Latent TAG derivations for Semantic Role Labeling

Anoop Sarkar
(Joint work with **Yudong Liu** & Gholamreza Haffari)

School of Computing Science
Simon Fraser University
British Columbia, Canada
http://natlang.cs.sfu.ca

Semantic Role Labeling

Semantic Role Labeling (SRL)

- For a given verb (predicate), SRL aims to identify and label all its arguments with semantic roles, such as Agent, Patient, and Theme

A0: seller

[*Ports of Call Inc.*] *reached agreements to* **sell** [*its remaining seven aircraft*] [*to buyers that weren't disclosed*].

A1: goods

A2: buyer

SRL in NLP applications

Document summarization

(SFU team: SQUASH: Melli et al. DUC-2005)

- sentence selection
- sentence compression

Question answering

(Shen and Lapata, 2007, Stenchikova et al 2005)

Information retrieval

(Surdeanu et al., 2003)

Semantic entailment

(Braz et al.,2005)

Verb sense disambiguation

(Dang and Palmer, 2005)

Machine translation

(Wu and Fung, 2009)

Automatic case marker prediction in Japanese

(Suzuki and Touranova, 2006)

Co-reference resolution

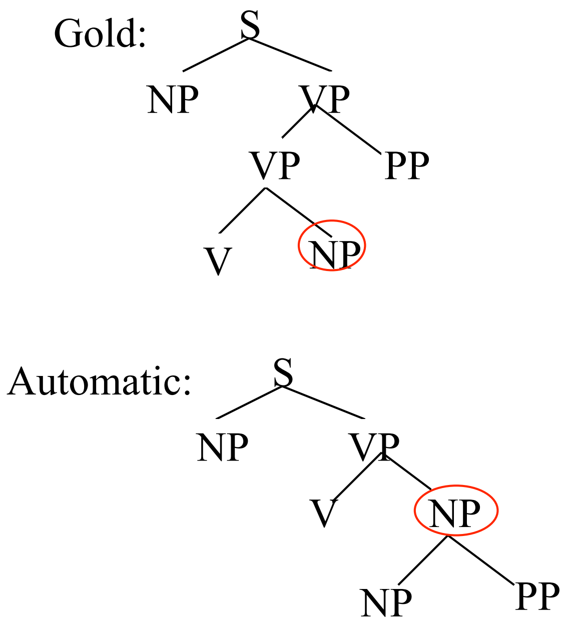
(Ponzetto and Strube, 2006)

…

High accuracy is achieved by

Proposing new types of features from different syntactic views

- chunks (Hacioglu et al., 2004)
- parses (Gildea and Jurafsky, 2002, Gildea and Palmer, 2002; Punyakanok et al., 2005)
- CCG derivations (Gildea and Hockenmaier, 2003)
- dependency trees (Hacioglu et al., 2004)



Modeling the predicate frameset between arguments: A0 A0 V A2 A1

(Gildea and Jurafsky, 2002; Pradhan et al., 2004; Toutanova et al., 2008; Punyakanok et al., 2008)

Dealing with incorrect parser output by using more than one parse

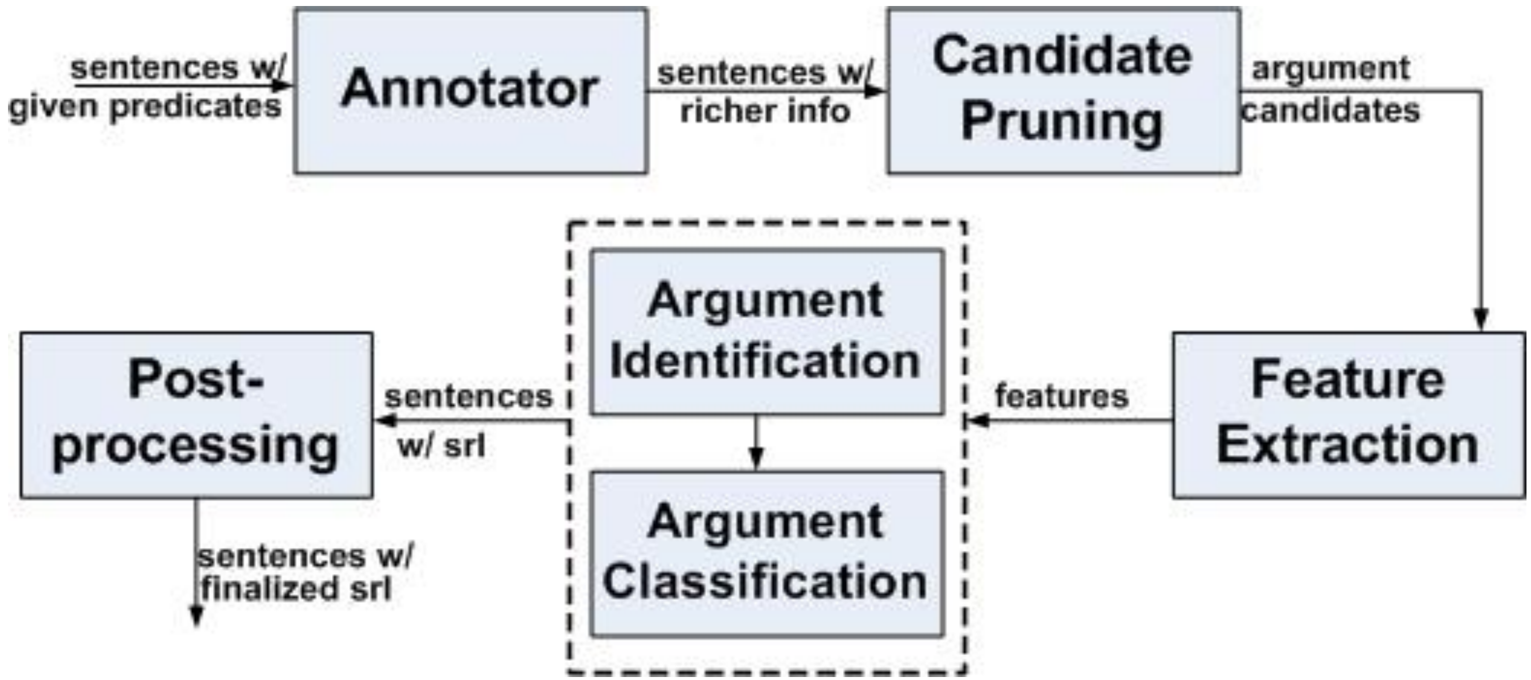
(Punyakanok et al., 2005; Toutanova et al., 2008; Pradhan et al., 2005)

Our work

Proposing new types of features from different syntactic views

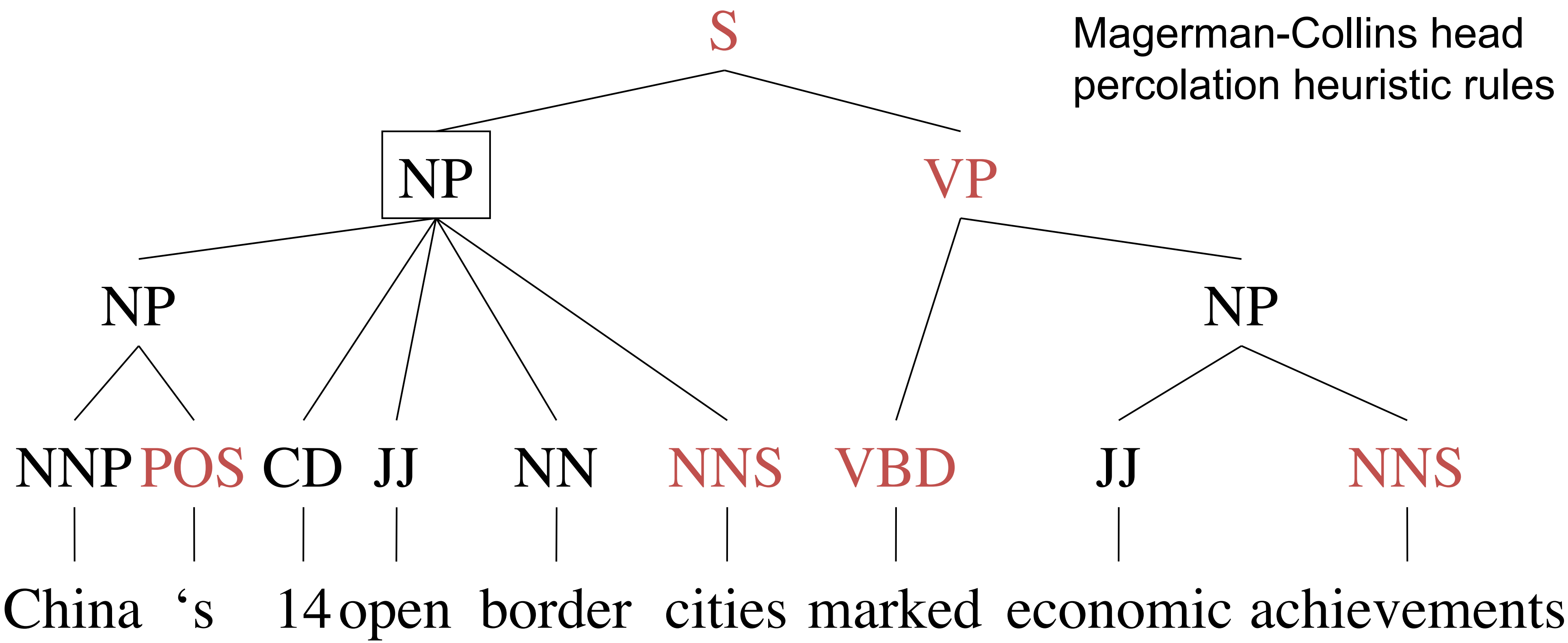
- chunks (Hacioglu et al., 2004)
- parses (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Punyakanok et al., 2005)
- CCG derivations (Gildea and Hockenmaier, 2003)
- dependency trees (Hacioglu et al., 2004)
- Lexicalized Tree Adjoining Grammars (TAG) derivations (Liu and Sarkar EMNLP 2007)

Architecture of our SRL system



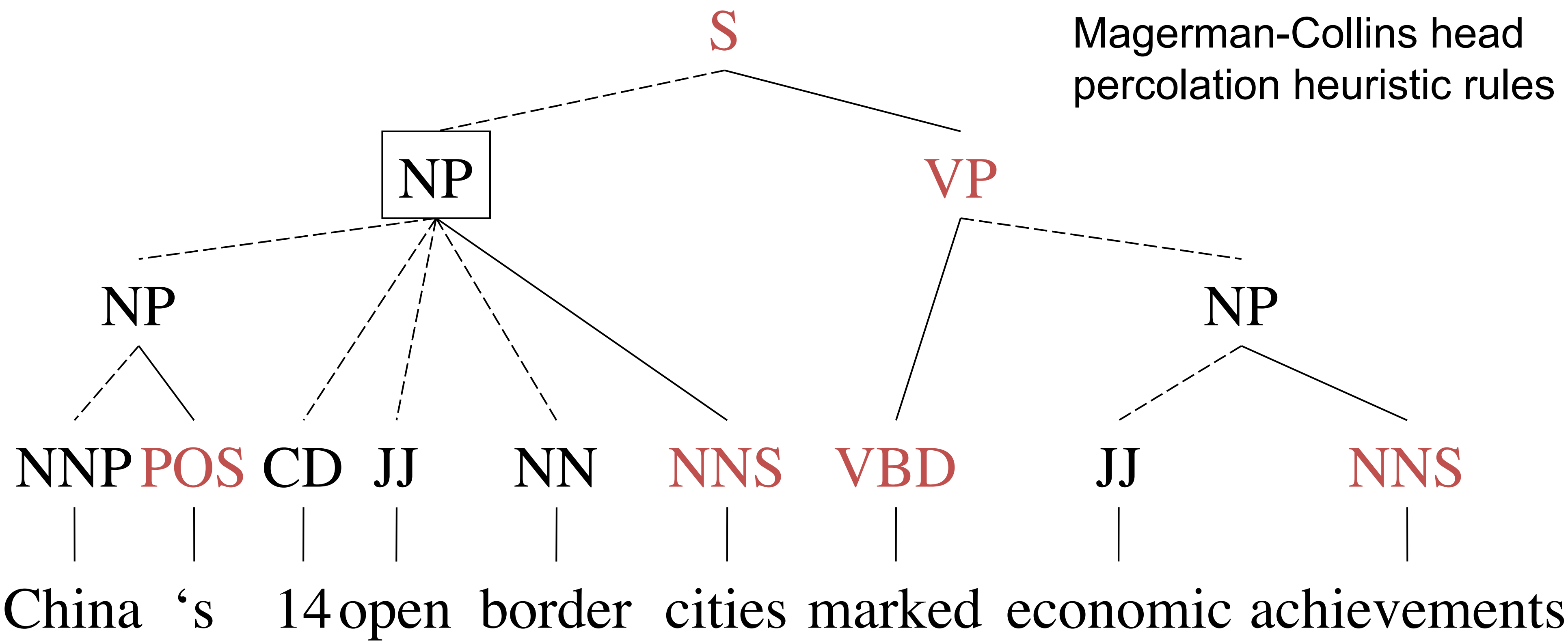
Tree adjoining Grammars (TAG)

