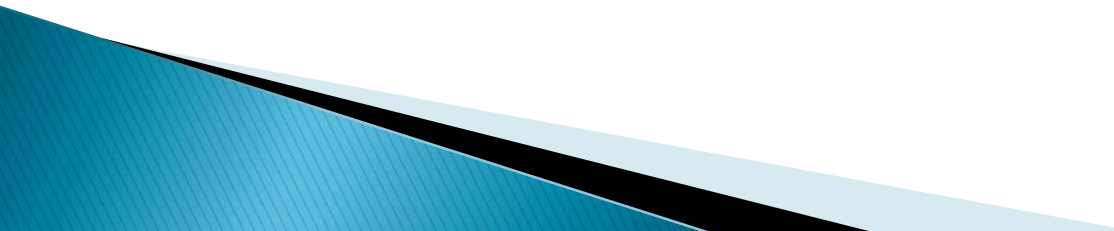


# Self Training

(...when is Self-Training Effective for Parsing?)

Marcel Köster (WS09/10)

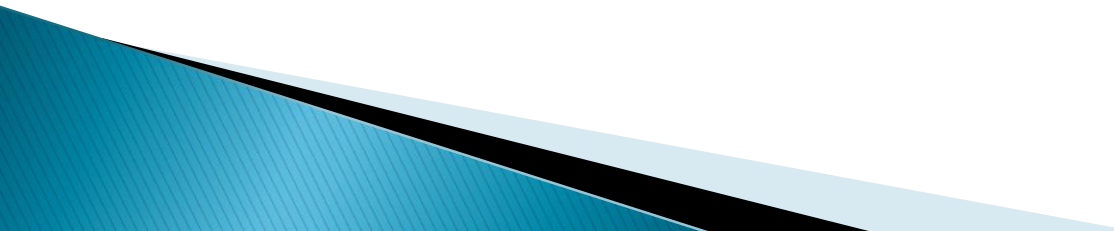
# Presentation Outline

- ▶ Introduction to Self Training
  - ▶ Previous applications of Self-Training
  - ▶ Why and when does self-training actually work for parsing and improve performance?
- 

# Self Training

»» Introduction to ST

# Definitions

- ▶  $\Theta$  = number of iterations performed
  - ▶  $\Omega$  = seed (amount of labeled training data)
  - ▶  $\Phi$  = set of sentences per iteration
  - ▶  $\Pi$  = probability with which a sentence is accepted during a ST-iteration
- 

# Revision / Introduction to ST

Function **Train**( $\Omega$ ,  $\Phi$ ,  $\Theta$ ,  $\Pi$ )

Model := **Train**( $\Omega$ )

TrainingData :=  $\Omega$

**For**  $i = 1$  **To**  $\Theta$

    LabeledData := **Use**(Model,  $\Phi[i]$ )

    TrainingData := **Combine**(TrainingData, LabeledData)

    Model := **Train**(TrainingData,  $\Pi$ )

**Next**

**Return** Model



# Self Trained Parsers

- » Previous applications of Self-Training  
(using generative parsers)

