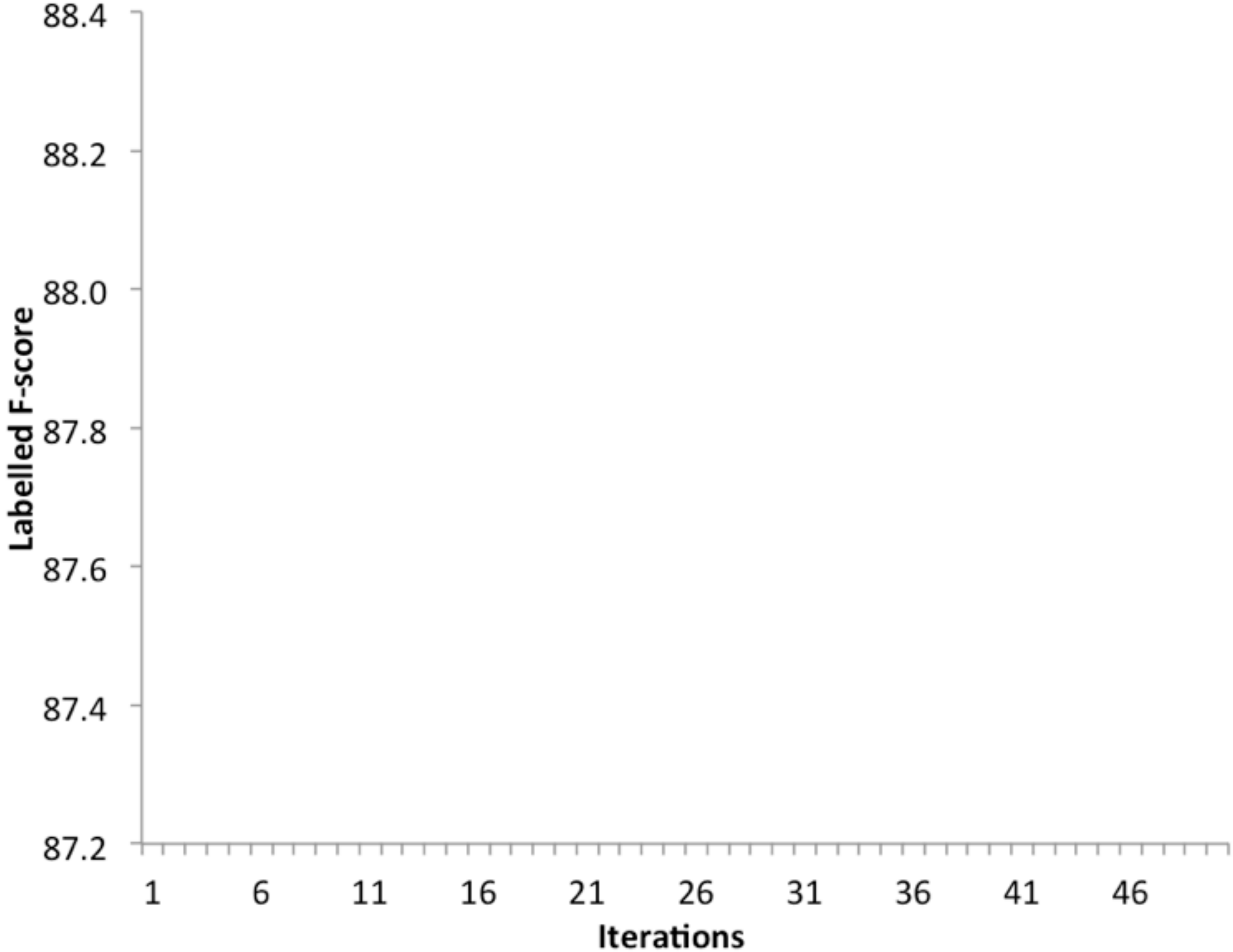
Dual Decomposition

- Computes *exact* maximum, *if* it converges.
    - Otherwise: return best parse seen (approximation).
- Complexity is additive: $O(Gn^3 + Gn)$
- Use to compute Viterbi solutions.

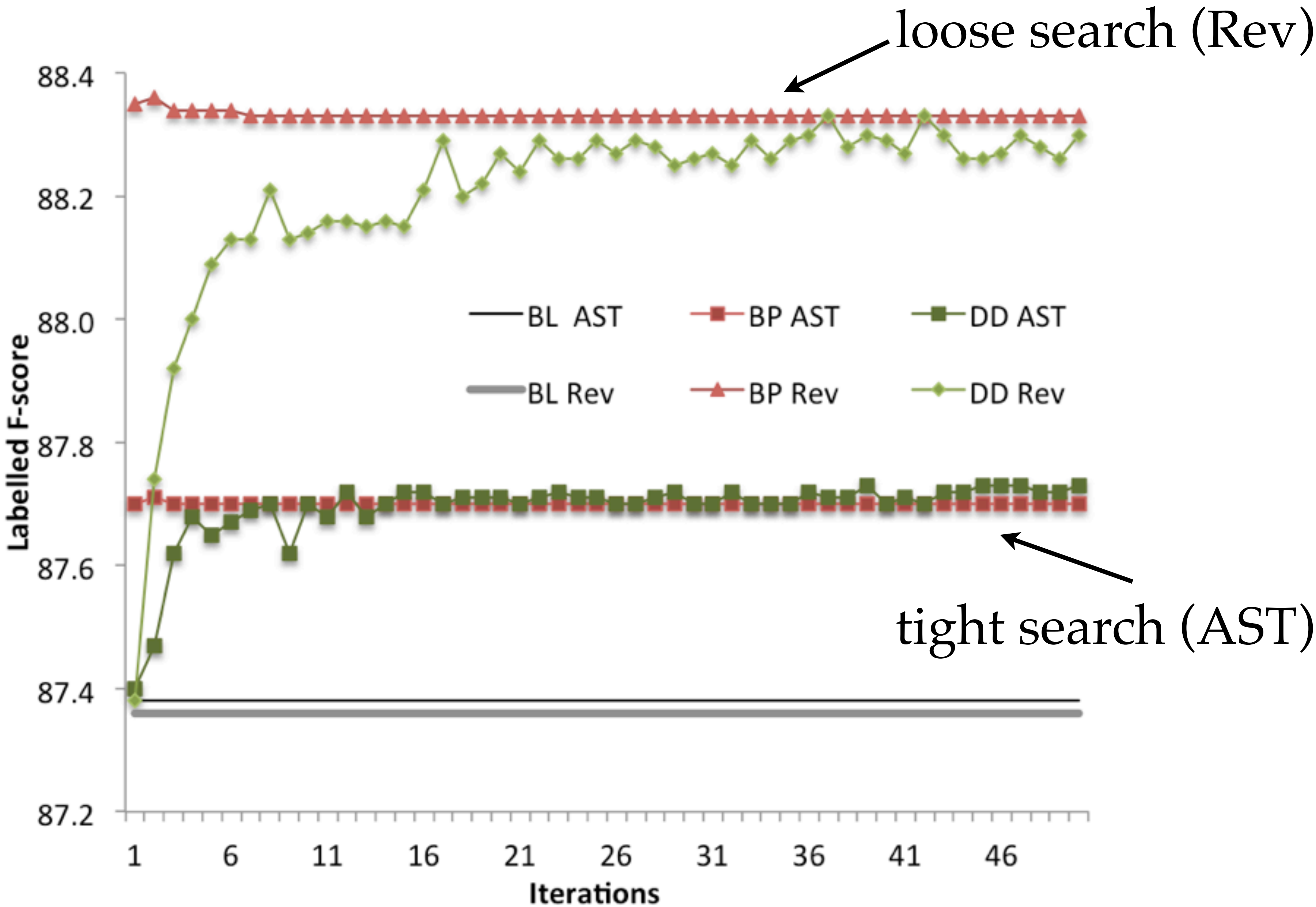
Experiments

- Standard parsing task:
  - C&C Parser and supertagger (Clark & Curran 2007).
  - CCGBank standard train/dev/test splits.
  - Piecewise optimisation (Sutton and McCallum 2005)
  - Approximate algorithms used to decode test set.