TAG derivations from Treebanks or phrase-structure parses



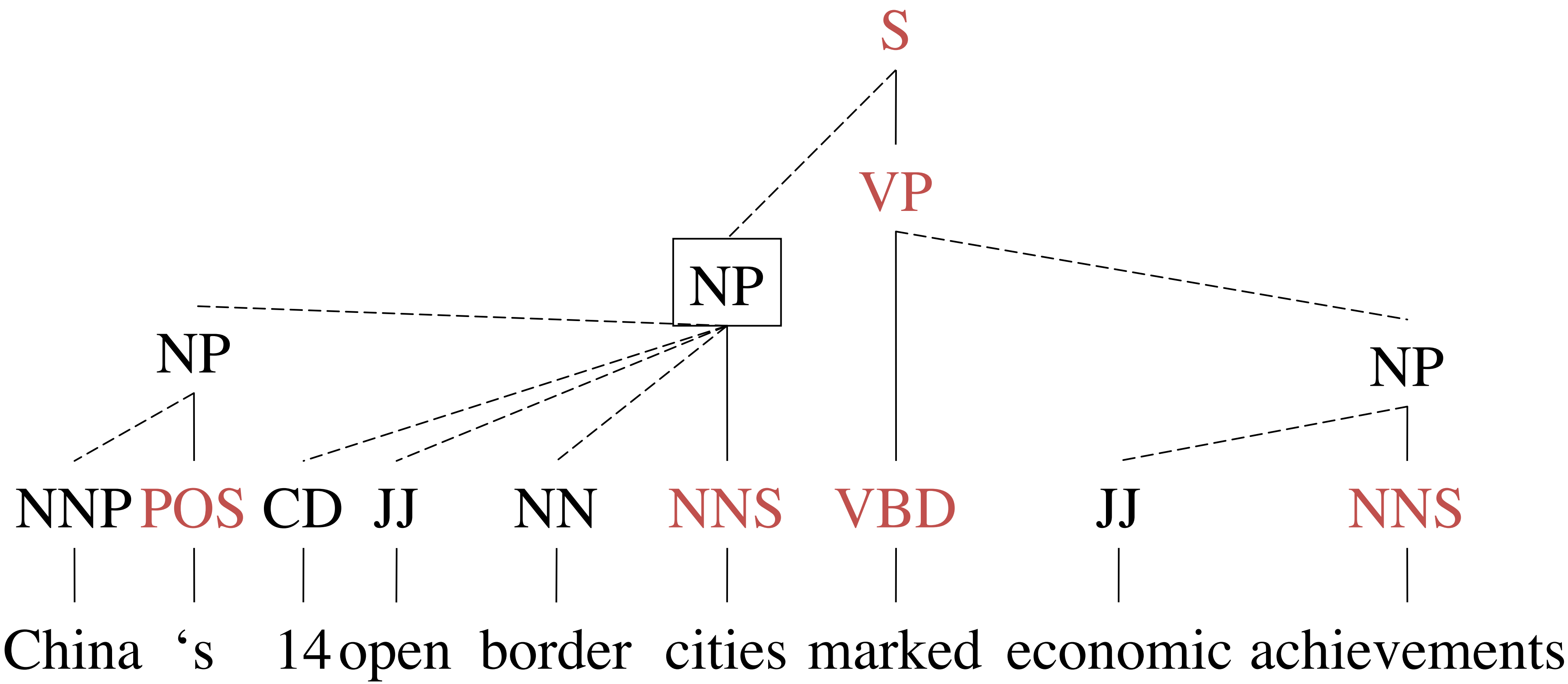
Magerman-Collins head percolation heuristic rules

TAG derivations from Treebanks or phrase-structure parses

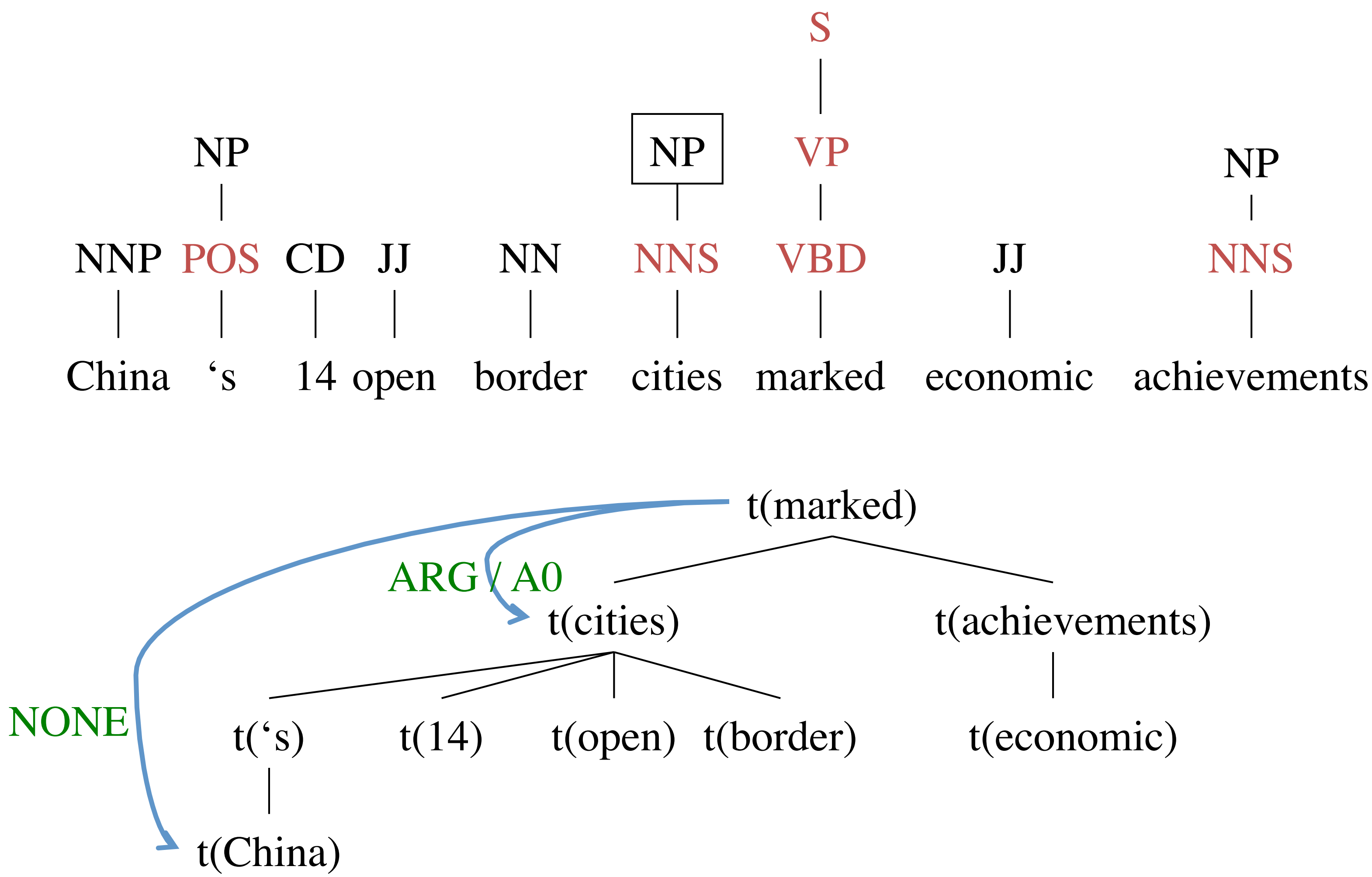


Magerman-Collins head percolation heuristic rules

TAG derivations from Treebanks or phrase-structure parses

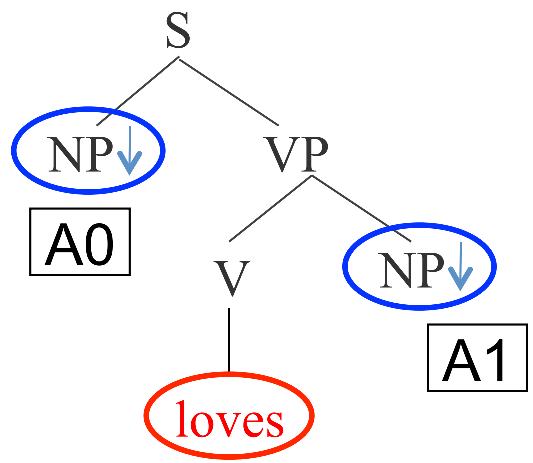


TAG derivations from Treebanks or phrase-structure parses

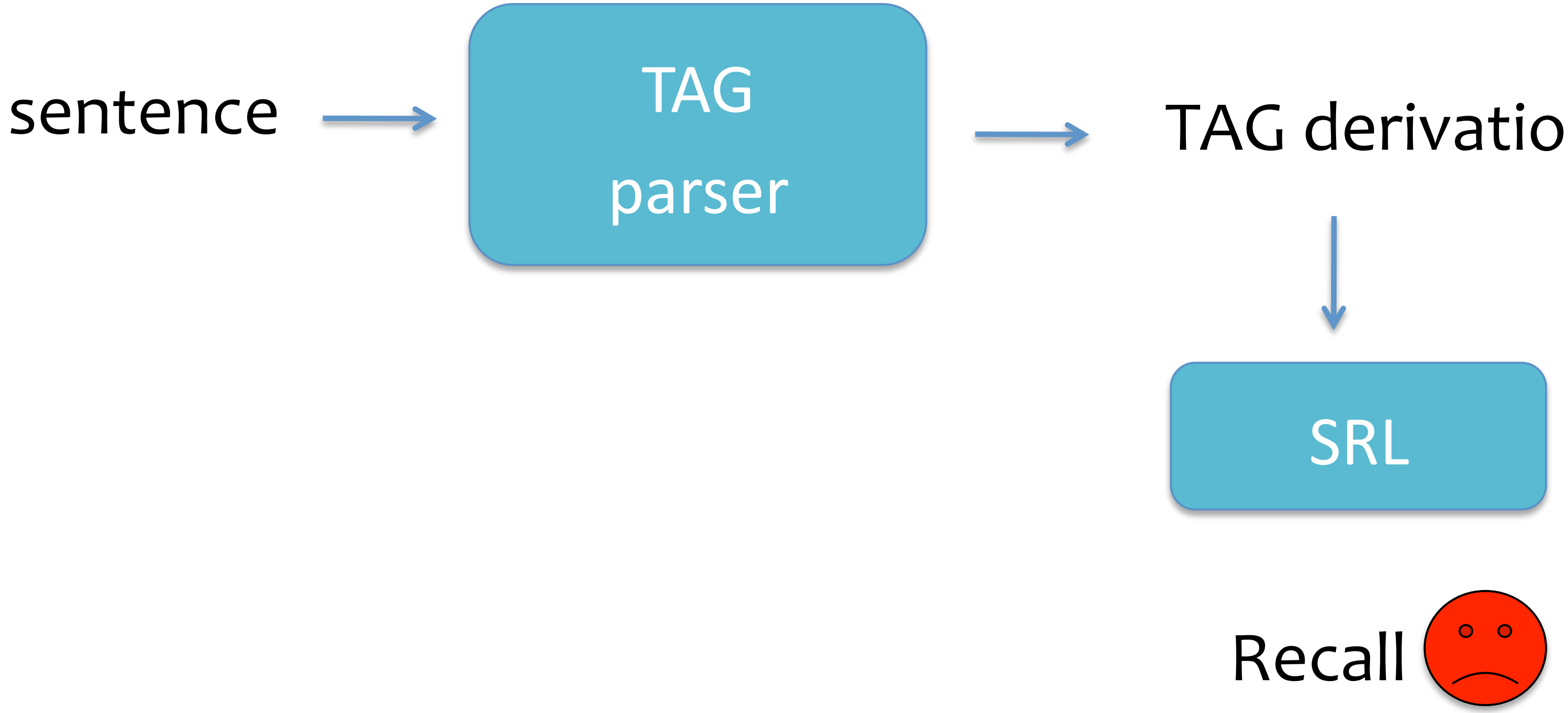


SRL and TAG

- TAG is closely related to SRL due to its *extended domain of locality*
- TAG provides an alternative syntactic view for SRL feature selection