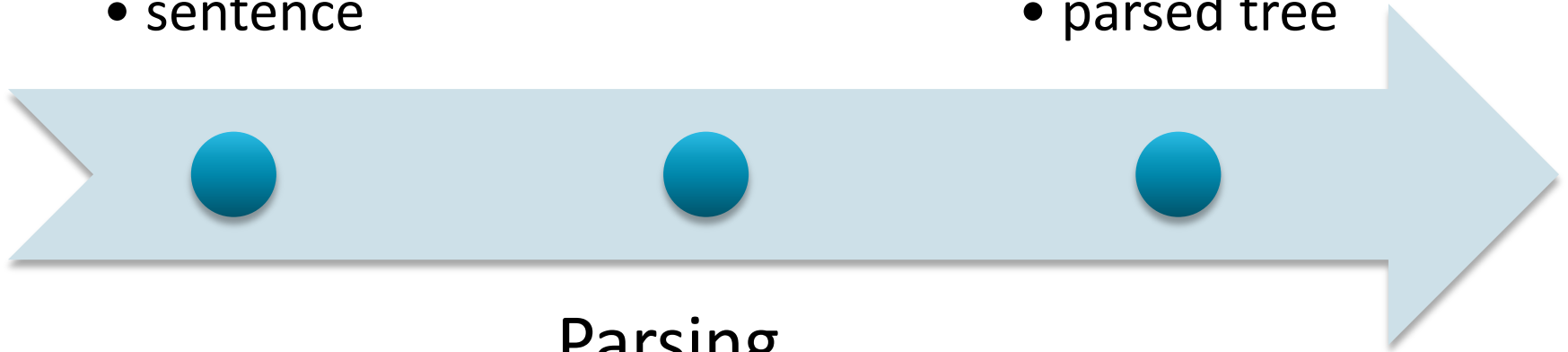
# Generative PCFG Parsers

Input

- sentence

Result

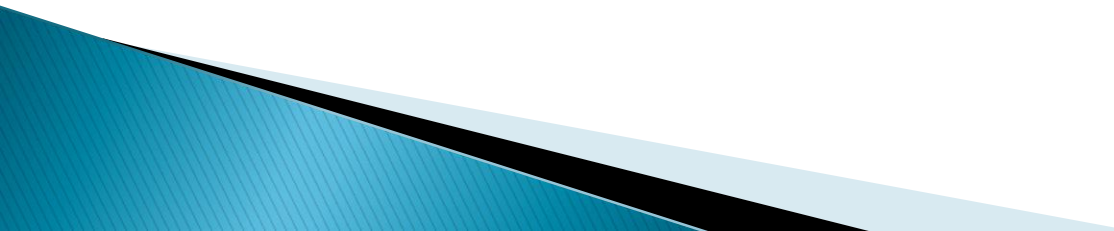
- parsed tree



Parsing

- generative  
Parser

# First reported use of Self-Training

- ▶ by Charniak (1997)
  - ▶ Parser trained on the Wall Street Journal
  - ▶ New model using 30 million words of unparsed WSJ text
  - ▶ Model getting worse and worse for a larger  $\Theta$
- 



# Exp. From the 2002 CLSP SW

- ▶ Used different trained models
- ▶ Performed many iterations of ST
- ▶  $\Phi = \sim 30$
- ▶  $\text{Max}(|\Omega|) = \sim 10.000$  sentences (WSJ)
- ▶  $\rightarrow$  training did not yield a significant gain

# Reichart & Rappoport 2007

- ▶  $|\Omega|$  is small
- ▶  $\Theta = 1$
- ▶ Model improved by using  $\Theta = 1$

# What do we know so far?

- ▶ Usage of generative PCFG parsers
- ▶ The larger  $\Theta$  gets the lower the performance
  - $\rightarrow \Theta = 1$  will result in better performance
- ▶ The full effect of  $|\Omega|$  is still unclear
- ▶ There were no qualified theories why and when the parameters influence the performance

