Recent Advances in Dependency Parsing

Qin Iris Wang
AT&T Interactive
qiniriswang@gmail.com

Yue Zhang
Cambridge University
frcchang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Topic-Author Clouds of NAACL-HLT 2010

Quite popular

Courtesy: http://www.wordle.net

Dependency Parsing Events in Recent Years

- CoNLL-X Shared Task: Multi-lingual Dependency Parsing in 2006
  - http://nextens.uvt.nl/~conll/

- Tutorial by Joachim Nivre and Sandra Kuebler at COLING-ACL in 2006
  - http://aclweb.org/mirror/acl2006/program/tutorials/dependency.html

- CoNLL Shared Task: Joint Parsing of Syntactic and Semantic Dependencies in 2008
  - http://www.yr-bcn.es/conll2008/

A Few Notes

- This tutorial is focused on **recent development** in dependency parsing
  - After 2006
- Although this tutorial is on dependency parsing, most approaches are applicable to other formalisms
  - E.g., phrase-structure parsing or synchronous parsing for MT
- The field is really parsing instead of dependency parsing
  - Read all the parsing papers if you can!

Tutorial Goals

- Introduce data-driven dependency parsing (graph-based, transition-based and integrated models)
- Improve dependency parsing via statistical machine learning approaches
  - Explore more features with better learning algorithms
  - Better parsing strategies (efficiency and accuracy)
  - Using extra information sources

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
- Part D: the integrated models
- Part E: other recent trends in dependency parsing

Part A: Introduction to Dependency Parsing

Qin Iris Wang

AT&T Interactive
qiniriswang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Outline

- Part A: introduction to dependency parsing
  - **Dependency syntax**
  - Dependency parsing approaches
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
- Part D: the integrated models
- Part E: other recent trends in dependency parsing

Ambiguities In NLP

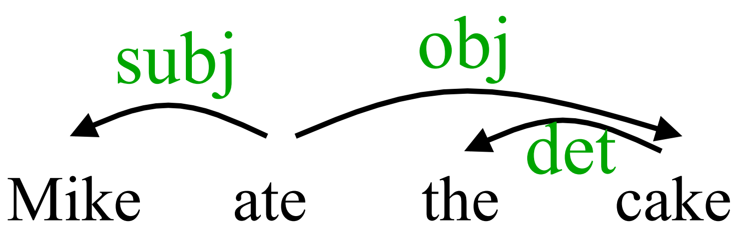
"I saw her duck."

How about

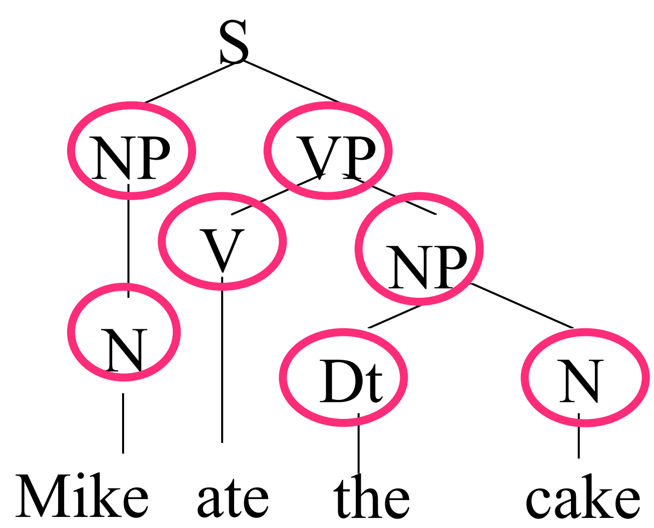
"I saw her duck with a telescope."

Dependency Structure vs. Constituency Structure

Parsing is one way to deal with the ambiguity problem in natural language.



Dependency structure



Constituency structure

Dependency Syntax

- A dependency structure represents syntactic relations (**dependencies**) between word pairs in a sentence
  - By drawing a link between the two words

- For the link: **a** **telescope**

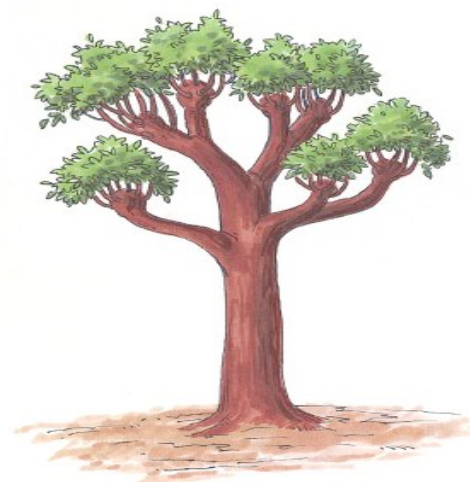
The **head** of ***a*** is ***telescope***

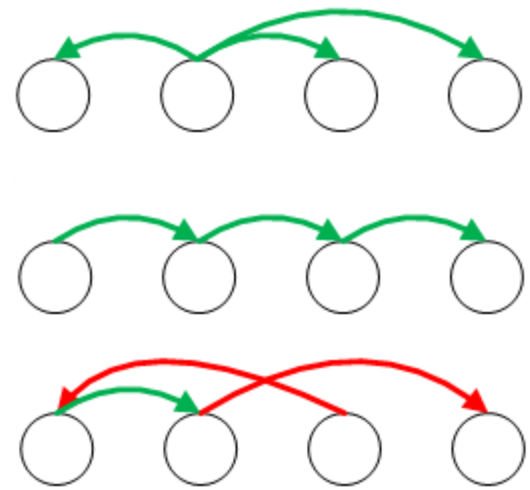
- Modifier
- Dependent
- Child

- Head
- Governor
- Parent

Dependency Graphs

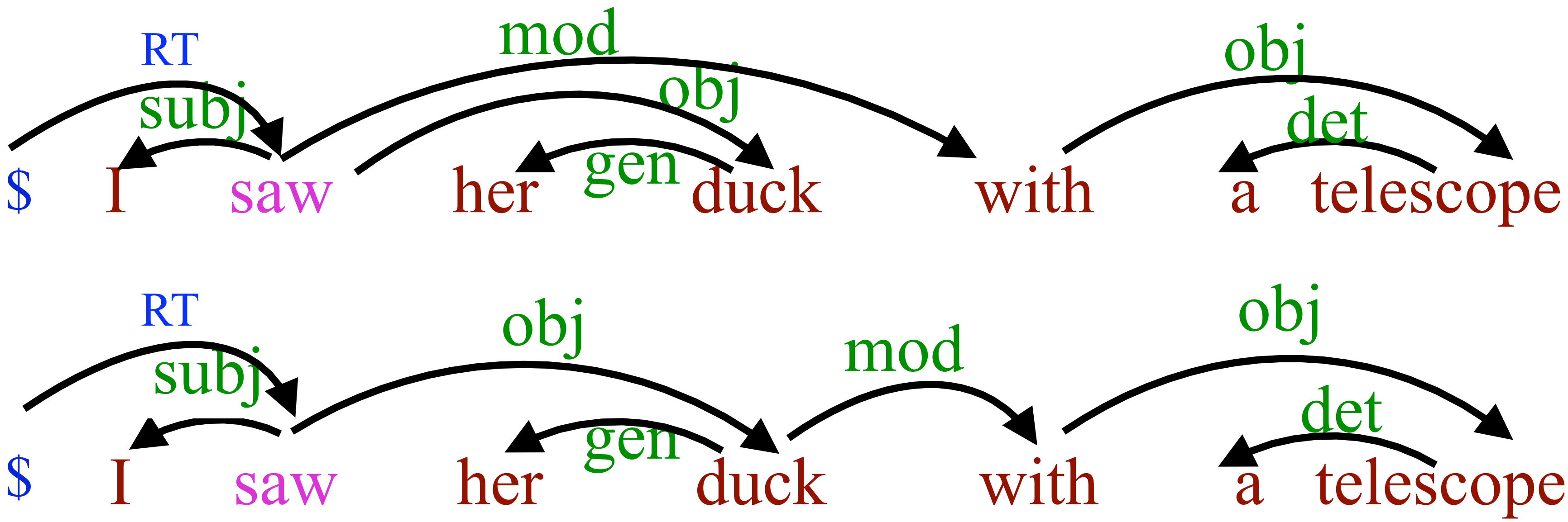
- A dependency structure is a directed graph G with the following constraints:
  - Connected
  - Acyclic
  - Single-head
  - Projective



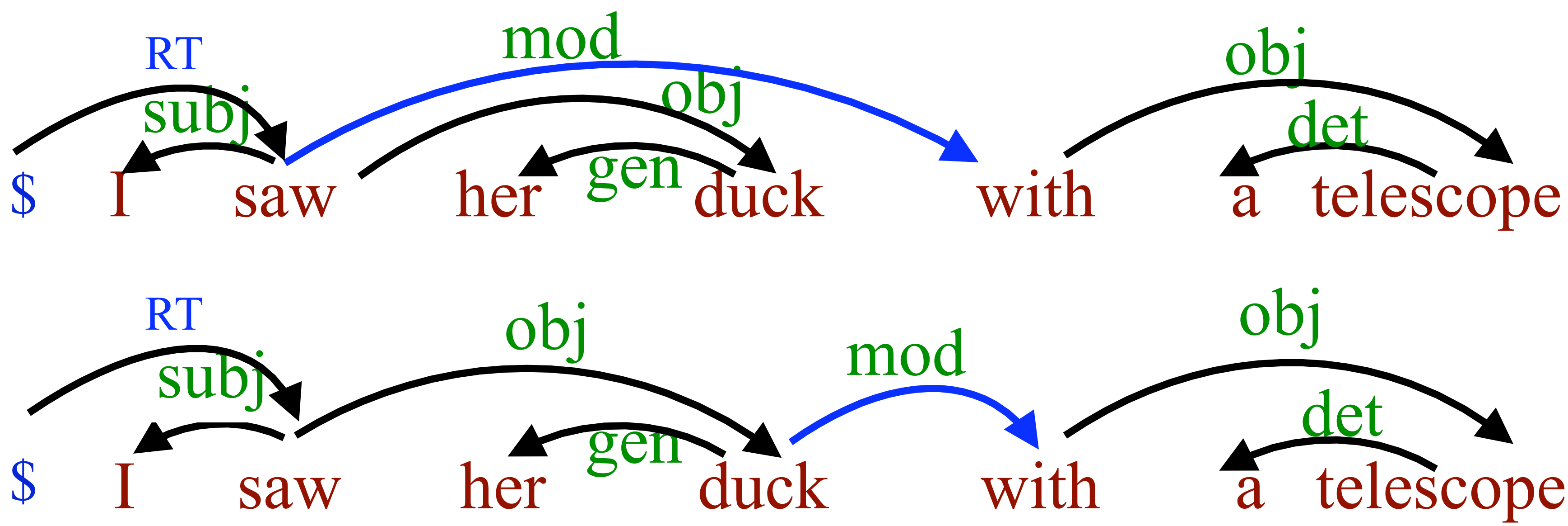


No crossing links (a word and its dependents form a contiguous substring of the sentence)

Dependency Trees

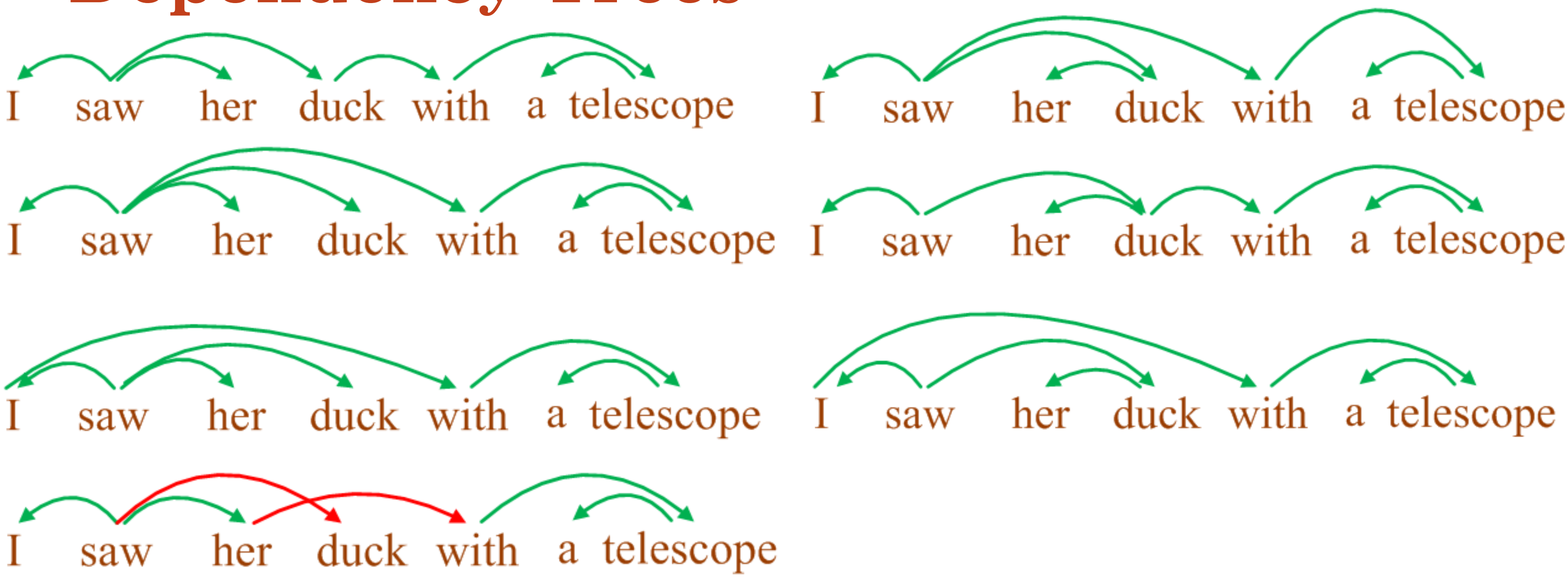


Dependency Trees



Dependency Trees

How many trees for a 20-word sentence?
Over one million!!



Over 100 possible trees for this seven-word sentence!

Non-projective Dependency Trees

- With crossing links
- Not so frequent in English
  - All the dependency trees from Penn Treebank are projective
- Common in other languages (Kuhlmann & Satta 09)
  - 23% sentences are non-projective in the Prague Dependency Treebank of Czech
  - Percentage in German and Dutch are even higher

• Long-distance dependencies
• Languages with free word order, such as German and Dutch

Outline

- Part A: introduction to dependency parsing
  - Dependency syntax
  - **Dependency parsing approaches**
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
- Part D: the integrated models
- Part E: other recent trends in dependency parsing

Dependency Parsing

- The problem:
  - Input: a sentence
  - Output: a dependency tree (**connected**, **acyclic**, **single-head**)

- Grammar-based parsing
  - Context-free dependency grammar
  - Constraint dependency grammar

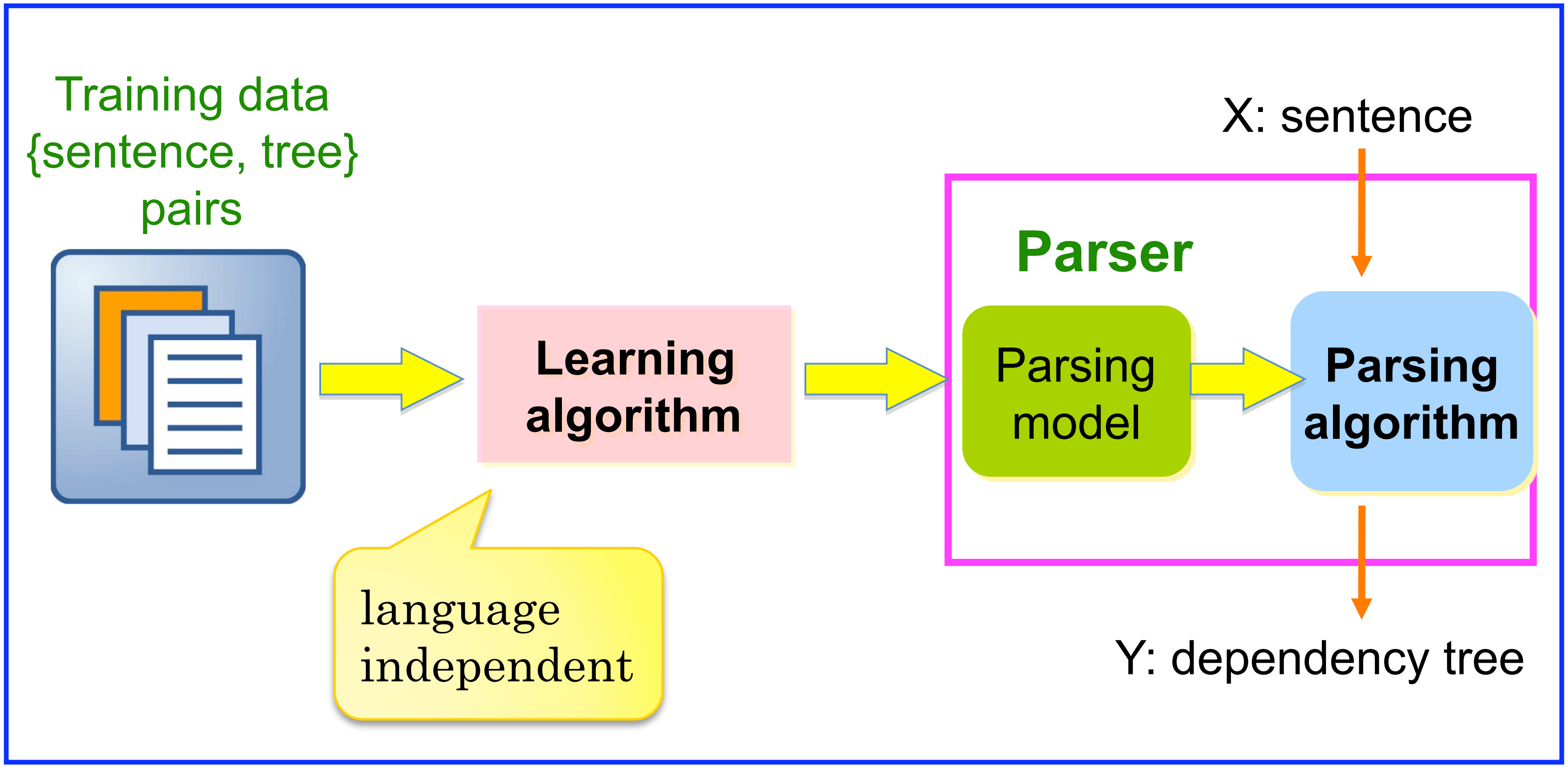
• **Ambiguities handling**
• **Incomplete search**

Data Driven Dependency Parsing

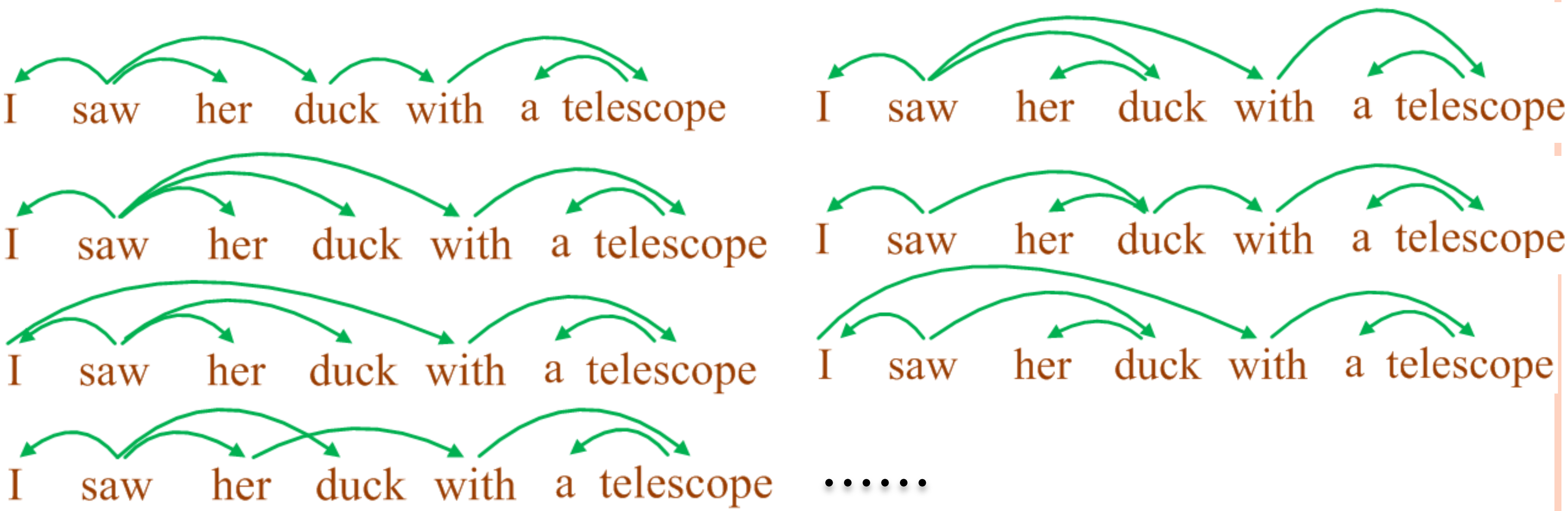
- **Data-driven parsing**
  - No grammar / rules needed; any tree is possible
  - Parsing decisions are made based on learned models
  - Can deal with ambiguities well

- **Three approaches**
  - Graph-based models
  - Transition-based models
  - Hybrid models

Data-driven Parsing Framework



Graph-based Models



- Score each possible output
- Search for a tree with the highest score
- Often use Dynamic Programming to explore search space

Graph-based Models

- Define a space of candidate dependency trees for a sentence
  - **Learning**: induce a model for scoring an entire tree
  - **Parsing**: find a tree with the highest score, given the induced model
  - Exhaustive search
  - Features are defined over a limited parsing history
  - Represented by Eisner 96, McDonald et al. 05a, McDonald et al. 05b and Wang et al. 07

Transition-based Models

- Define a transition system for mapping a sentence to its dependency tree
  - Predefine some **transition actions**
  - **Learning**: induce a model for predicting the next state transition, given the transition history
  - **Parsing**: construct the optimal transition sequence, given the induced model
  - Greedy search / beam search
  - Features are defined over a richer parsing history
  - Represented by Yamada & Matsumoto 03, Nivre & Scholz 04, Zhang & Clark 08, Huang et al. 09

Comparison

- Graph-based models
  - Find the optimal tree from all the possible ones
  - Global, exhaustive
- Transition-based models
  - Predefine some actions (shift and reduce)
  - Find the optimal action sequence
  - Local, Greedy or beam search
- The two models produce different types of errors
  - Error distribution (McDonald & Nivre 07)
  - Have complementary strengths

Hybrid Models

- Three integration methods
  - Ensemble approach: parsing time integration (Sagae & Lavie 2006)
  - Feature-based integration (Nivre & Mcdonald 2008)
  - Single model combination (Zhang & Clark 2008)
- Advantages
  - Gain benefits from both models

Summary – Introduction to Dependency Parsing

- Dependency Syntax

- Dependency parsing approaches
  - Graph-based models
  - Transition-based models
  - Hybrid models

References

- J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.
- L. Huang, W. Jiang, and Q. Liu. 2009. Bilingually-Constrained (Monolingual) Shift-Reduce Parsing. In *Proc. EMNLP*.
- M. Kuhlmann and G. Satta. 2009. Treebank Grammar Techniques for Non-Projective Dependency Parsing. In *Proc. EACL*.
- R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. ACL*.
- R. McDonald, F. Pereira, K. Ribarov and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. HLT-EMNLP*.
- R. McDonald and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proc. EMNLP-CoNLL*.
- J.Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. ACL-HLT*.