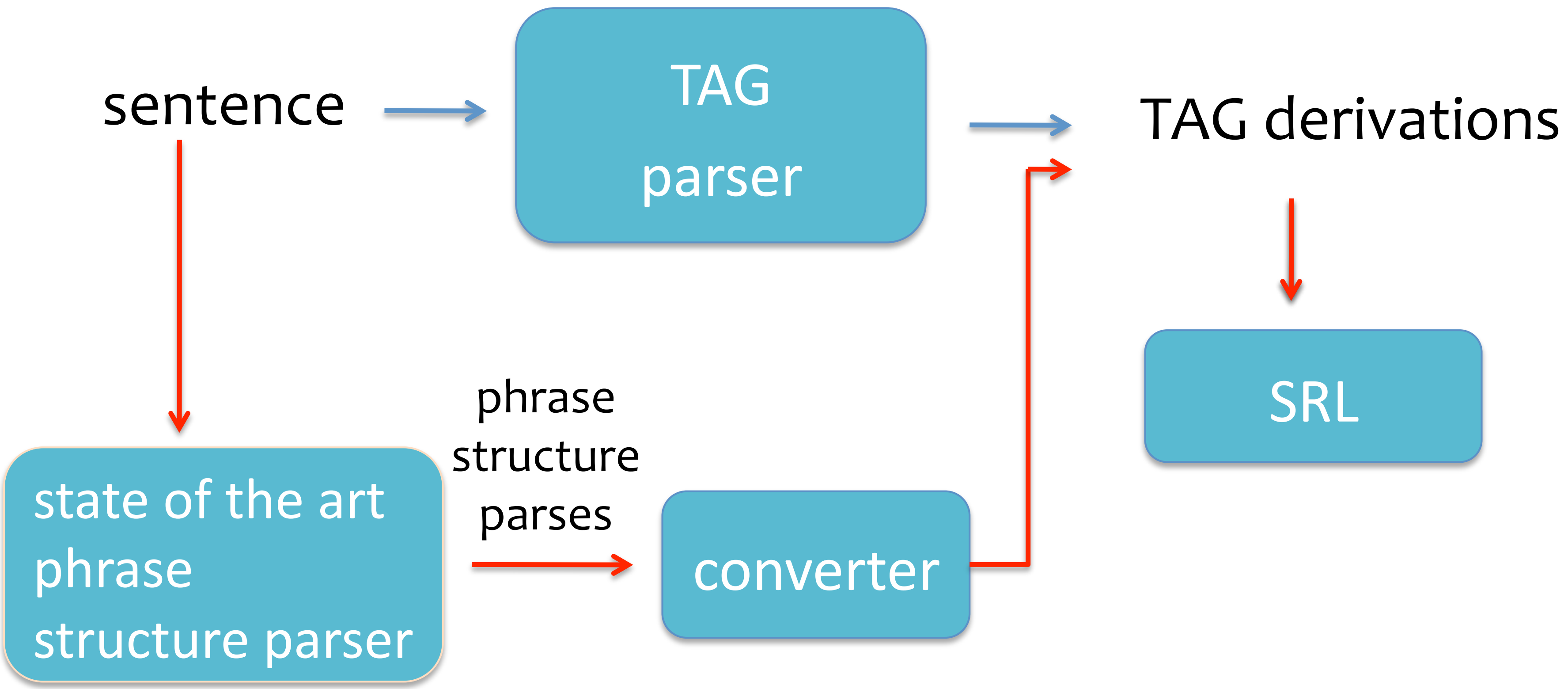


TAG derivations for SRL

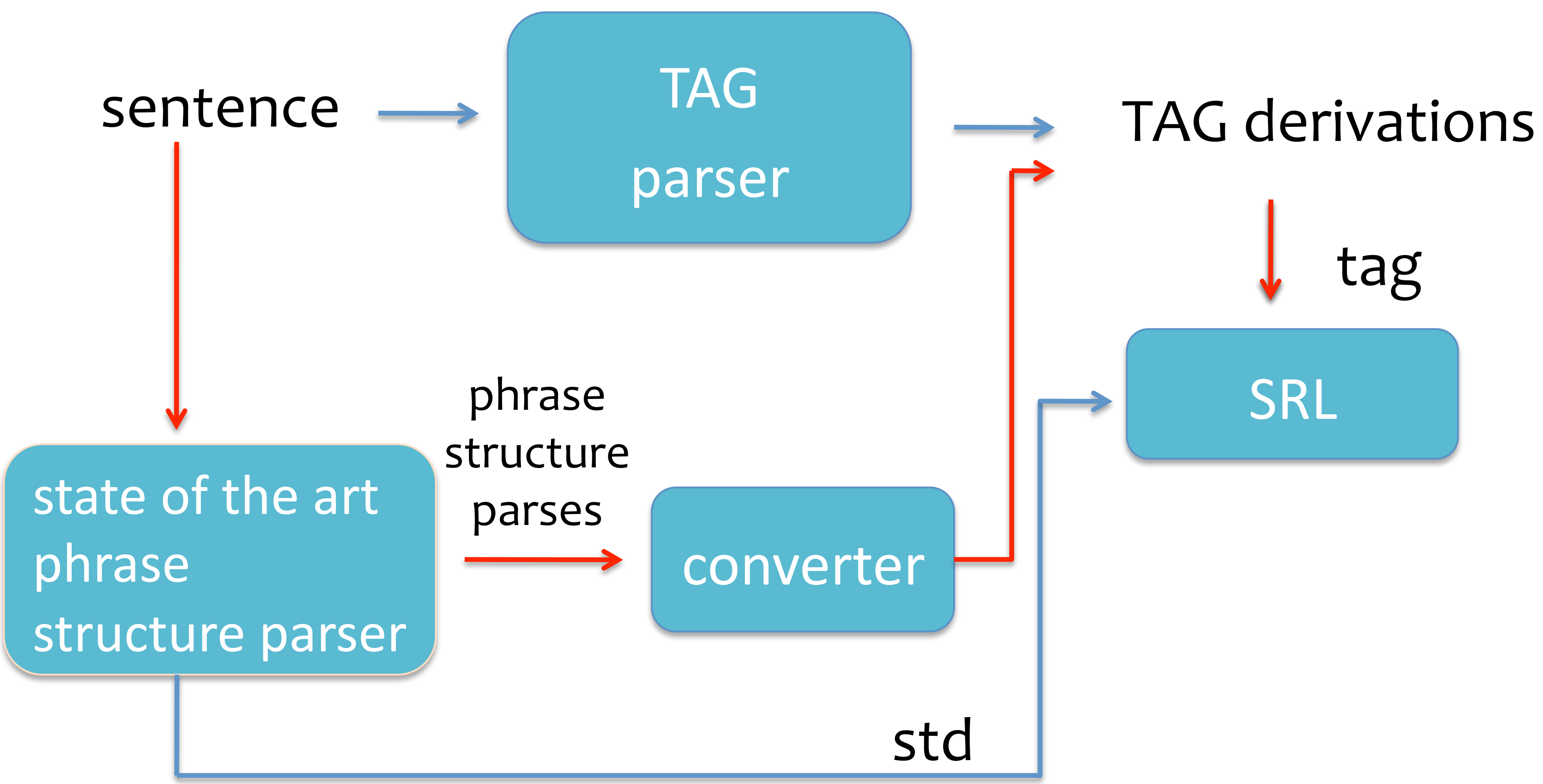


	Precision	Recall
phrase-structure	85.8	87.7
TAG	85.8	85.6

TAG derivations for SRL

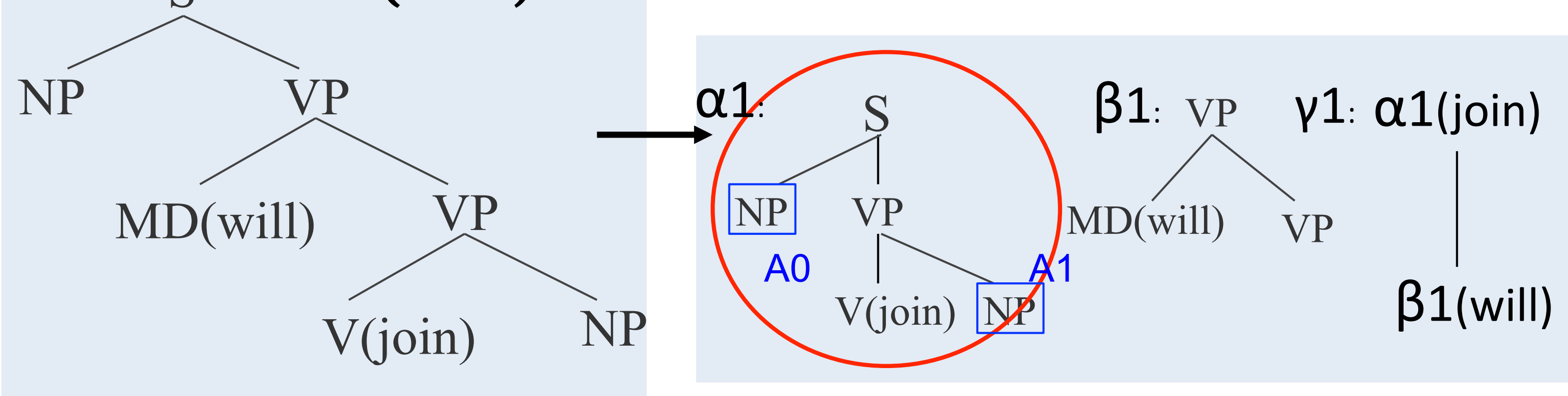


TAG derivations for SRL



TAG for SRL

Extended domain of locality (EDL)

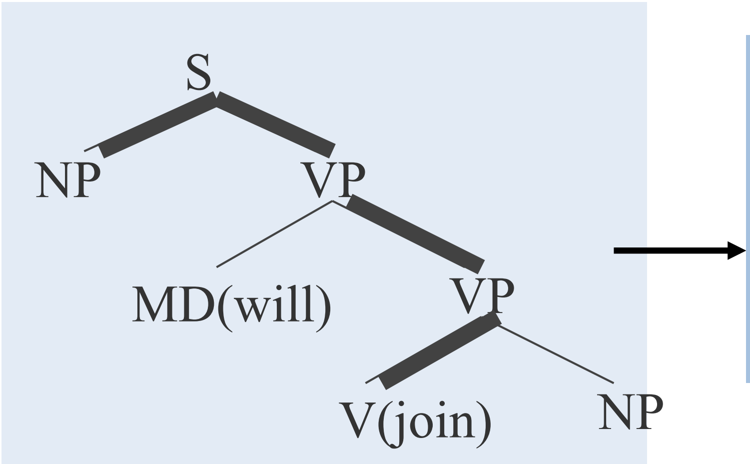


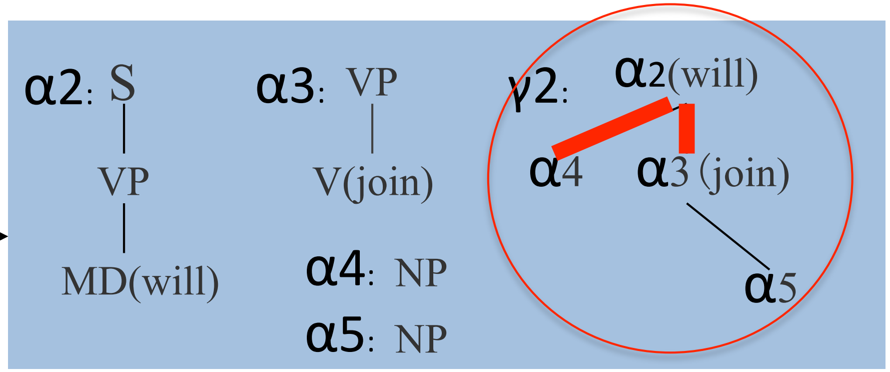
(Chen and Rambow, 2003)

only ~87% of dependencies between predicate and (core) argument are captured in gold trees.

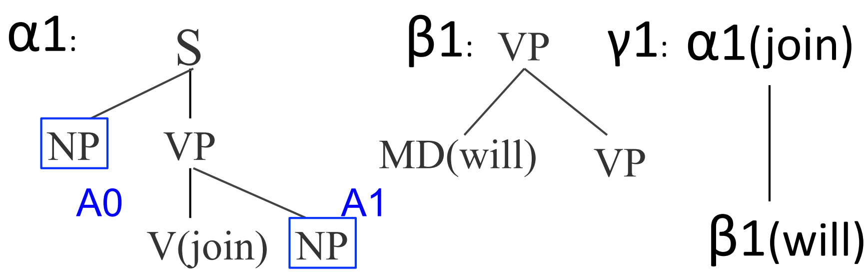
TAG for SRL

(Liu and Sarkar, EMNLP 2007)