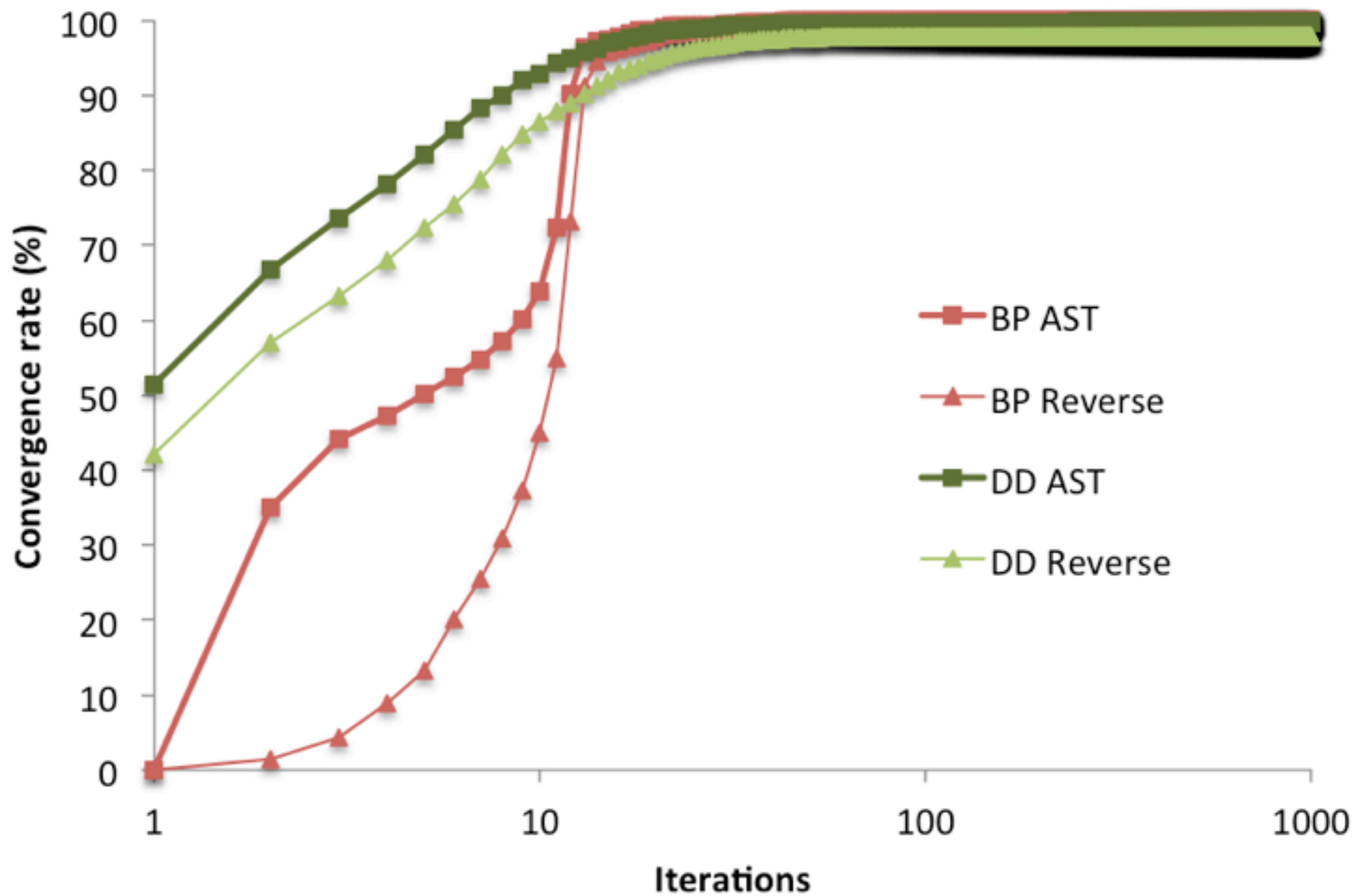
Experiments: Accuracy over time



Experiments: Accuracy over time

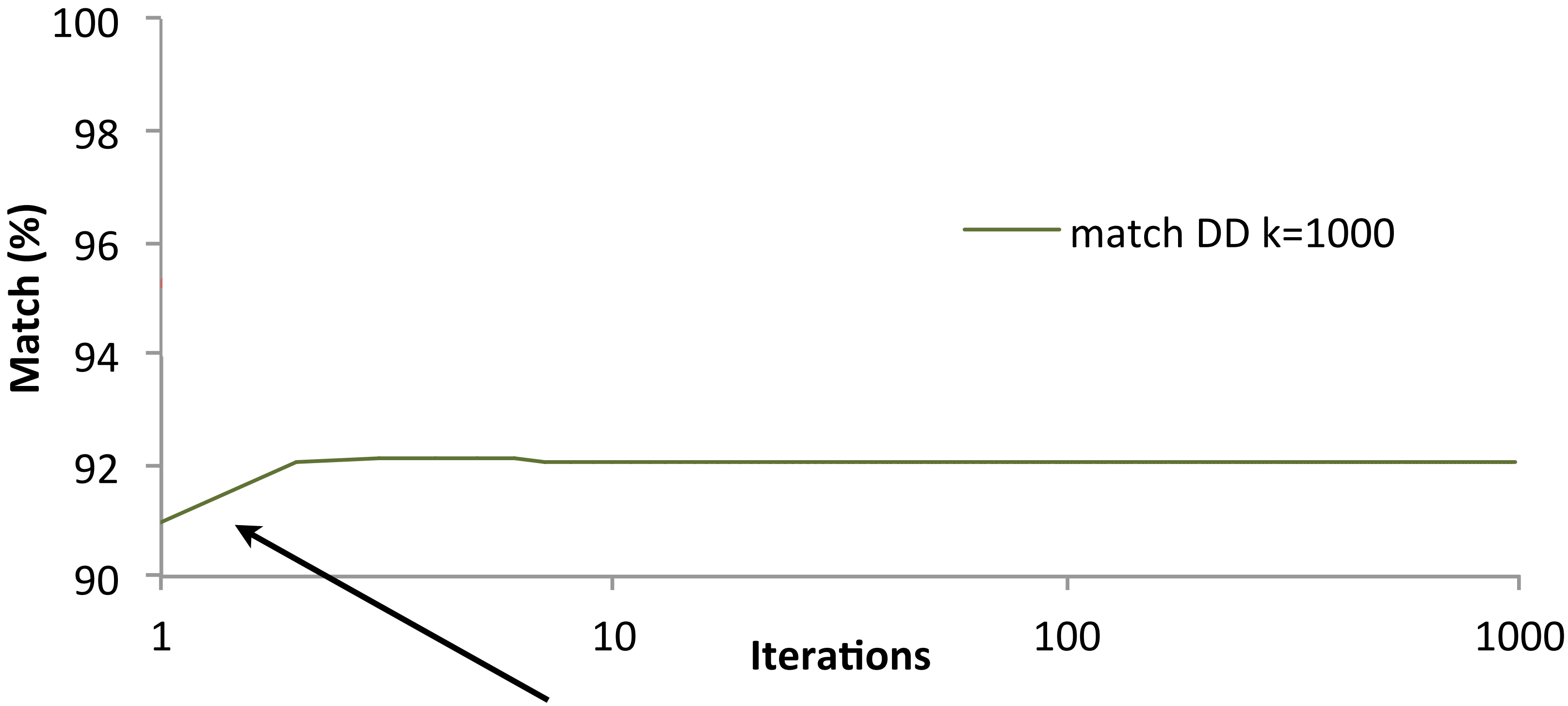


Experiments: Convergence



Dual decomposition exact in 99.7% of cases
What about belief propagation?

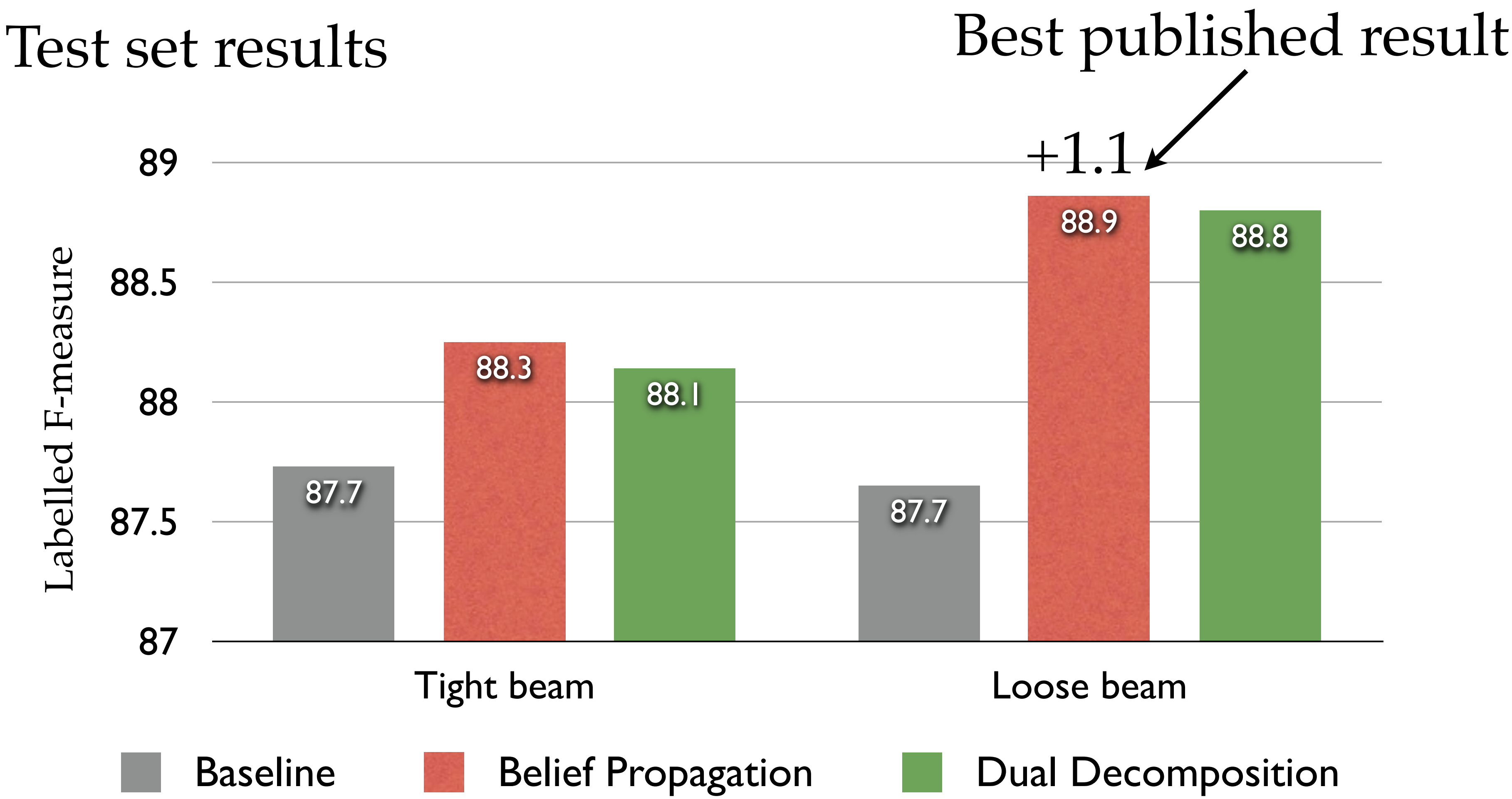
Experiments: BP Exactness



Instantly, 91% match final DD solutions!

Takes DD 15 iterations to reach same level.

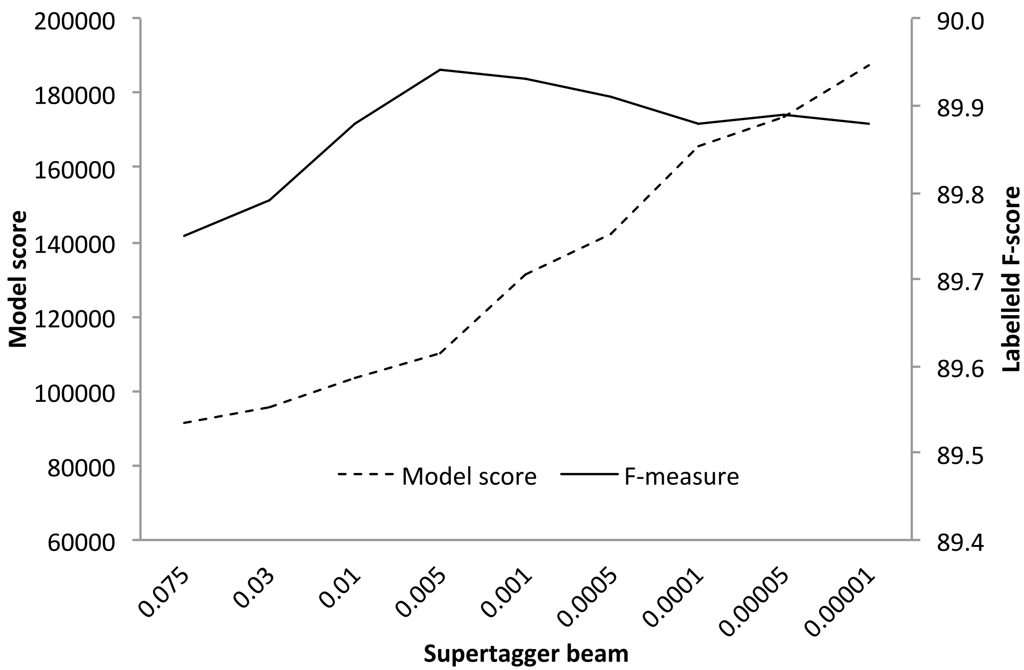
Experiments: Accuracy



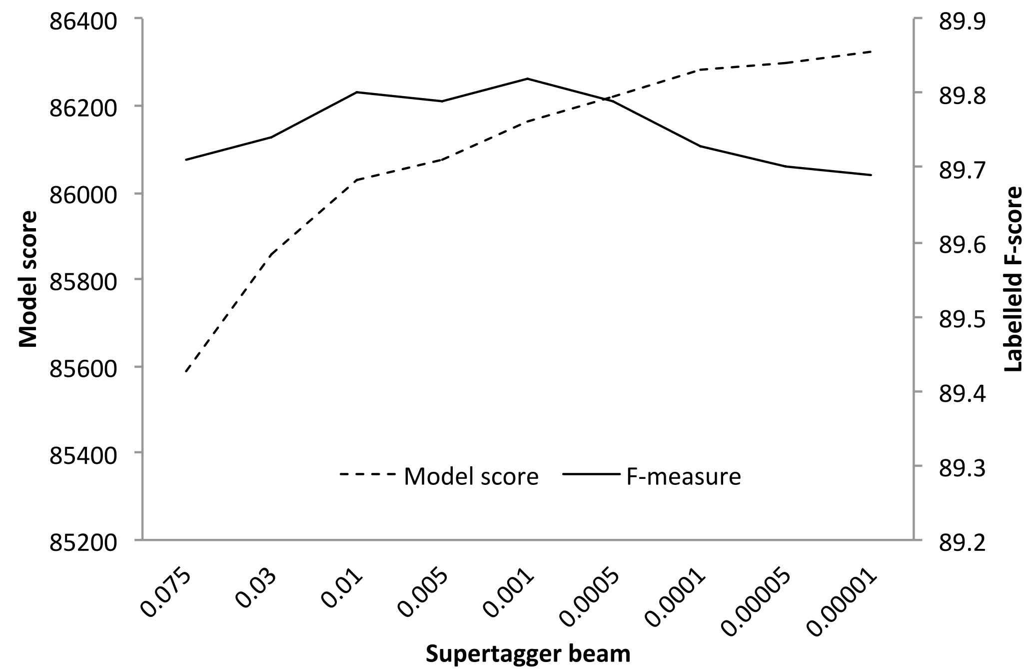
Note: BP accuracy after 1 iteration; DD accuracy after 25 iterations

Oracle Results Again

Belief Propagation



Dual Decomposition



Summary so far

- Supertagging efficiency comes at the cost of accuracy.
- Interaction between parser and supertagger can be exploited in an integrated model.
- Practical inference for complex integrated model.
- First empirical comparison between dual decomposition and belief propagation on NLP task.
- Loopy belief propagation is fast, accurate and exact.