- J. Nivre and M. Scholz. 2004. Deterministic Dependency Parsing of English Text. In *Proc. COLING*.
- K. Sagae and A. Lavie. 2006a. Parser combination by reparsing. In *Proc. HLT-NAACL*.
- Q. I. Wang, D. Lin and D. Schuurmans. 2007. Simple Training of Dependency Parsers via Structured Boosting. In *Proc. IJCAI*.
- H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.
- Y. Zhang and S. Clark. 2008. A tale of two parsers: investigating and combining graph- based and transition-based dependency parsing us- ing beam-search. In *Proc. EMNLP*.

Recent Advances in Dependency Parsing

Qin Iris Wang

AT&T Interactive

qiniriswang@gmail.com

Yue Zhang

Cambridge University

frcchang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Part B: Graph-based Dependency Parsing Models

Qin Iris Wang

AT&T Interactive
qiniriswang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
  - **Dependency parsing model**
  - Parsing algorithms
  - Features
  - Learning approaches
- Part C: transition-based models
- Part D: the combined models
- Part E: other recent trends in dependency parsing

Dependency Parsing Model

- $\boldsymbol{X}$: an input sentence
- $\boldsymbol{Y}$: a candidate dependency tree
- $\boldsymbol{x_i \rightarrow x_j}$: a dependency link from word $\boldsymbol{i}$ to word $\boldsymbol{j}$
- $\Phi(X)$: the set of possible dependency trees over $\boldsymbol{X}$

Edge/link based factorization (Eisner 96)

$$Y^* = \arg\max_{Y \in \Phi(X)} score(Y \mid X)$$
$$= \arg\max_{Y \in \Phi(X)} \sum_{(x_i \rightarrow x_j) \in Y} score(x_i \rightarrow x_j)$$

- Applicable to both **probabilistic** and **non-probabilistic** models

Edge Based Factorization

$$Y^* = \arg\max_{Y \in \Phi(X)} \sum_{(x_i \rightarrow x_j) \in Y} score(x_i \rightarrow x_j)$$

$$score(x_i \rightarrow x_j) = \vec{f}(x_i \rightarrow x_j) \cdot \vec{\theta}$$

Standard linear classifier

A vector of features

A vector of feature weights

- The score of a link is dot product between feature vector and feature weights
  - What features we can use? (later)
  - What learning approaches can lead us to find the best tree with the highest score (later)

Score of a Link

- The score of each link is based on the features
- The features for the word pair: (***saw***, ***duck***)
  - (***saw***, ***duck***) = 1
  - POS (***saw***, ***duck***): (VBD, NN) = 1
  - PMI (***saw***, ***duck***) = 0.27 (PMI: pointwise mutual information)
  - dist (***saw***, ***duck***) = 2 dist2(***saw***, ***duck***) = 4

$$score\ (saw, duck)$$
$$= 1 * \theta_{(saw,\ duck)} + 1 * \theta_{(VB,\ NN)} + 0.27 * \theta_{PMI} + 2 * \theta_{dist} + 4 * \theta_{dist2}$$

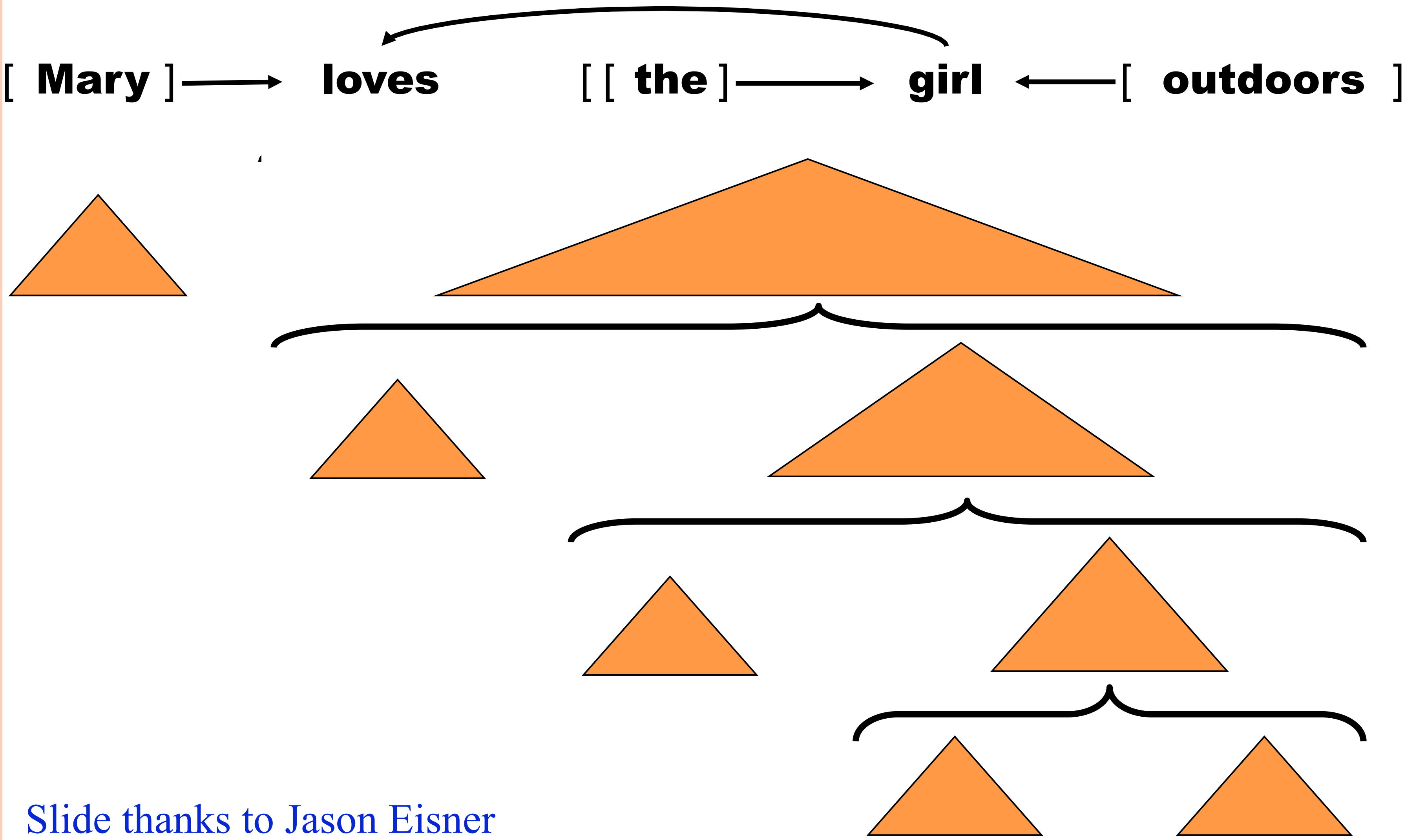
Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
  - Dependency parsing model
  - **Parsing algorithms**
  - Features
  - Learning approaches
- Part C: transition-based models
- Part D: the combined models
- Part E: other recent trends in dependency parsing

Comparison of Some Popular Dependency Parsing Algorithms

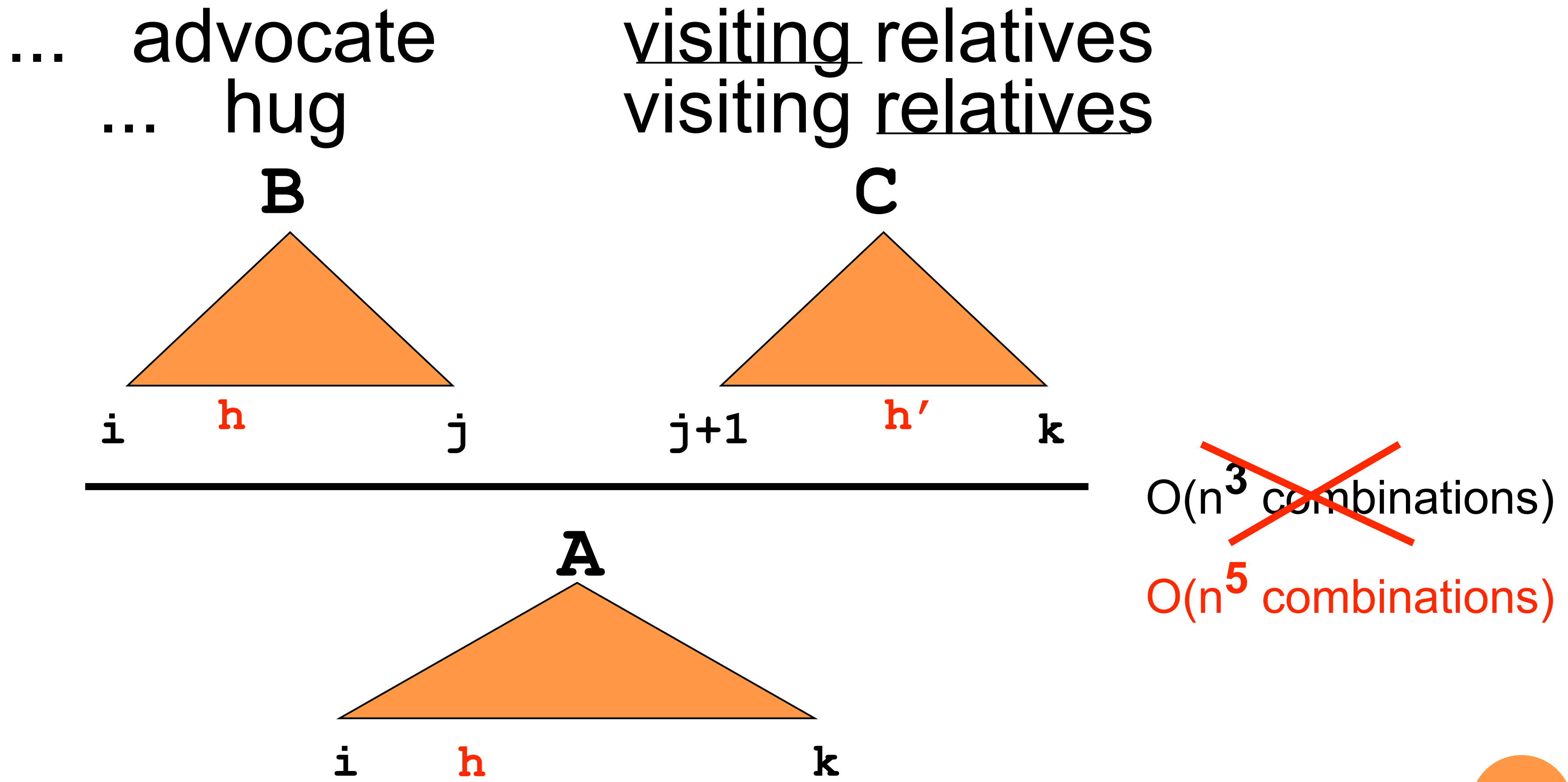
Name	Inventor	Projectivity	Complexity
CKY-style chart parsing	Cocke–Younger–Kasami	Projective	$O(n^5)$
Eisner $O(n^3)$ parsing alg.	Eisner (96)	Projective	$O(n^3)$
Maximum Spanning Tree	Chu-Liu-Edmonds (65, 67)	Non-projective	$O(n^2)$
Shift-Reduce style parsing	Yamada, Nivre	Projective	$O(n)$

The CKY-style algorithm O(n^5)



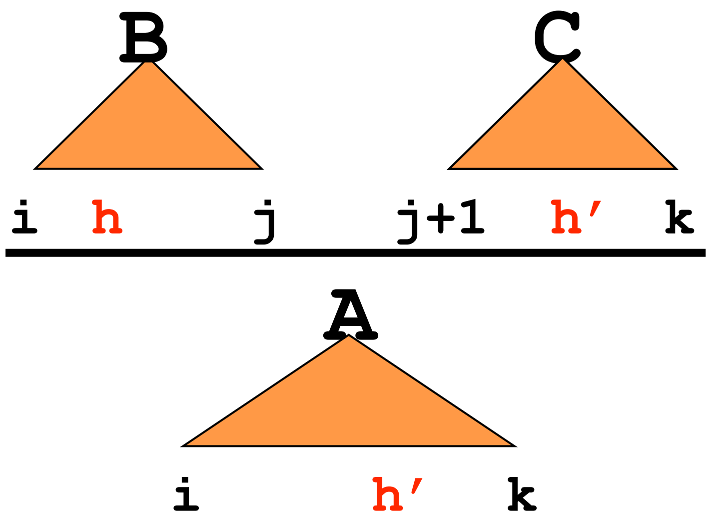
Slide thanks to Jason Eisner

Why CKY is $O(n^5)$ not $O(n^3)$



Slide thanks to Jason Eisner

$O(n^4)$ Parsing Algorithm
(Eisner&Satta 99)



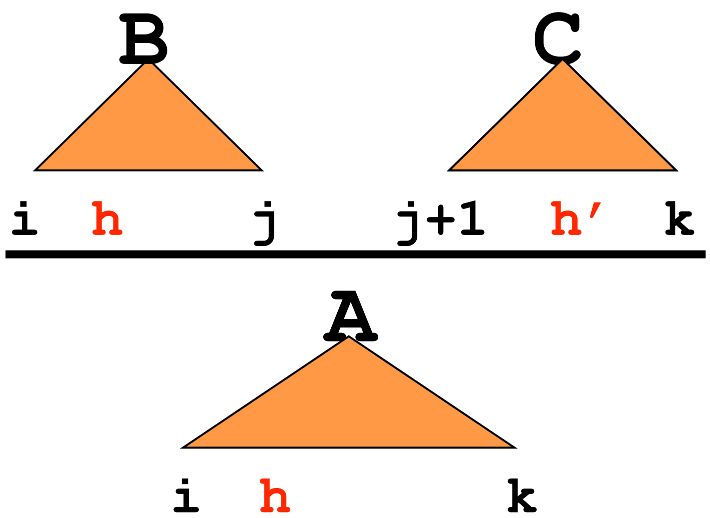
- Combine what B and C?
  - must try different-width C's (vary **k**)
  - must try different midpoints **j**
  - Separate these!

Slide thanks to Jason Eisner

$O(n^4)$ Parsing Algorithm

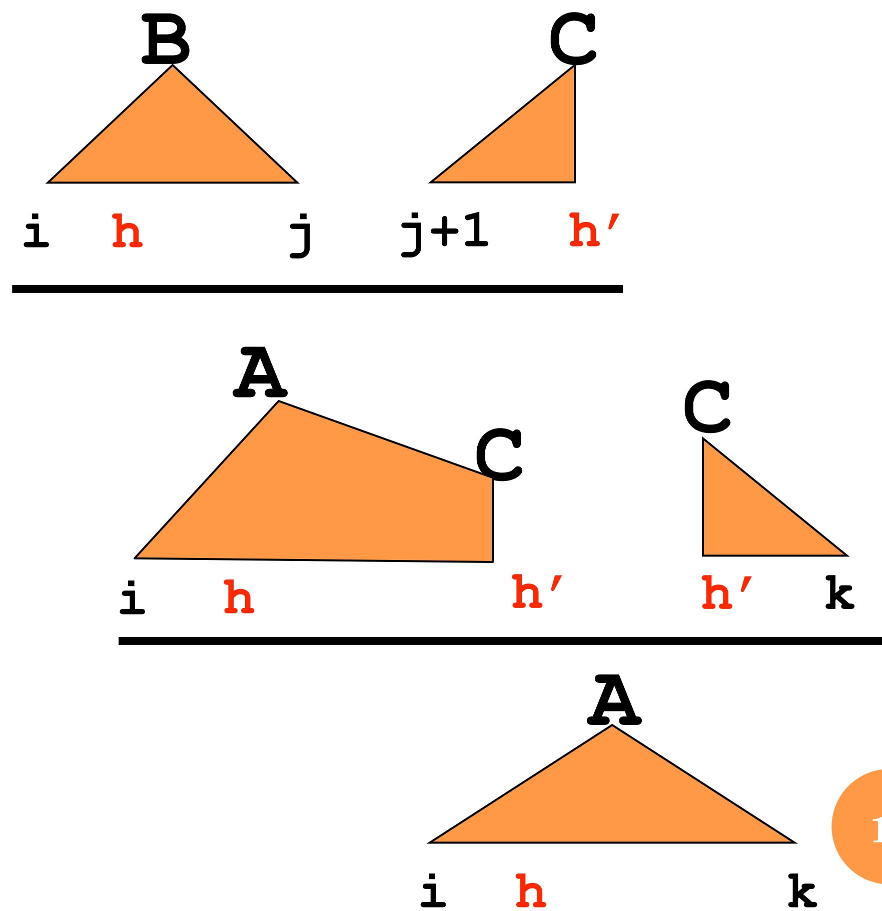
(Eisner&Satta 99)

(the old CKY way)



Step 1: (i, j, h, h')
$O(n^4)$

Step 2: (i, h, h', k)
$O(n^4)$



Slide thanks to Jason Eisner

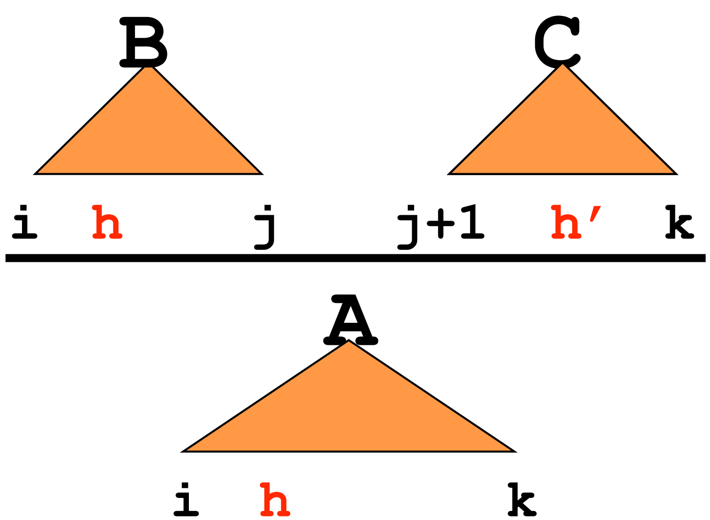
We Can Do Better

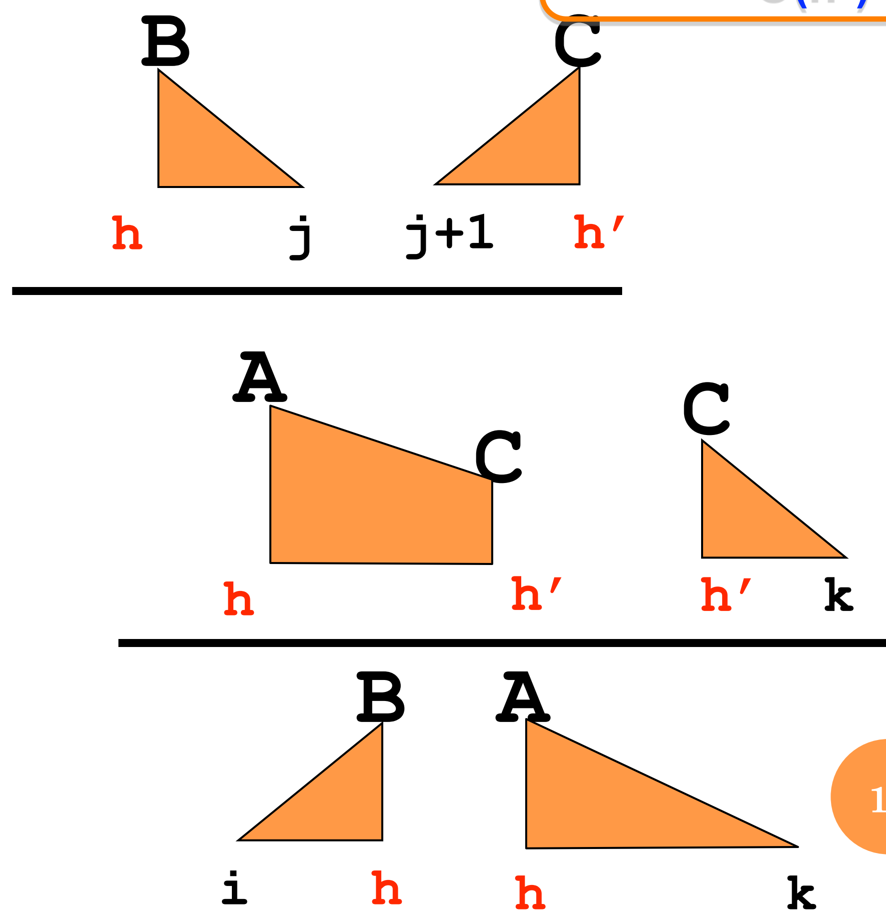
Step 1: (j, h, h’) $O(n^3)$

Step 2: (h, h’, k) $O(n^3)$

Step 3: (i, h, k) $O(n^3)$

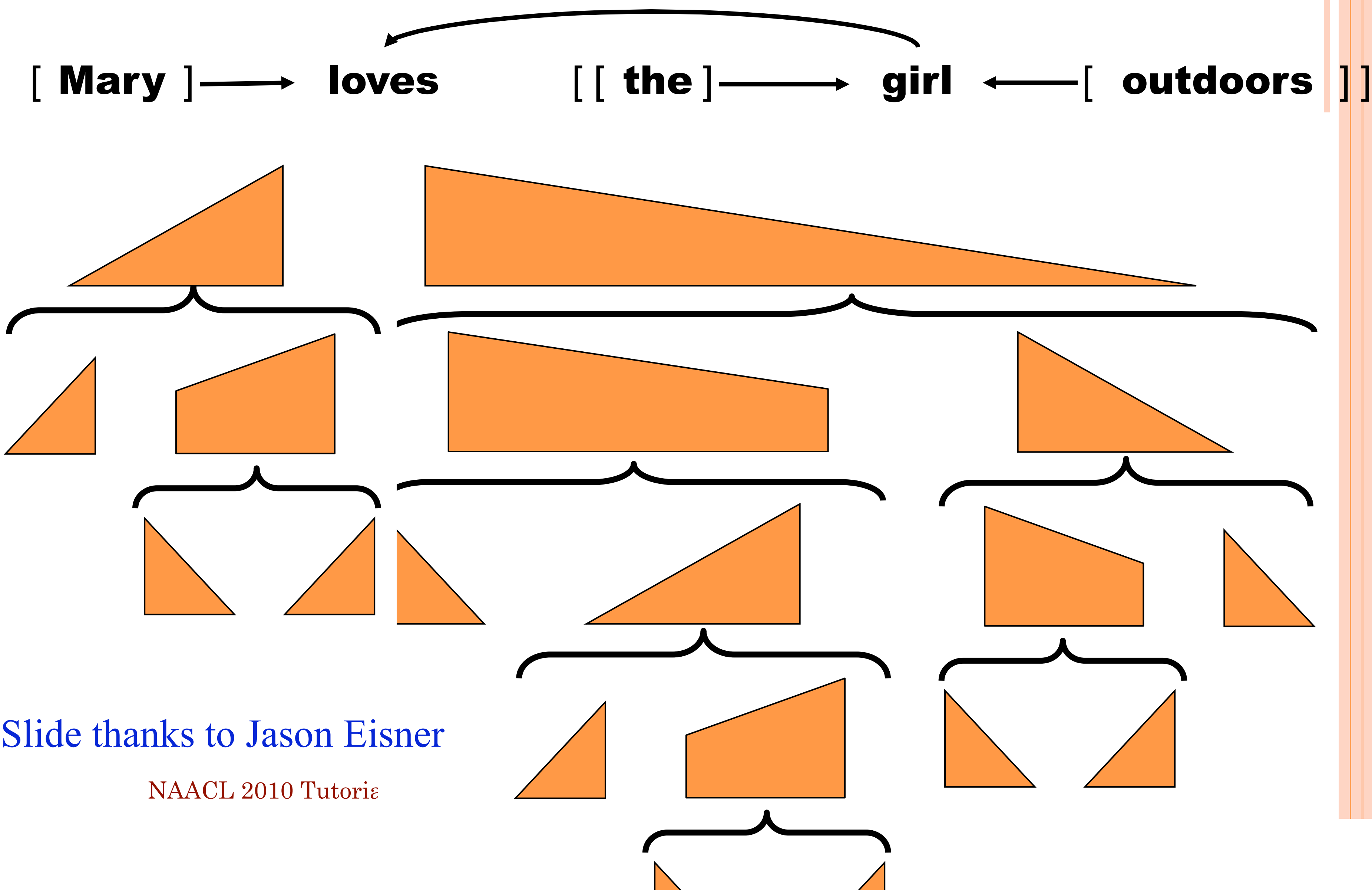
(the old CKY way)