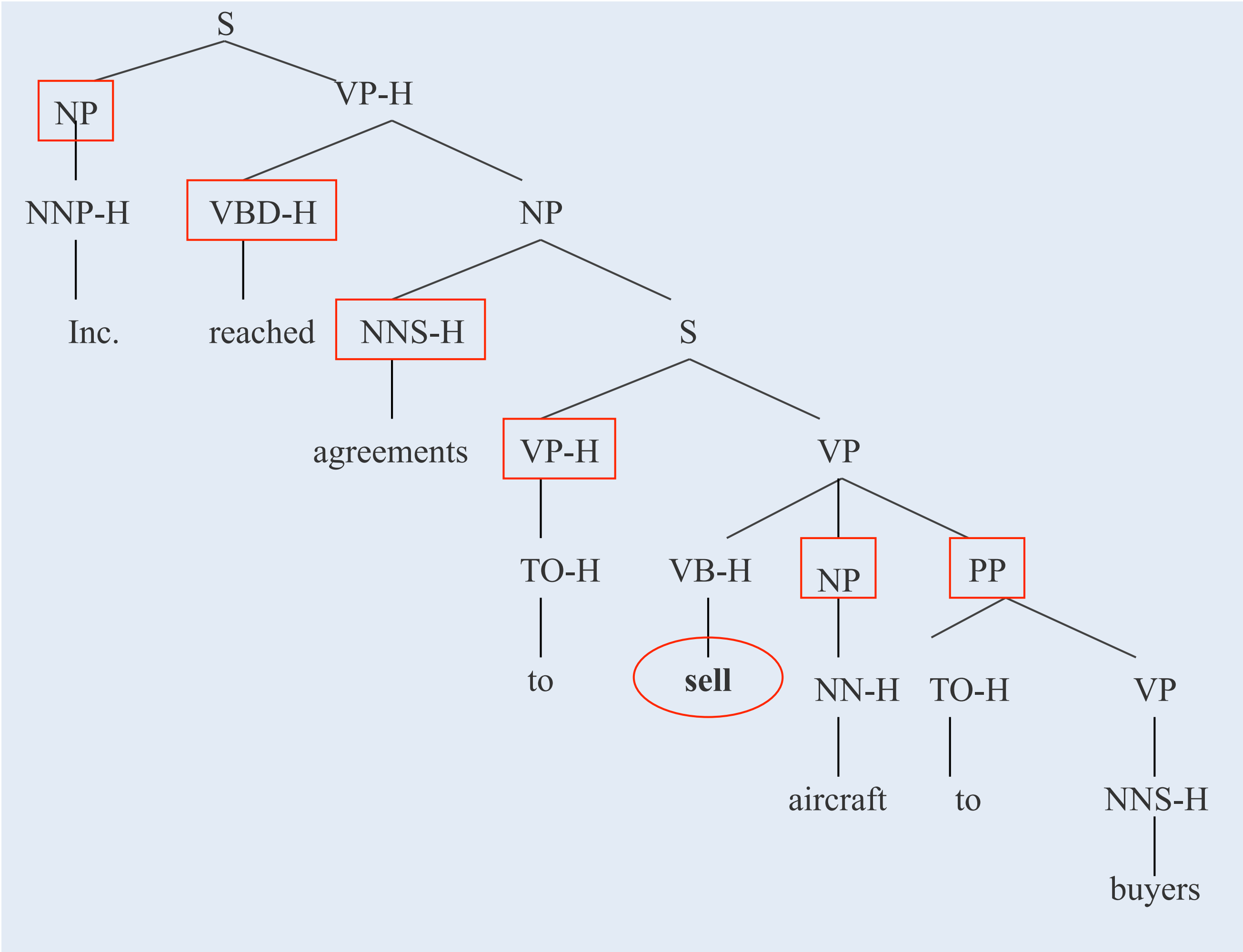


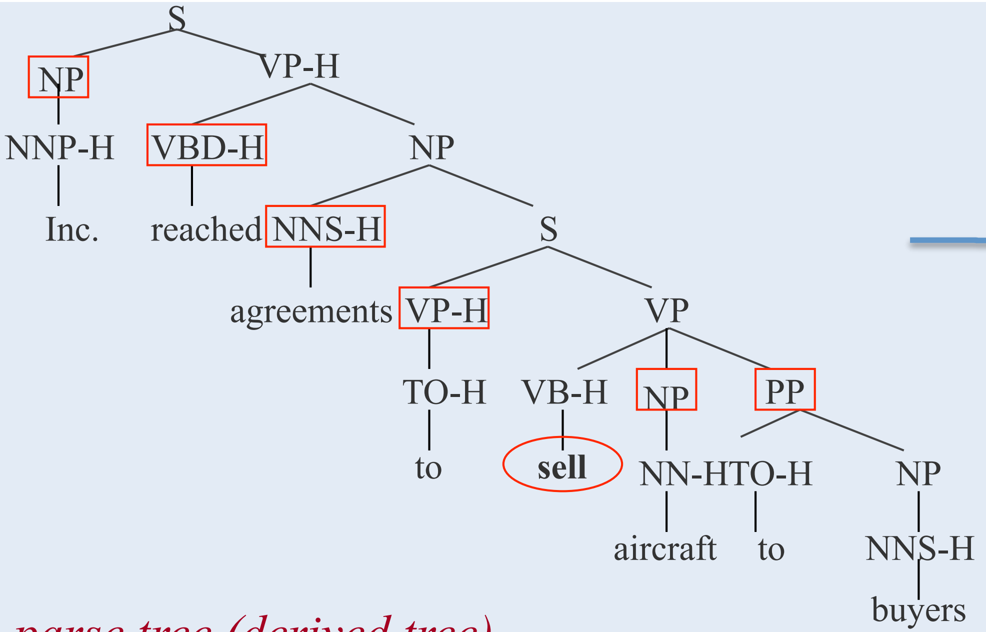
(Chen and Rambow, 2003)



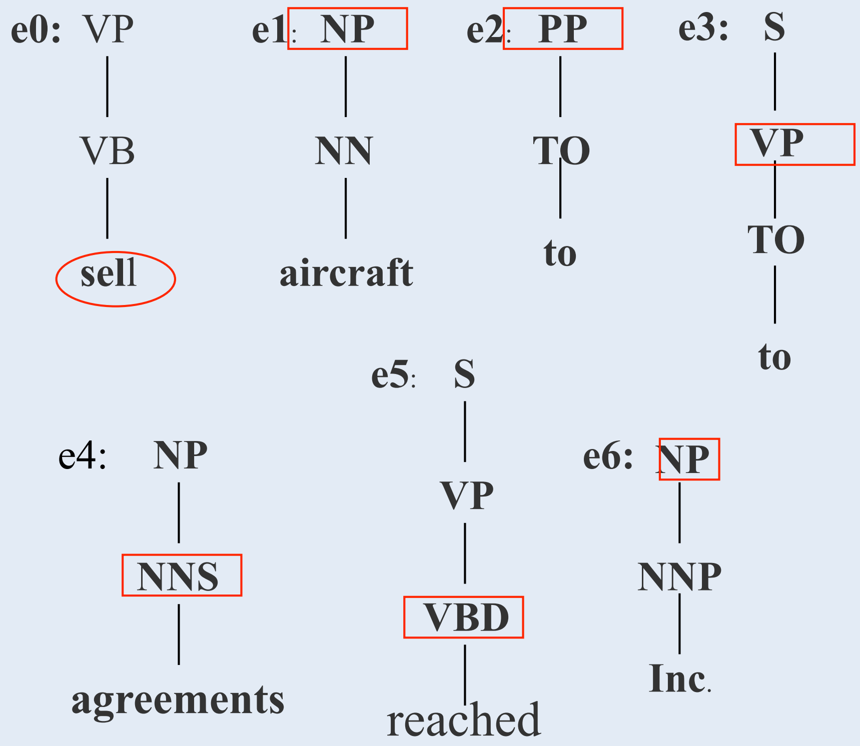
- Magerman-Collins head percolation rules (Chiang, 2000)
- Sister-adjunction operation (Schabes and Shieber, 1994)
- path feature less sparse

the example revisited: [seller *Ports of Call Inc.*] *reached agreements to* **sell** [goods *its remaining seven aircraft*] [buyer *to buyers that weren't disclosed*].

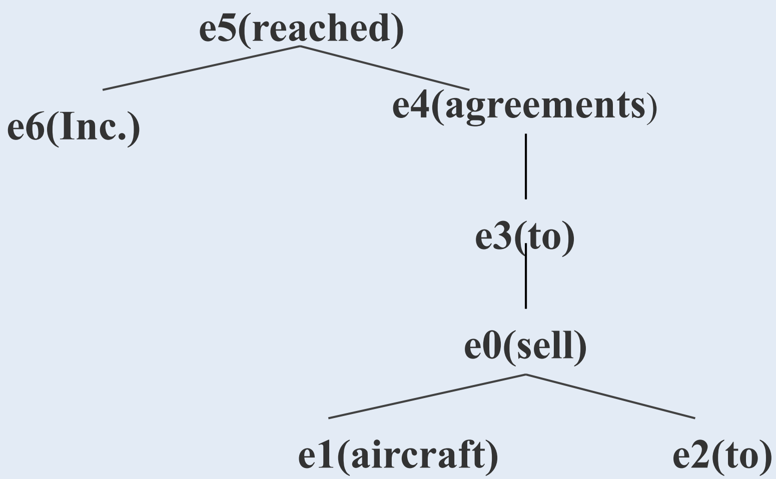




parse tree (derived tree)



elementary trees



derivation tree

[seller *Ports of Call Inc.*] *reached agreements to* **sell** [goods *its remaining seven aircraft*] [buyer *to buyers that weren't disclosed*].

Argument-adjunct distinction:

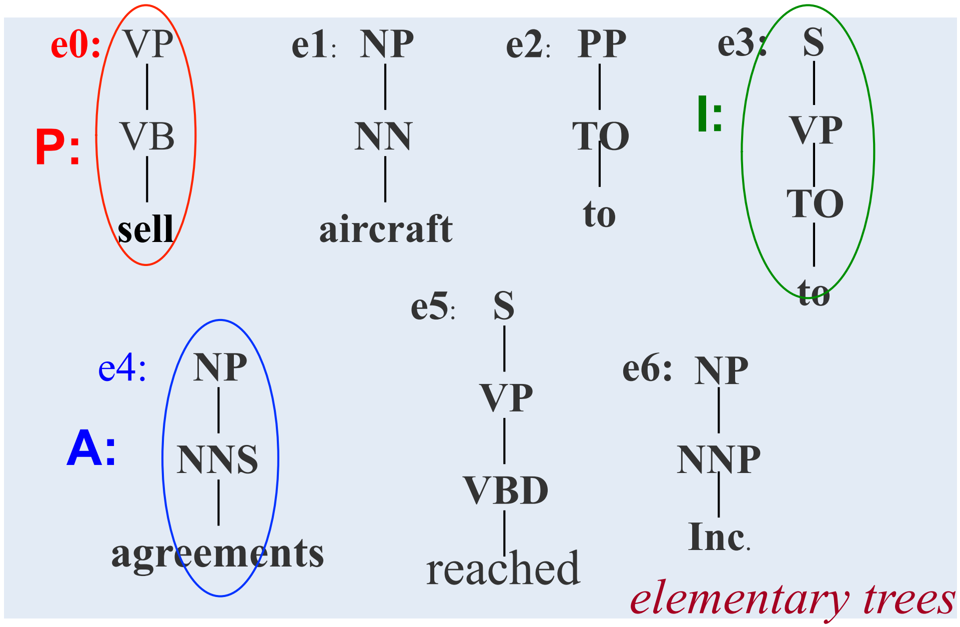
All elementary trees are in *spinal form*