Overview

- Analysis of state-of-the-art approach

  Trade-off between efficiency and accuracy (ACL 2011a)

- Integrated supertagging and parsing

  with Loopy Belief Propagation and Dual Decomposition (ACL 2011b)

- Training the integrated model

  with Softmax-Margin towards task-specific metrics (EMNLP 2011)

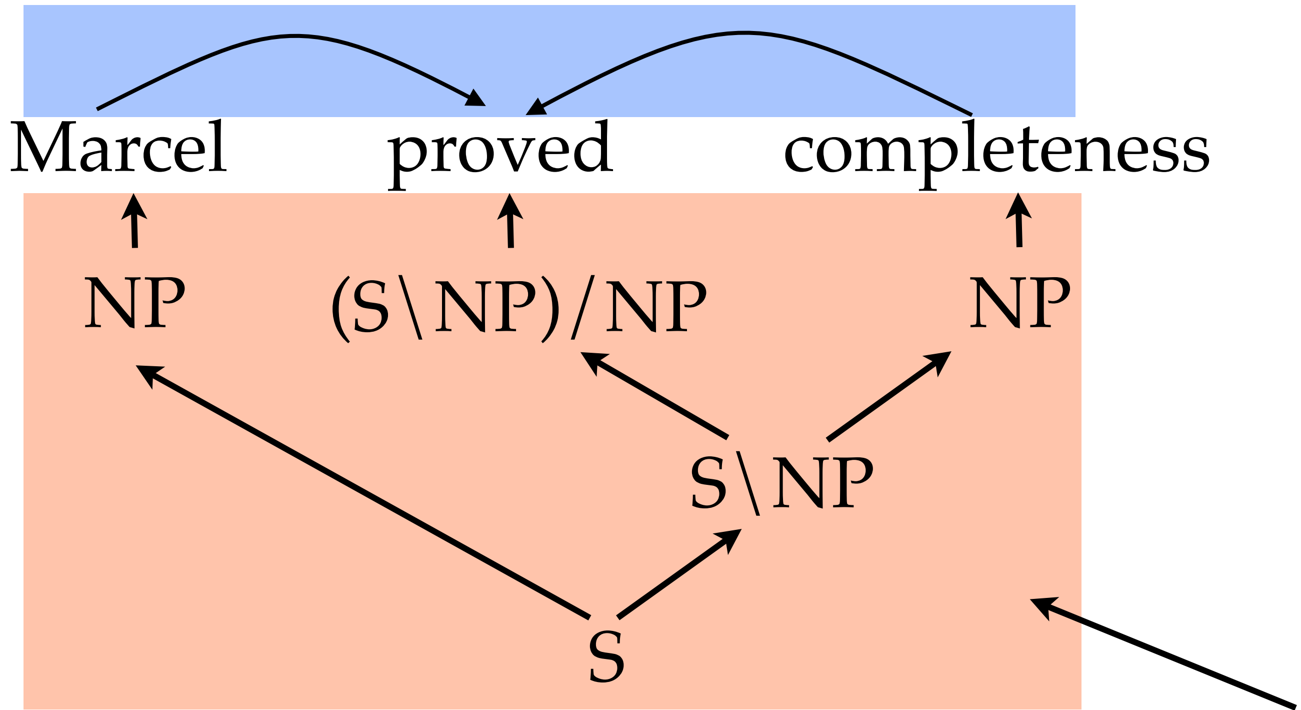
Training the Integrated Model

- So far optimised Conditional Log-Likelihood (CLL).
- Optimise towards task-specific metric e.g. F_1 such as in SMT (Och, 2003).
- Past work used approximations to Precision (Taskar et al. 2004).
- Contribution: Do it **exactly** and verify approximations.

Parsing Metrics

CCG: Labelled, directed dependency recovery

(Clark & Hockenmaier, 2002)



Not this!

<proved, (S\NP)/NP, completeness>

<proved, (S\NP)/NP, Marcel>

Evaluate this

Parsing Metrics

y = dependencies in ground truth

y' = dependencies in proposed output

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

Precision $$P(y, y') = \frac{|y \cap y'|}{|y'|} = \frac{n}{d}$$

Recall $$R(y, y') = \frac{|y \cap y'|}{|y|} = \frac{n}{|y|}$$

F-measure $$F_1(y, y') = \frac{2PR}{P + R} = \frac{2|y \cap y'|}{|y| + |y'|} = \frac{2n}{d + |y|}$$

Softmax-Margin Training

(Sha & Saul, 2006; Povey & Woodland,2008; Gimpel & Smith, 2010)

- Discriminative.
- Probabilistic.
- Convex objective.
- Minimises bound on expected risk for a given loss function.
- Requires little change to existing CLL implementation.

Softmax-Margin Training

training examples weights features input *true* output possible outputs *proposed* output

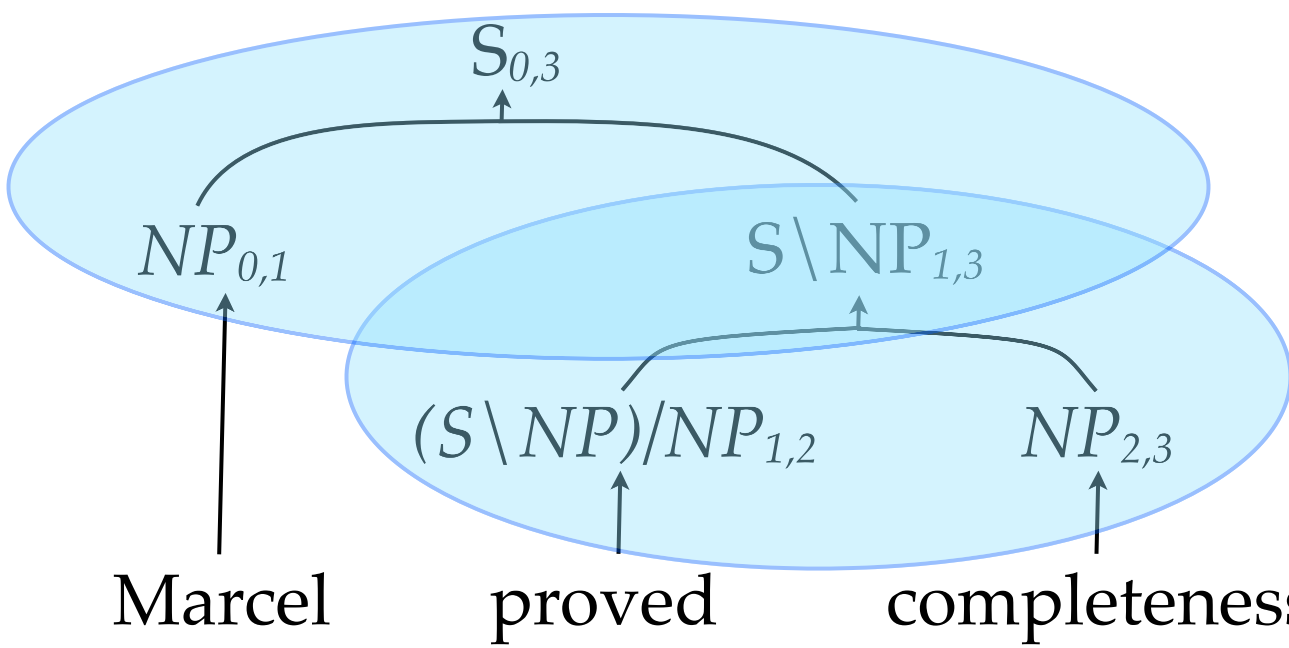
CLL: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\top} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\top} f(x^{(i)}, y)\} \right]$$

SMM: $$\min_{\theta} \sum_{i=1}^{m} \left[-\theta^{\top} f(x^{(i)}, y^{(i)}) + \log \sum_{y \in \mathcal{Y}(x^{(i)})} \exp\{\theta^{\top} f(x^{(i)}, y) + \ell(y^{(i)}, y)\} \right]$$

- Penalise high-loss outputs.
- Re-weight outcomes by *loss function*.
- Loss function an *unweighted feature* -- if **decomposable.**

Decomposability

- CKY assumes weights factor over substructures (node + children = substructure).
- A *decomposable* loss function must factor identically.



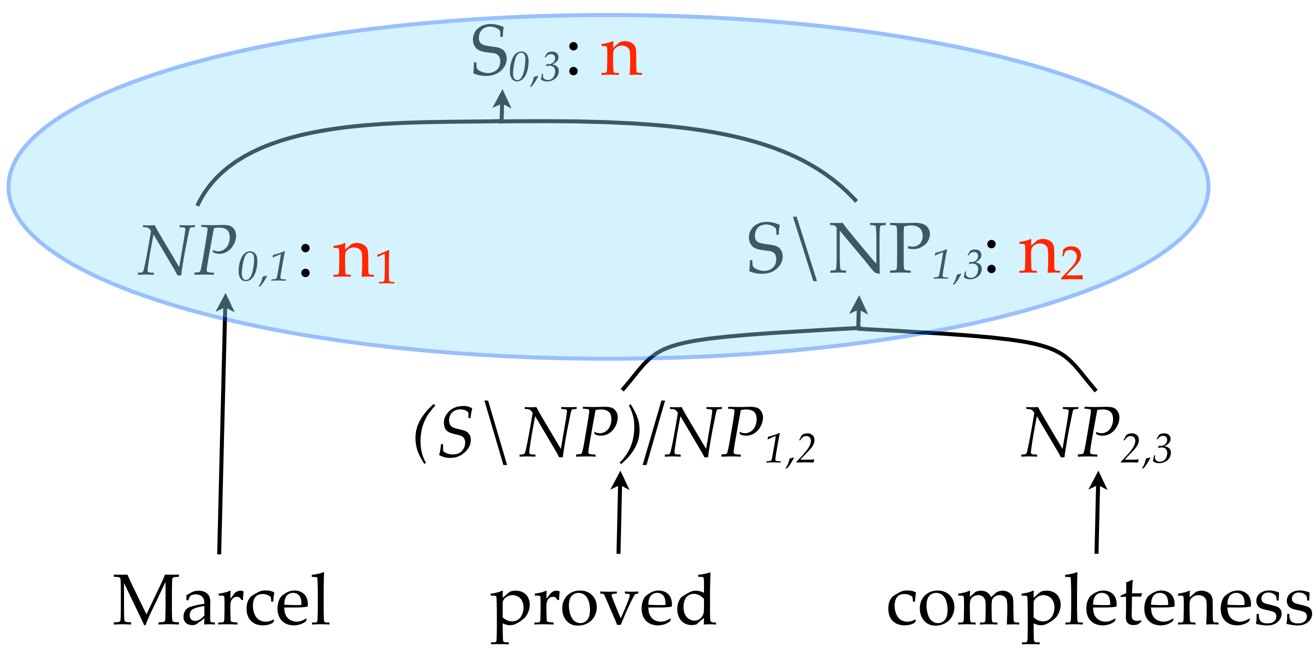
Decomposability

Correct dependency counts

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

$n = n_1 + n_2$



Decomposability

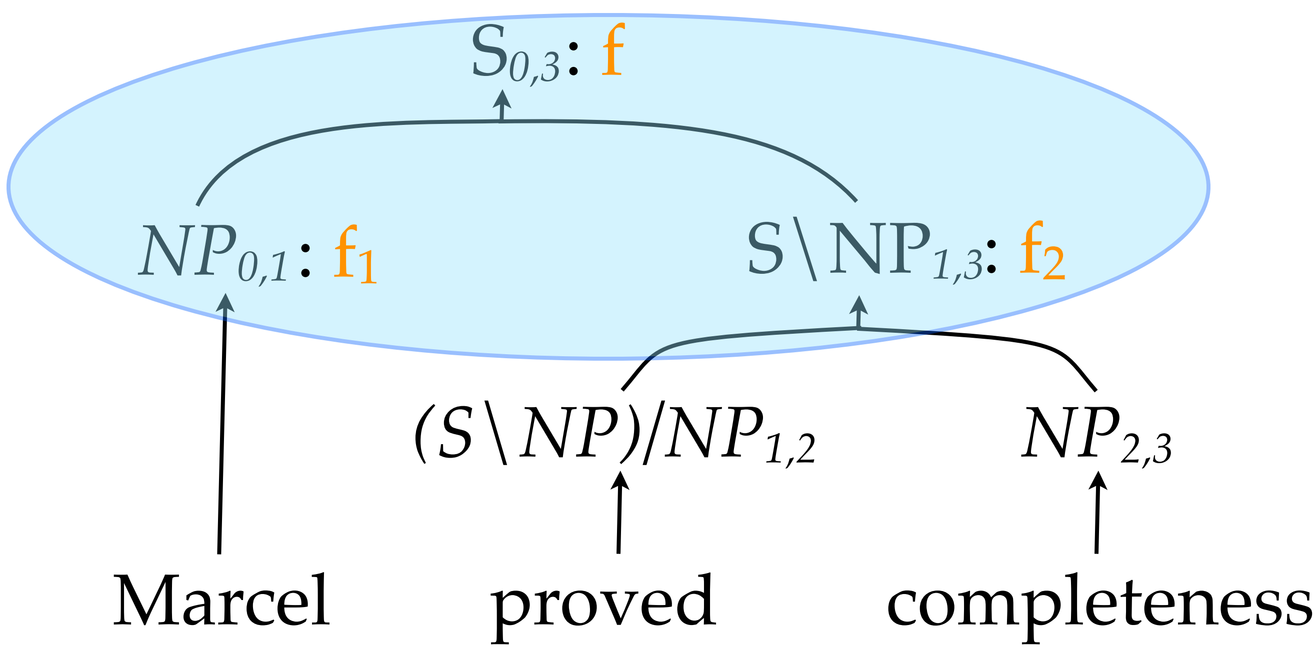
F-measure

$|y \cap y'| = n$ correct dependencies returned

$|y'| = d$ all dependencies returned

$f = f_1 \otimes f_2$

Approximations!