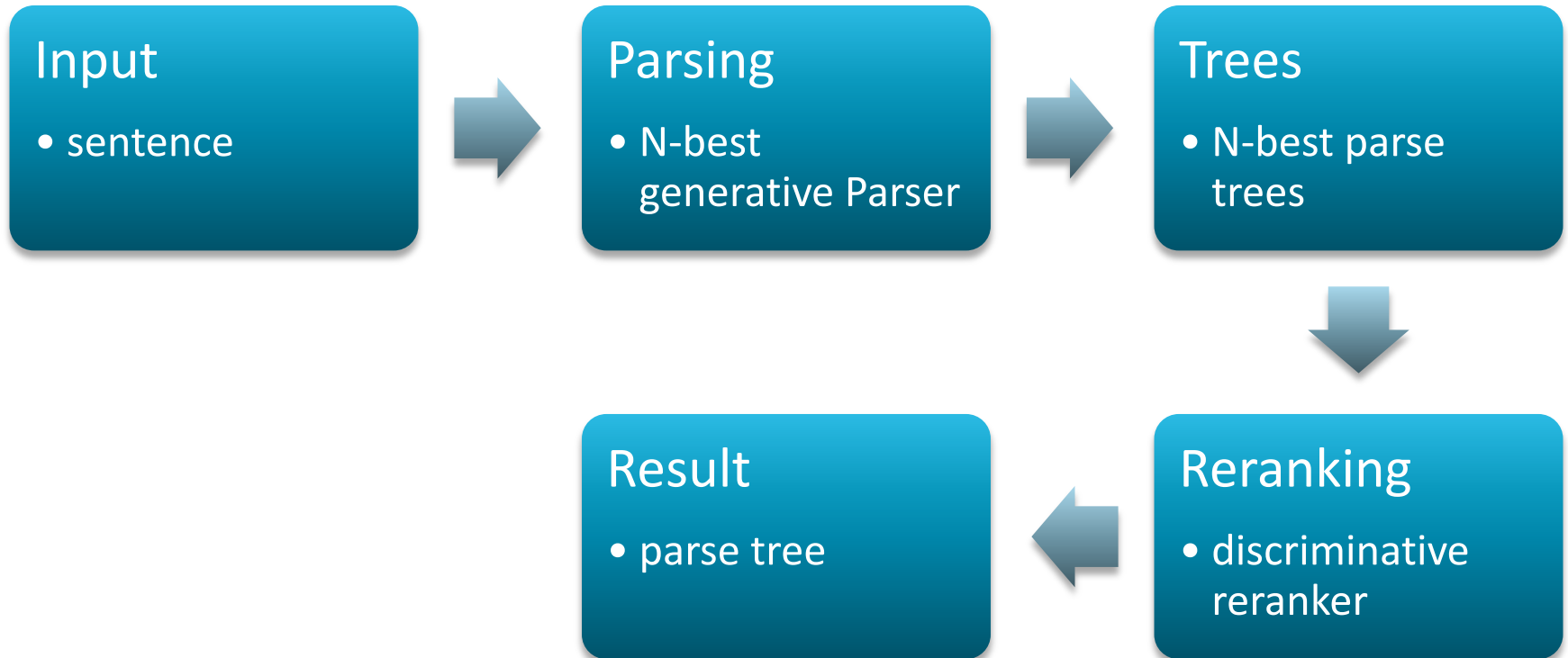
# Self Trained Parsers

»» ST with Reranking Parsers

# Reranking Parsers

- ▶ Consists of two stages:
  - Generative lexicalized PCFG parser  
(which proposes a list of the n-best parse trees)
  - Discriminative reranker  
(which reorders the n-best list to extract the best parse tree)

# Reranking Parsers



# ST of Reranking Parsers


- ▶ Self-Training using this two-stage processing improves parsing accuracy  
(McClosky et. Al 2008)
- ▶  $|\Omega|$  is quite large
- ▶  $\Theta = 1$
- ▶ Model was improved

# Self Trained Parsers

- »» Why & when is ST useful  
(with Reranking Parsers)