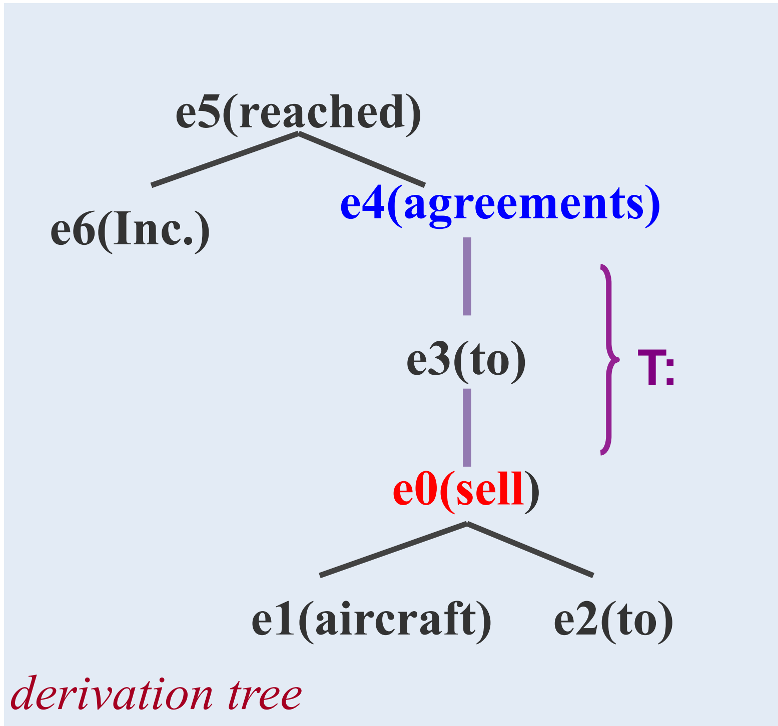
Sister-adjunction

TAG features

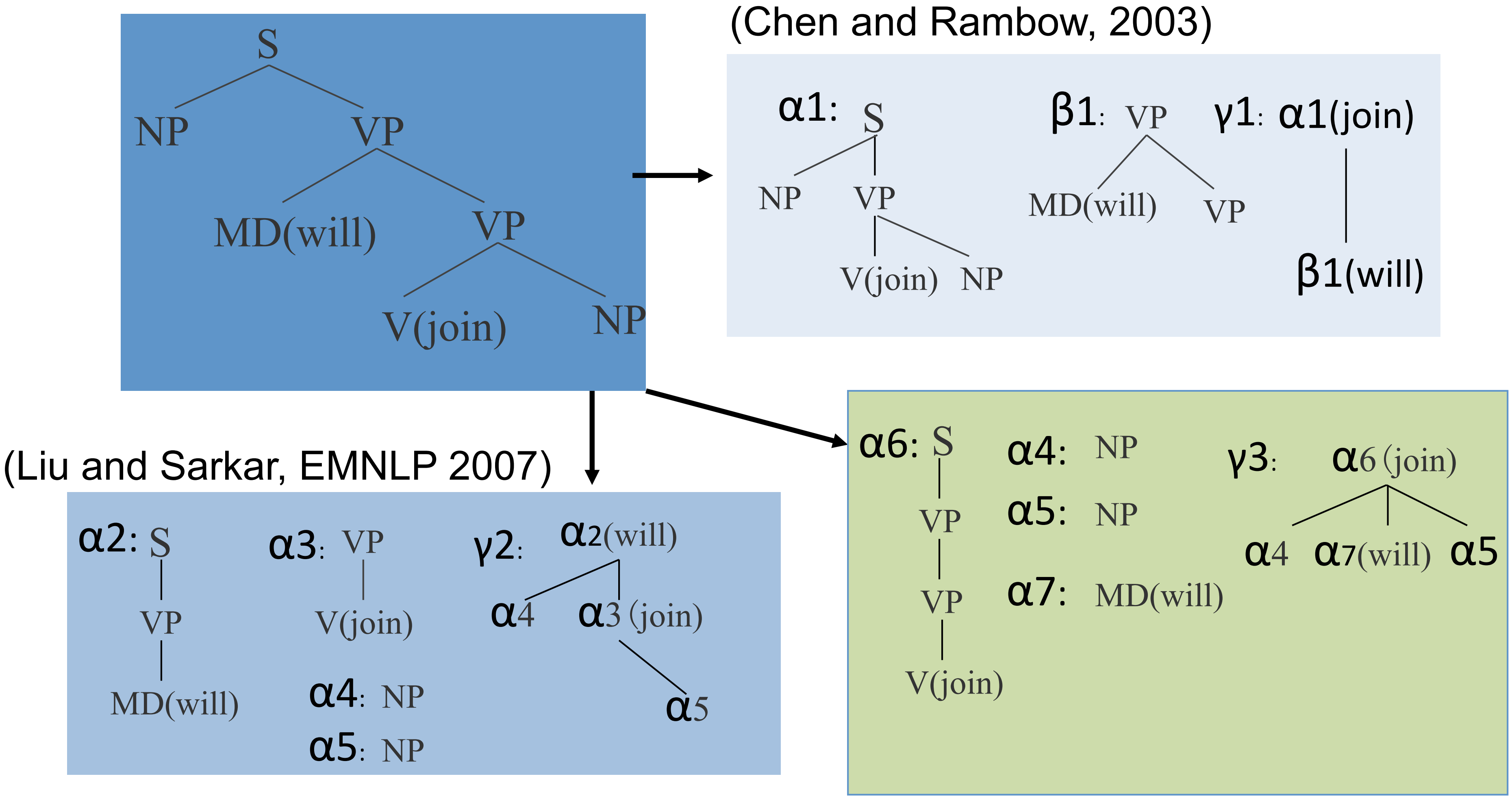
Example <sell, NP(agreement)>

- *P*redicate elementary tree features
- *A*rgument elementary tree features
- *I*ntermediate elementary tree features
- *T*opological relations in TAG derivations:
  - distance between e-trees
  - relative position
  - modifying relations.
- Feature analysis shows that adding all feature types improves accuracy





Motivation for latent derivations



Motivation for latent derivations

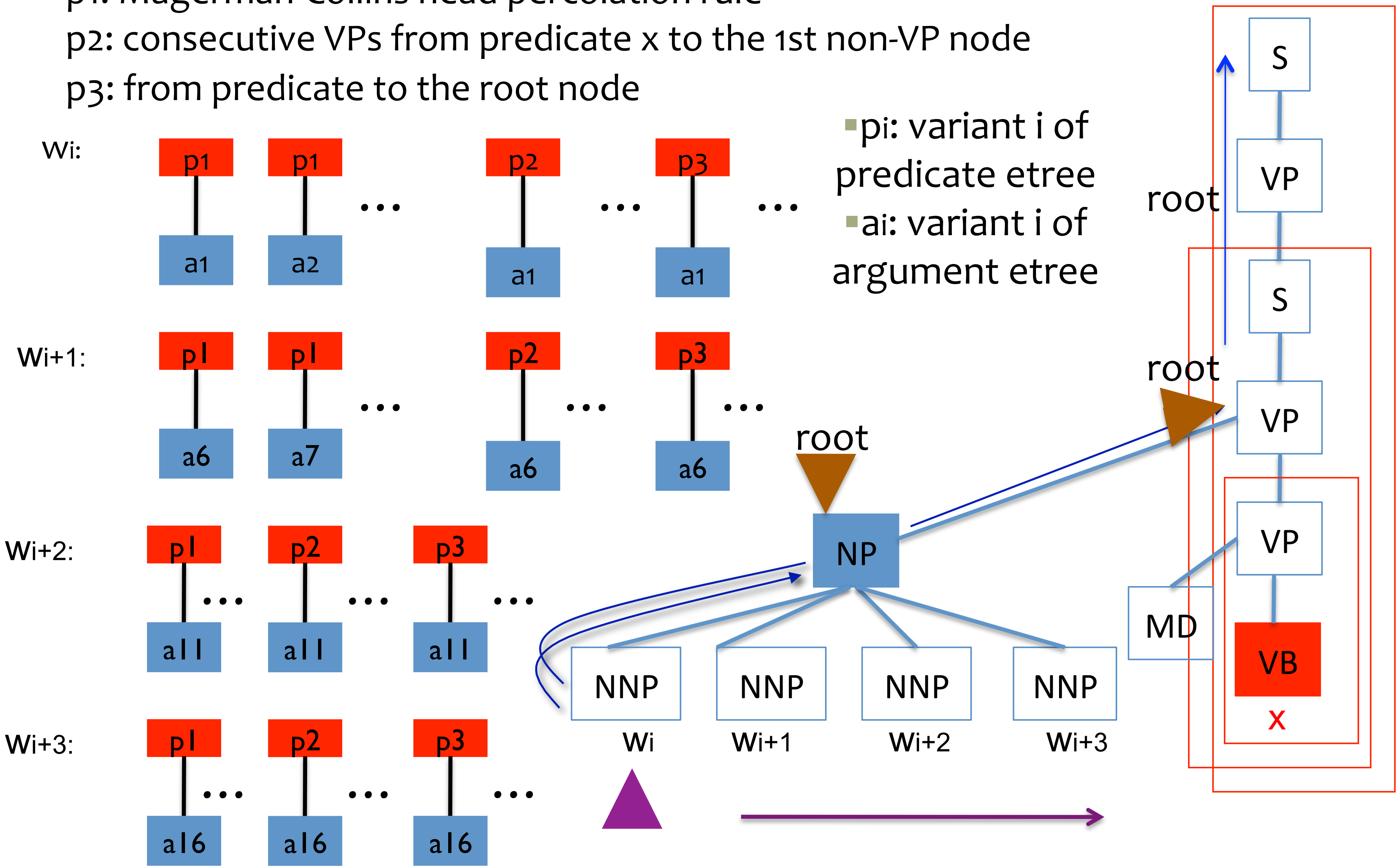
- Observations:
  - For a single derived tree, **multiple** TAG derivations exist and can be treated as latent structures
  - TAG derivations can **localize** long distance dependencies and provide useful features for SRL
- Hypothesis:
  - For different SRL instances, possibly **different** latent TAG derivations can provide discriminative features
  - Use TAG features to search for **more accurate SRL** classifiers. Do **not** search for "good" TAG derivations.
- Head choice in head-percolation and Lexical choice:
  - Extend head-percolation heuristics to generate multiple *predicate* e-trees and associated argument e-trees
  - Enumerate *all possible lexical heads* for *argument* constituents

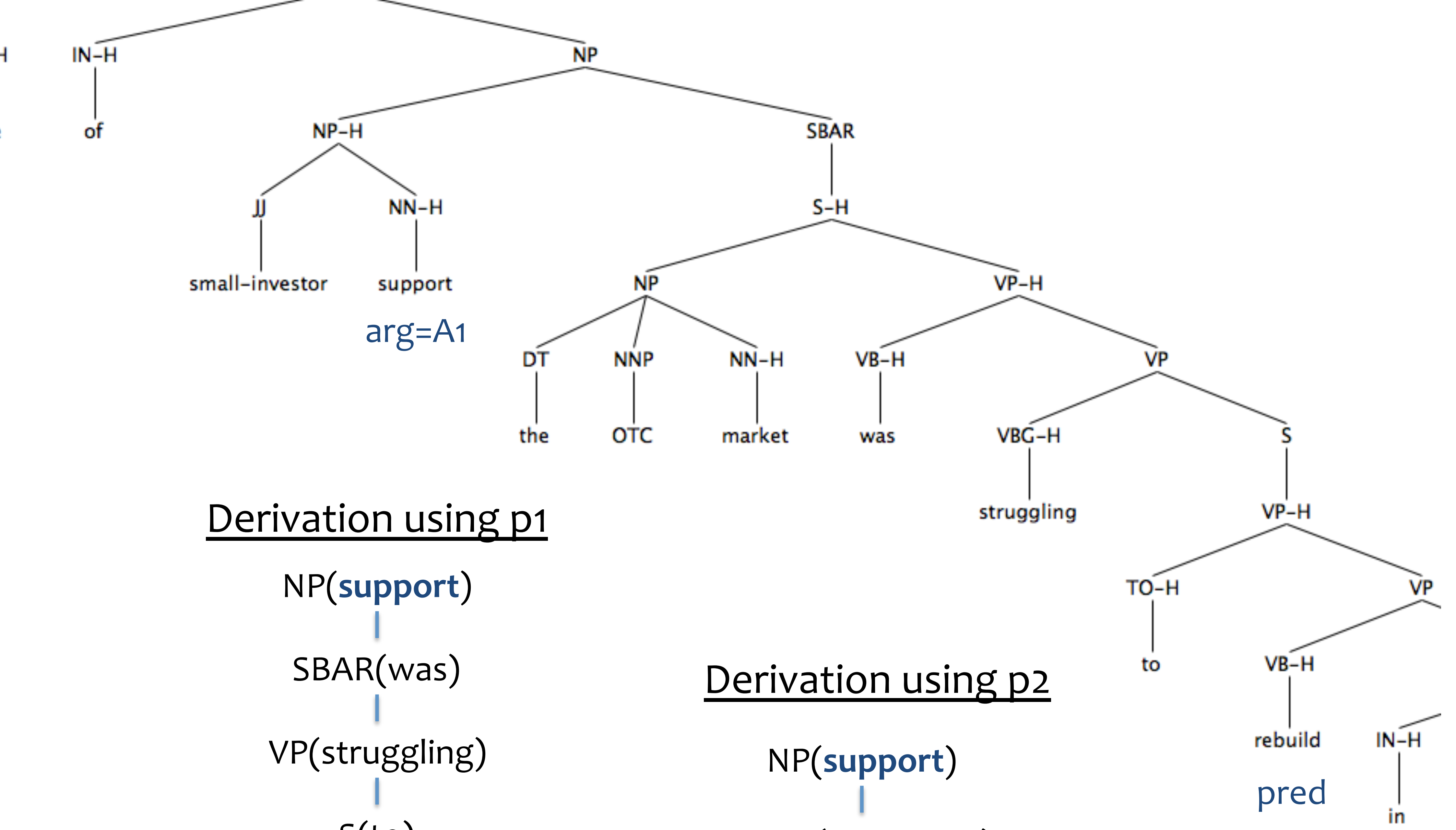
Generating latent TAG derivations <x,NP>

p1: Magerman-Collins head percolation rule

p2: consecutive VPs from predicate x to the 1st non-VP node

p3: from predicate to the root node





Derivation using p1

NP(**support**)
|
SBAR(was)
|
VP(struggling)
|
S(to)
|
VP(***rebuild***)

Derivation using p2

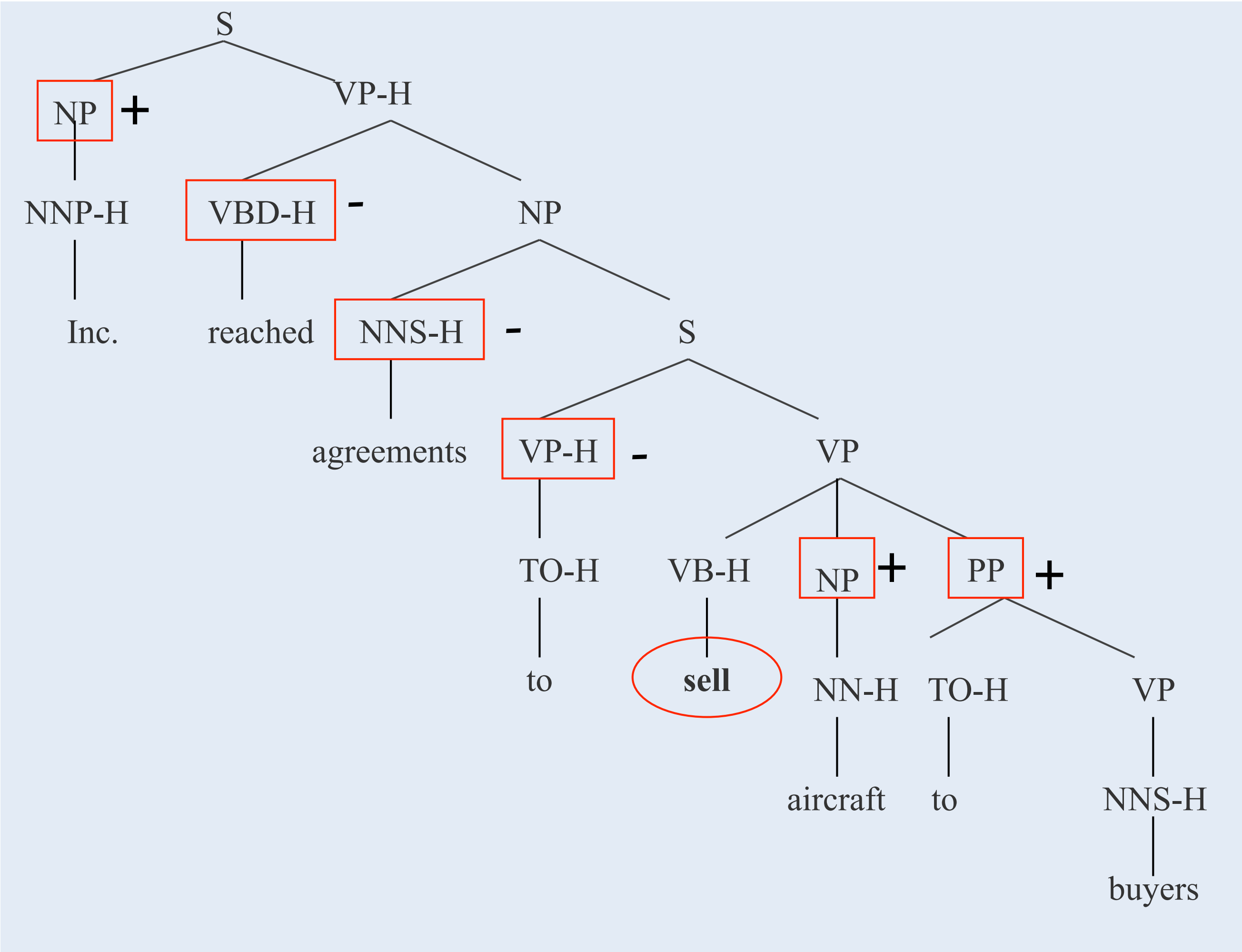
NP(**support**)
|
SBAR(struggling)
|
VP(***rebuild***)