Approximate Loss Functions

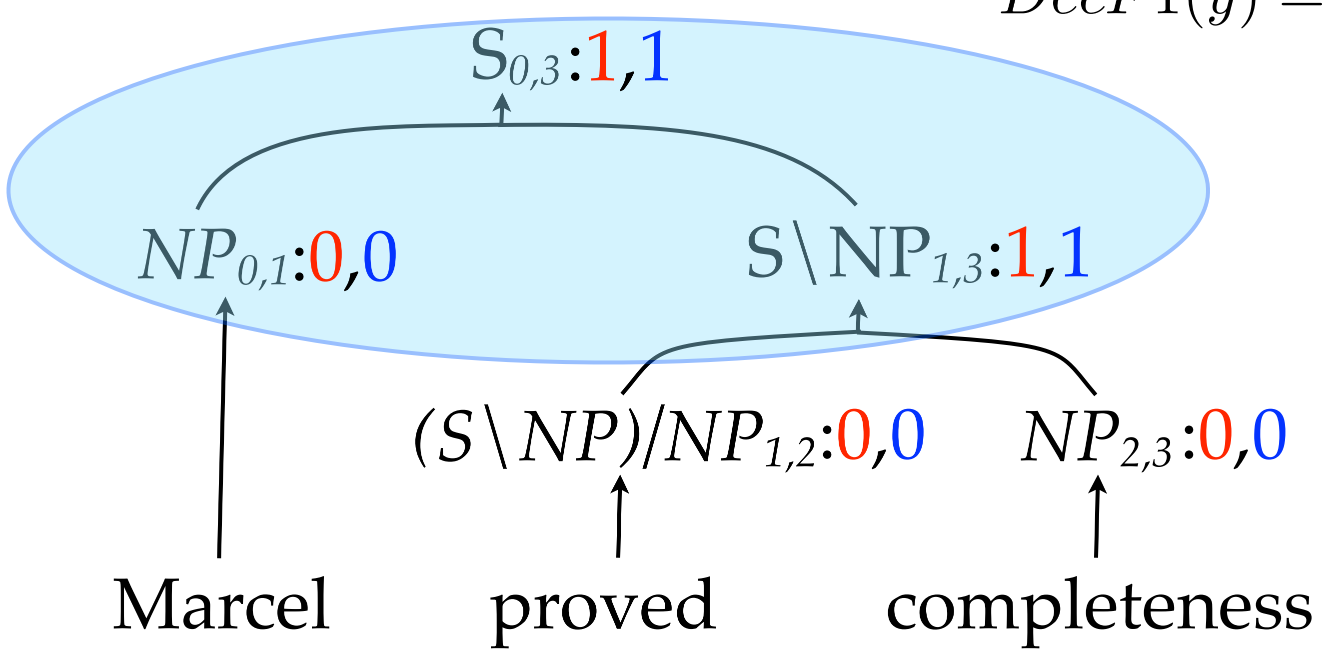
for each substructure:

- n_+ correct dependencies
- d_+ all dependencies
- c_+ gold dependencies

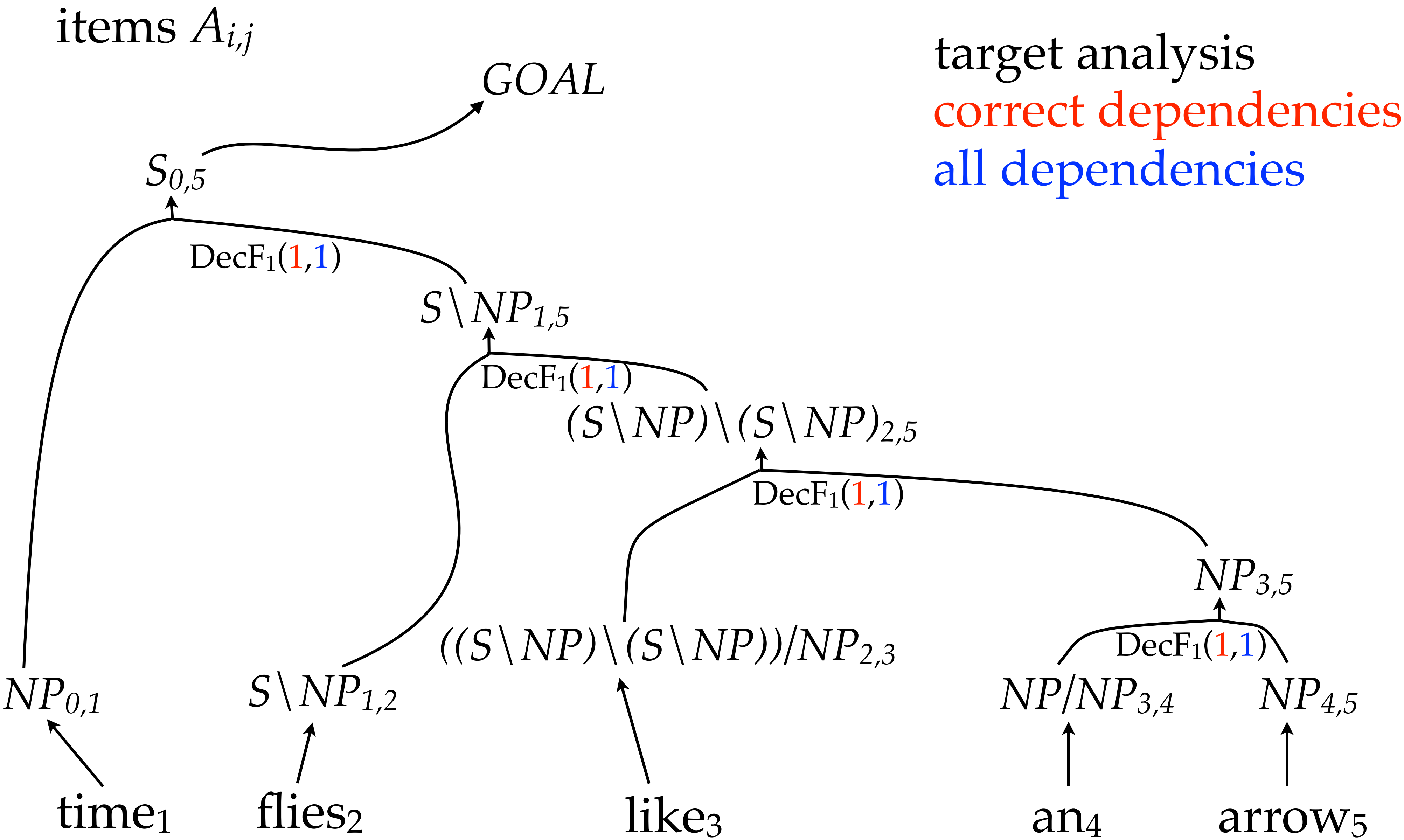
$$DecP(y) = \sum_{t \in T(y)} d_+(t) - n_+(t)$$