Slide thanks to Jason Eisner

The O(n^3) Half-Tree Parsing Algorithm (Eisner 96)

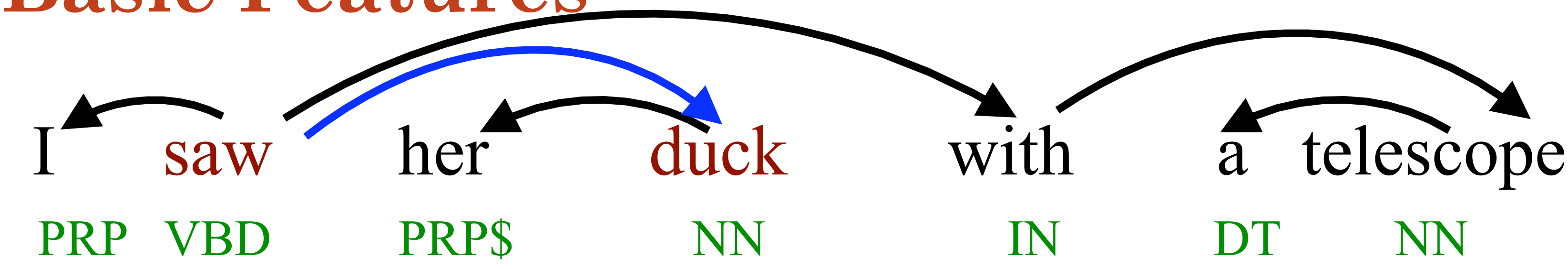


Slide thanks to Jason Eisner

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
  - Dependency parsing model
  - Parsing algorithms
  - **Features**
  - Learning approaches
- Part C: transition-based models
- Part D: the combined models
- Part E: other recent trends in dependency parsing

Basic Features



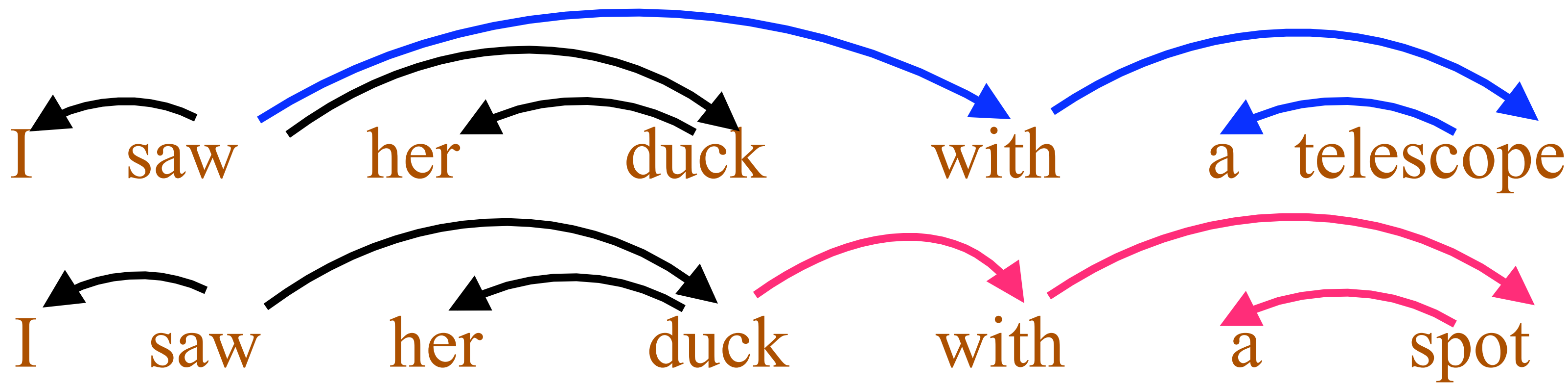
- Uni-gram features
- Bi-gram features
- In between POS features
- Surrounding word POS features

Saw_VBD, saw, VBD duck_NN, duck, NN
saw_VBD_duck_NN, VBD_duck_NN, saw_duck_NN, saw_VBD_NN, saw_VBD_duck, Saw_duck, VBD_NN
VBD_PRP$_NN
VBD_PRP$_PRP$_NN, PRP_VBD_PRP$_NN, VBD_PRP$_NN_IN, PRP_VBD_NN_IN

Non-local Features

- Also known as **dynamic features**
- Take into account the link labels of the surrounding word-pairs when predicting the label of current pair
  - Commonly used in sequential labeling (McCallum et al. 00, Toutanova et al. 03)
- A simple but useful idea for improving parsing accuracy
  - Wang et al. 05
  - McDonald and Pereira 06

Non-local Features



- A word's children are generated first, before it modifies another word
  - Define a canonical order

- "with telescope/with spot" are the dynamic features for deciding whether generating a link between "saw & with" or "duck & with"

Features from Other Resources

- Cluster-based features (Wang et al. 05, Koo et al. 08)
- Subtrees from auto-parsed data (W. Chen et al. 09)
- Alignment features from bilingual data (Huang et al. 09)

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
  - Dependency parsing model
  - Parsing algorithms
  - Features
  - **Learning approaches**
- Part C: transition-based models
- Part D: the combined models
- Part E: other recent trends in dependency parsing

Learning Approaches for Dependency Parsing

- Local learning approaches
  - Learn a local link classifier given a set of features defined on the **local training examples**

> Word pairs along with corresponding features extracted from the training data

- Global learning approaches
- Unsupervised/Semi-supervised learning approaches
  - Use both annotated training data and un-annotated raw text

Local Training Examples

L: left link
R: right link
N: no link

- Given training data {(X, Y)}

local examples

Word-pair	Link-label	Instance_weight	Features
The-boy	L	1	W1_The, W2_boy, W1W2_The_boy, T1_DT, T2_NN, T1T2_DT_NN, Dist_1, …
boy-skipped	L	1	W1_boy, W2_skipped, …
skipped-school	R	1	W1_skipped, W2_ school, …
skipped-regularly	R	1	W1_skipped, W2_regularly, …
The-skipped	N	1	W1_The, W2_skipped, …
The-school	N	1	W1_The, W2_school, …
…			

Local Training Methods

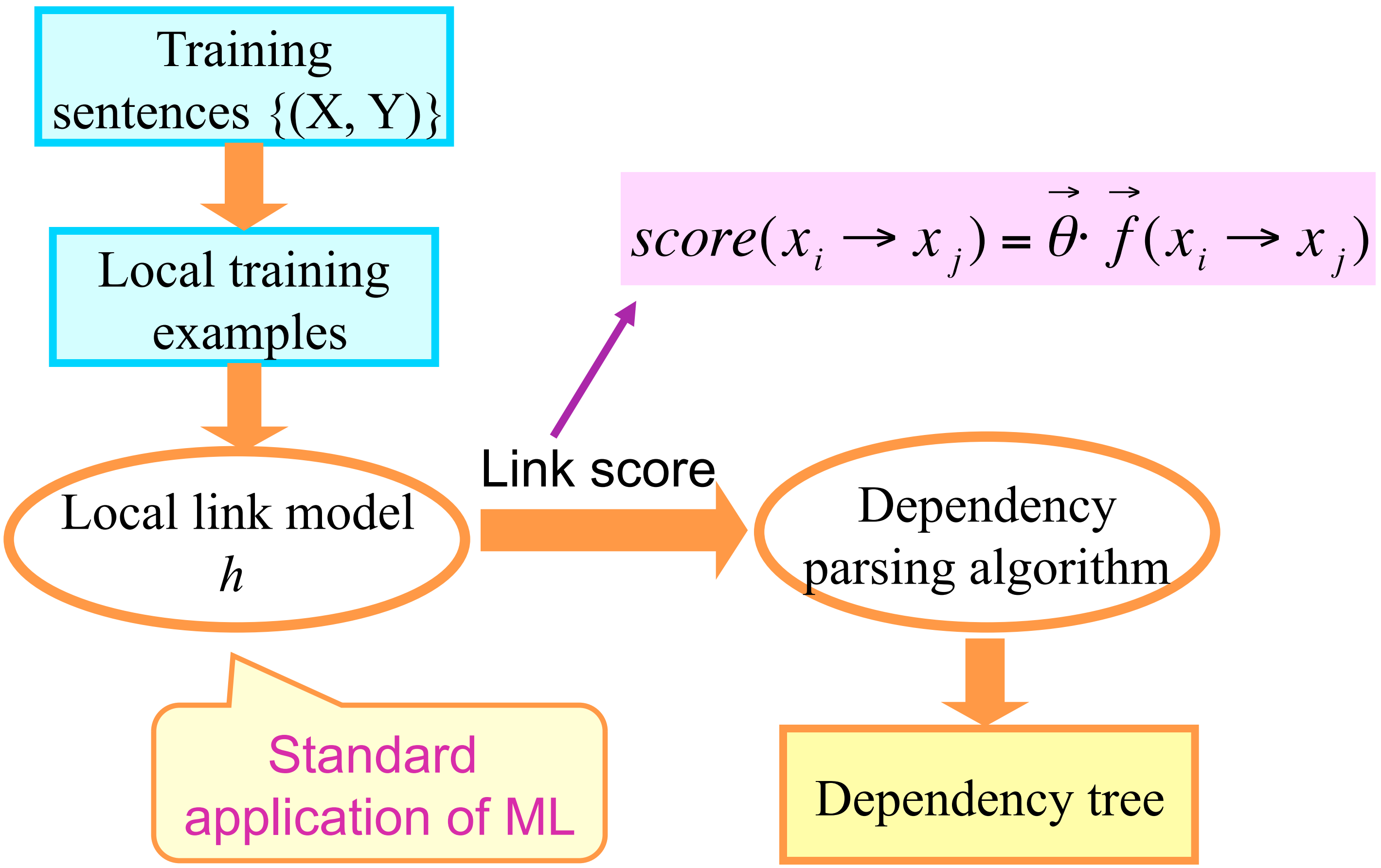
- Learn a local link classifier given a set of features defined on the local examples

- For each word pair in a sentence
  - No link, left link or right link ?
  - 3-class classification

- Efficient *O(n)* local training

- Any classifier can be used as a link classifier for parsing

Combine Local Training with a Parsing Algorithm



Parsing With a Local Link Classifier

- Learn the weight vector $\vec{\theta}$ over a set of features defined on the local examples
- Generative approaches
  - Maximum entropy models (Ratnaparkhi 99, Charniak 00)
- Discriminative approaches
  - Support vector machines (Yamada & Matsumoto 03)
  - Use a richer feature set!

- Each link is scored separately, instead of being computed in coordination with other links in a sentence

Global Training for Parsing

- Directly capture the relations between the links of an output tree

- Incorporate the effects of the parser directly into the training algorithm
  - Structured SVMs (Tsochantaridis et al. 04)
  - Max-Margin Parsing (Taskar et al. 04)
  - Improved large-margin training (Wang et al. 06)
  - Online large-margin training (McDonald et al. 05a)

Standard Large Margin Training

$$\min_{\theta} \frac{\beta}{2}\theta^T\theta + \sum_i \xi_i \qquad subject\ to$$

$$\xi_{i,Y} \geq L(Y_i, Y) - (score(X_i, Y_i) - score(X_i, Y))$$

$$for\ all\ i, Y \in \Phi(X_i)$$

Exponential constraints!

- Having been used for parsing
  - Tsochantaridis et al. 04, Taskar et al.04
- State of the art performance in dependency parsing
  - McDonald et at. 05a

Online Large-Margin Training (McDonald et al. 05a)

- For each training instance (X_i, Y_i)
  - Find current k best trees:
  - Create constraints using these k *best* ...
  - Small number of constraints for each QP

$$\theta = \arg\min_{\theta^*} \|\theta^* - \theta\|$$
$$s.t.\ score(X_i, Y_i) - score(X_i, Y) \geq L(Y_i, Y)$$
$$\forall\ Y \in k-best-trees(X_i)$$

Add *O(klogk)* (Huang& Chiang 05)

MIRA (crammer & Singer 03

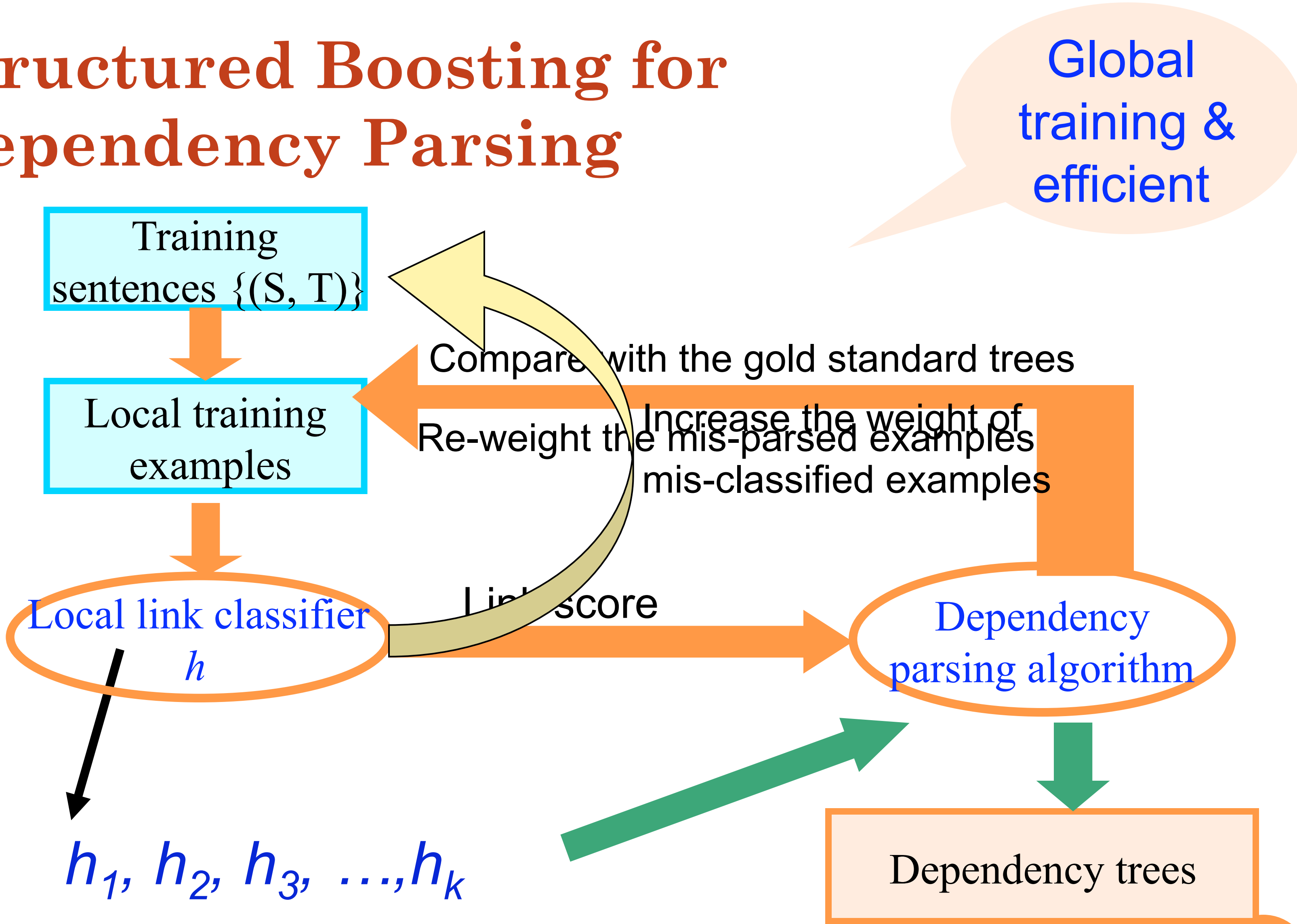
Only *k* constraints for each QP

Structured Boosting (Wang et al. 07)

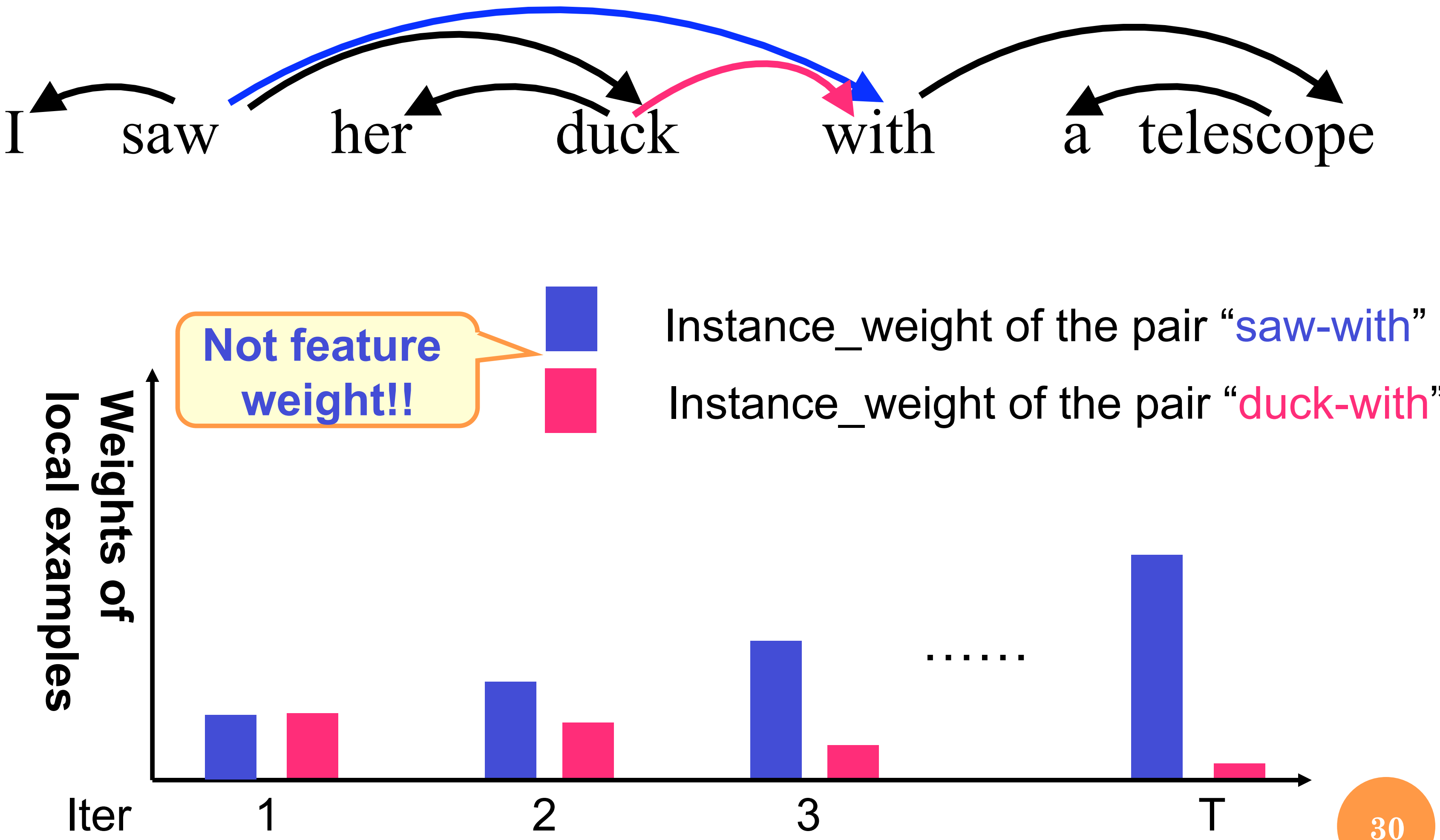
- A simple approach to training structured classifiers by applying a boosting-like procedure to standard supervised training methods
  - A simple variant of standard boosting algorithms Adaboost M1 (Freund & Schapire 97)

- Advantages
  - Global optimization
  - Simple, as efficient as local methods
  - General, can use any local classifier
  - Besides dependency parsing, it can be easily applied to other tasks

Structured Boosting for Dependency Parsing



Structured Boosting (An Example)



From Supervised to Semi/unsupervised learning

- The Penn Treebank
  - 4.5 million words
  - About 200 thousand sentences
  - A
    p
    s

Limited & Human-labor expensive!