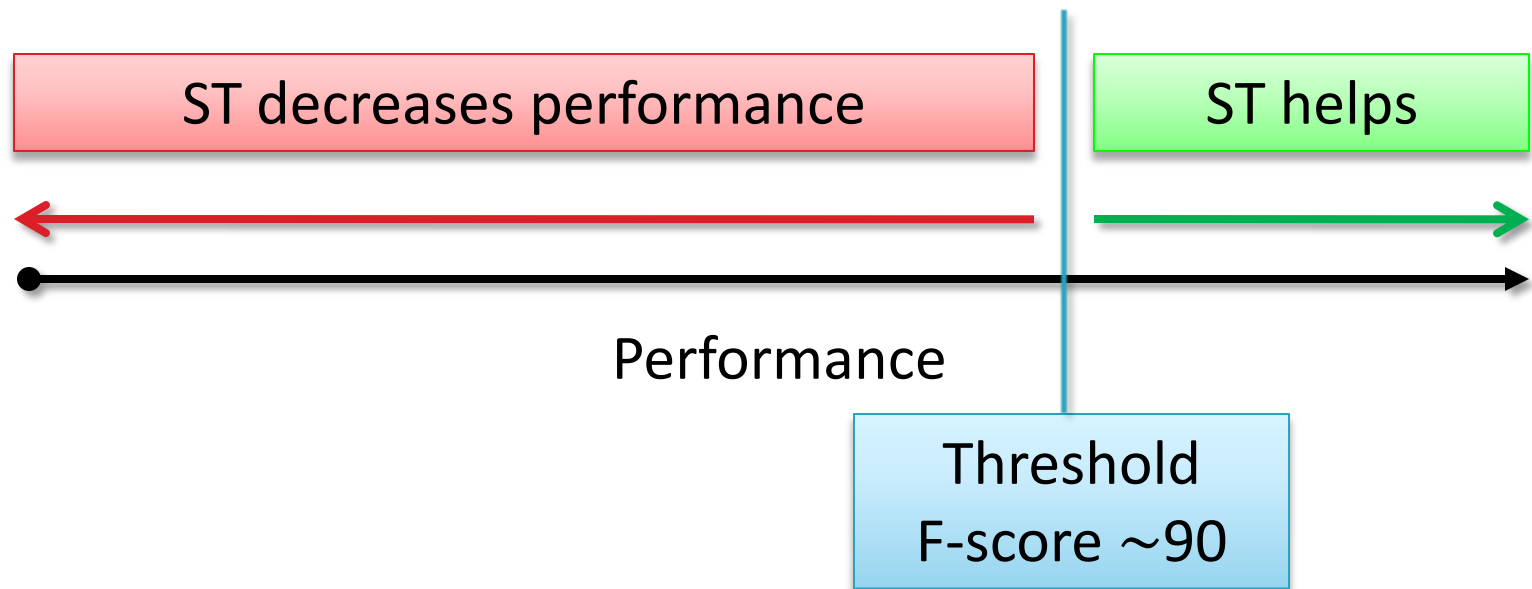
# Investigation of 4 hypotheses

1. ST helps after a phase transition
  2. ST reduces search errors
  3. ST improves more using specific classes of reranker features
  4. ST helps to handle new combinations of known (and unknown) words (so called new bilexical dependencies)
- 

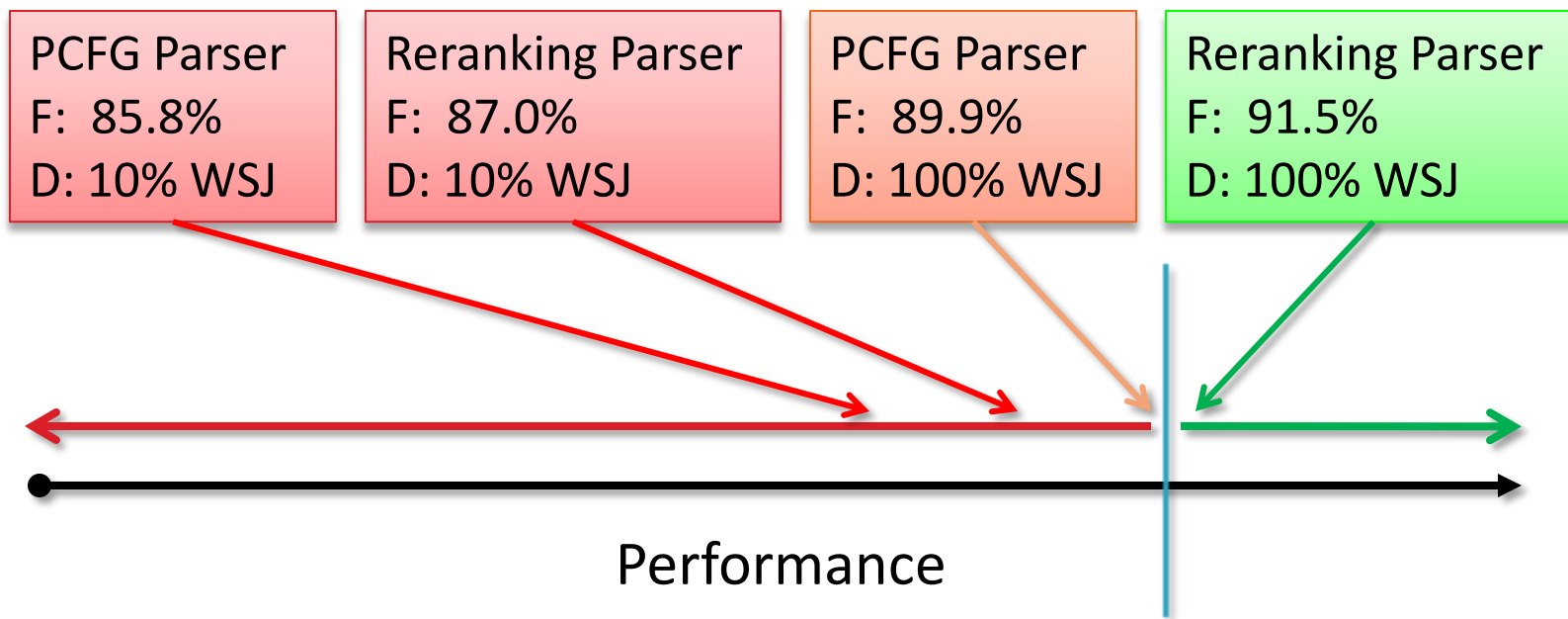
# 1. Phase Transition HT

- ▶ The hypothesis:
  - If a parser has achieved a certain (very) high threshold of performance the labels will be “good enough” for ST.
- ▶ Verification:
  - 2 tests with  $|\Omega| = 10\%$  &  $|\Omega| = 100\%$  from WSJ  
[  $S \in \Omega : |S| \leq 100$  ]
  - $\Theta = 1$  &  $\Phi = \sim 1$  Million from NANC,  $\Pi = 0.1$