$$DecR(y) = \sum_{t \in T(y)} c_+(t) - n_+(t)$$

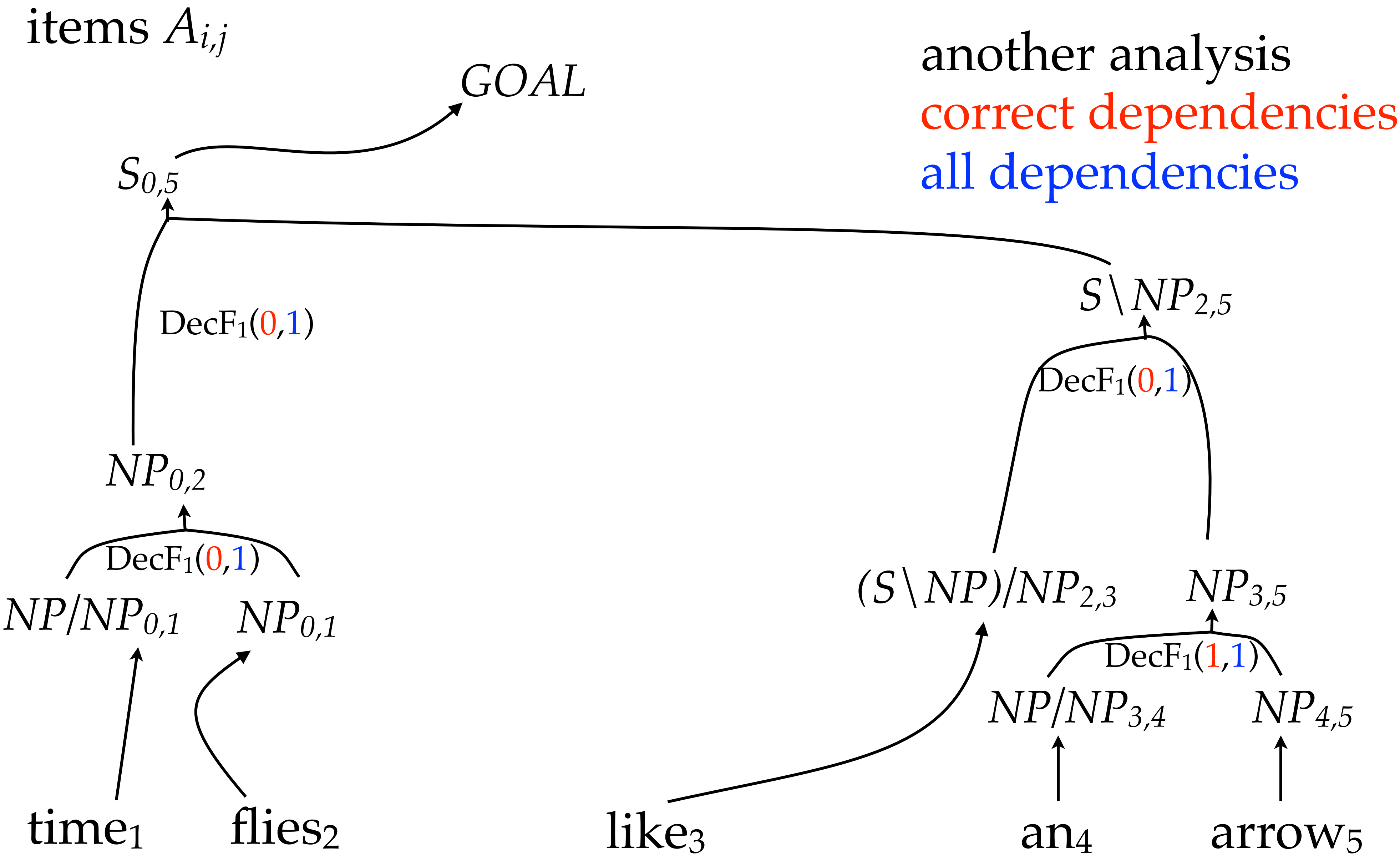
$$DecF1(y) = DecP(y) + DecR(y)$$



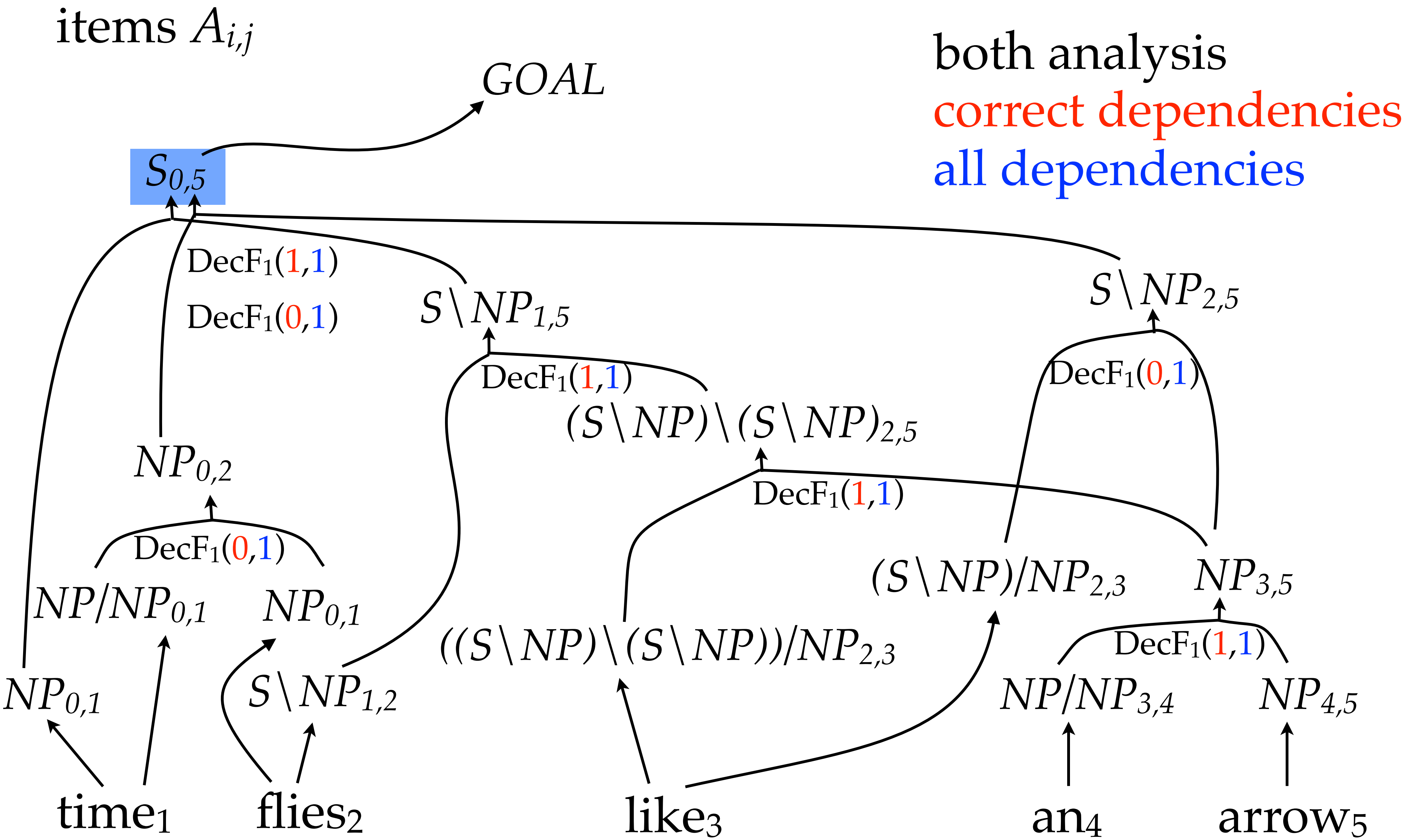
Approximate Losses with CKY



Approximate Losses with CKY



Approximate Losses with CKY



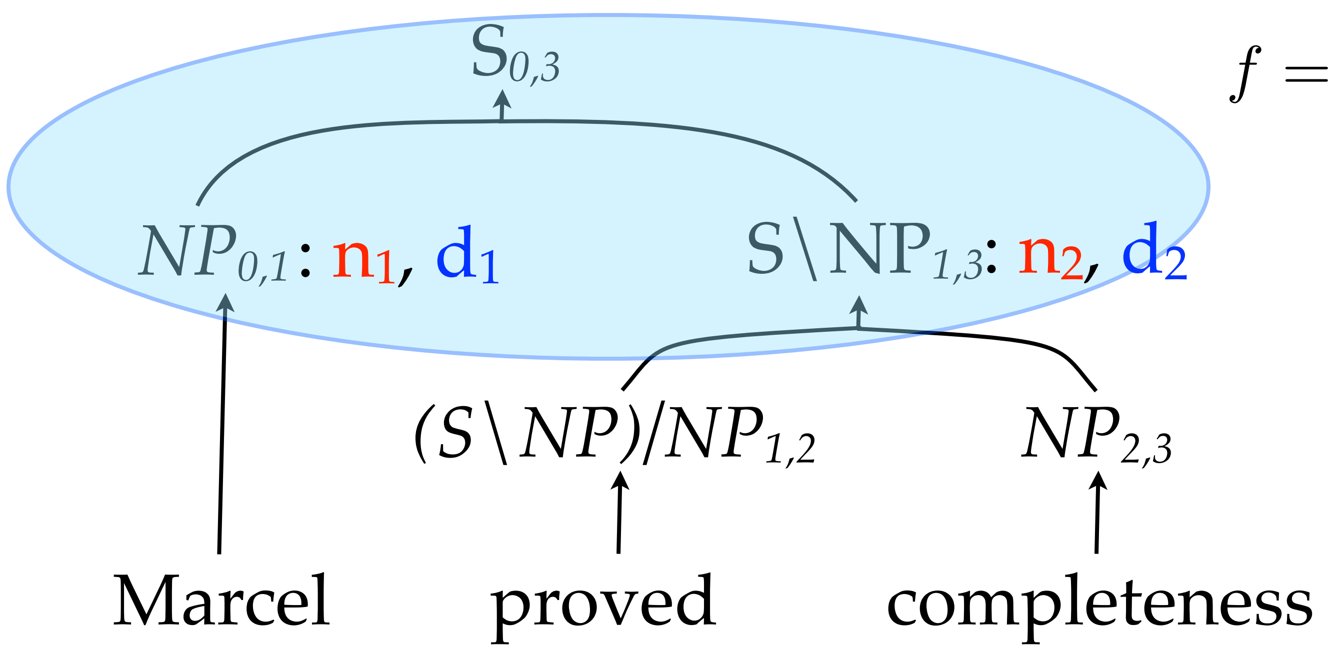
Decomposability Revisited

F-measure

$$|y \cap y'| = n \qquad \text{correct dependencies returned}$$

$$|y'| = d \qquad \text{all dependencies returned}$$

$$F_1(y, y') = \frac{2n}{d + |y|}$$

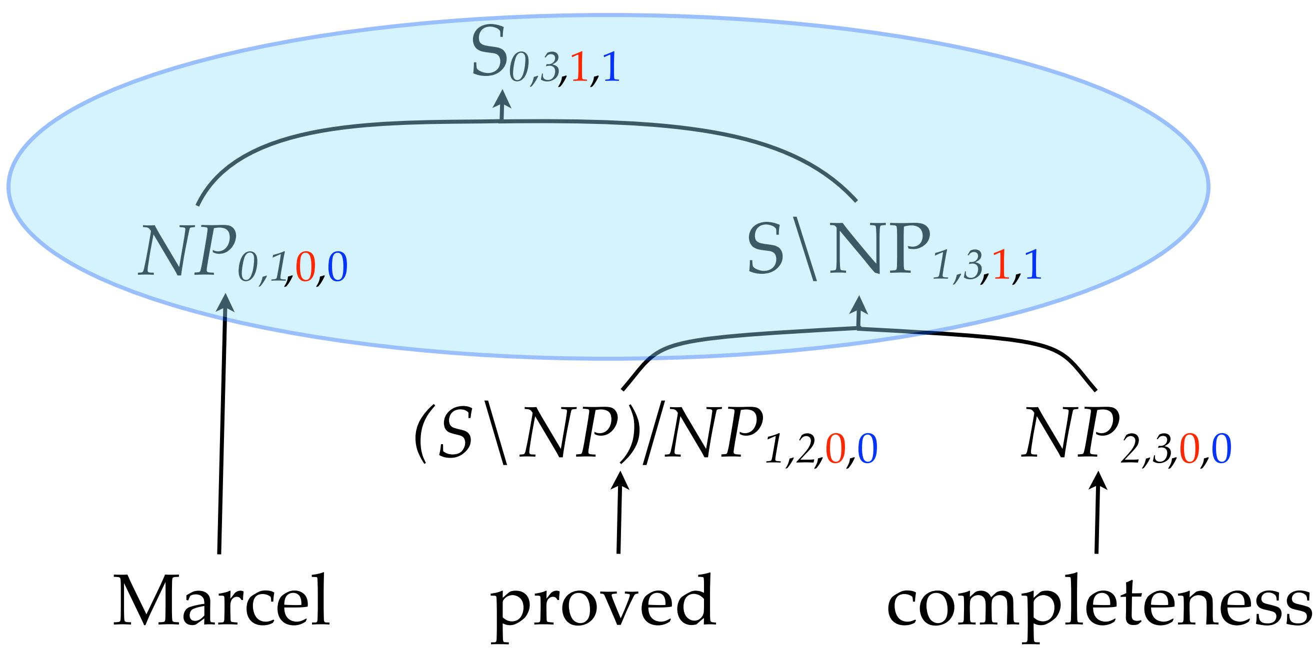


$$f = \frac{2n_1}{d_1 + |y|} \otimes \frac{2n_2}{d_2 + |y|}$$

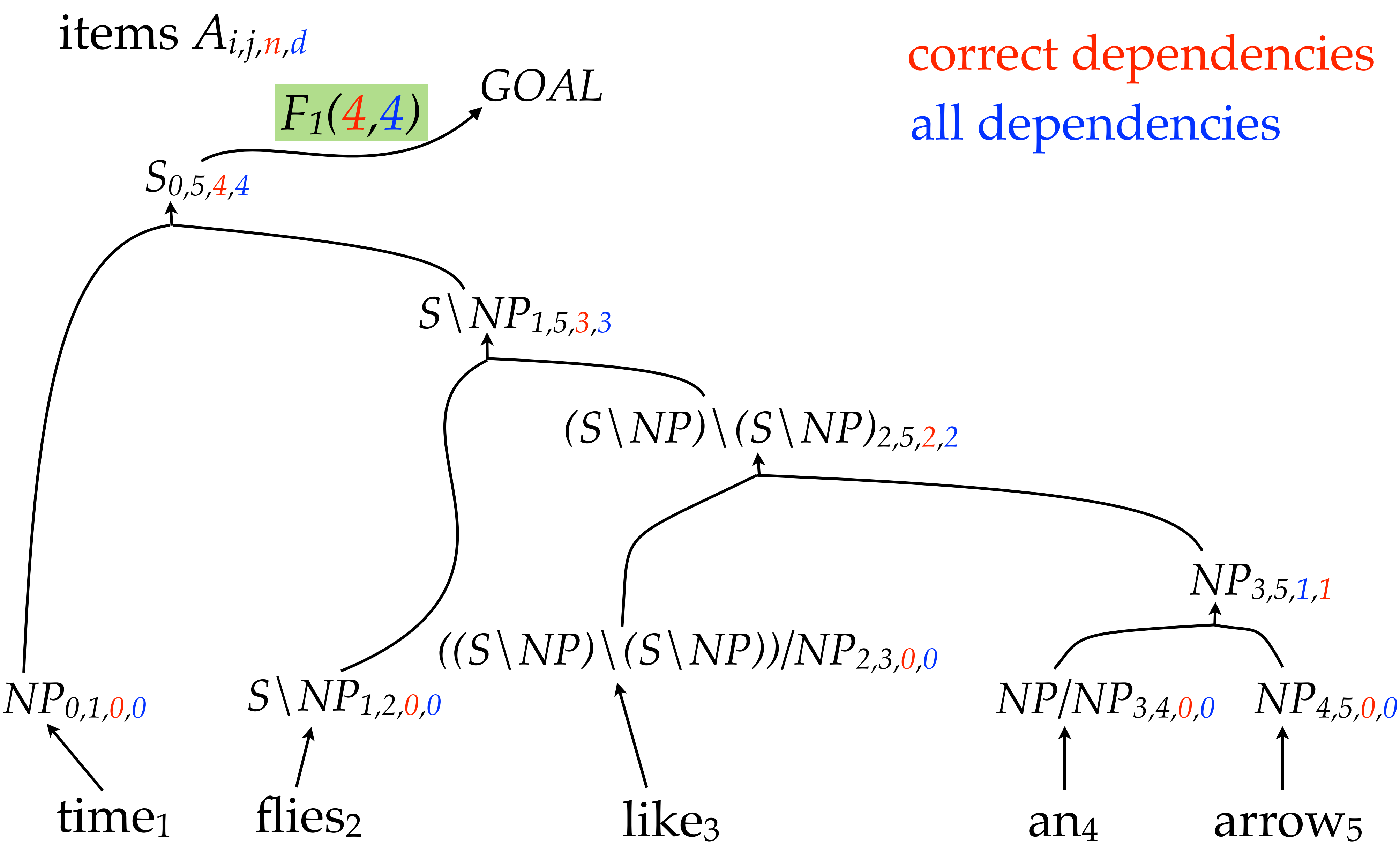
$$= \frac{2(n_1 + n_2)}{d_1 + d_2 + 2|y|}$$