- Raw text data
  - News wire
  - Wikipedia
  - Web resources
  -

Plentiful & Free!

Semi/unsupervised learning

Unsupervised/Semi-supervised learning approaches

- Self-training
  - Not very effective
  - Until recently (McClosky et al. 06a, McClosky et al. 06b)
- Generative models (EM)
  - Local optima
  - The disconnection between likelihood and accuracy
  - Same mistakes can be amplified at next iteration
- Semi-supervised Structured SVM (S^3VM)
  - Global optimum
  - Incorporate the effects of the parser directly into the training algorithm

Semi-supervised Structured SVM (S^3VM)

- The objective of the standard S^3VM is a combination of
  - Structured loss on labeled data (convex)
  - Structured loss on un-labeled data (non-convex)
- Convex + non-convex is non-convex
  - Local optima
- Complex and expensive to solve
  - Too complicated to apply it to parsing

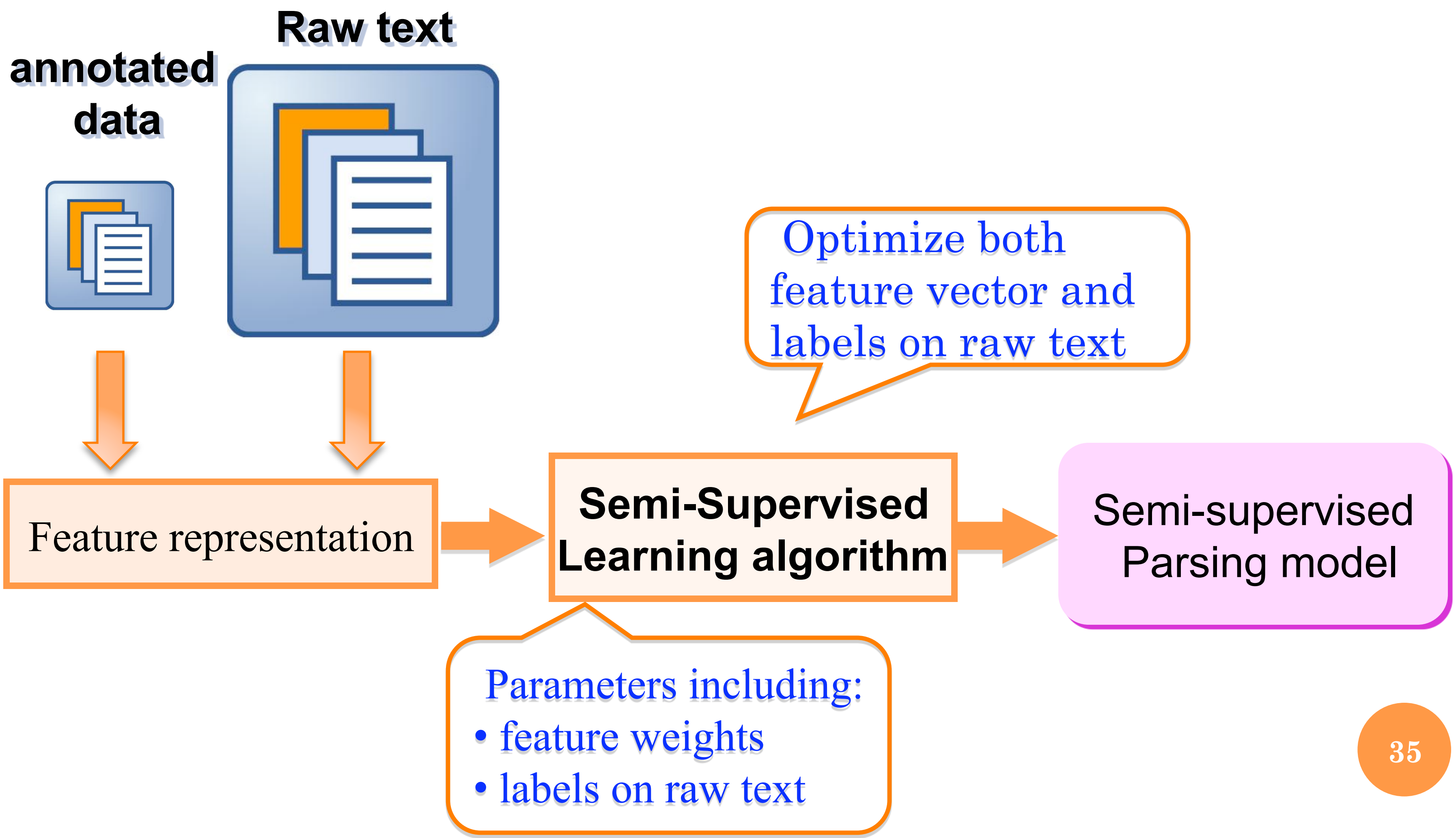
Semi-supervised Convex Training Dependency Parsing (Wang et al. 08)

- The objective is a combination of
  - Structured loss on labeled data (convex)
  - Least square loss on un-labeled data (convex)

convex + convex is convex

- Using a **stochastic gradient descent** approach
  - Parameters are updated locally on each sentence
  - Converge after a few iterations
- This convex training approach:
  - Focused on semi-supervised learning instead of feature engineering
  - Used only basic features due to the complexity issue

Semi-supervised Convex Training Dependency Parsing (Wang et al. 08)

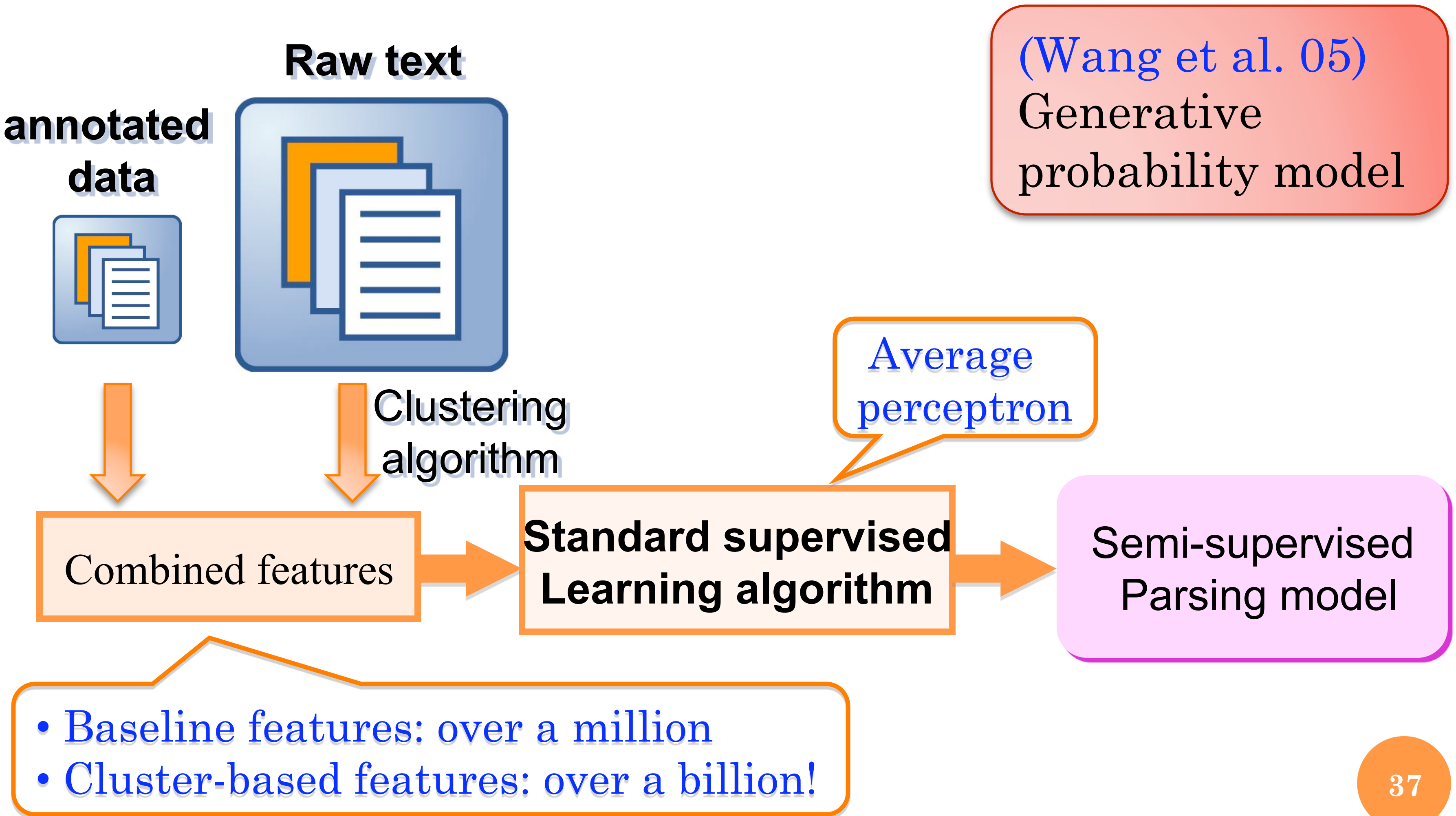


Simple Semi-supervised Dependency Parsing (Koo et al. 08)

- Extract features from unlabeled data
  - Instead of solving the complex S^3VM, add features derived from a large unannotated corpus

- Combining word clusters with discriminative learning (Miller et al. 04)
  - Incorporate word clusters derived from a large unannotated corpus via unsupervised learning
  - Using both the baseline and cluster-based features
  - Average perceptron learning algorithm (fast)
  - Achieve substantial improvement on dependency parsing over competitive baseline

Simple Semi-supervised Dependency Parsing (Koo et al. 08)



Summary – Graph-based Models

- Dependency parsing model
- Dependency parsing algorithms
- Features
- Learning algorithms

References

- E. Charniak. 2000. A maximum entropy inspired parser. In *Proc. NAACL.*
- J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. COLING*.
- Y. Freund and R. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Computer and System Sciences.*
- T. Koo, X. Carreras and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. ACL-HLT*.
- A. McCallum, D. Freitag and F. Pereira. 2000. Maximum Entropy Markov Methods for Information Extraction and Segmentation. In *Proc. ICML*.
- D. McClosky, E. Charniak and M. Johnson. 2006. Effective Self-Training for Parsing. In *Proc. HLT-NAACL*.
- D. McClosky, E. Charniak and M. Johnson. 2006. Reranking and Self-Training for Parser Adaptation. In *Proc. COLING-ACL*.
- R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. ACL*.

- R. McDonald and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *Proc. EACL.*
- S. Miller, J. Guinness and A. Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Train- ing. In *Proc. HLT-NAACL.*
- A. Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning.*
- B. Taskar, D. Klein, M. Collins, D. Koller and C. Manning. 2004. Max-margin parsing. In *Proc. EMNLP.*
- K. Toutanova, D. Klein, C. Manning and Y. Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proc. HLT-NAACL.*
- I. Tsochantaridis, T. Hofmann, T. Joachims and Y. Altun. 2004. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *Proc. ICML.*
- Q. Wang, C. Cherry, D. Lizotte and D. Schuurmans. 2006. Improved Large Margin Dependency Parsing via Local Constraints and Laplacian Regularization. In Proc. CoNLL.
- Q. Wang, D. Schuurmans and D. Lin. Strictly Lexical Dependency Parsing. 2005. In *Proc. IWPT.*
- Q. Wang, D. Lin and D. Schuurmans. 2007. Simple Training of Dependency Parsers via Structured Boosting. In *Proc. IJCAI.*

- Q. Wang, D. Lin and D. Schuurmans. Semi-supervised Convex Training for Dependency Parsing . In *Proc. ACL*.
- H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. IWPT*.

Recent Advances in Dependency Parsing

Qin Iris Wang

AT&T Interactive

qiniriswang@gmail.com

Yue Zhang

Cambridge University

frcchang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Part C: Transition-based Dependency Parsing Models

Yue Zhang

Cambridge University

frcchang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
  - **Transition-based parsing processes**
  - Decoding algorithms
  - Learning algorithms and feature templates
- Part D: the integrated models
- Part E: other recent trends in dependency parsing

Overview

- Graph-based parsers
  - Enumerate all possible graphs
  - Score each candidate according to graph-based features
  - Choose the highest scored one
- Transition-based parsers
  - Build a candidate output using a stack and a set of actions
  - The stack used to hold partially-built parses
  - The input tokens are put into a queue

A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse

I like playing table-tennis with her .

A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse

I	like playing table-tennis with her .

A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse

A transition-based parsing process

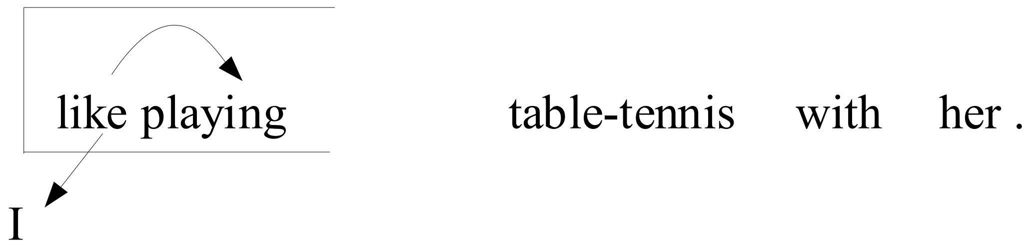
- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse

like playing table-tennis with her .

I

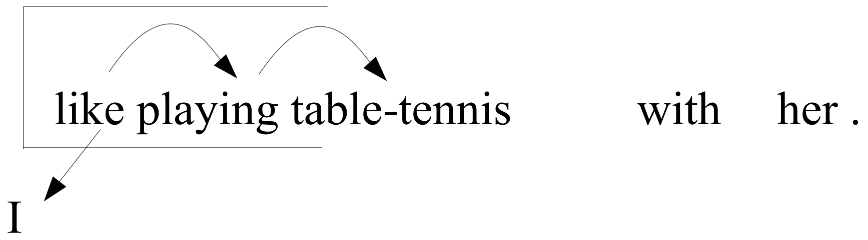
A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse



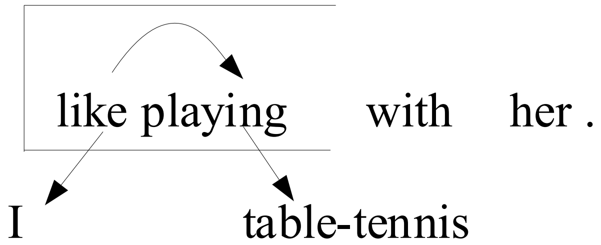
A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse



A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse



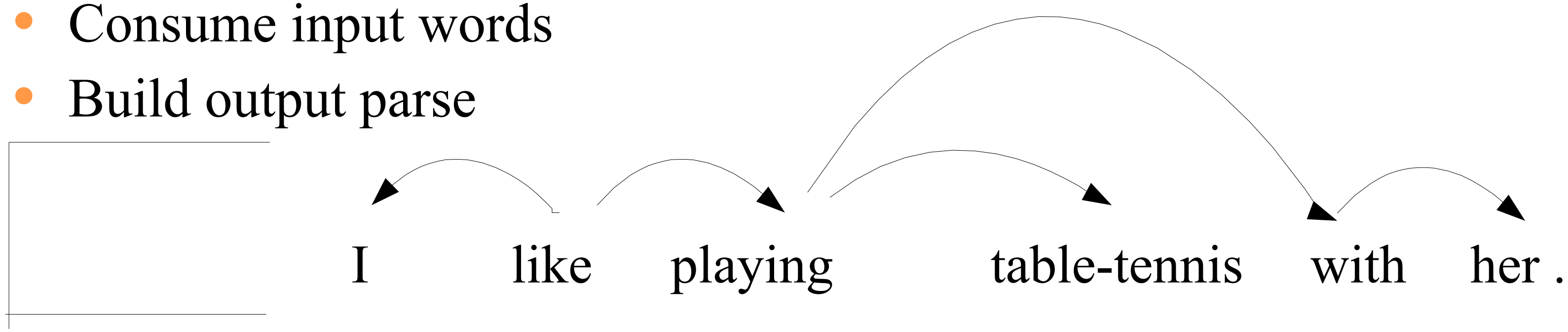
A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse

...

A transition-based parsing process

- Stack holds partially built parses
- Queue contains unprocessed words
- Transition-actions
  - Consume input words
  - Build output parse

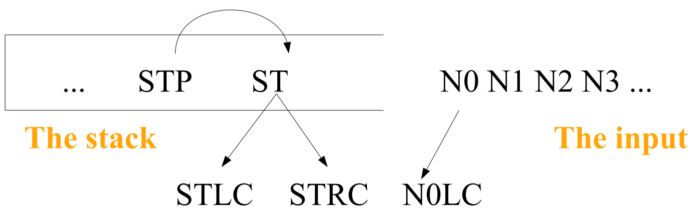


The arc-eager parser

- Arc-eager parser
  - A stack to hold partial candidates
  - A queue of next incoming words
  - Four transition-actions
    - SHIFT, REDUCE, ARC-LEFT, ARC-RIGHT
  - Examples
    - MaltParser (Nivre et al., 2006)
    - Johansson and Nugues (2007)
    - Zhang and Clark (2008)

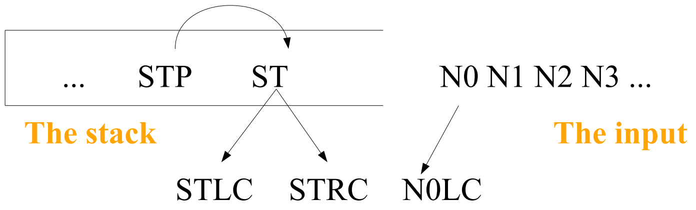
The arc-eager parser

- The context



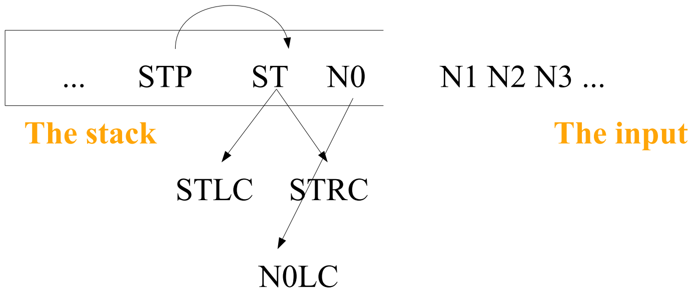
The arc-eager parser

- Transition actions
  - Shift



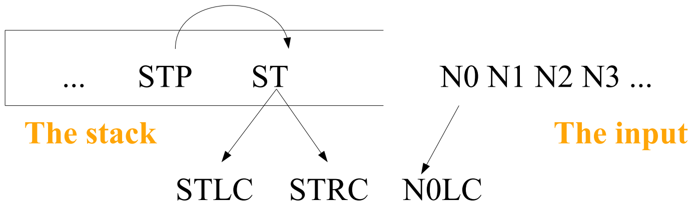
The arc-eager parser

- Transition actions
  - Shift
    - Pushes stack



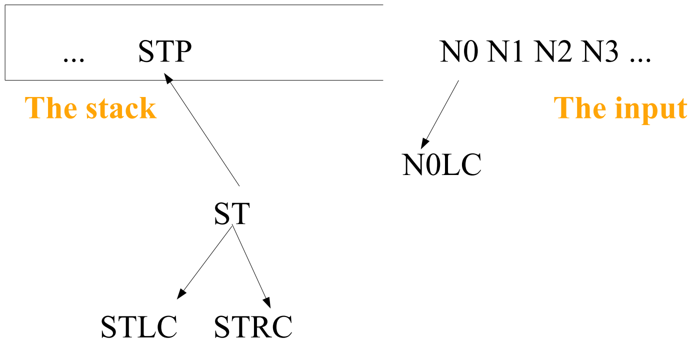
The arc-eager parser

- Transition actions
  - Reduce



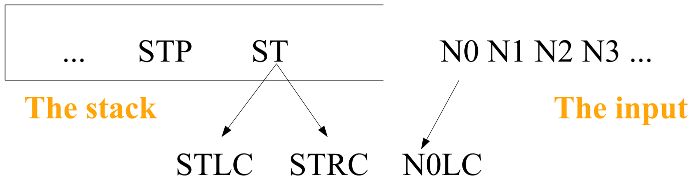
The arc-eager parser

- Transition actions
  - Reduce
    - Pops stack



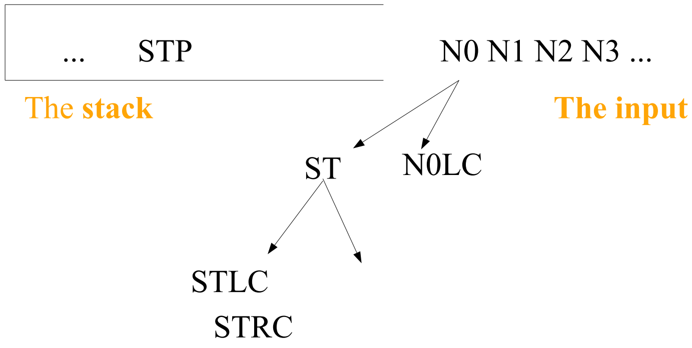
The arc-eager parser

- Transition actions
  - Arc-Left



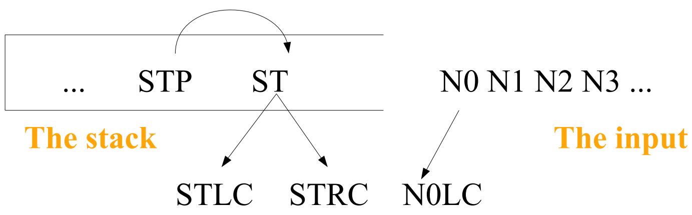
The arc-eager parser

- Transition actions
  - Arc-Left
    - Pops stack
    - Adds link



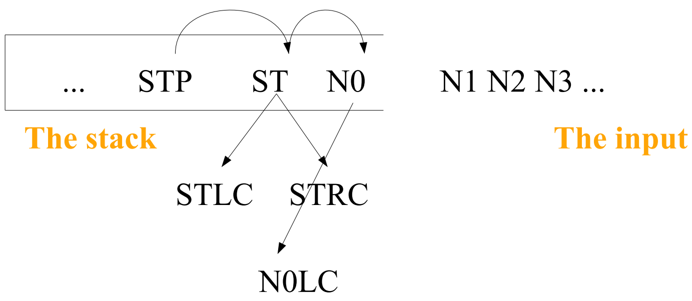
The arc-eager parser

- Transition actions
  - Arc-right



The arc-eager parser

- Transition actions
  - Arc-right
    - Pushes stack
    - Adds link



The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight

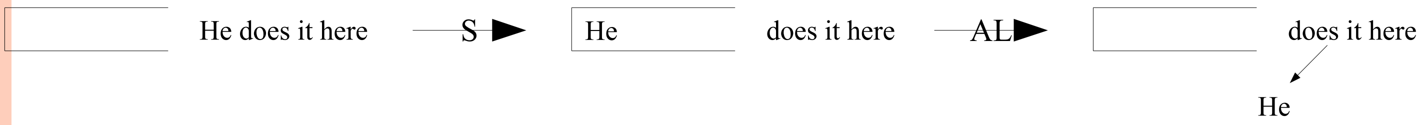
He does it here

The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight

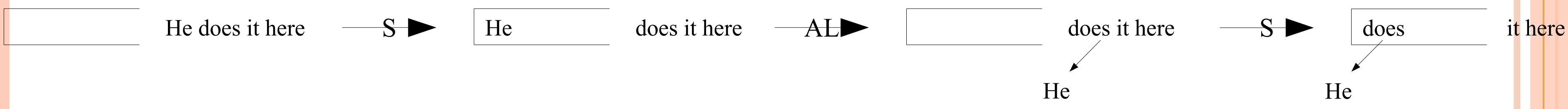
The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight



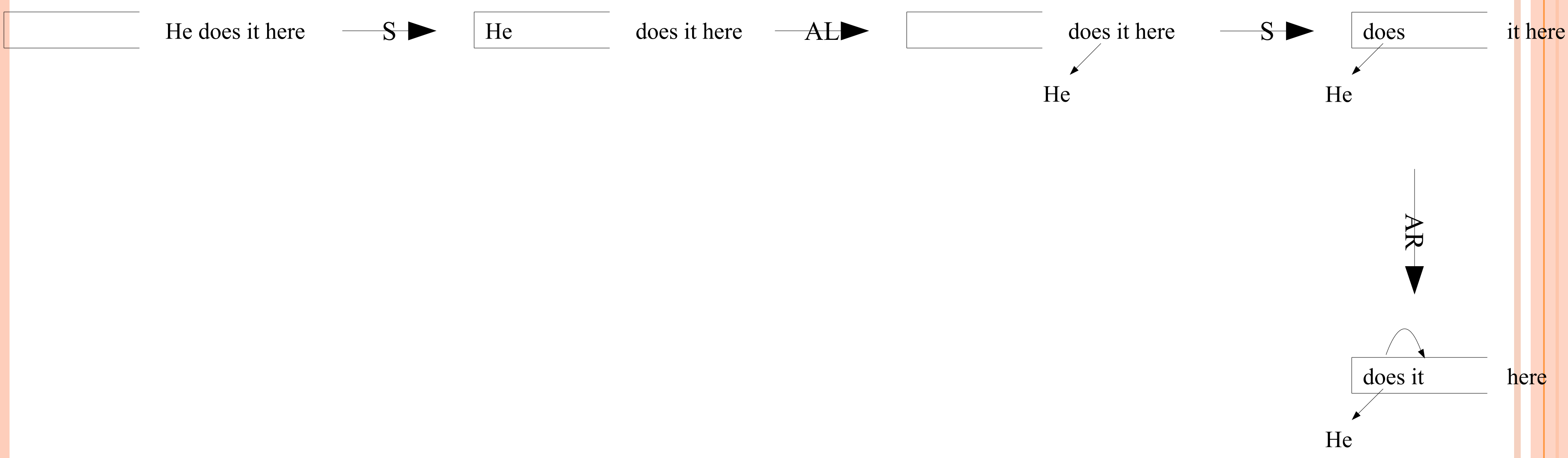
The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight



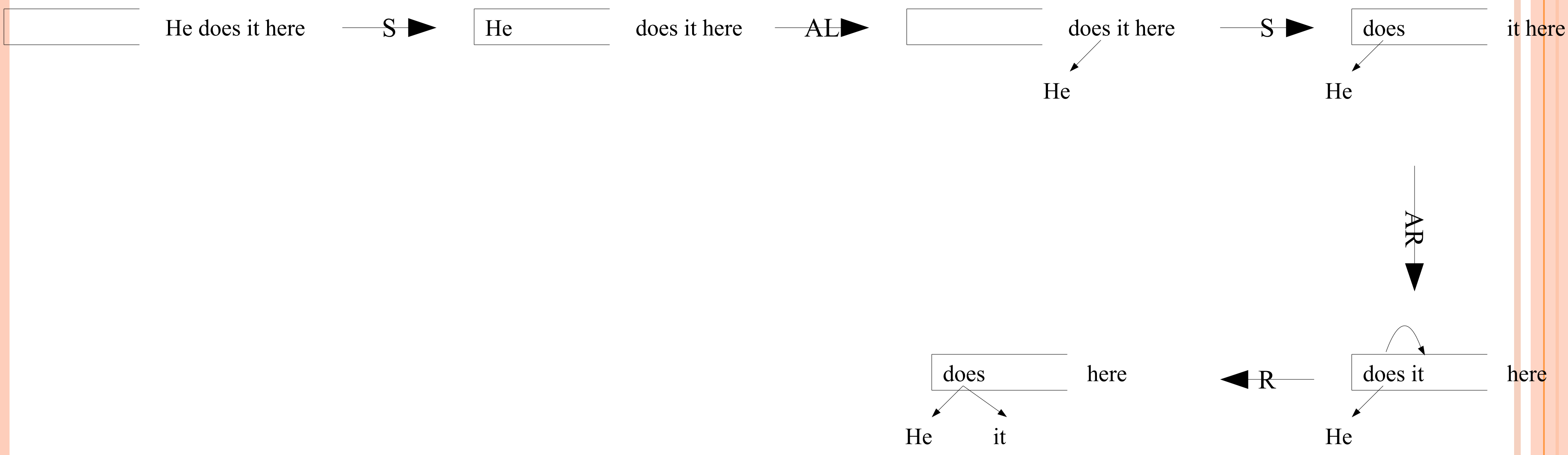
The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight



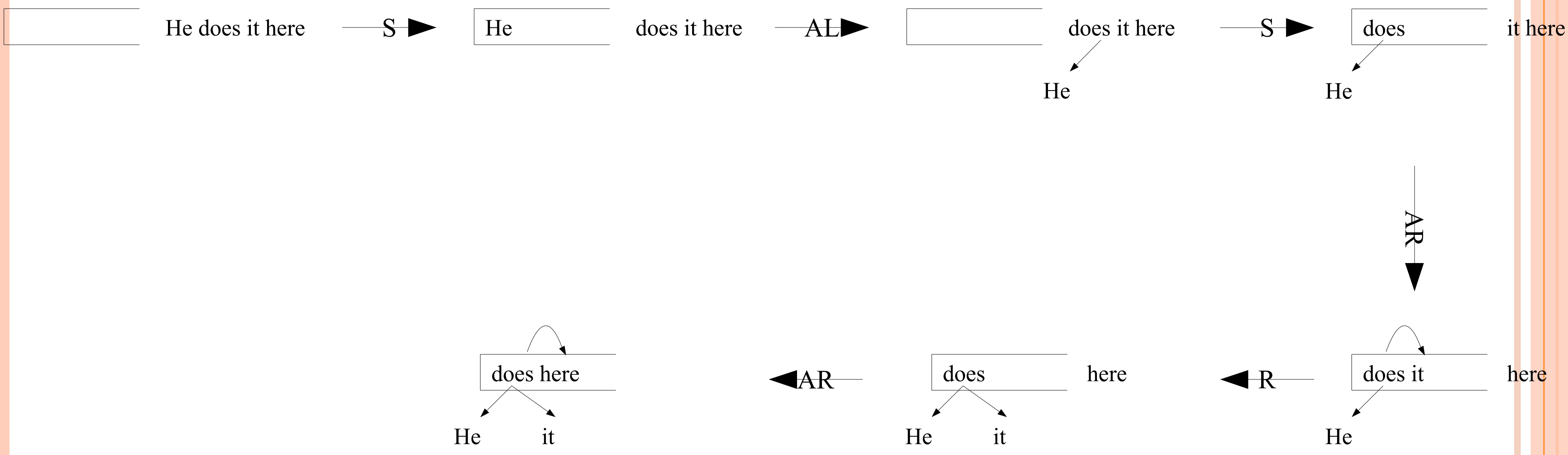
The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight



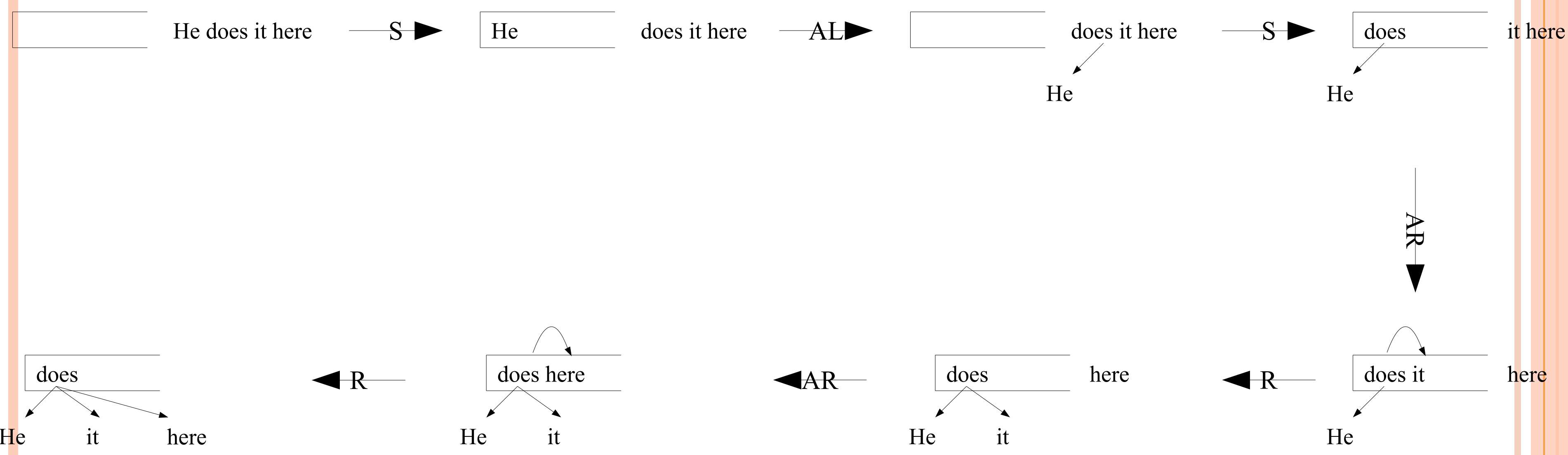
The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight



The arc-eager parser

- An example
  - S – Shift
  - R – Reduce
  - AL – ArcLeft
  - AR – ArcRight



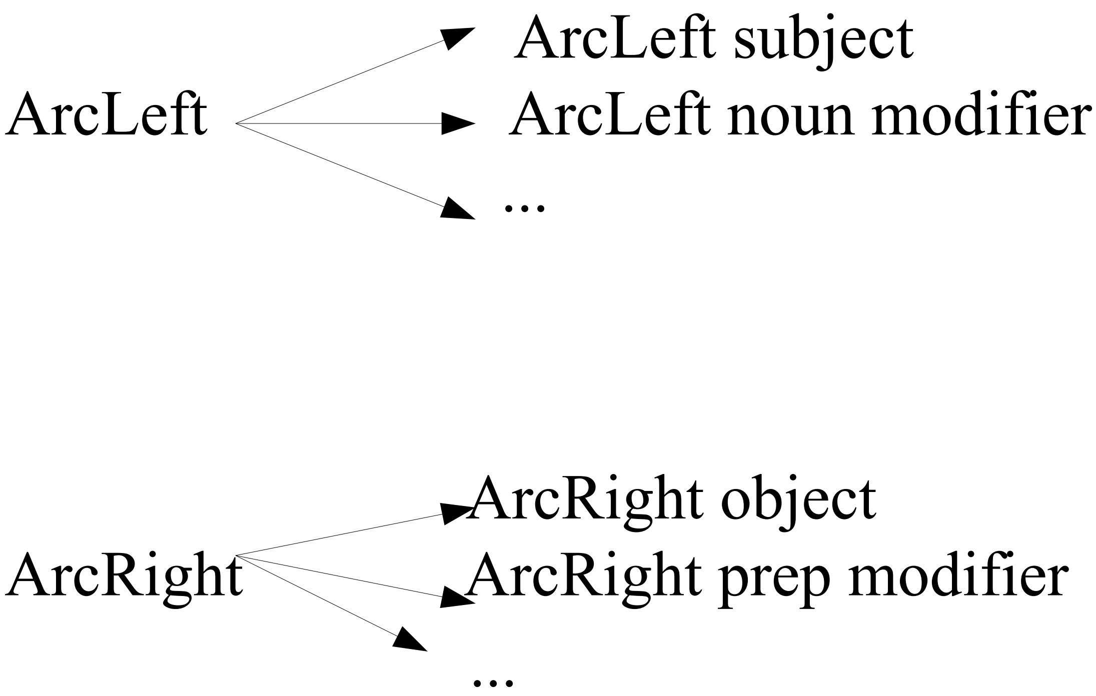
31

The arc-eager parser

- Arc-eager parser
  - Time complexity: linear
    - Every word is pushed once onto the stack
    - Every word except the root is popped once
  - Links are added between ST and N0
    - As soon as they are in place
    - 'eager'

The arc-eager parser

- Arc-eager parser
  - Labeled parsing?

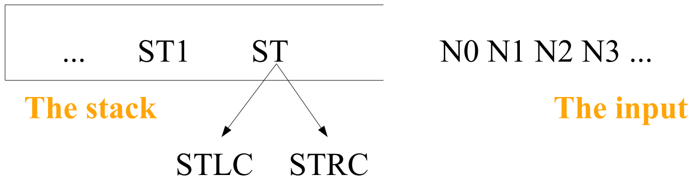


The arc-standard parser

- Arc-standard parser
  - Same as previously
    - A stack to hold partial candidates
    - A queue of next incoming words
  - Different from previously
    - Transition actions: SHIFT LEFT RIGHT
  - Examples
    - Yamada and Matsumoto (2003)
    - Huang et al. (2009)

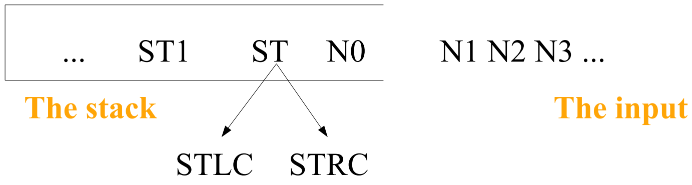
The arc-standard parser

- Transition actions
  - Shift



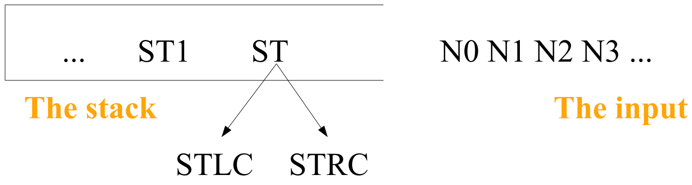
The arc-standard parser

- Transition actions
  - Shift
    - Pushes stack



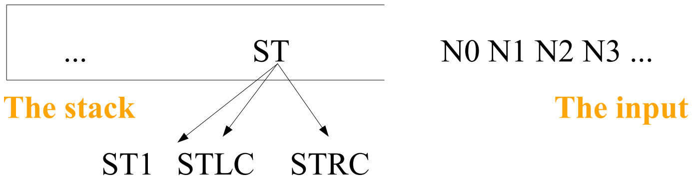
The arc-standard parser

- Transition actions
  - Left



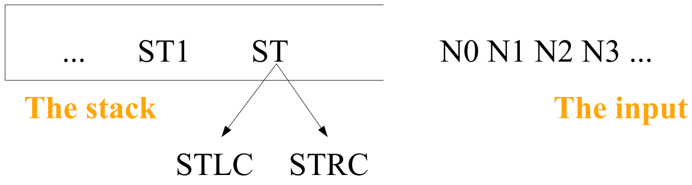
The arc-standard parser

- Transition actions
  - Left
    - Pops stack
    - Adds link



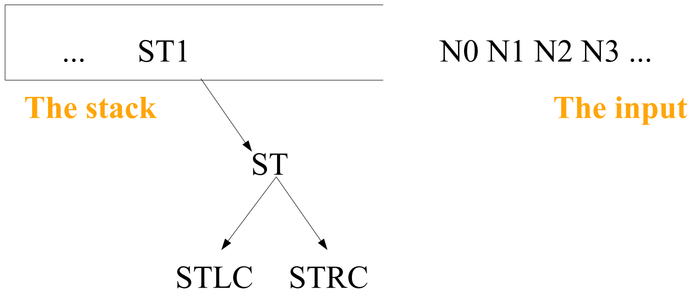
The arc-standard parser

- Transition actions
  - Right



The arc-standard parser

- Transition actions
  - Right
    - Pops stack
    - Adds link



The arc-standard parser

- Arc-standard parser
  - Time complexity: linear
    - Every word is pushed once onto the stack
    - Every word except the root is popped once
  - Links are added between ST and ST1
- Standard or eager?
  - empirical

The arc-standard parser

- Arc-standard parser
  - Similarity to shift-reduce phrase-structure parsing
    - Sagae and Lavie (2005)
    - Wang et al. (2006)
    - Zhang and Clark (2009)

Non-projectivity

- Problem

A meeting was scheduled for this today.

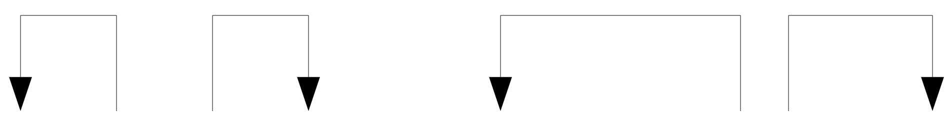
- Neither parsers solves it
  - Word orders are kept
  - Links added between neighbors (on stack)

Non-projectivity

- Problem

A meeting was scheduled for this today.

- One Solution



A meeting for this was scheduled today.

Non-projectivity

- Online reordering (Nivre 2009)
  - Add an extra action to the parser: swap
    - Pops the second word off stack
    - The other transitions are the same

Non-projectivity

- An extra transition action
  - swap

A meeting was scheduled for this today.

Non-projectivity

- An extra transition action
  - swap

A meeting was scheduled for this today.

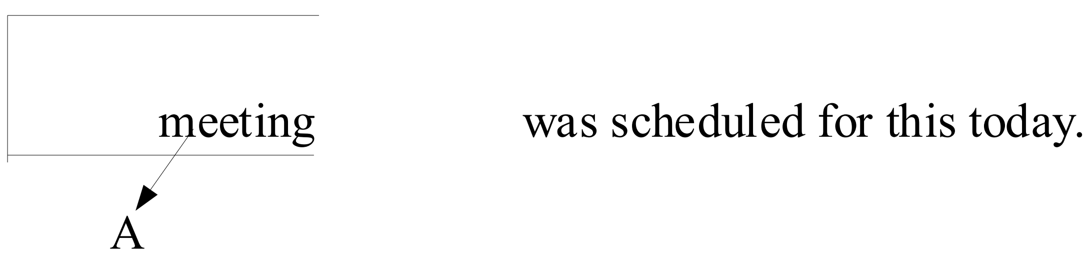
Non-projectivity

- An extra transition action
  - swap

A meeting was scheduled for this today.