Generating latent TAG derivations

- The set of features includes the three intermediate elementary trees closest to the predicate (if they exist)
- The average number of TAG derivations per SRL decision is ~130
- Problems with using latent features:
  - Scaling to millions of features and unlimited input length
  - Effectively use such a large number of latent features
  - Focus on discriminative features for each SRL instance
- Solution:
  - Latent support vector machines (LSVM)
  - Train several binary classifiers using LSVM and combine them using one vs. all for the full SRL task

Latent Support Vector Machines

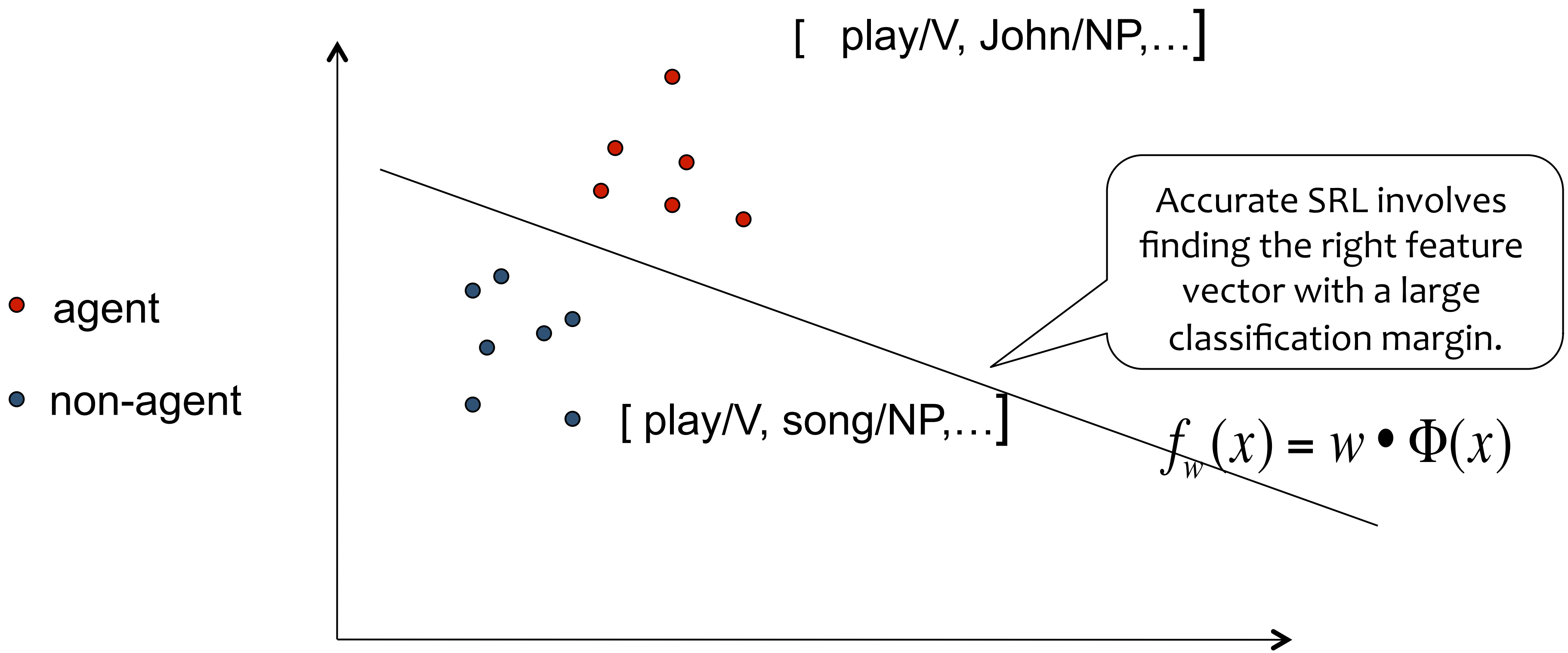
SRL as binary classification



- <*pred, arg candidate*> pair
- arg candidates taken from the original derived tree
- all depth-1 node in the pruned tree

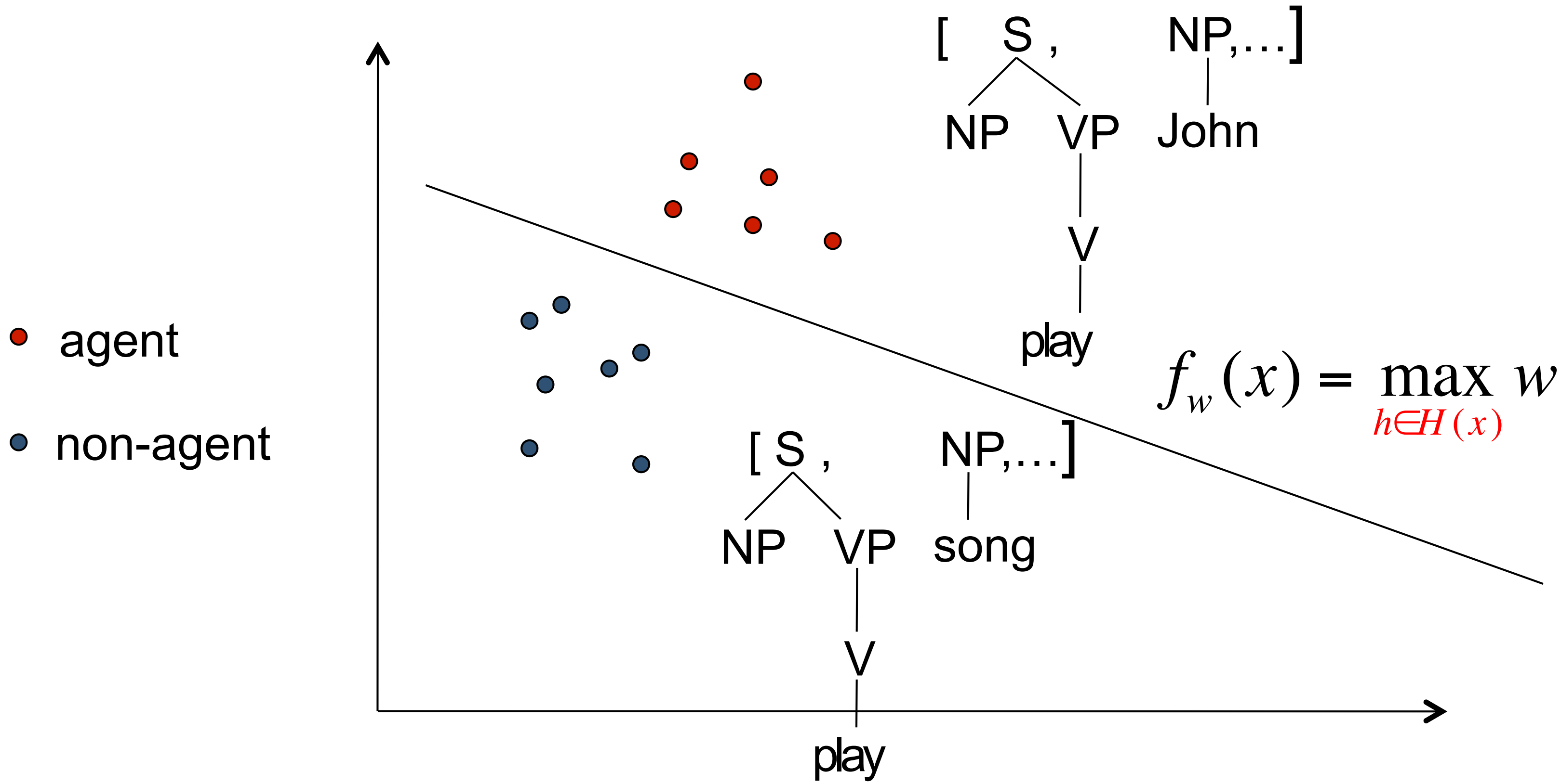
SRL as binary classification

$$L_D(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\max(0,1 - y_i f_w(x_i))$$



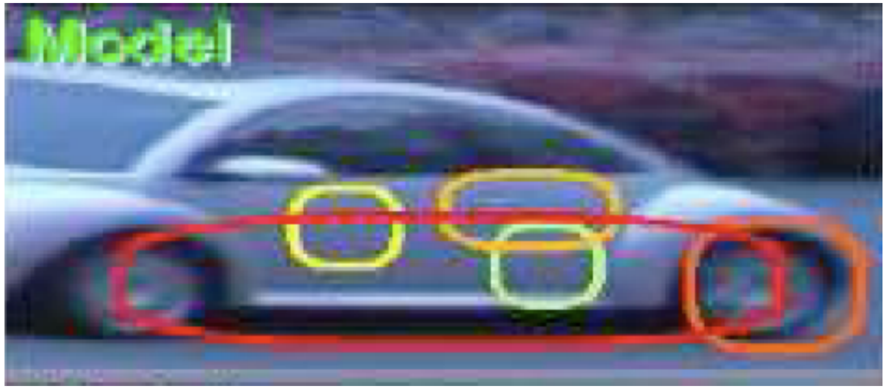
Latent SVM

$$L_D(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i))$$



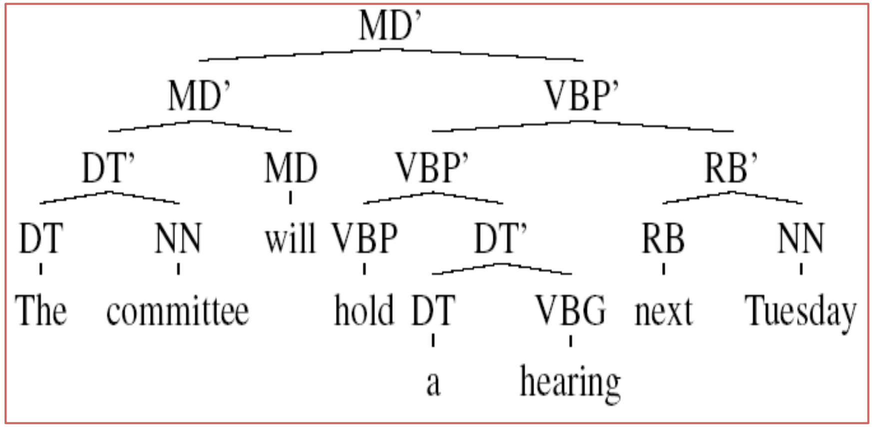
$$f_w(x) = \max_{h \in H(x)} w \bullet \Phi(x, h)$$

Previous work



Object detection in images

[P. Felzenszwalb et al. 2008]



Sentence classification for language modeling (in MT)

[Cherry & Quirk, 2008]

Semi-convexity (Felzenszwalb et al. 2008)

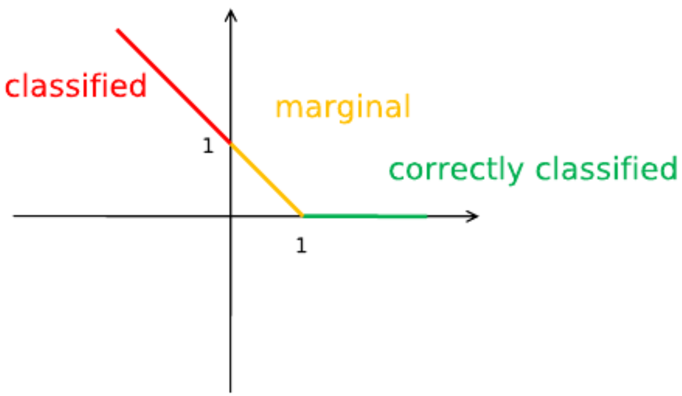
- $f_w(x) = \max_{h \in H(x)} \boxed{w \bullet \Phi(x, h)}$
- Maximum of convex function is convex, thus

  $f_w(x) = \max_{h \in H(x)} w \bullet \Phi(x, h)$ is convex in w, thus

  $\max(0, 1 - y_i f_w(x_i))$ is convex for negative examples

- $L_D(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f_w(x_i))$

Objective $L_D(w)$ becomes convex if we fix the latent structure h for positive examples.