Exact Loss Functions

- Treat sentence-level F_1 as *non-local feature* dependent on n, d.
- Result: new dynamic program over items $A_{i,j,n,d}$



Exact Losses with State-Split CKY



Exact Losses with State-Split CKY

items $A_{i,j,n,d}$

correct dependencies

all dependencies

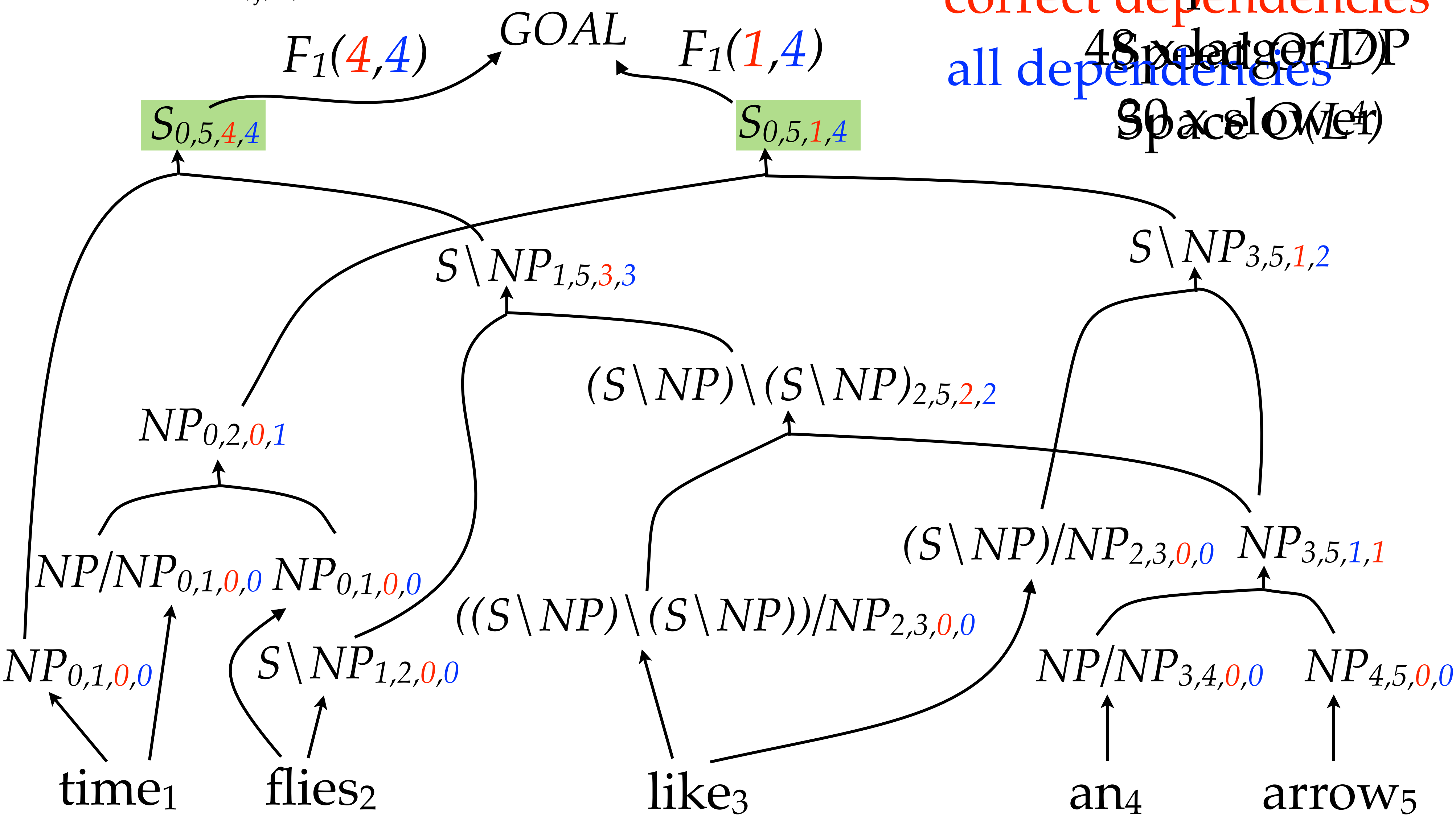
in practice

48x larger DP

30x slower

Speed $O(L^7)$

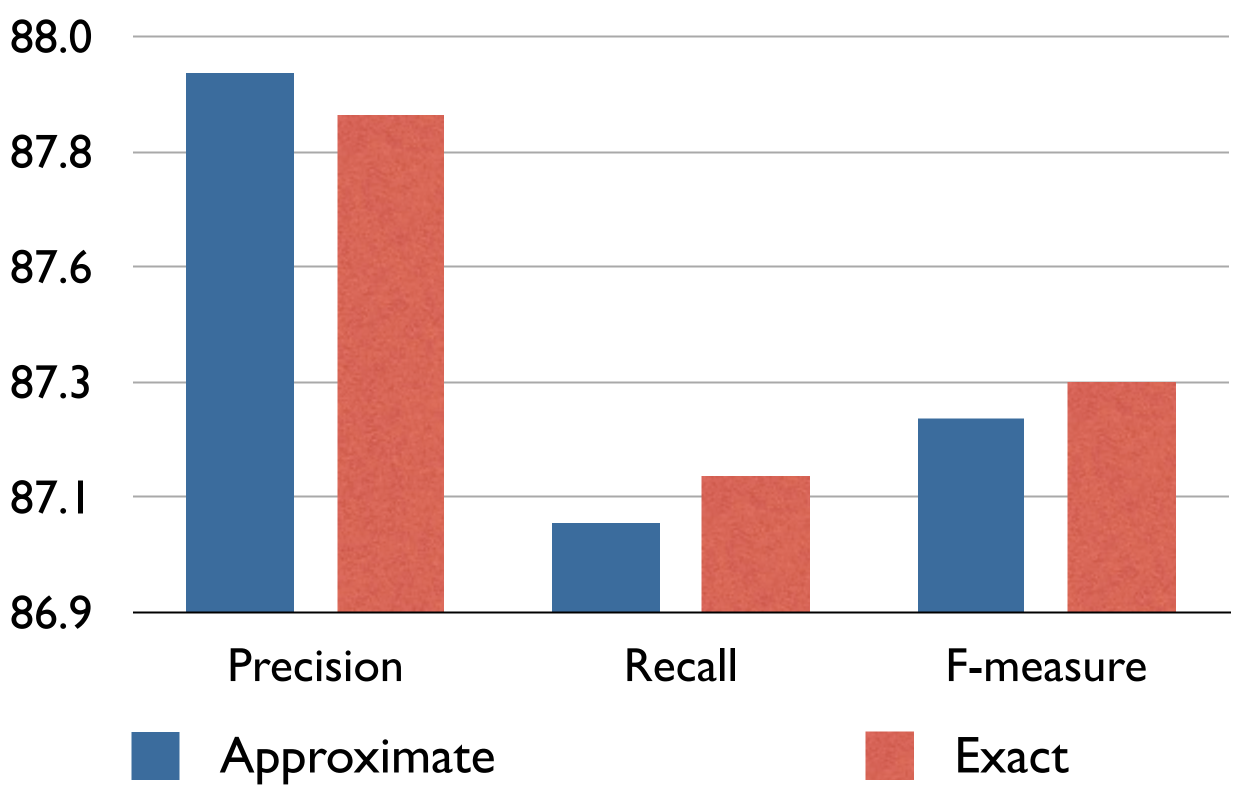
Space $O(L^4)$



Experiments

- Standard parsing task:
  - C&C Parser and supertagger (Clark & Curran 2007).
  - CCGBank standard train/dev/test splits.
  - Piecewise optimisation (Sutton and McCallum 2005)

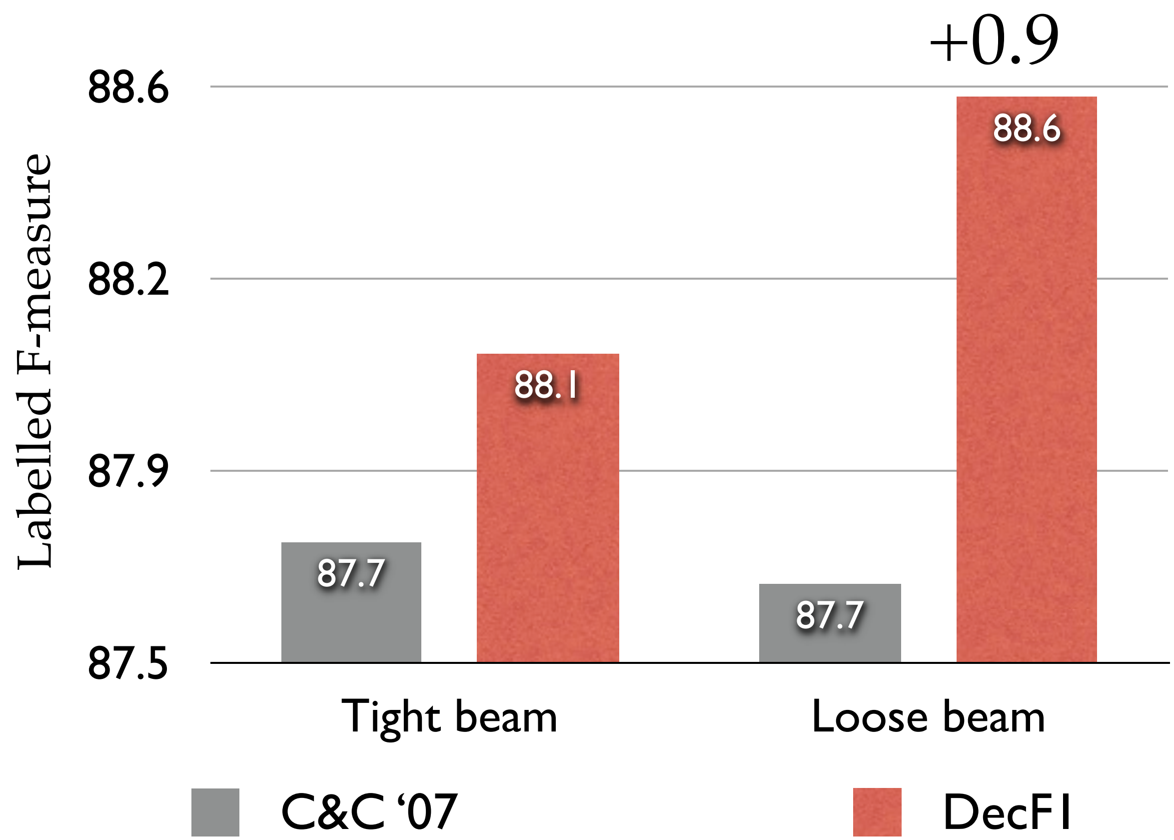
Exact versus Approximate



Approximate loss functions work, and much faster!