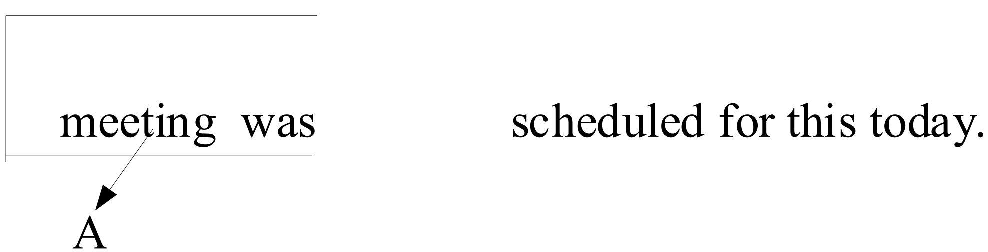
A transition-based parsing process

- An extra transition action
  - swap



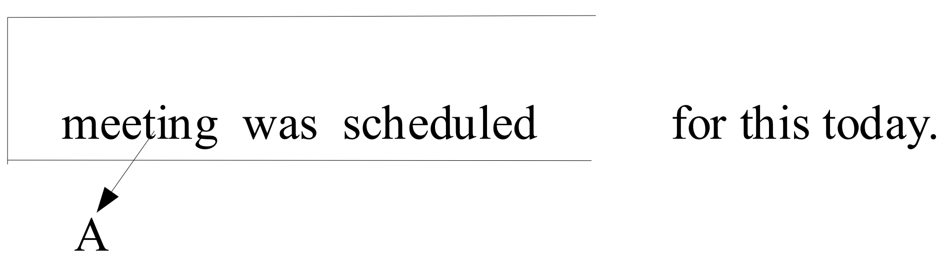
A transition-based parsing process

- An extra transition action
  - swap



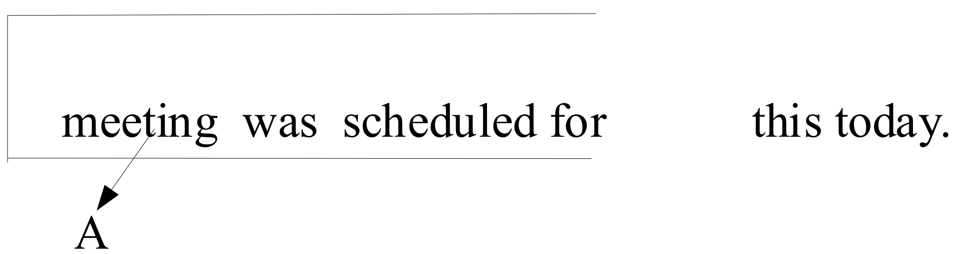
A transition-based parsing process

- An extra transition action
  - swap



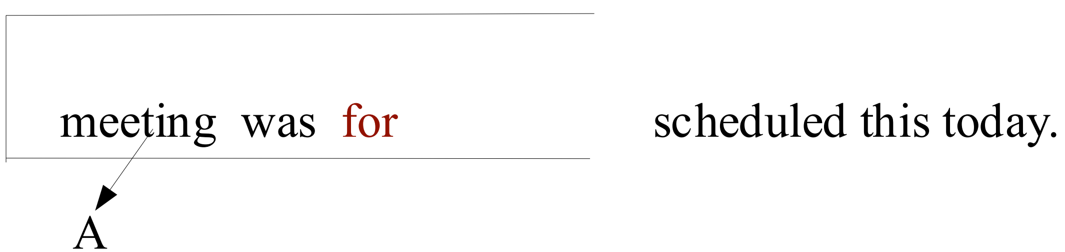
A transition-based parsing process

- An extra transition action
  - swap



A transition-based parsing process

- An extra transition action
  - **swap**



A transition-based parsing process

- An extra transition action
  - **swap**

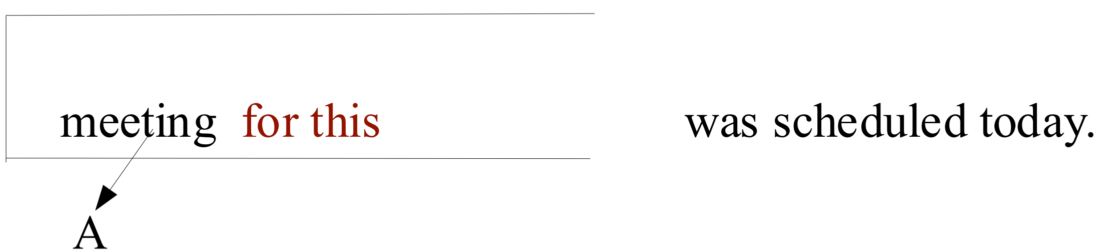
A transition-based parsing process

- An extra transition action
  - **Swap**

…

A transition-based parsing process

- An extra transition action
  - **swap**



Non-projectivity

- Online reordering (Nivre 2009)
  - Add an extra action to the parser: swap
  - Not linear any more
    - Can be N-square
    - Expected linear time

Transition-based parsing processes

- Summary
  - Build the output using
    - A stack
    - A set of transition actions
  - Different types
    - Arc-eager
    - Arc-standard
    - More?

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
  - Transition-based parsing processes
  - **Decoding algorithms**
  - Learning algorithms and feature templates
- Part D: the integrated models
- Part E: other recent trends in dependency parsing

Decoding algorithms

- Goal
  - Search for one sequence of transition-action to build the parse
  - Done by scoring transition action given context
  - Models talked about in the next section
- Comparison with graph-based
  - Search for one graph from candidates

Decoding algorithms

- Candidate item

  <S, G, Q>

Decoding algorithms

- Greedy local search
  - Initialize a start item

    S=empty, G=empty, Q=input sentence
  - Define a final item

    S=[root], G=tree, Q=[]
  - Pick up one transition-action at a time by score

Greedy local search

- Malt parser (Nivre et al., 2006)
  - Arc-eager transitions
    - Pushing actions: SHIFT, ARC-RIGHT
    - Popping actions: REDUCE, ARC-LEFT
    - Links are added with ARC-
  - Start state
    - Stack empty, no word has been processed by now
  - Finish state
    - Stack contains only root, all processed
  - Greedily picks up one transition action after another from start to finish

Score(action)

Greedy local search

- Malt parser

He does it here

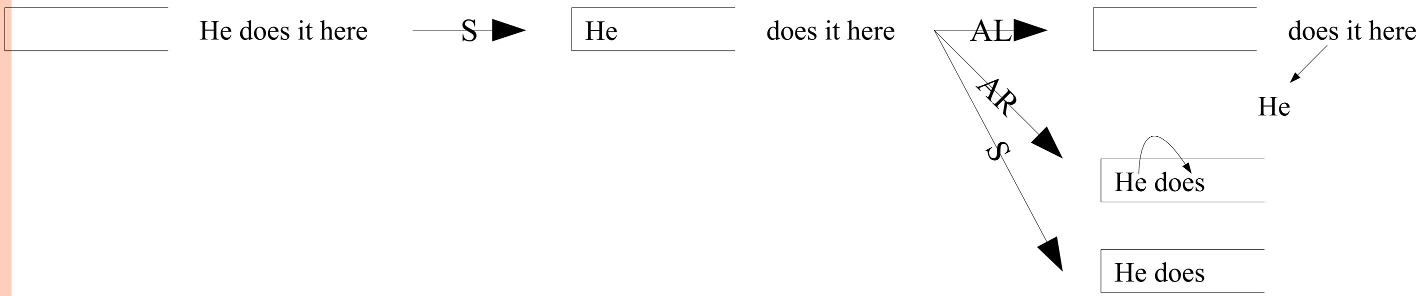
Greedy local search

- Malt parser

He does it here —S→ He | does it here

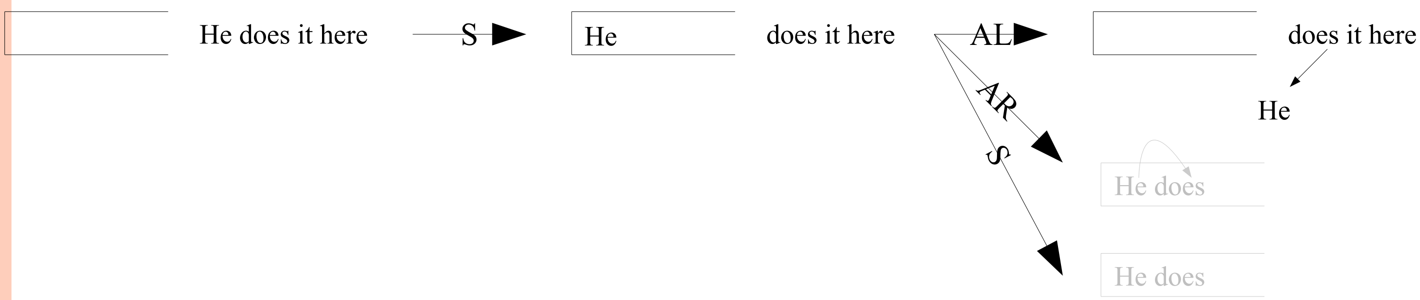
Greedy local search

- Malt parser



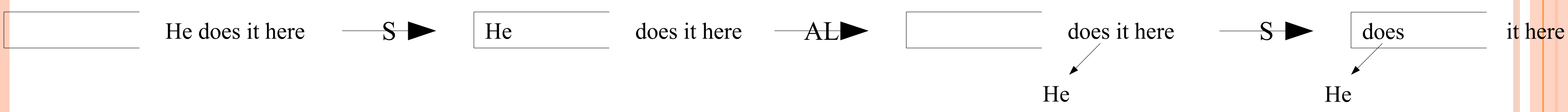
Greedy local search

- Malt parser



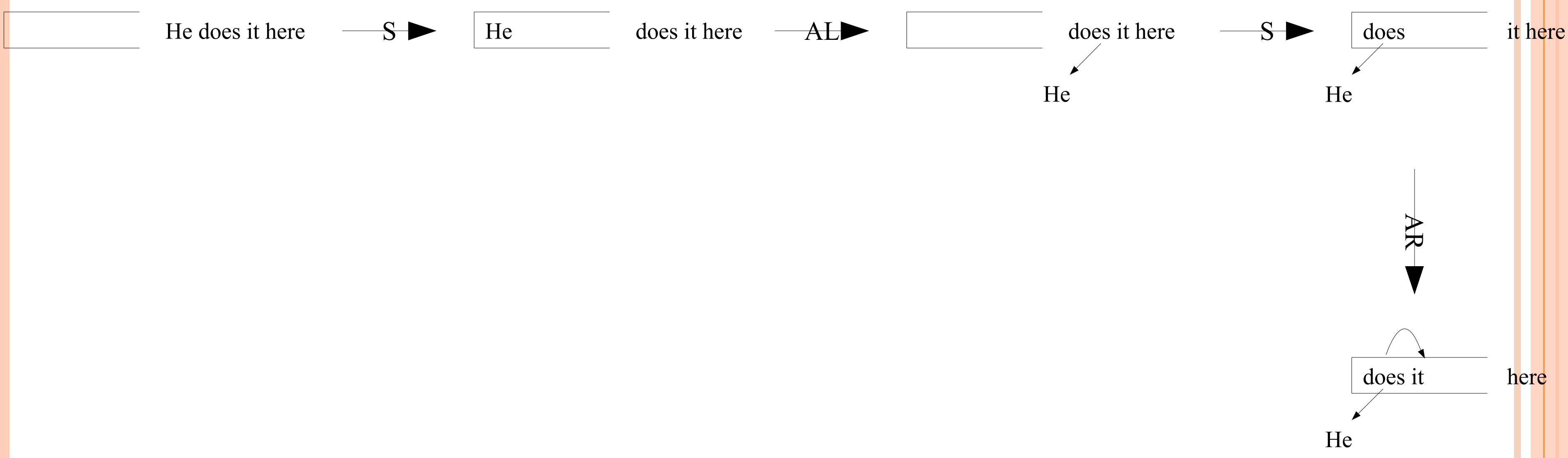
Greedy local search

- Malt parser



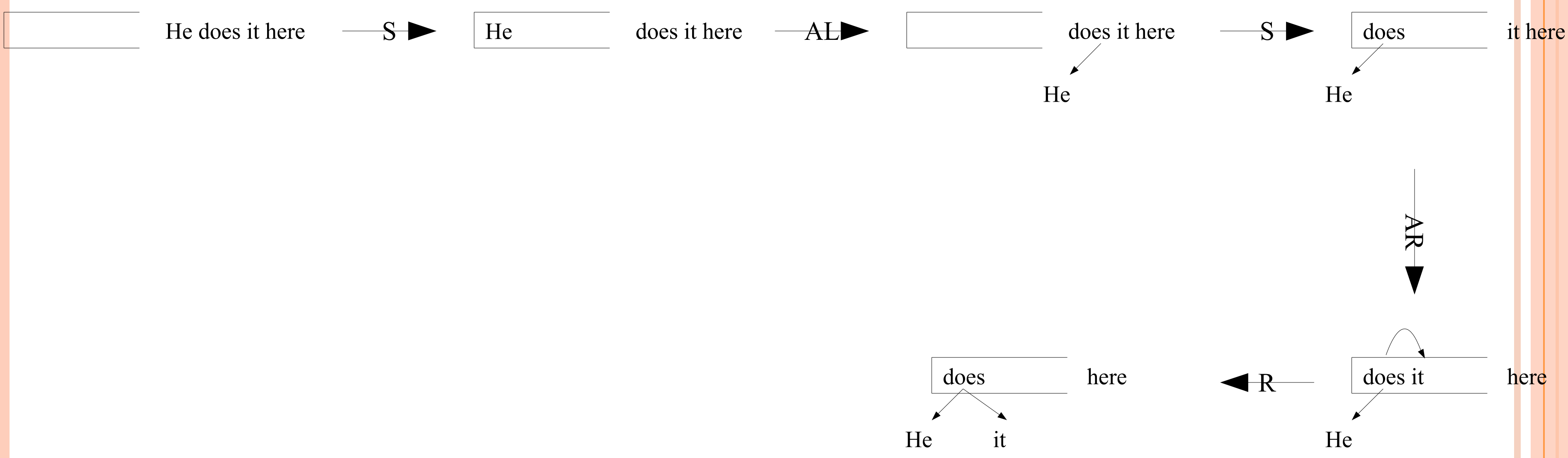
Greedy local search

- Malt parser



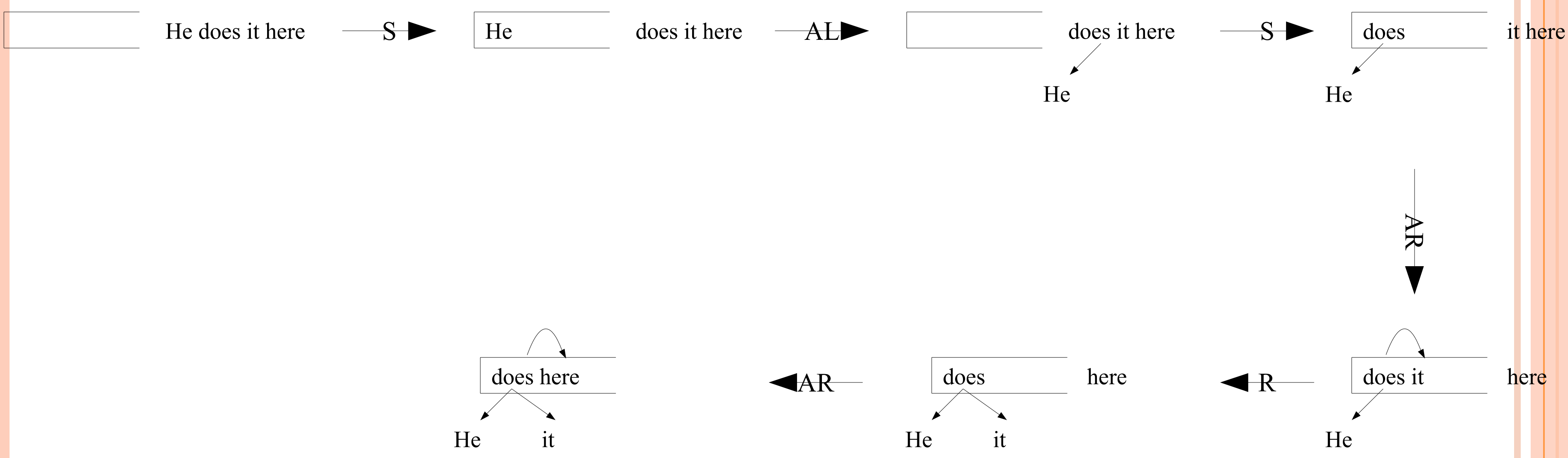
Greedy local search

- Malt parser



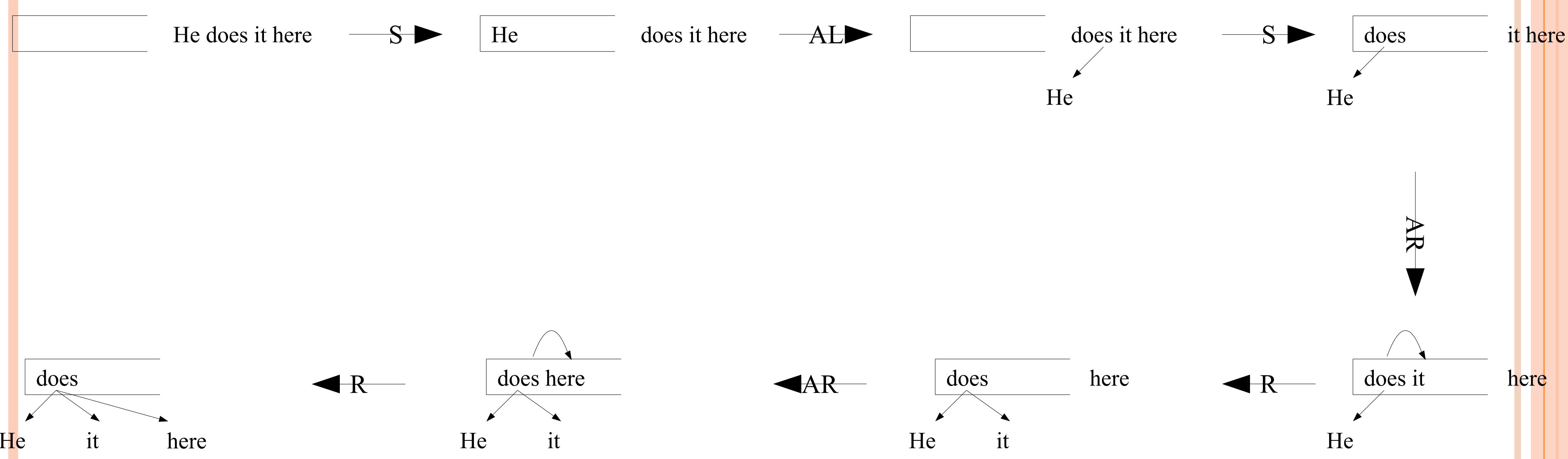
Greedy local search

- Malt parser



Greedy local search

- Malt parser



Decoding algorithms

- Greedy local search
  - Problem:
    one error leads to incorrect parse

Decoding algorithms

- Beam search

  - Keeps N different partial state items in agenda.
  - Use the total score of all actions to rank state items

$$Score(parse) = \sum_{action \in parse} Score(action)$$

  - Avoid error propagations from early decisions

Beam search

- Example work
  - Johansson and Nugues (2007)
  - Zhang and Clark (2008)