# 1. Phase Transition HT

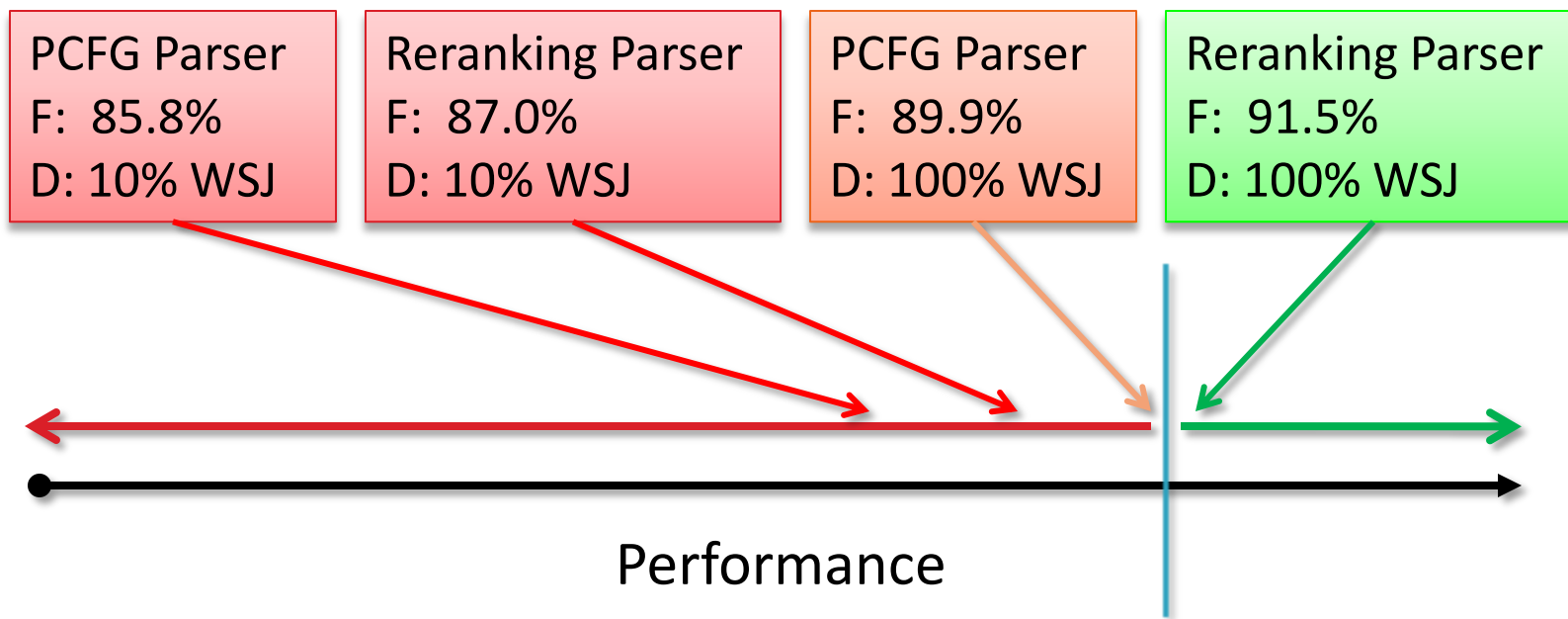


# 1. Phase Transition HT



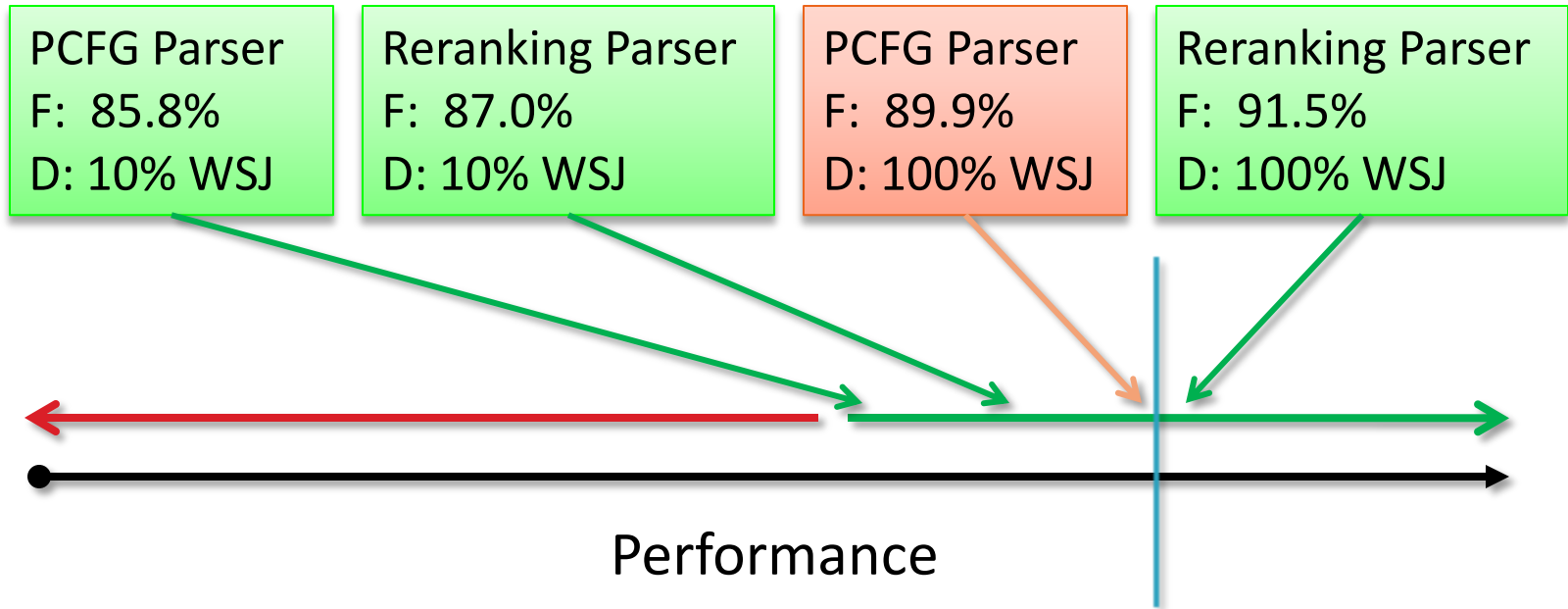
% WSJ	Parser f-score	Reranking P. f-score
10 (3,995 sentences)	85.8	87.0
100 (39,832 sentences)	89.9	91.5

# 1. Phase Transition HT



% WSJ	Reranker used?	Parser f-score	Reranking P. f-score
10	No	87.7 (+1.9)	88.7 (+1.7)
10	Yes	88.4 (+2.6)	89.0 (+2.0)

# 1. Phase Transition HT



% WSJ	Reranker used?	Parser f-score	Reranking P. f-score
10	No	87.7 (+1.9)	88.7 (+1.7)
10	Yes	88.4 (+2.6)	89.0 (+2.0)

→ the hypothesis is false

## 2. ST Reduces search errors

### ▶ What is a search error?



### ▶ Verification: investigate n-best lists

## 2. ST Reduces search errors

- ▶ Let's introduce some notations

Notation	Description
O	Parses from the original parser
S	Parses from the ST parser
topS (X)	Best parse determined by reranker from the ST model on a given set X of trees
topO (X)	Best parse determined by reranker from the original model on a given set X of trees
eval[M] (X)	Evaluates the model's (M's) f-score related to the given set of trees



## 2. ST Reduces search errors

Expression	Value
$0 \cap S$	66.0%
$\text{topS}(S) = \text{topO}(0)$	42.4%
$\text{topS}(S) \in 0$	60.3%
<b>Search Errors</b>	<b>2.5%</b>

### Search Errors

$\text{topS}(S) \notin 0$  and  $\text{topO}(S \cup 0) = \text{topS}(S)$

## 2. ST Reduces search errors

▶ F-score evaluation:

Parser	F-score
eval[O] (O)	91.5%
eval[O] (O $\cup$ S)	91.7%
eval[S] (S)	92.0%

→ the hypothesis is true

# 3. Reranker features

- ▶ 2 classes of reranker features:
  - GEN
    - Features that are (roughly) captured by the parser
      - CFG rule rewrites
      - Head-child dependencies
  - EDGE
    - Features that are (not) captured by the parser
      - Grammatical relations (also captured by the parser but.. uses combinations of labels as paths from one (non-)terminal node to another node)
      - Functional dependencies between individual and grouped terminals
- ▶ Test setting:  $\Theta = 1$  &  $\Phi = \sim 1$  Million from NANC,  $\Pi = 0.1$