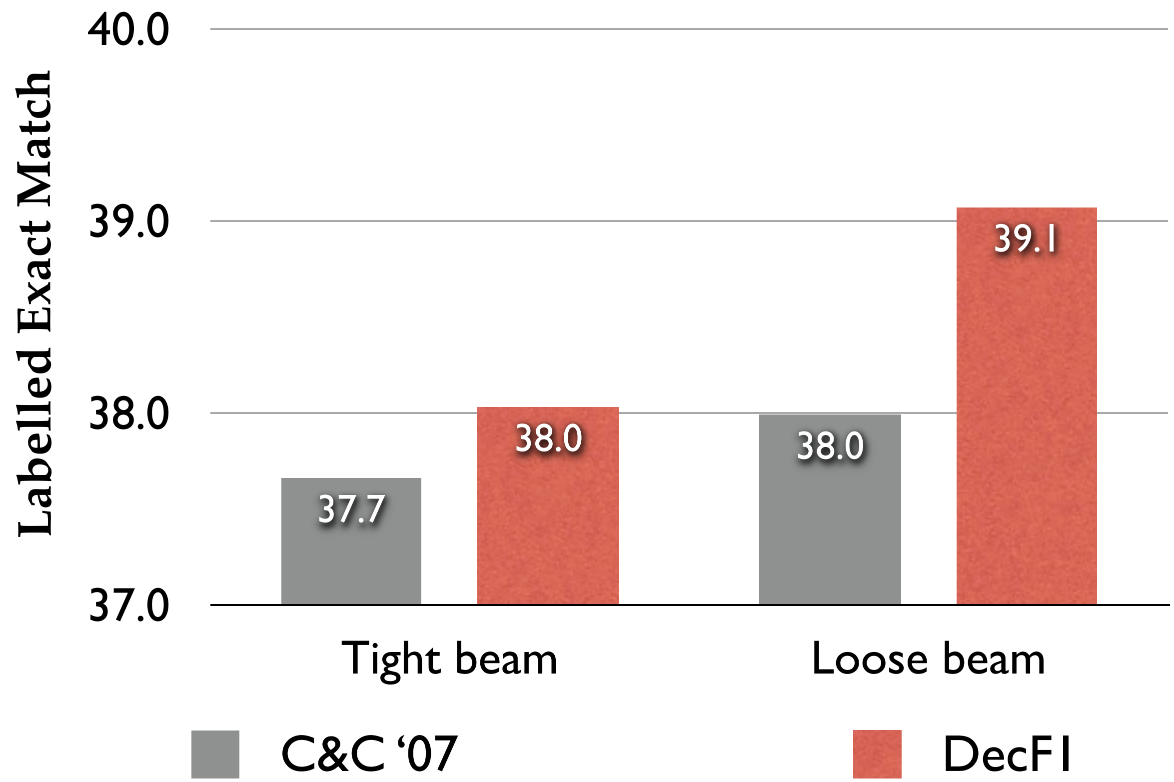
Softmax-Margin beats CLL

Test set results

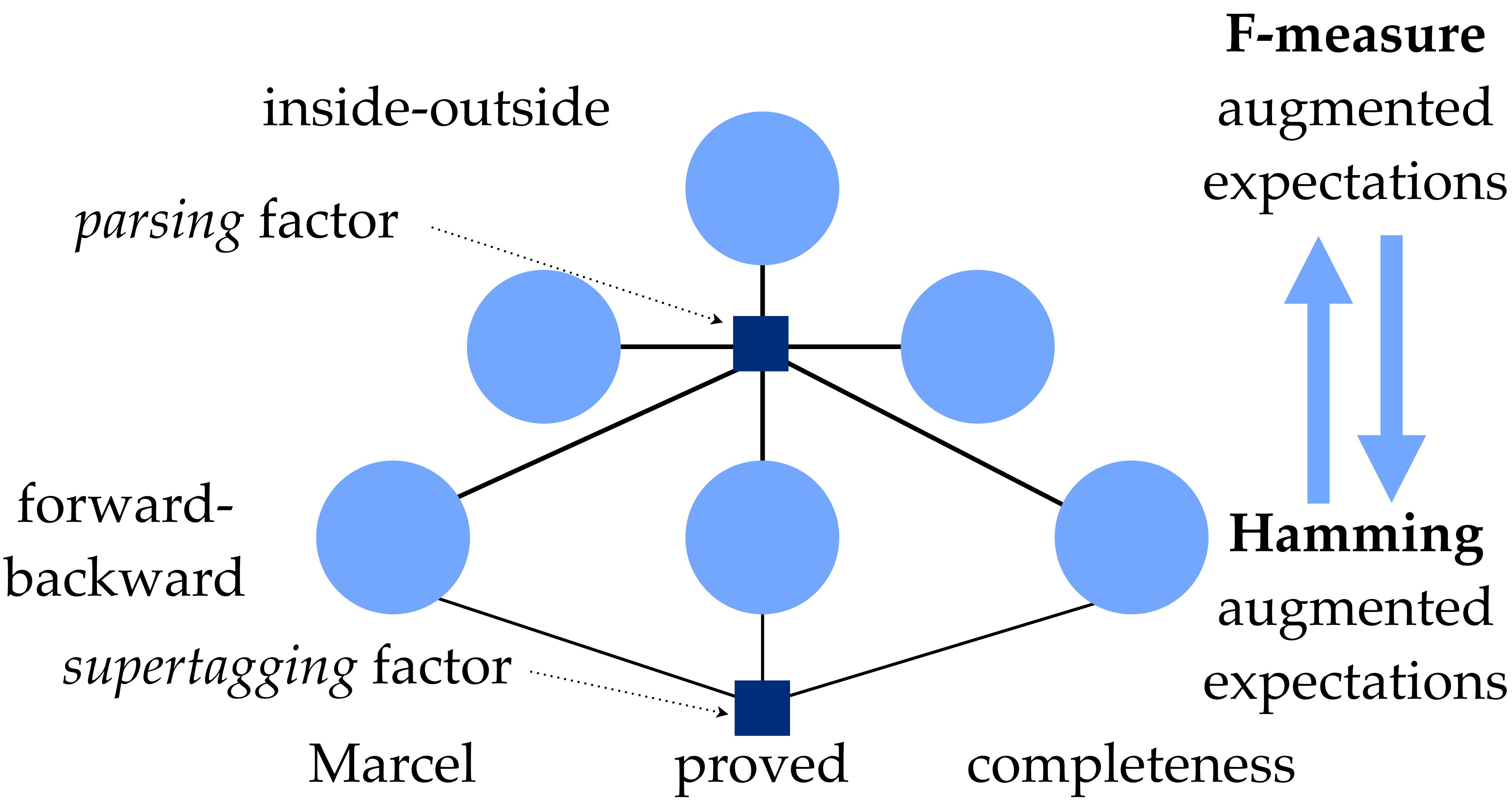


Softmax-Margin beats CLL

Does task-specific optimisation degrade accuracy on other metrics?

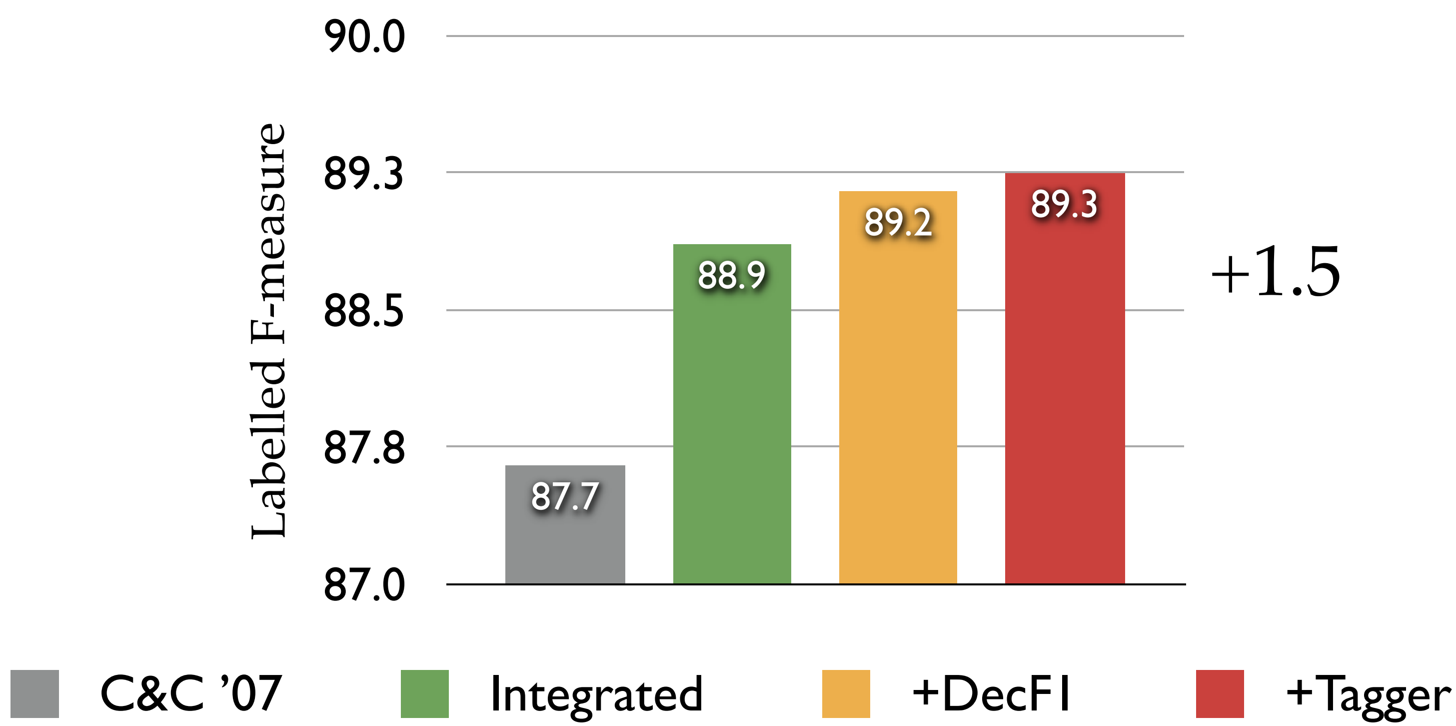


Integrated Model + SMM



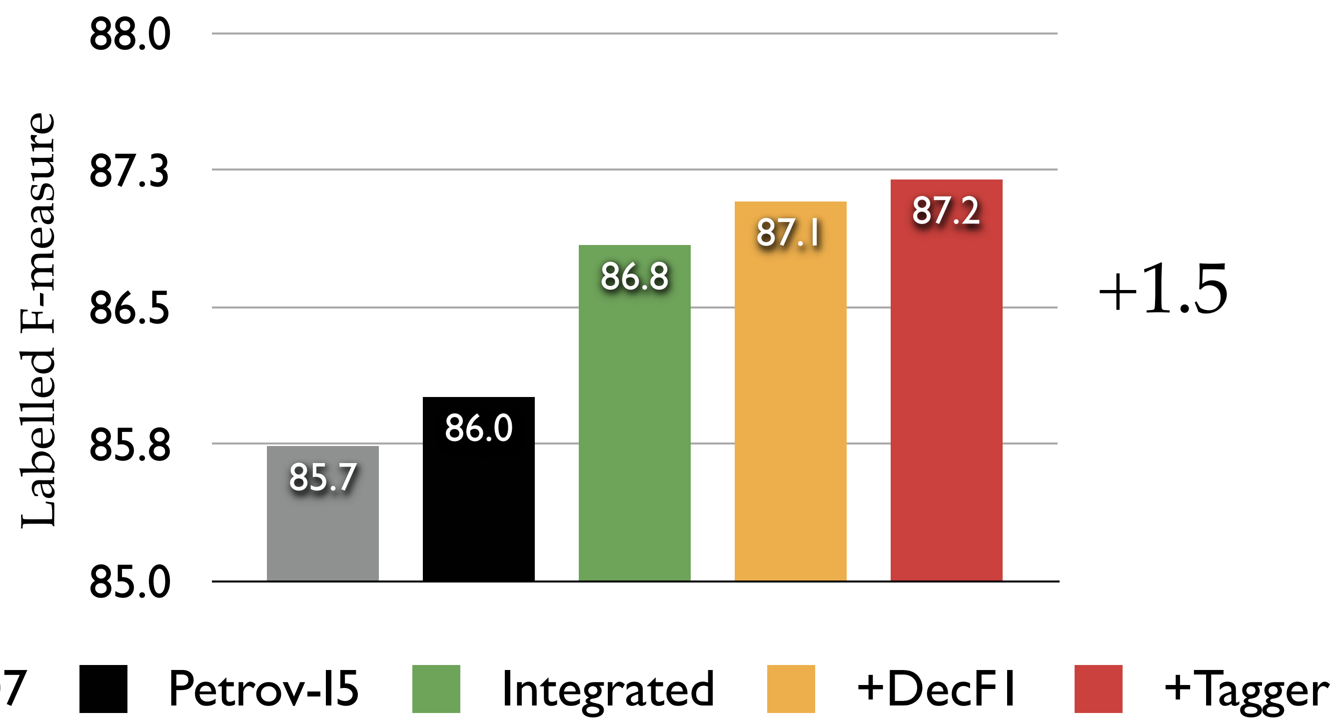
Results: Integrated Model

- F-measure loss for parsing sub-model (+DecF$_1$).
- Hamming loss for supertagging sub-model (+Tagger).
- Belief propagation for inference.