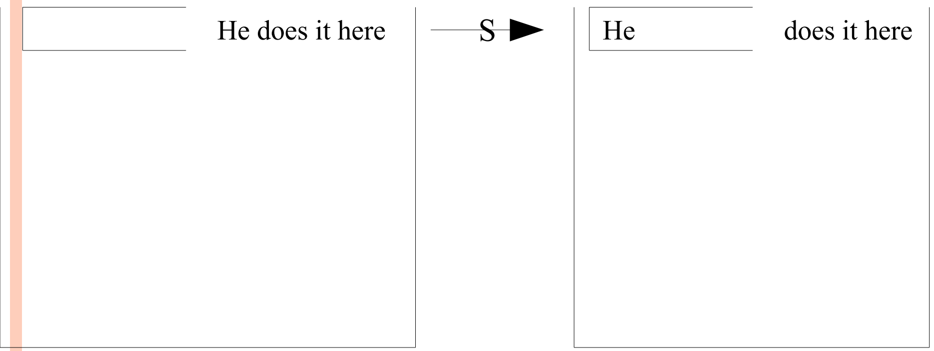
Beam search

- An example

He does it here

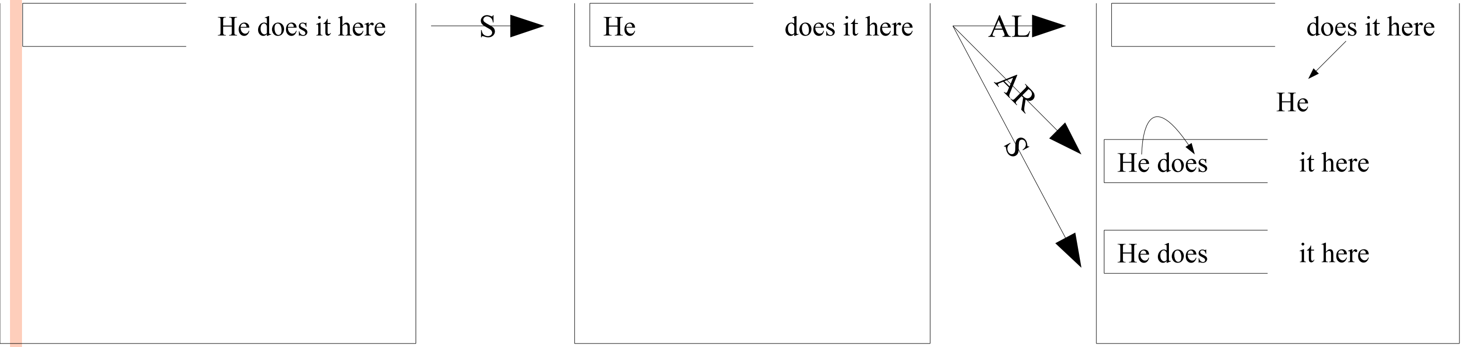
Beam search

- An example



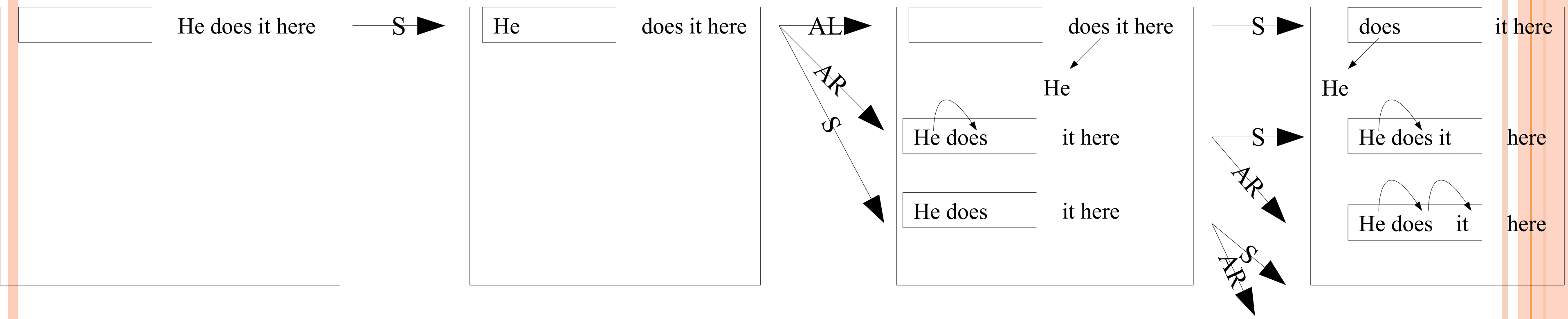
Beam search

- An example



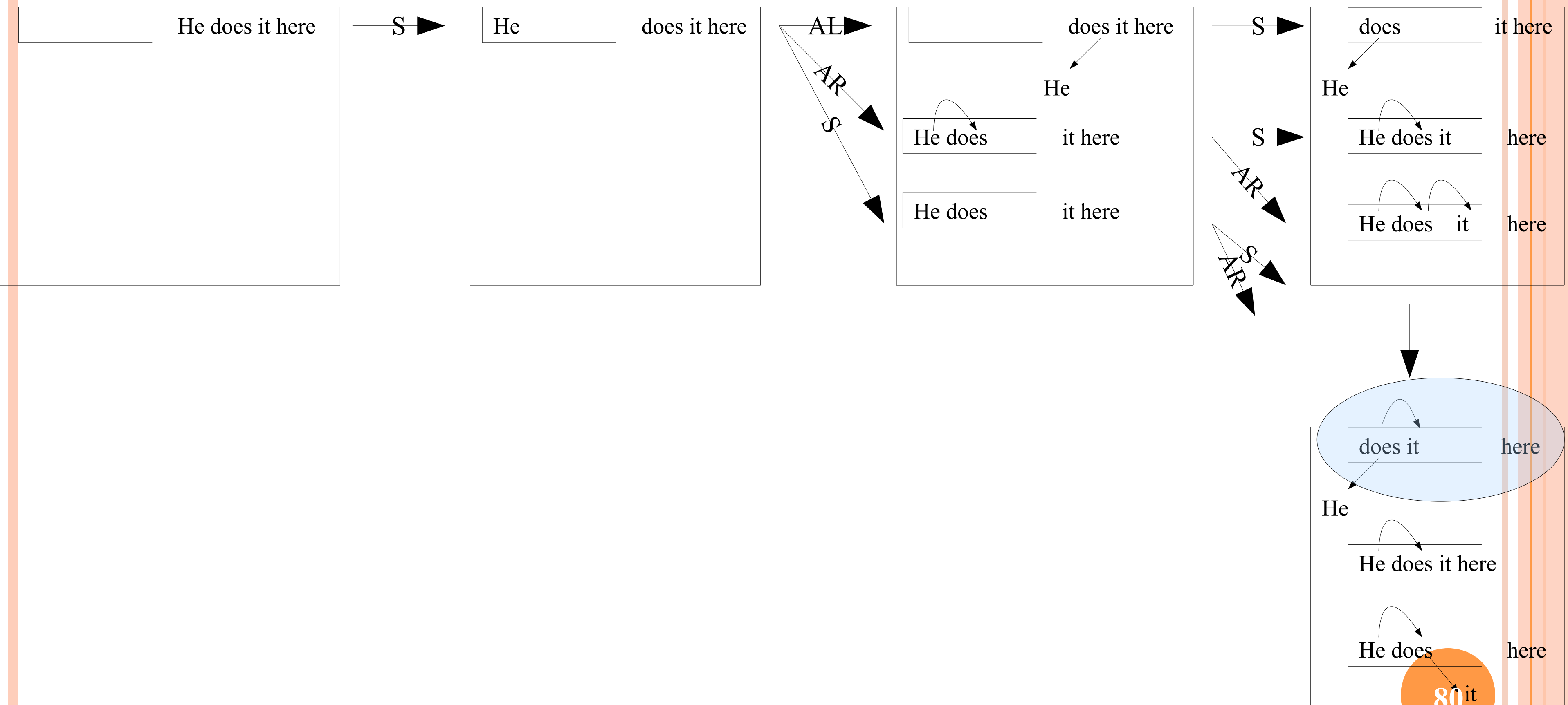
Beam search

- An example



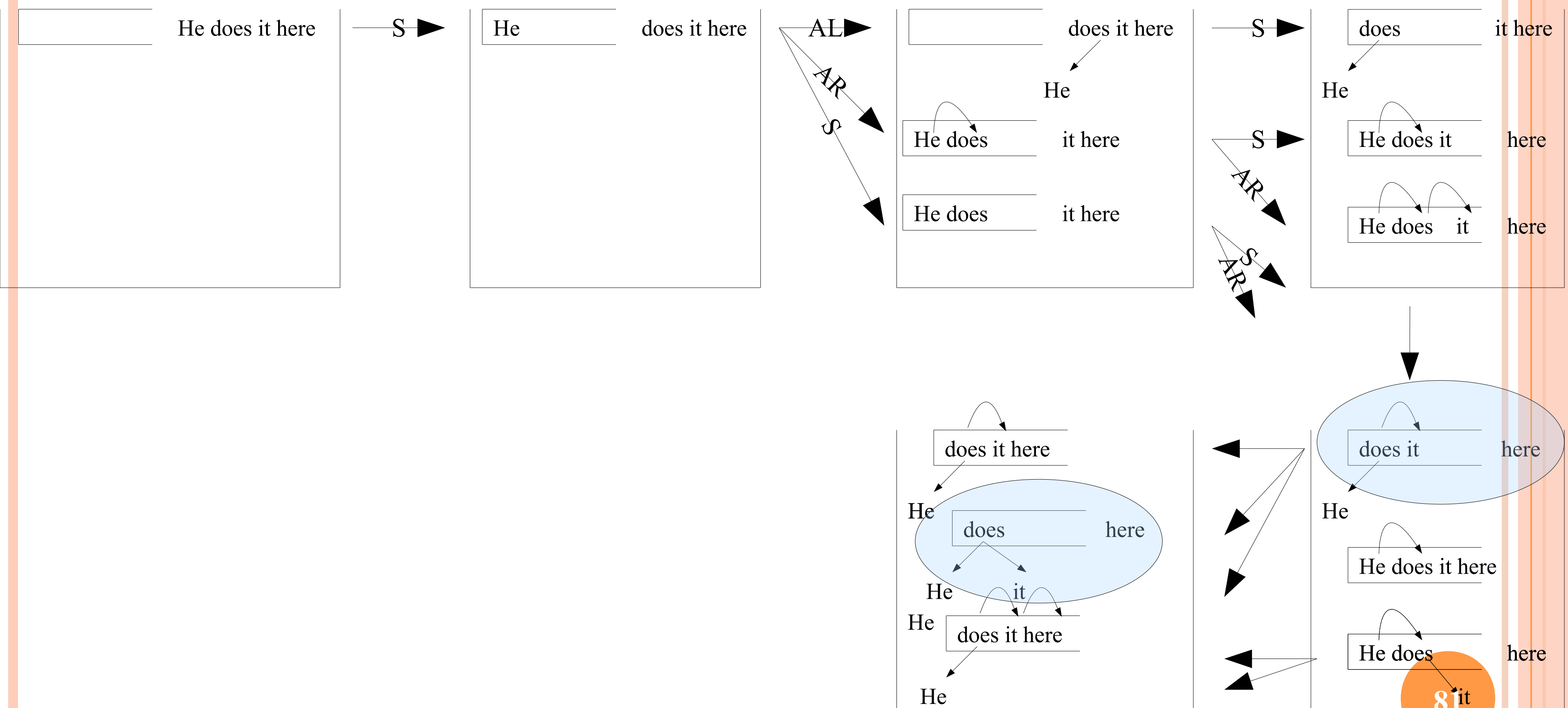
Beam search

- An example



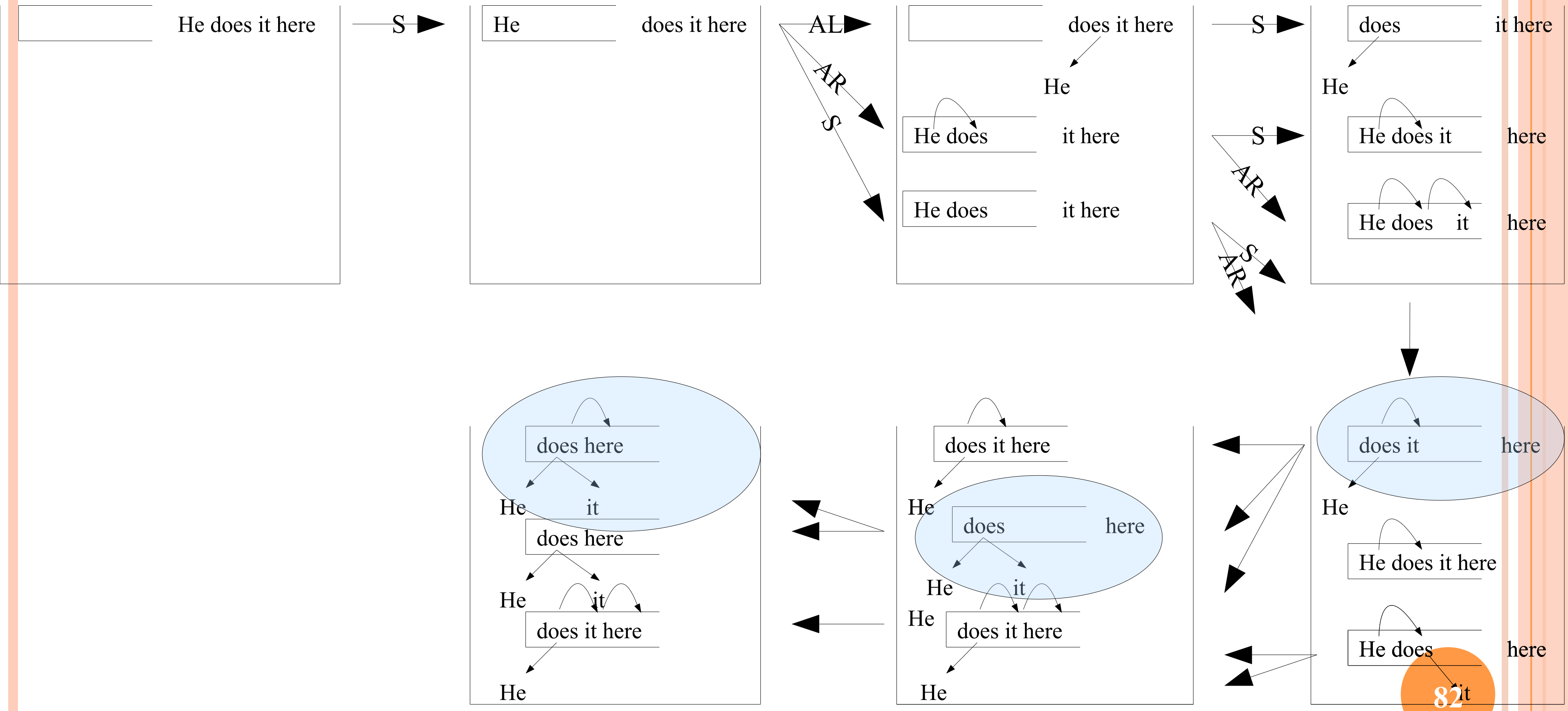
Beam search

- An example



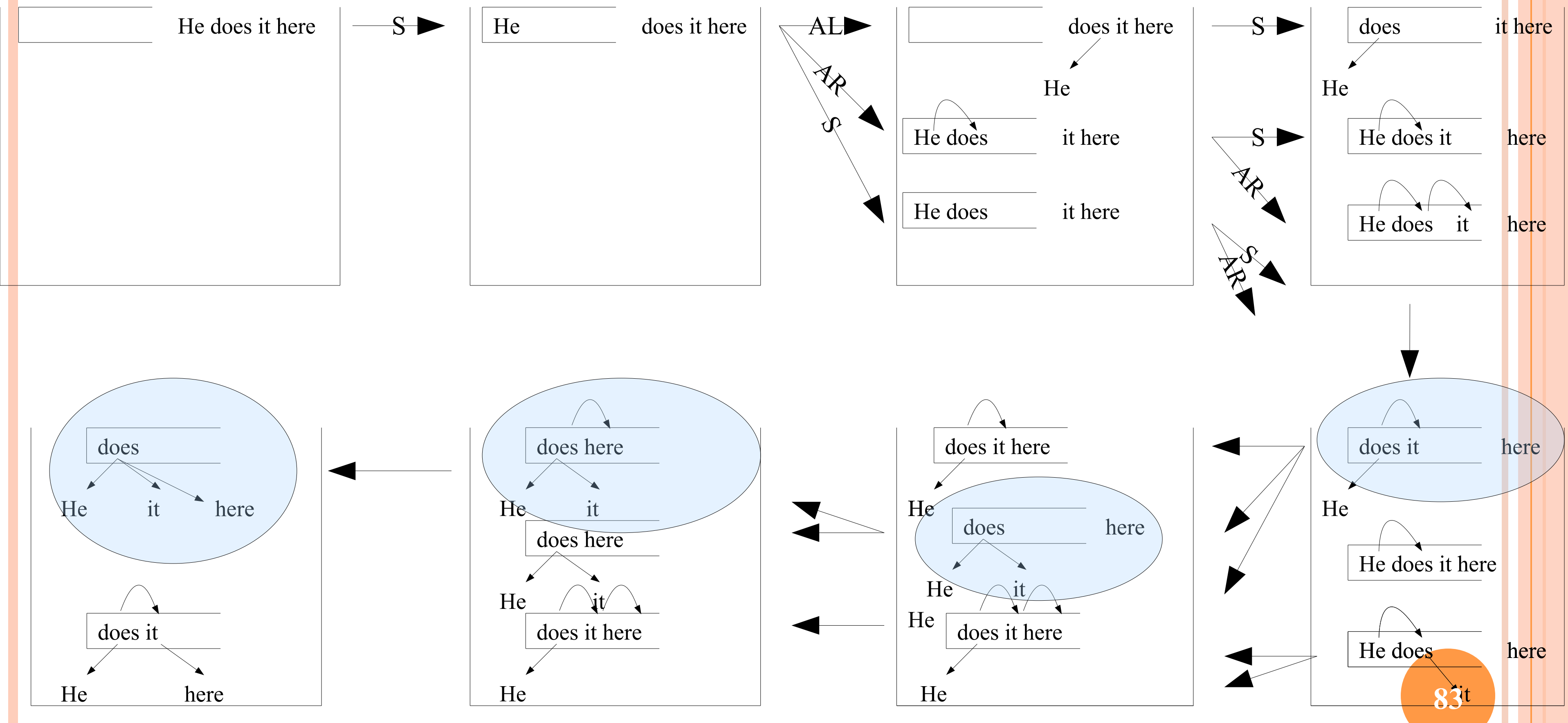
Beam search

- An example



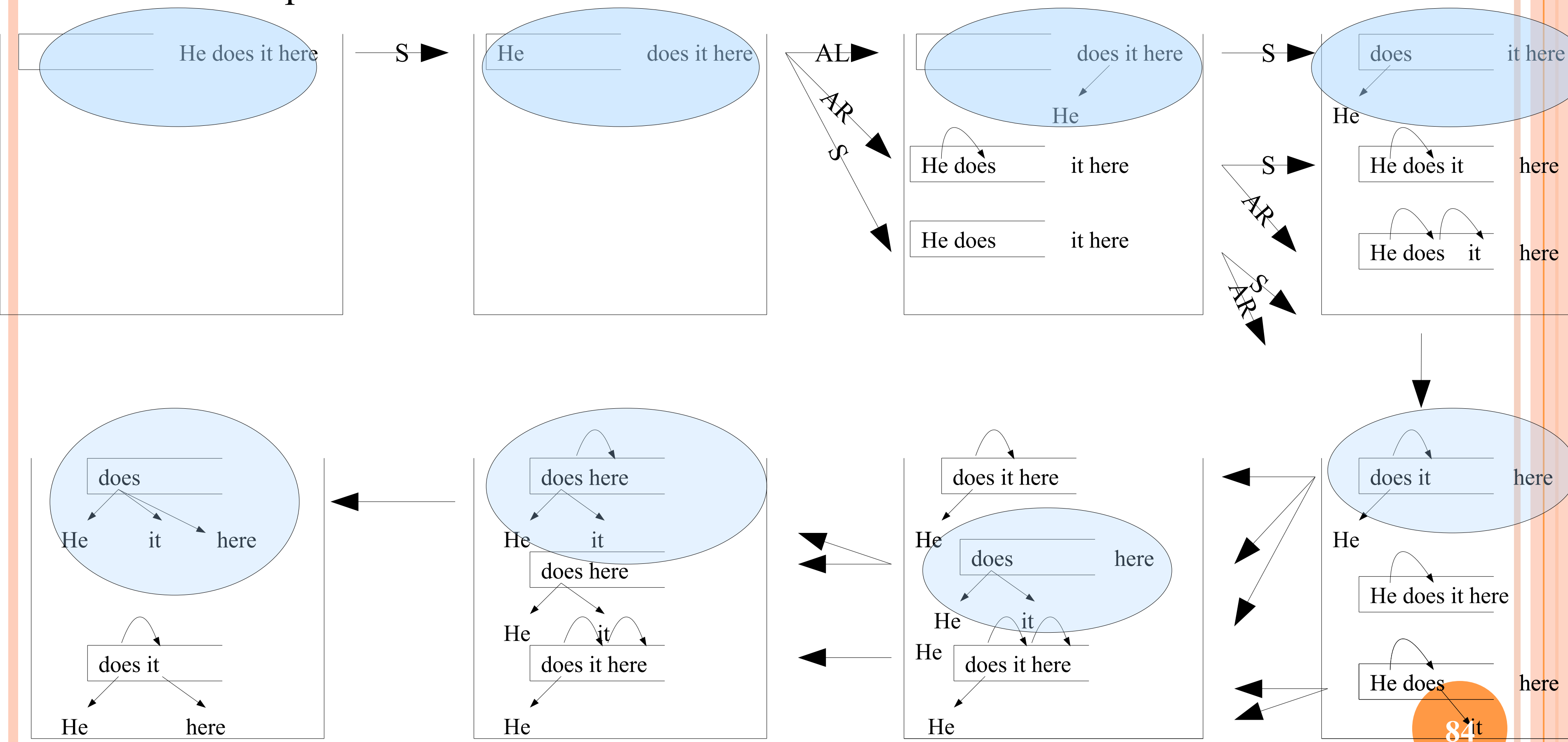
Beam search

- An example



Beam search

- An example



Parsing algorithms

- Search strategies
  - Greedy local search
  - Beam search
  - Best-first
    - Duan et al. (2007)

Parsing algorithms

- Search strategies
  - Greedy local search
  - Beam search
  - Best-first
  - Other strategies?
    - Huang and Sagae (2010)

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
  - Transition-based parsing processes
  - Decoding algorithms
  - **Learning algorithms and feature templates**
- Part D: the integrated models
- Part E: other recent trends in dependency parsing

Models

- The way we score transition actions
  - Linear models

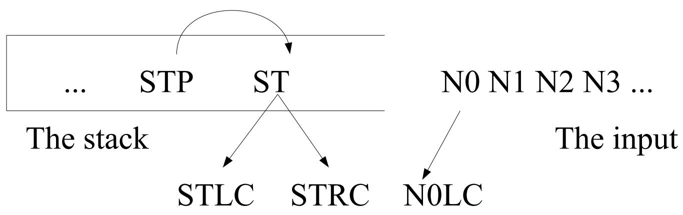
$$Score(action) = \sum_{feature \in features\ with\ context} feature \times weight(feature)$$

  - Non-linear models
    - SVM

      non-linear kernels

Learning algorithms

- Locally learn for each transition action
  - SVM

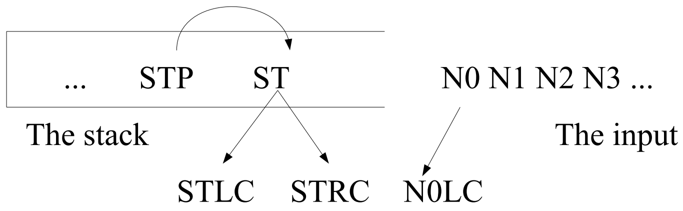


  - Examples
    - MaltParser (Nivre et al., 2006)
    - Johansson and Nugues (2007)
    - Duan (2007)

  - LIBSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)

Learning algorithms

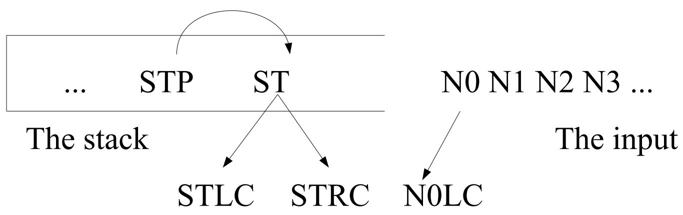
- Feature templates



- Example templates
  - STw, STp,
  - N0w, N0p,
  - ST N0 distance,
  - STLCw, STLCp,
  - N1w, N1p
  - ...

Learning algorithms

- Feature templates



- Example templates
  - STw, STp,
  - N0w, N0p,
  - ST N0 distance,
  - STLCw, STLCp,
  - N1w, N1p
  - …
- A second order polynomial kernel will combine individuals

Learning algorithms

- Globally learn the best sequence of actions
  - Linear model to score actions
  - Globally search for the best sequence of actions, globally learn

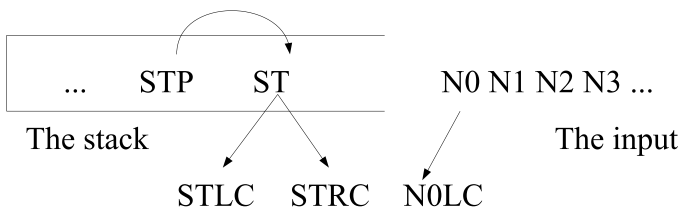
$$
\begin{aligned}
& Score(parse) \\
= \; & \sum_{action \in parse} Score(action) \\
= \; & \sum_{action \in parse} \sum_{feature \in status\ for\ action} feature \times weight(feature)
\end{aligned}
$$

Learning algorithms

- Globally learn the best sequence of actions
  - Zhang and Clark (2008)
  - Use the generalized perceptron learning algorithm (Collins, 2002)

Learning algorithms

- Feature templates



- Example templates
  - STw, STp,
  - N0w, N0p,
  - ST N0 distance,
  - STwSTp, STwN0w, STwpN0wp
  -
  - …
- Manual combination of information; linear model.

References

Xiangyu Duan, Jun Zhao, Bo Xu, 2007. Probabilistic Models for Action-Based Chinese Dependency Parsing. In proceedings of ECML , pages 559-566
Liang Huang, Wenbin Jiang, and Qun Liu, 2009. Bilingually-Constrained (Monolingual) Shift-Reduce Parsing. In Proceedings of EMNLP 2009.
Richard Johansson and Pierre Nugues. 2007. Incremental Dependency Parsing Using Online Learning. In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL, pp. 1134–1138, Prague, June 2007.
Joakim Nivre and Ryan McDonald. 2008. Integrating Graph-Based and Transition-Based Dependency Parsers. In Proceedings of ACL-08: HLT, pages 950–958, Columbus, USA. June.
Joakim Nivre, Johan Hall, Jens Nilsson, Gulson Erygit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL), pages 221–225.
Joakim Nivre. 2009. Non-Projective dependency parsing in expected linear time. In Proceedings of the 47th meeting of the ACL and the 4th international conference on Natural Language Processing of the FNLP, pages 351-359.
Kenji Sagae and Alon Lavie. 2006. Parser Combination by Reparsing. In Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 129–132, New York, June 2006.
Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT), pages 195–206.
Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 562–571.
Yue Zhang and Stephen Clark. Transition-Based Parsing of the Chinese Treebank using a Global Discriminative Model.In proceedings of IWPT 2009. Paris, France. October.