Latent SVM training

- Two-step optimization algorithm (Felzenszwalb et al. 2008):
- Initialize w and iterate:
  1. Pick best h for each positive example. For each training example x pick $h = argmax_h\, w \bullet \Phi(x,h)$
  2. Find w for objective function with fixed h optimized using online learning (stochastic gradient descent)
     - *svmsgd* (Léon Bottou)
- In our implementation:
  - examples: <predicate, argument candidate> pair
  - h: best latent TAG derivation, picked for each positive and negative SRL instance

Latent SVM training

- For each training example, the phrase-structure tree remains fixed
  - Gold Treebank phrase-structure tree is used for training
  - Charniak parser output (from CoNLL 2005 shared task) is used for test data
- All the latent TAG derivations for a given sentence produce the *same* phrase-structure tree
- Each word lexicalizes one tree each and so all derivations have same number of steps

Experimental Results

Experimental Setup

- Data:
  - CoNLL-2005 shared task released data
  - PropBank Section 02-21 for training, 23 for testing
- Argument Set Under Consideration:
  - {A0, A1, A2, A3, A4, A5, AM-*, R-A*}
- Model: one-vs-all binary classifiers
  - Svmsgd (linear kernel)
- Evaluation metrics: Precision/Recall/F-score
- Baseline1: std
- Baseline2: std + tag (Liu and Sarkar 2007)
- Initial weights for LSVM iterations are from Baseline2

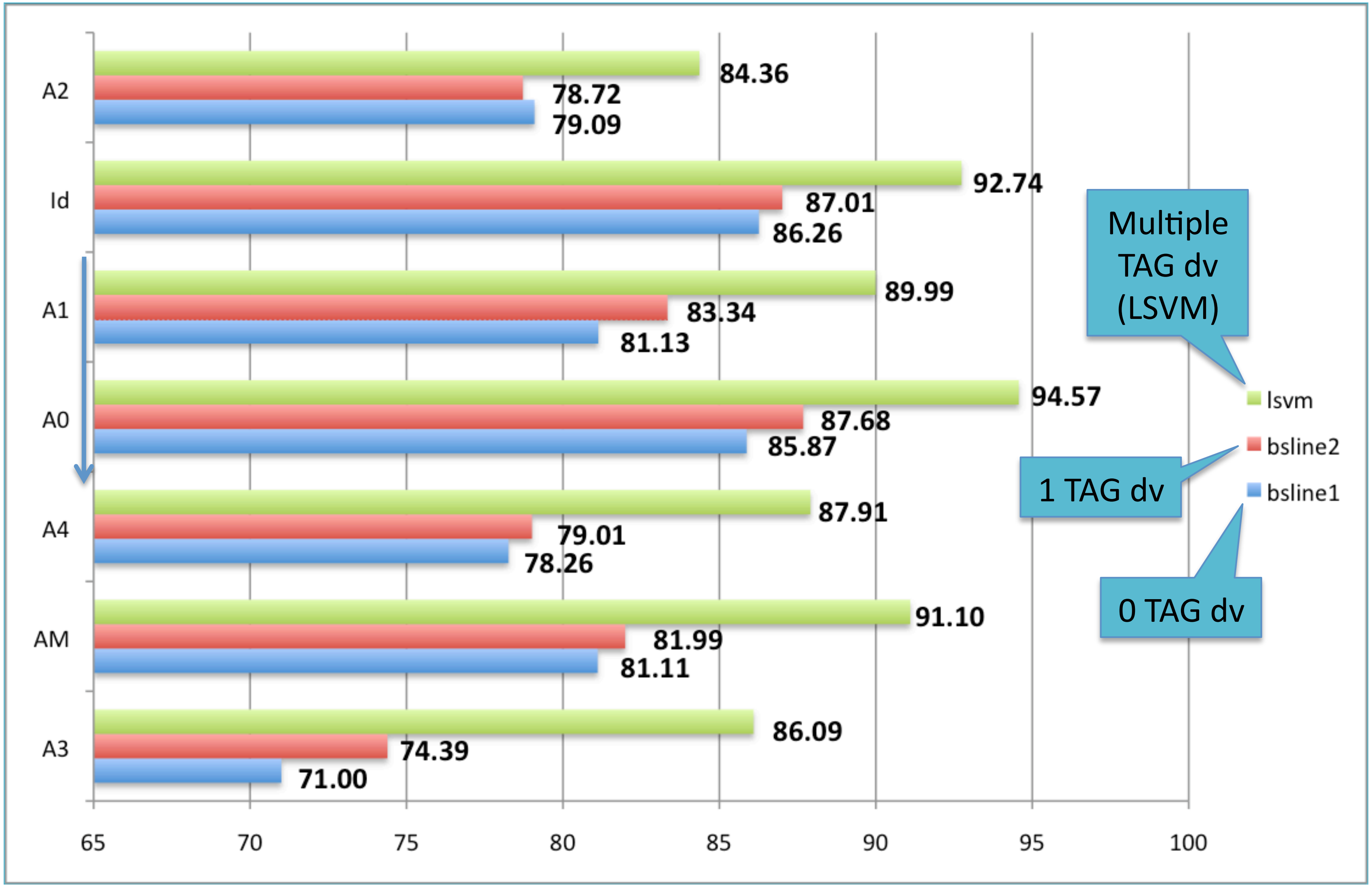
Architecture of our SRL system

- On a given parse tree, run the pruning component: some candidate spans are potential arguments, the others are labeled NONE
- Run a binary classifier for **identification** and have some spans labeled ARG and the rest NONE
- Run binary classifiers for **classification**: A0 vs not-A0, A1 vs not-A1, etc. on the nodes labeled ARG
- Combine output of binary classifiers using one vs all
  - for each ARG node pick binary classifier with highest confidence and decide the label of each node: A0, A1, A2, …
- Convert output to CoNLL 2005 shared task format and run CoNLL05 evaluation script.

CoNLL 2005 Shared Task / Charniak parser

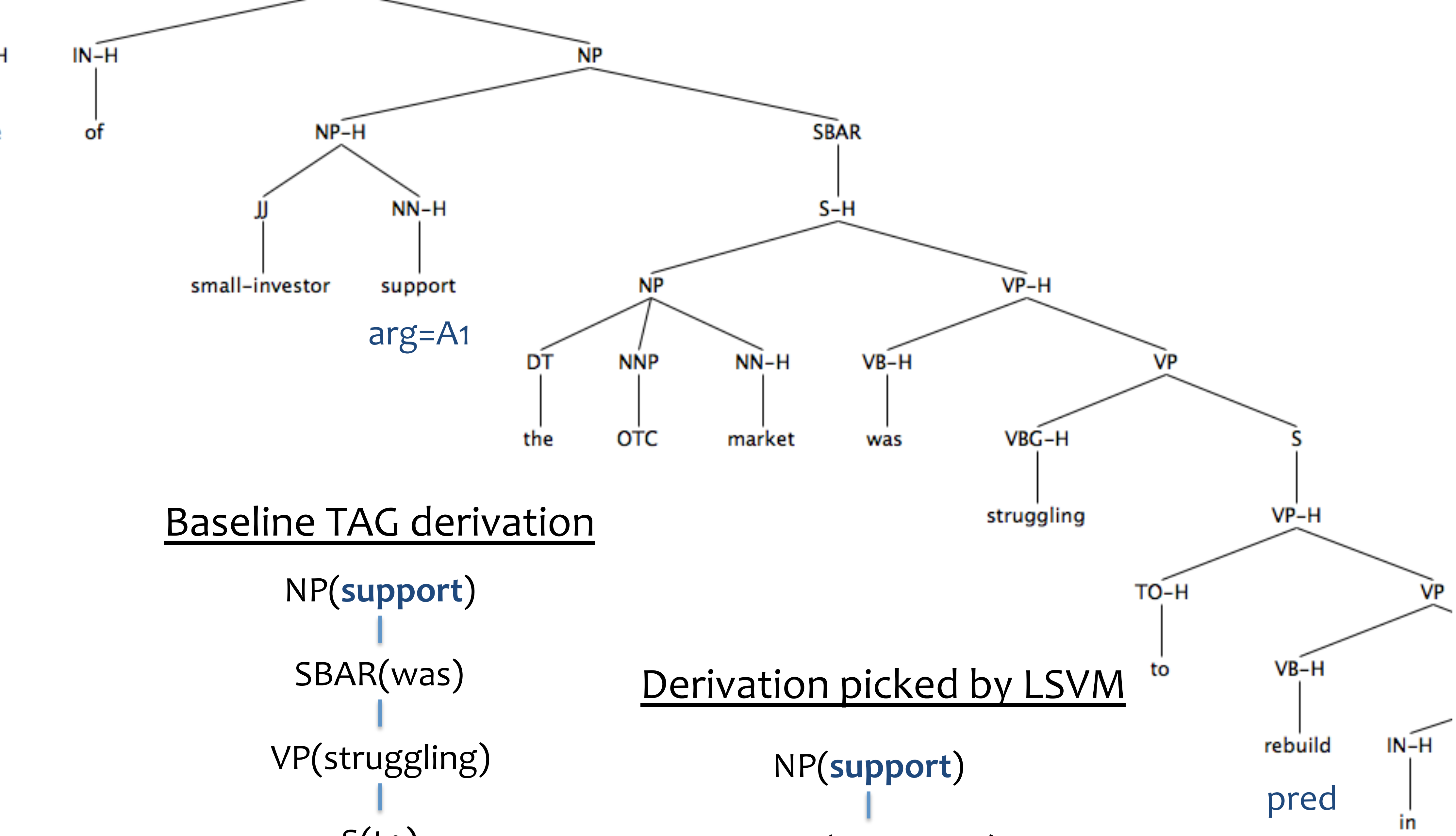
	Toutanova et al. (2008)			LSVM-SRL		
	Prec.	Rec.	F1	Prec.	Rec.	F1
Overall	81.90	78.81	80.32	95.90	84.05	**89.59**
A0	88.37	88.91	88.64	98.78	93.24	**95.93**
A1	81.50	81.27	81.38	91.95	79.30	**85.16**
A2	73.44	68.74	71.01	99.27	73.96	**84.77**
A3	75.00	55.49	63.79	90.85	74.57	**81.90**
A4	74.74	69.61	72.08	95.24	78.43	**86.02**
A5	100.00	80.00	**88.89**	40.00	80.00	53.33
AM-*	78.19	69.98	73.86	97.60	83.51	**90.01**
R-AM-*	73.91	61.44	67.10	70.76	99.02	**82.54**

Experimental results: F-score



Analysis: Individual binary classifiers, id, A0 vs. not-A0, etc.

class	No TAG (p/r%)		1 TAG deriv		Latent TAG derivs		stop iter	Recall bound
id	87.71	84.86	89.00	85.21	98.96	86.38	12	86.90
A0	86.46	85.30	87.87	87.50	99.26	90.31	18	94.24
A1	78.70	83.72	84.56	82.16	99.69	82.00	22	84.37
A2	85.04	73.91	83.00	74.86	99.26	73.35	26	76.24
A3	77.04	65.82	83.46	67.09	98.48	76.47	18	78.24
A4	77.42	79.12	90.14	70.33	98.77	79.21	20	80.20
AM	80.85	81.39	82.10	81.87	97.73	85.31	22	85.75



Baseline TAG derivation

NP(**support**)
|
SBAR(was)
|
VP(struggling)
|
S(to)
|
VP(***rebuild***)

Derivation picked by LSVM

NP(**support**)
|
SBAR(struggling)
|
VP(***rebuild***)

(S (PP (IN In)
(NP a Madrid hotel room))
(NP (DT a)
(NN **viewer**)) arg=A1
(VP (VB caught)
(NP a TV show ending)
(PU ,)
(S (ADVP only)
(VP (TO to)
(VP (VB be)
(VP (VBN urged)
(PP (IN by)
(NP the announcer)
(S (VP (TO to)
(VP (PU ``)
(VP (VB ***stay***) pred
(ADJP (VBN tuned))
(PP (IN for)
(NP another show)))
(PU ''))))))))))

Baseline TAG derivation

S(caught)

NP(**viewer**) S(to)

VP(be)

VP(urge)

S(to)