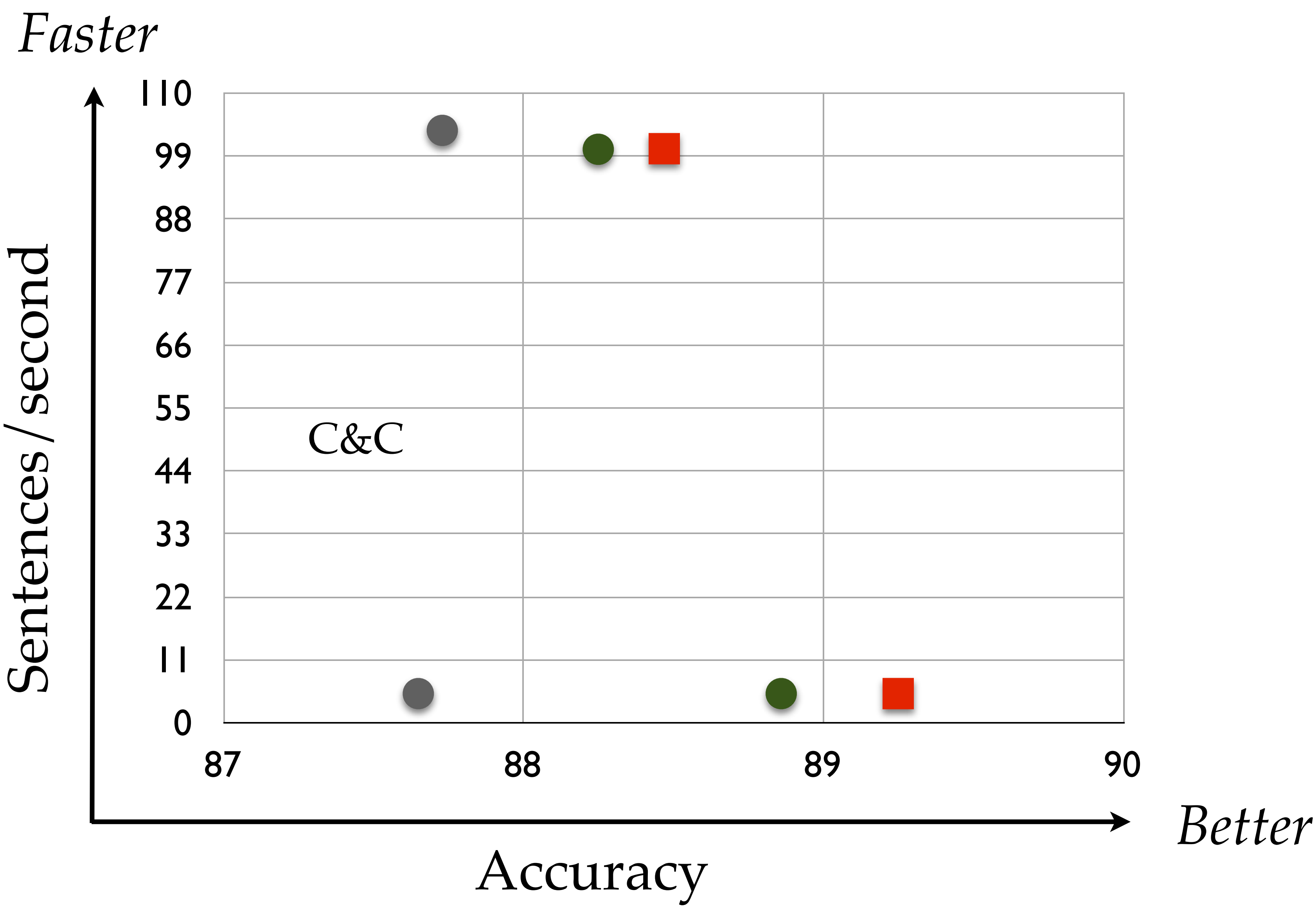
Results: Automatic POS

- F-measure loss for parsing sub-model (+DecF$_1$).
- Hamming loss for supertagging sub-model (+Tagger).
- Belief propagation for inference.

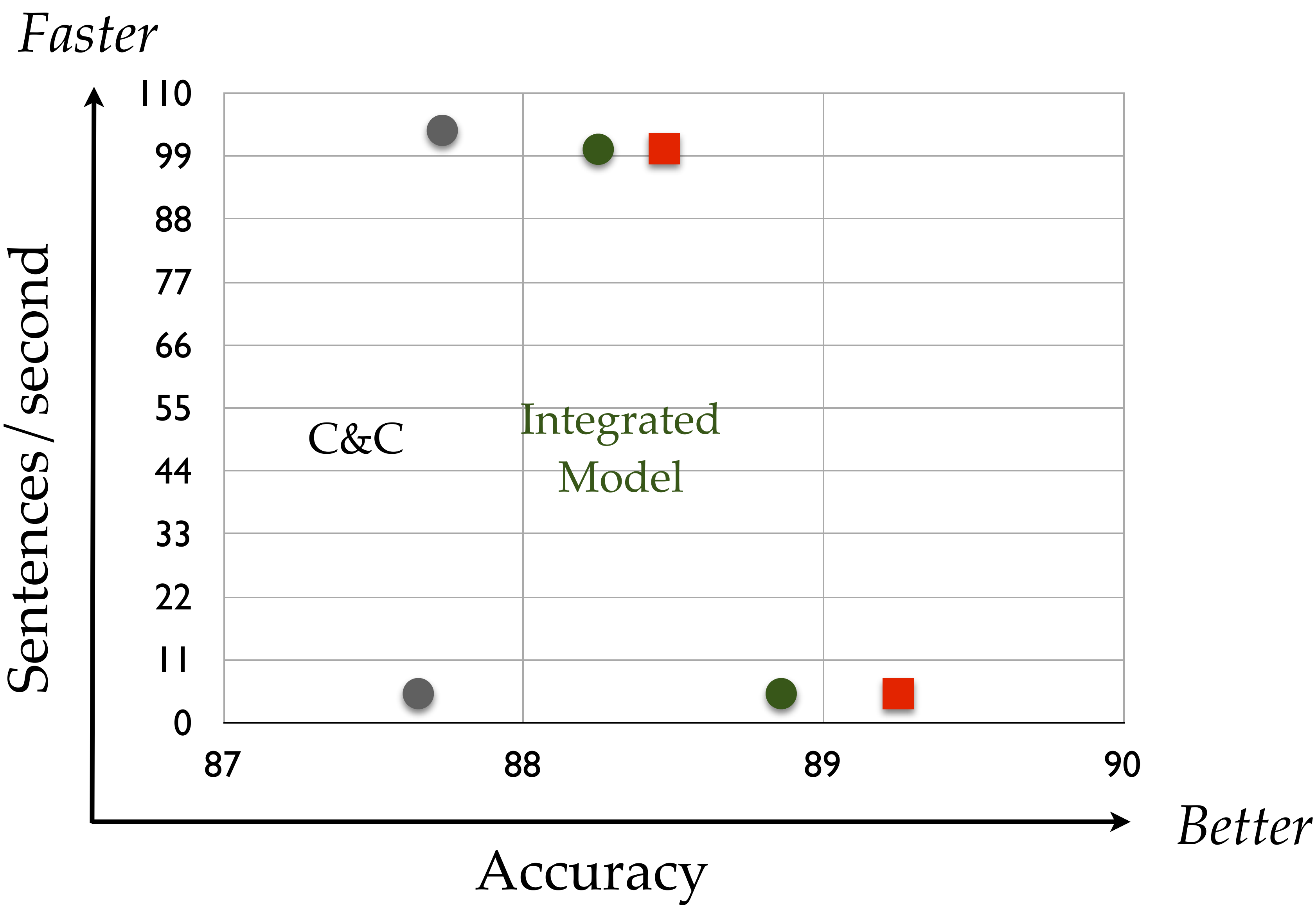


Fowler & Penn (2010)

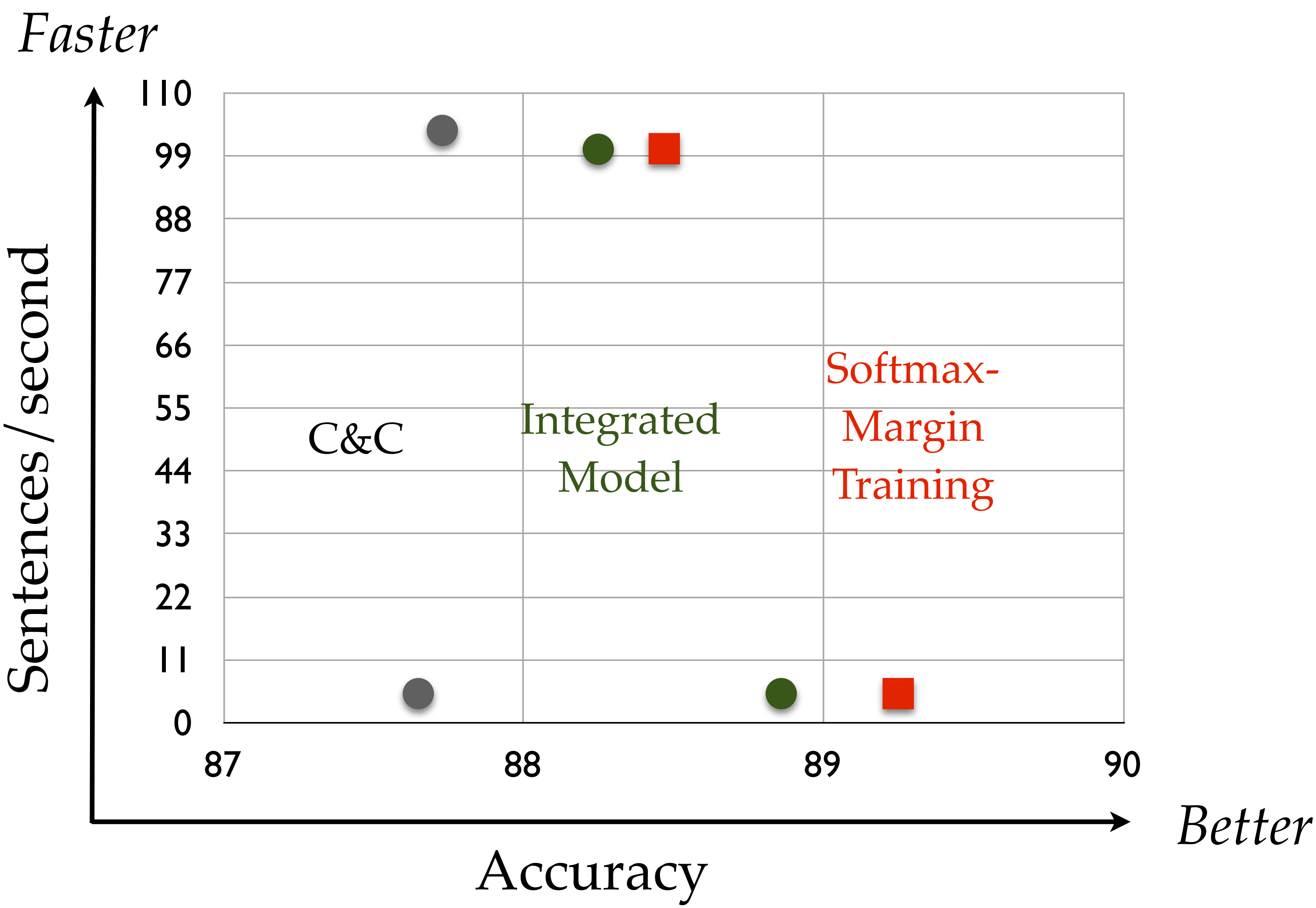
Results: Efficiency vs. Accuracy



Results: Efficiency vs. Accuracy



Results: Efficiency vs. Accuracy



Summary

- Softmax-Margin training is easy and improves our model.
- Approximate loss functions are fast, accurate and easy to use.
- Best ever CCG parsing results (87.7 $\rightarrow$ 89.3).