Recent Advances in Dependency Parsing

Qin Iris Wang
AT&T Interactive
qiniriswang@gmail.com

Yue Zhang
Cambridge University
frcchang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Part D: The Combination of Different Models

Yue Zhang

Cambridge University

frcchang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

The combined models

- Motivation
  - Parsers make different mistakes, each having a particular strength
    - McDonald and Nivre (2007)
  - Combined parser lead to superior accuracies than individual parsers

Overview

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
- Part D: the integrated models
  - **The ensemble approach**
  - The stacking approach
  - The single-model approach
- Part E: other recent trends in dependency parsing

The ensemble method

- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

The ensemble method

- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

  - m parses for a single input
  - Combine all parses
    - Calculate link weights according to each parse
    - Add m numbers
    - Links from different parser outputs weighted equally or differently according to various configurations

The ensemble method

- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

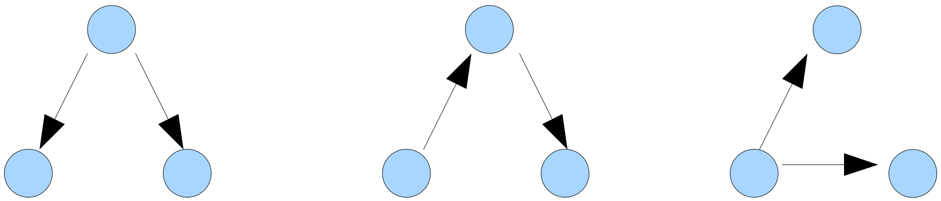
  - m parses for a single input
  - Combine all parses
    - Calculate link weights according to each parse
    - Add m numbers
    - Links from different parser outputs weighted equally or differently according to various configurations

  - Find the MST according to these weights

The ensemble method

- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

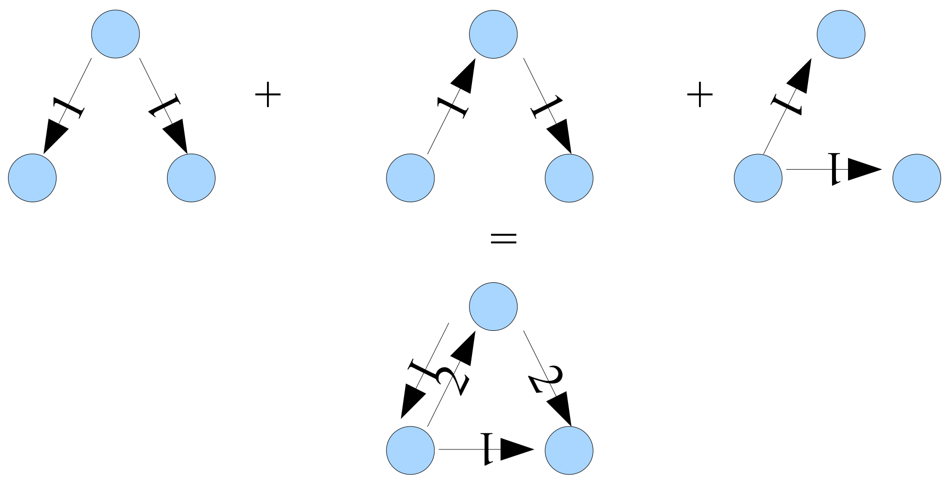
  - m parses for a single input



The ensemble method

- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

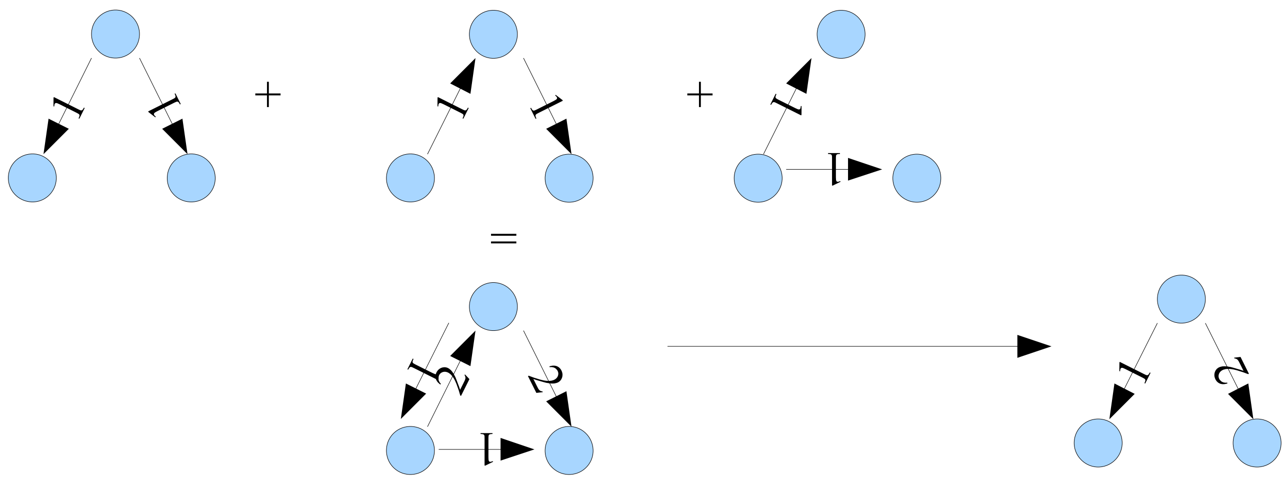
  - m parses for a single input
  - Combine all parses



The ensemble method

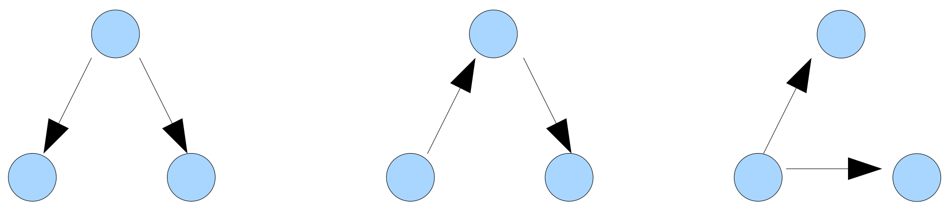
- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

  - m parses for a single input
  - Find MST



The ensemble method

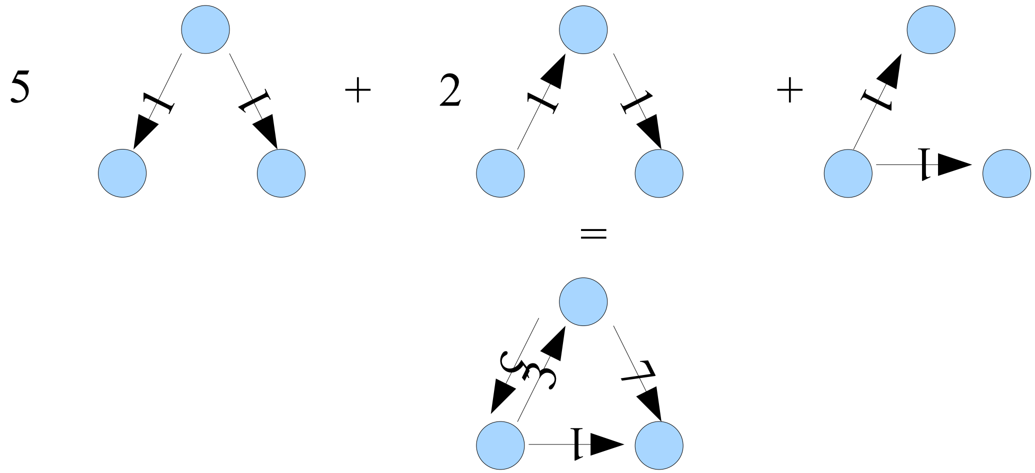
- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately
  - m parses for a single input



The ensemble method

- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

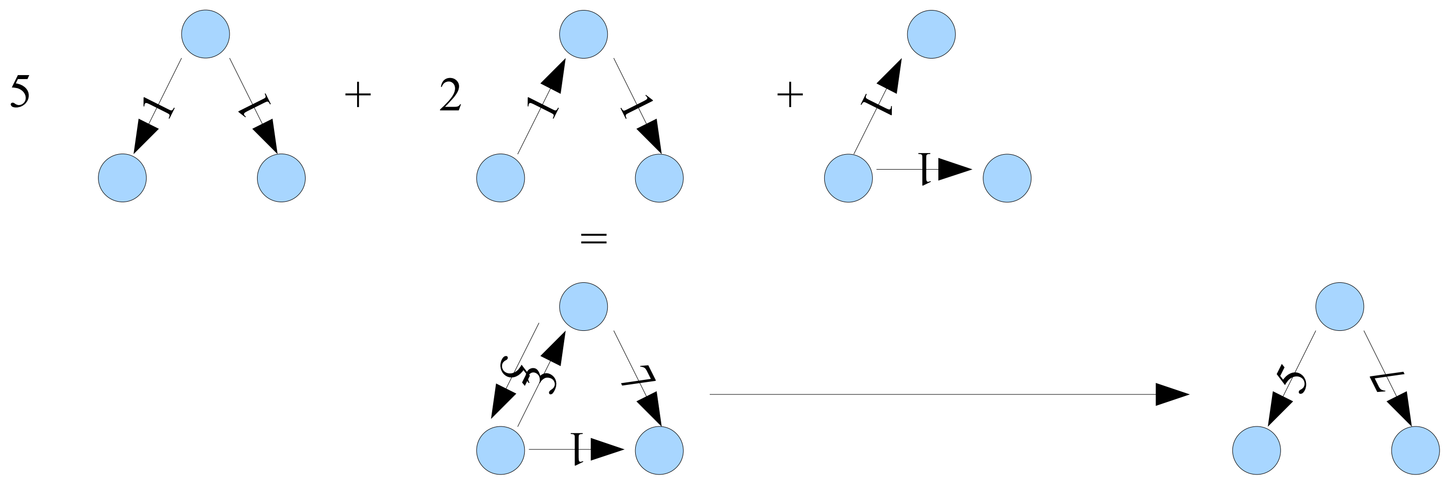
  - m parses for a single input
  - Combine all parses by weighted sum of them



The ensemble method

- Sagae and Lavie (2006)
  - m parsers
  - Each different and trained separately

  - m parses for a single input
  - Find the output



Overview

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
- Part D: the integrated models
  - The ensemble approach
  - **The stacking approach**
  - The single-model approach
- Part E: other recent trends in dependency parsing

The stacking method

- Nivre and McDonald (2008)
  - Combination of
    - Graph-based MSTParser
    - Transition-based MaltParser
  - Stacking

The stacking method

- Nivre and McDonald (2008)
  - Train one parser first
    - Parser1
  - Let the other parser (i.e. parser2) consult parser1 when it does parsing
  - Two resulting parsers (Malt-MST, and MST-Malt)

The stacking method

- Nivre and McDonald (2008)
  - During test
  - Use parser1 to parse input
  - Parser2 extract features from parser1 output
  - Take parser2 output as the result

The stacking method

- Nivre and McDonald (2008)
  - During training
  - Use parser1 to parse training data
  - Parser2 extract features from parser1 output
  - Train parser2 with the additional features

The stacking method

- Nivre and McDonald (2008)
  - During training
  - *Use parser1 to parse training data*
  - Parser2 extract features from parser1 output
  - Train parser2 with the additional features

The stacking method

- Nivre and McDonald (2008)
  - During training
  - *Use parser1 to parse training data*
    - Can't train parser1 on the training data (same set)
    - Solution
      - 10-fold cross-validation
      - Take a tenth of the training data as the "test" data
      - Use the other nine tenths to train parser1
      - Generate parser1 output for the "test" sent
      - Repeat 10 times to get parser1 output for all training sentences
  - Parser2 extract features from parser1 output
  - Train parser2 with the additional features

Overview

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based dependency parsing models
- Part D: the integrated models
  - The ensemble approach
  - The stacking approach
  - **The single-model approach**
- Part E: other recent trends in dependency parsing

The single-model method

- Zhang and Clark (2008)
  - Combine graph-based and transition-based parsers
    - Same as just now
  - Two parsers are treated equally
    - Graph-based and transition-based information in a single model
    - Trained together
    - Used together for decoding
  - They become one
    - single-model

The single-model method

- Zhang and Clark (2008)
  - Challenges:
    - Decoder combination
      - Graph-based parsers typically take dynamic programming
      - Transition-based features hard to be accommodated by DP at the same time
    - Model combination
      - How to use both kinds of information in a single model?
    - Training combination

The single-model method

MSTParser

Graph-based

Exact search

- Accurate
- Local features

MaltParser

Transition-based

Greedy (no search)

- Less accurate
- Non-local features

The single-model method

<u>MSTParser</u>

Graph-based

Exact search

- Accurate
- Local features

Beam search (approximate)

- Some search
- Non-local features

<u>MaltParser</u>

Transition-based

Greedy (no search)

- Less accurate
- Non-local features

The single-model method

MSTParser		MaltParser
Graph-based	Combine	Transition-based
Exact search	Beam search (approximate)	Greedy (no search)
Accurate	Some search	Less accurate
Local features	Non-local features	Non-local features

The single-model method

- Zhang and Clark (2008)
  - Decoder combination
    - The beam-search decoder for the transition-based parser
      - Provides transitions;
      - Provides graph (partial parse in candidate item <S, Q, G>);
      - Does not restrict features – we use non-local graph-features too.
  - Model combination
  - Training methods of the combined model

The single-model method

- Zhang and Clark (2008)
  - Decoder combination
  - Model combination (linear models)
    - $Score_{COMBINED}(parse) = Score_{GRAPH}(parse) + Score_{TRANSITION}(parse)$
    - $Score_{GRAPH}(parse) = \sum_{feature \in parse} feature \times weight(feature)$
    - $Score_{TRANSITION}(parse)$

$$= \sum_{action \in parse} Score(action)$$

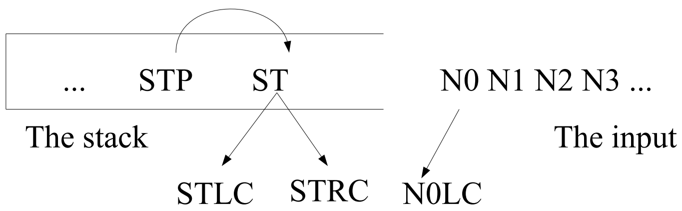
$$= \sum_{action \in parse} \sum_{feature \in status\ for action} feature \times weight(feature)$$

$$Score_{COMBINED}(parse) = \sum_{feature \in graph + action} feature \times weight(feature)$$

  - Training methods of the combined model

The single-model method

- Zhang and Clark (2008)
  - Decoder combination
  - Model combination



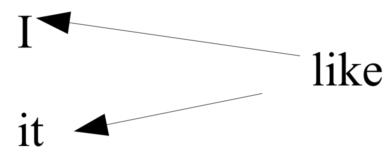
  - Transition feature templates (w – word, t – POS tag)
    - **Stack top**: STwt; STw; STt
    - **Current word**: N0wt; N0w; N0t
    - **Next word**: N1wt; N1w; N1t
    - **Stack top and current word**: STwtN0wt; STwtN0w; ...
    - **POS bigram**: N0tN1t
    - **POS trigrams**: N0tN1tN2t; STtN0tN1t; ...
    - **N0 word + POS bigrams**: N0wN1tN2t; STtN0wN1t; ...
  - Training methods of the combined model

The single-model method

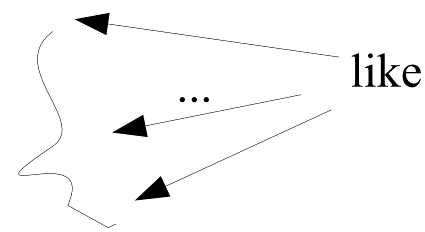
- Zhang and Clark (2008)
  - Decoder combination
  - Model combination
    - Graph feature templates
      - From MSTParser
        - **Head**: Head word, head tag, head word + tag
        - **Modifier**: Modifier word, modifier tag, modifier word + tag
        - **Head + modifier:** word / tag combinations
        - **Between**: Any tag between head and modifier
        - **Surrounding**: Tags on the left / right of head / modifier
        - **Sibling**: word / tag combinations
      - Extra features
        - **Two links**: Tags of parent, child and grandchild
        - **Arity** + head word / tag Transition feature templates (w – word, t – POS tag)
  - Training methods of the combined model

The single-model method

- Zhang and Clark (2008)
  - Decoder combination
  - Model combination
    - Graph feature templates
      - From MSTParser



      - Extra features



  - Training methods of the combined model

The single-model method

- Zhang and Clark (2008)
  - Decoder combination
  - Model combination
  - Training methods of the combined model
    - Perceptron – allowed by the linear model

The combined models

- Comparison
  - Ensemble method: decoding time combination
  - Stacking method: decoding and training time combination, but separately
  - Single method: complete combination
- One recent study about ensemble / stacking
  - Surdeanu and Manning (2010)

References

Xiangyu Duan, Jun Zhao, Bo Xu, 2007. Probabilistic Models for Action-Based Chinese Dependency Parsing. In proceedings of ECML , pages 559-566

Liang Huang, Wenbin Jiang, and Qun Liu, 2009. Bilingually-Constrained (Monolingual) Shift-Reduce Parsing. In Proceedings of EMNLP 2009.

Richard Johansson and Pierre Nugues. 2007. Incremental Dependency Parsing Using Online Learning. In Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL, pp. 1134–1138, Prague, June 2007.

Joakim Nivre and Ryan McDonald. 2008. Integrating Graph-Based and Transition-Based Dependency Parsers. In Proceedings of ACL-08: HLT, pages 950–958, Columbus, USA. June.

Joakim Nivre, Johan Hall, Jens Nilsson, Gulson Erygit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL), pages 221–225.

Joakim Nivre. 2009. Non-Projective dependency parsing in expected linear time. In Proceedings of the 47th meeting of the ACL and the 4th international conference on Natural Language Processing of the FNLP, pages 351-359.

Kenji Sagae and Alon Lavie. 2006. Parser Combination by Reparsing. In Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 129–132, New York, June 2006.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In Proceedings of the 8th International Workshop on Parsing Technologies (IWPT), pages 195–206.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 562–571.

Yue Zhang and Stephen Clark. Transition-Based Parsing of the Chinese Treebank using a Global Discriminative Model.In proceedings of IWPT 2009. Paris, France. October.

Recent Advances in Dependency Parsing

Qin Iris Wang
AT&T Interactive
qiniriswang@gmail.com

Yue Zhang
Cambridge University
frcchang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Part E: Other Recent Trends in Dependency Parsing

Qin Iris Wang

AT&T Interactive
qiniriswang@gmail.com

NAACL Tutorial, Los Angeles
June 1, 2010

Outline

- Part A: introduction to dependency parsing
- Part B: graph-based dependency parsing models
- Part C: transition-based models
- Part D: the combined models
- Part E: other recent trends in dependency parsing
  - Explore higher order features
  - Use extra information source
  - Better parsing strategies

Other Recent Trends in Dependency Parsing

- Explore higher order features
- Use extra information sources
  - Raw data
  - Bilingual data
  - Linguistic rules
- Better parsing strategies

Explore Higher-Order Features (1)

- Dependency Parsing by Belief Propagation (Smith & Eisner, 08)
  - Has a first order baseline parser
  - Using a BP network to incorporate higher order features into this first order parser approximately

- Integration of graph-based and transition-based models (Zhang & Clark, 08)
  - Approximation by beam-search

Explore Higher-Order Features (2)

- Concise Integer Linear Programming Formulations for Dependency Parsing (Martins et al. 09)
  - Formulate dependency parsing as a polynomial-sized integer linear program
  - Integer linear programming in NLP tutorial this afternoon

Use Extra Information Source –Raw Data

- Improving dependency parsing with subtrees from auto-parsed data (W. Chen et al. 09)
  - Using a base parser to pare large scale unannotated data
  - Extract subtrees from the auto-parsed data
- Simple semi-supervised dependency parsing (Koo et al. 08)
- Semi-supervised convex dependency parsing (Wang et al. 08)

Use Extra Information Source – Bilingual Data

- Bilingually-constrained monolingual shift-reduce parsing (Huang et al. 09)
  - A novel parsing paradigm that is much simpler than bi-parsing
  - Enhance a shift-reduce dependency parser with alignment features to resolve shift-reduce conflicts

Use Extra Information Source – Linguistic Rules

- Semi-supervised Learning of Dependency Parsers using Generalized Expectation Criteria (Druck et al. 09)
  - Directly use linguistic prior knowledge as a training signal
  - Model parameters are estimated using a generalized expectation (GE) objective function that penalizes the mismatch between model predictions and linguistic expectation constraints.

Better Parsing Strategies

- Non-projective shift-reduce parsing (Nivre, 09)
  - Expected linear time
- Easy-First Non-Directional Dependency Parsing (Goldberg and Elhadad, 10)
  - Inspired by Shen et al. 07
  - Use an easy-first order instead, *O(nlogn)* complexity
  - Allows using more context at each decision
- Dynamic programming for incremental parsing (Huang & Sagae, 10)
  - Linear time

References

- W. Chen, J. Kazama, K. Uchimoto and K. Torisawa. 2009. Improving Dependency Parsing with Subtrees from Auto-Parsed Data. In *Proc. EMNLP*.
- Gregory Druck; Gideon Mann; Andrew McCallum. 2009. Semi-supervised Learning of Dependency Parsers using Generalized Expectation Criteria. In *Proc. ACL*.
- Y. Goldberg and M. Elhadad. 2010. An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. In *Proc. NAACL*.
- L. Huang, W. Jiang, and Q. Liu. 2009. Bilingually-Constrained (Monolingual) Shift-Reduce Parsing. In *Proc. EMNLP*.
- L. Huang and K. Sagae. 2010. Dynamic Programming for Linear-time Incremental Parsing. In *Proc. ACL*.
- W. Jiang and Q. Liu. 2010. Dependency Parsing and Projection Based on Word-Pair Classification. In *Proc. ACL*.
- M. Kuhlmann and G. Satta. 2009. Treebank Grammar Techniques for Non-Projective Dependency Parsing. In *Proc. EACL*.

References

- A. Martins, N. Smith and E. Xing. 2009. Concise Integer Linear Programming Formulations for Dependency Parsing. *In Proc. ACL-IJCNLP*.
- J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. ACL-HLT*.
- S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. EMNLP*.
- L. Shen, G. Satta and A. K. Joshi. 2007. Guided learning for bidirectional sequence classifica- tion. In *Proc. ACL*.
- D. Smith and J. Eisner. 2008. Dependency Parsing by Belief Propagation. In *Proc. EMNLP*.

Thanks!