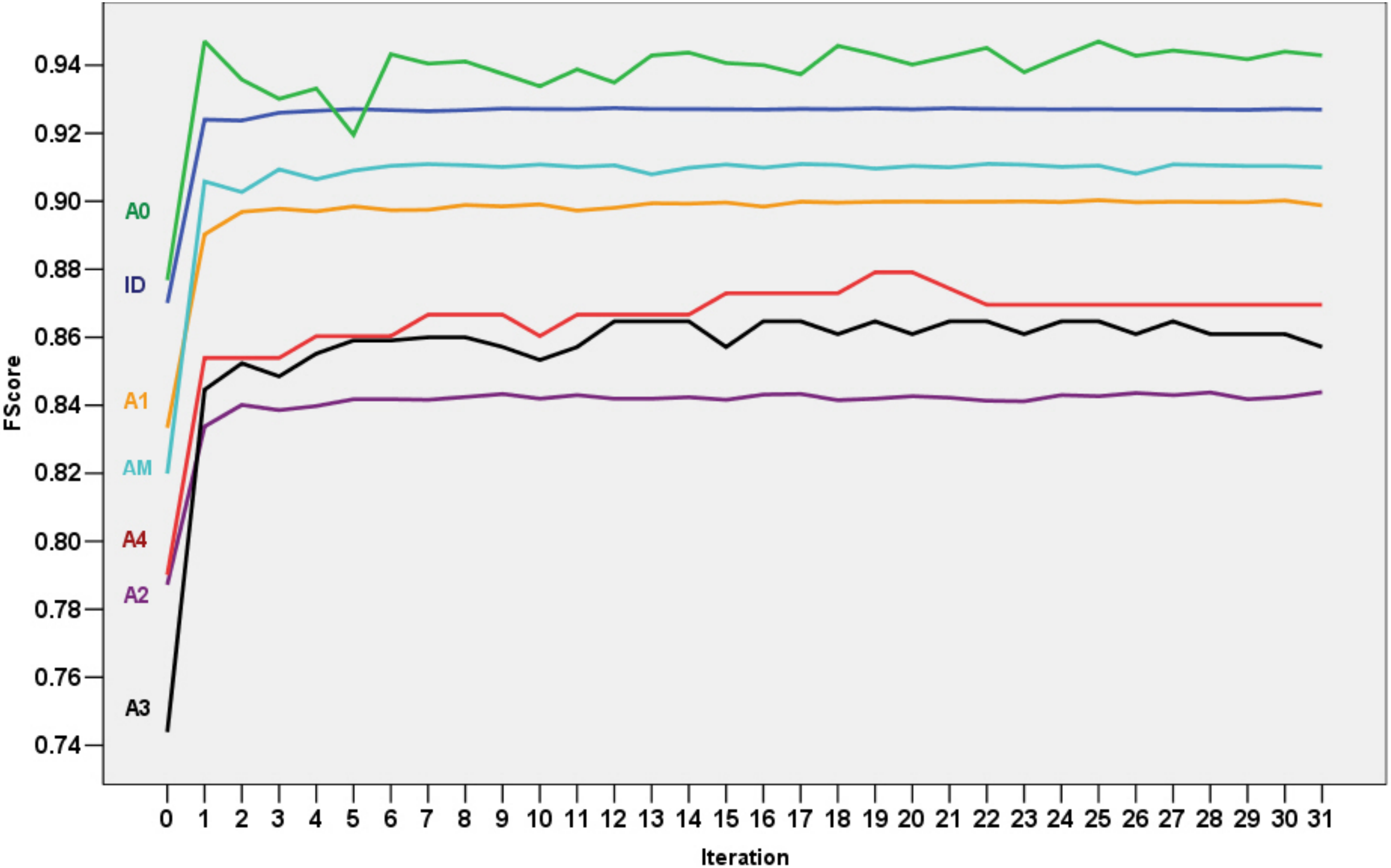
VP(***stay***)

Derivation picked by LSVM

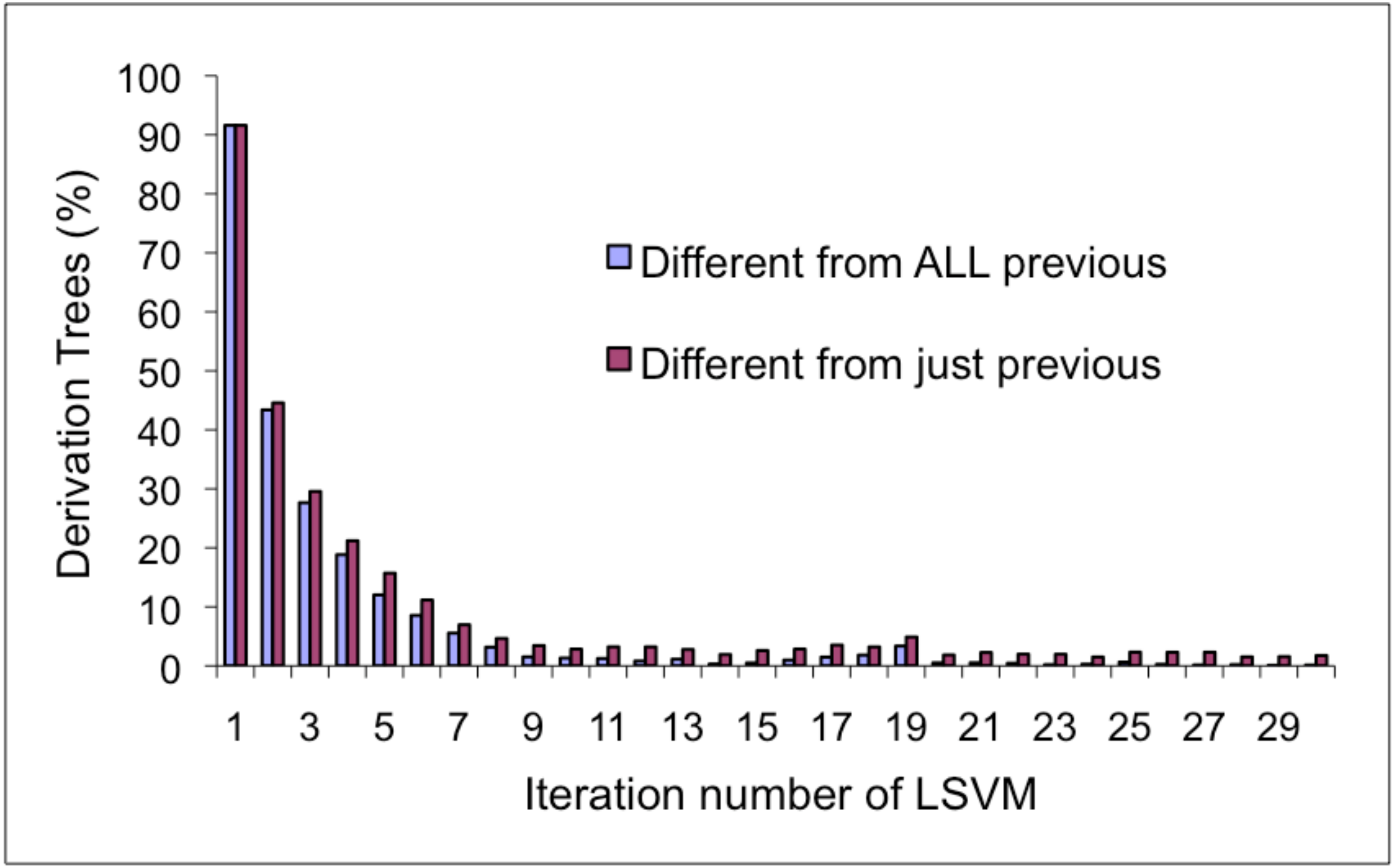
PP(In)

NP(**viewer**) S(***stay***)

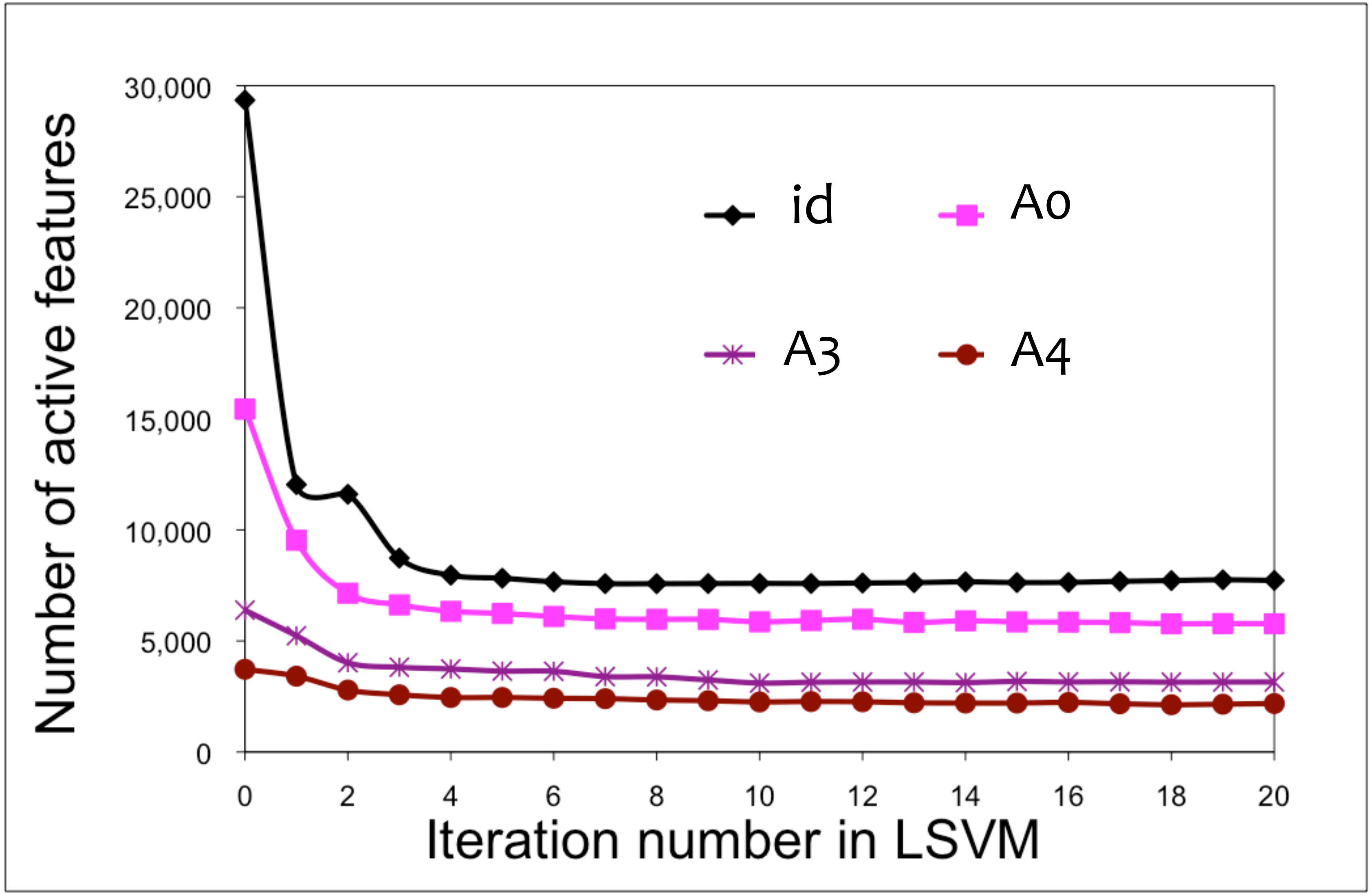
Analysis: F-scores across LSVM iterations



Analysis: change of derivation trees over LSVM iterations



Analysis: distribution of active features over LSVM iterations



Summary

- Latent TAG derivations and LSVM provide predictive features and very high precision and similar recall.
- LSVM boosts SRL F-score from 80% to 89%
- LSVM picked 8K features from the pool of 1,242,869 all possible.
- Further analysis of LSVM derivation trees is in our NAACL 2010 paper
  - Careful v.s. random initialization in LSVM training
  - How is LSVM taking advantage of the latent derivations?

Current Work

- Log linear models: sum over all latent TAG derivations per SRL decision
- Release of code and output of our system on CoNLL dev and test data
- Learn something about the SRL task from the derivations selected by LSVM
- LSVM is a general learning framework that can be potentially applied to other NLP tasks
- LSVM for (TAG) parsing

Thank you!

Analysis: Does initialization matter?

- Does picking the initial derivation tree carefully matter? Or can we simply select one at random.
- We compared A0-vs-not-A0 classifier with and without random choice of initial TAG derivation with identification classifier remaining the same

1 TAG dv, Magerman-Collins (A0: Baseline2)			1 TAG dv, Random (A0: avg over 5 runs)		
Precision	Recall	F-score	Precision	Recall	F-score
87.87	87.50	87.68	71.15±.79	86.11±.29	77.92±.36

Magerman-Collins init+LSVM (A0: Baseline2)			Random init+LSVM (A0: avg over 5 runs)		
Precision	Recall	F-score	Precision	Recall	F-score
99.26	90.31	94.57	84.54±8.00	86.44±2.12	85.33±4.49

Analysis: Why LSVM does better?

- Compare the LSVM argmax derivation tree with the Baseline2 Magerman-Collins derivation tree.
- Track changes when LSVM was correct and Baseline2 was incorrect.
- Five major categories of changes in derivation trees.

	ID	A0	A1	A2	A3	A4
Lexical choice	26.1	**44.5**	**47.7**	**74.4**	28.6	18.8
Distance	**79.9**	18.4	15.9	5.3	3.9	6.3
Predicate etree	**90.7**	**43.4**	**58.1**	**74.6**	**80.5**	**81.3**
Argument etree	47.1	**56.6**	**65.8**	**80.6**	28.6	25.0
Intermediate etree	5.7	17.1	14.7	2.3	14.3	6.2

CoNLL 2005 Shared Task / Charniak parser

	Corr.	Excess	Missed	Prec.	Rec.	F1
Overall	11360	506	2245	95.90	84.05	89.59
A0	3322	41	241	98.78	93.24	95.93
A1	3907	342	1020	91.95	79.30	85.16
A2	821	6	289	99.27	73.96	84.77
A3	129	13	44	90.85	74.57	81.90
A4	80	4	22	95.24	78.43	86.02
A5	4	6	1	40.00	80.00	53.33
AM-*	3101	76	612	97.60	83.51	90.01
R-AM-*	305	126	3	70.76	99.02	82.54
V	5126	141	141	97.32	97.32	97.32