# 3. Reranker features

- ▶ Evaluation of the test:

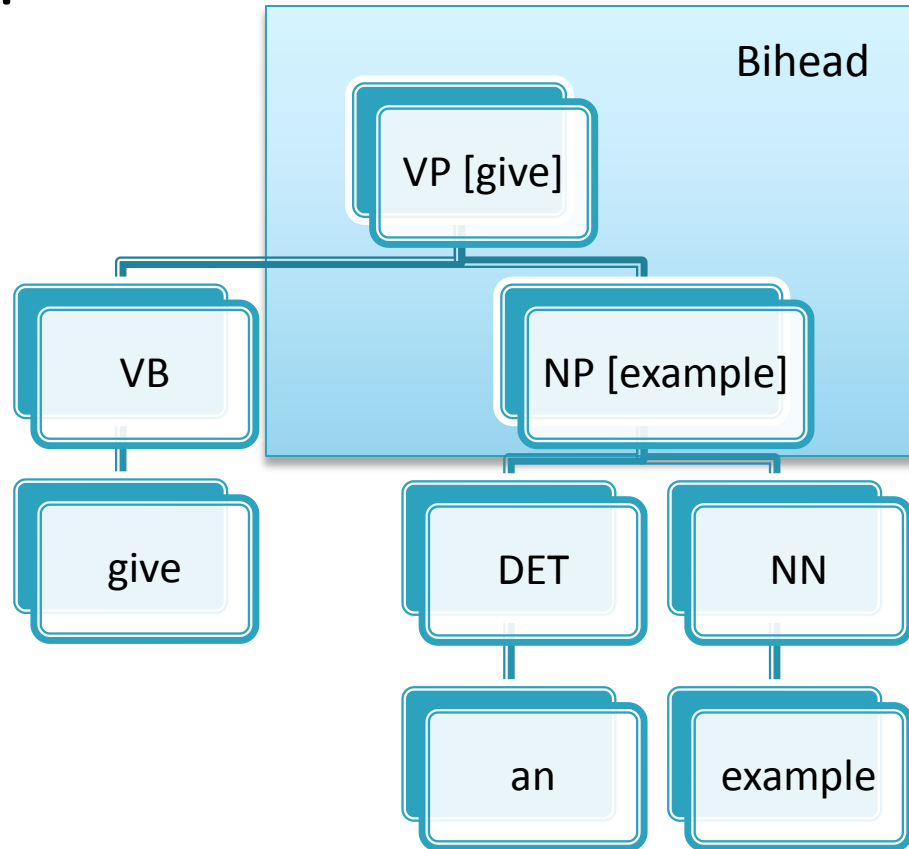
Feature set	# Number features	F-score
GEN	448.000	90.4
EDGE	885.000	91.0
ALL	~1.3 million	91.3
Reranking Parser without ST	/	90.5

- ▶ when using the features not covered by the first stage (EDGE features) the performance increases

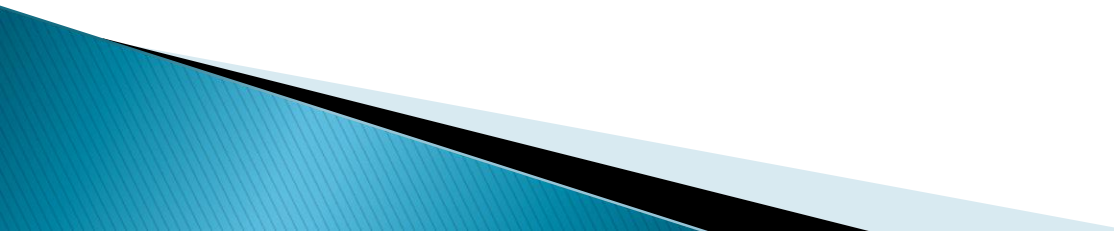
→the hypothesis is true

# 4. Bilexical Dependencies

► Revision:

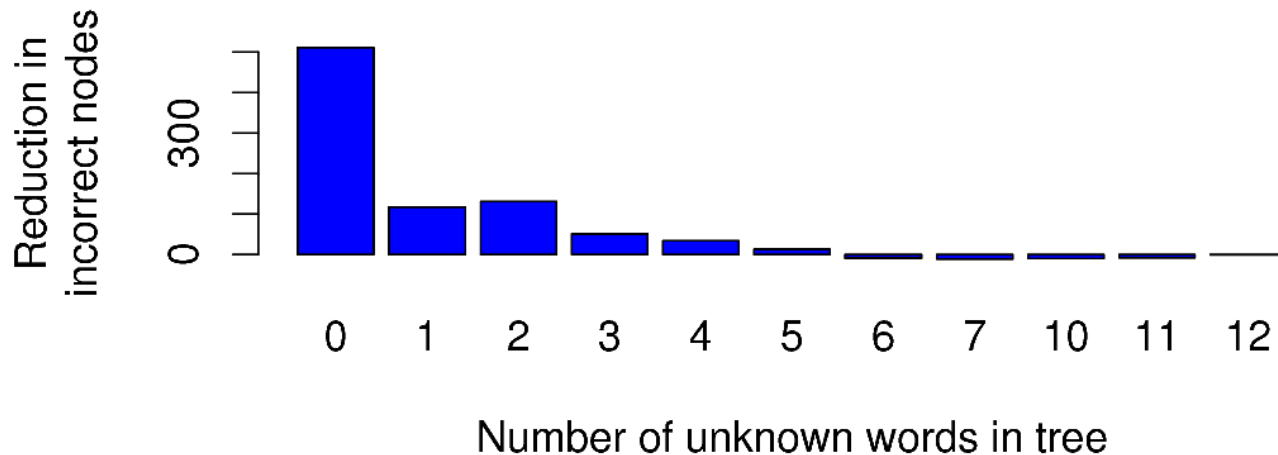


# 4. Bilexical Dependencies

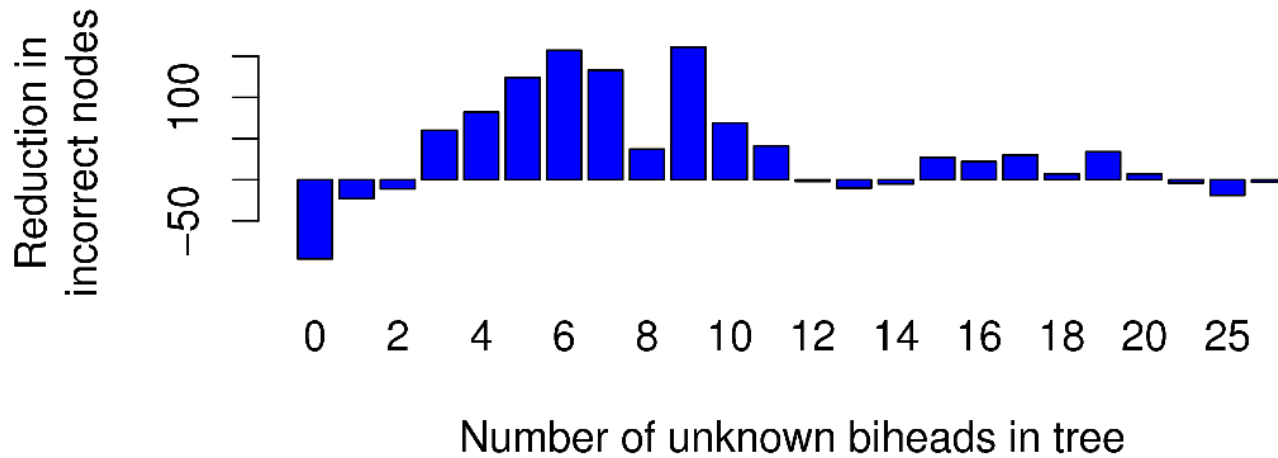
- ▶ Let's assume
    - $|\Omega|$  is relatively small
    - $\Phi$  is really large
  - ▶ Obviously the training set contains an immense number of unknown (never seen before) words
  - ▶ Vocabulary grows about 86-90% during a training step
- 

# 4. Bilexical Dependencies

- ▶ Verification: Factor Analysis  
(PCA is a special case of Factor Analysis)



# 4. Bilexical Dependencies



→ the hypothesis is true



# Summary

Hypothesis	Evaluation	Conclusion
Phase transition	Wrong	High performance is not necessary for improvements
Reduced search errors	True	Responsible for some improvements
Reranker features	True	When using features not covered by the first parser stage this will lead to improvements
Bilexical Dependencies	True	Unknown Biheads are responsible for improvements

- ▶ There is no complete explanation when and why ST actually works!

# References

## ▶ **When is Self-Training Effective for Parsing?**

- David McClosky
- Eugene Charniak
- Mark Johnson
  
- Brown Laboratory for Linguistic Information Processing (BLLIP)