Transition-Based Dependency Parsing

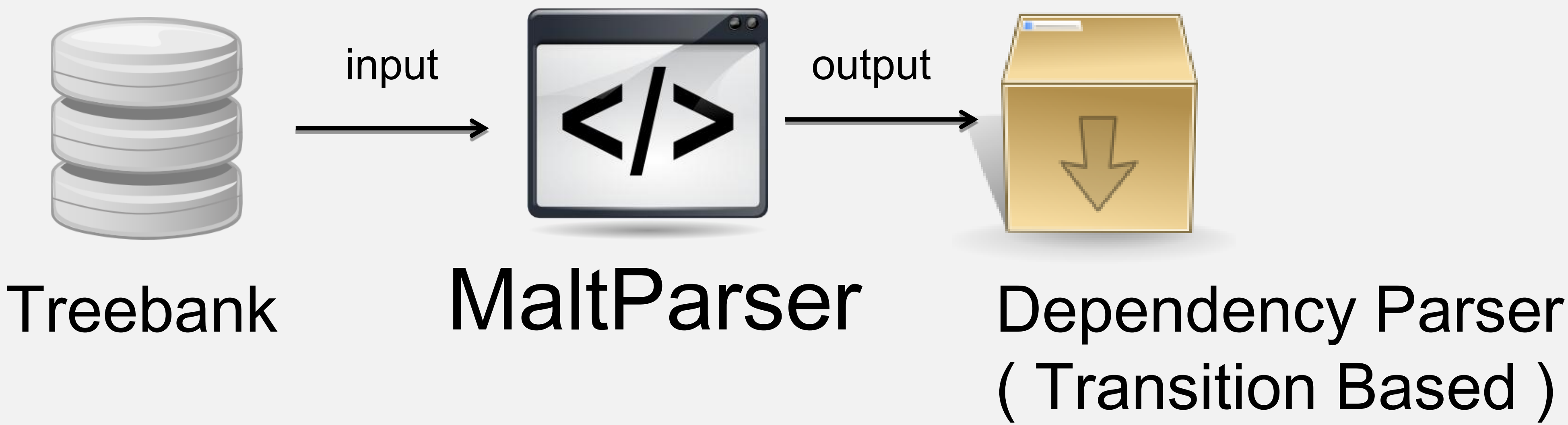
Saarbrücken, December 23rd 2011

David Przybilla – davida@coli.uni-saarland.de

Outline

1. MaltParser
2. Transition Based Parsing
   a. Example
   b. Oracle
3. Integrating Graph and Transition Based
4. Non –Projective Dependency Parsing

MaltParser



- Different Languages (No tuning for an Specific Lang)
- Language independent: accurate parsing for a wide variety of languages
- Accuracy between 80% and 90%
- Deterministic

Transition Based Parsing

Transitions

- Shift
- Left-arc
- Right-arc
- Reduction

Actions

Stack

W_k

W_x

..

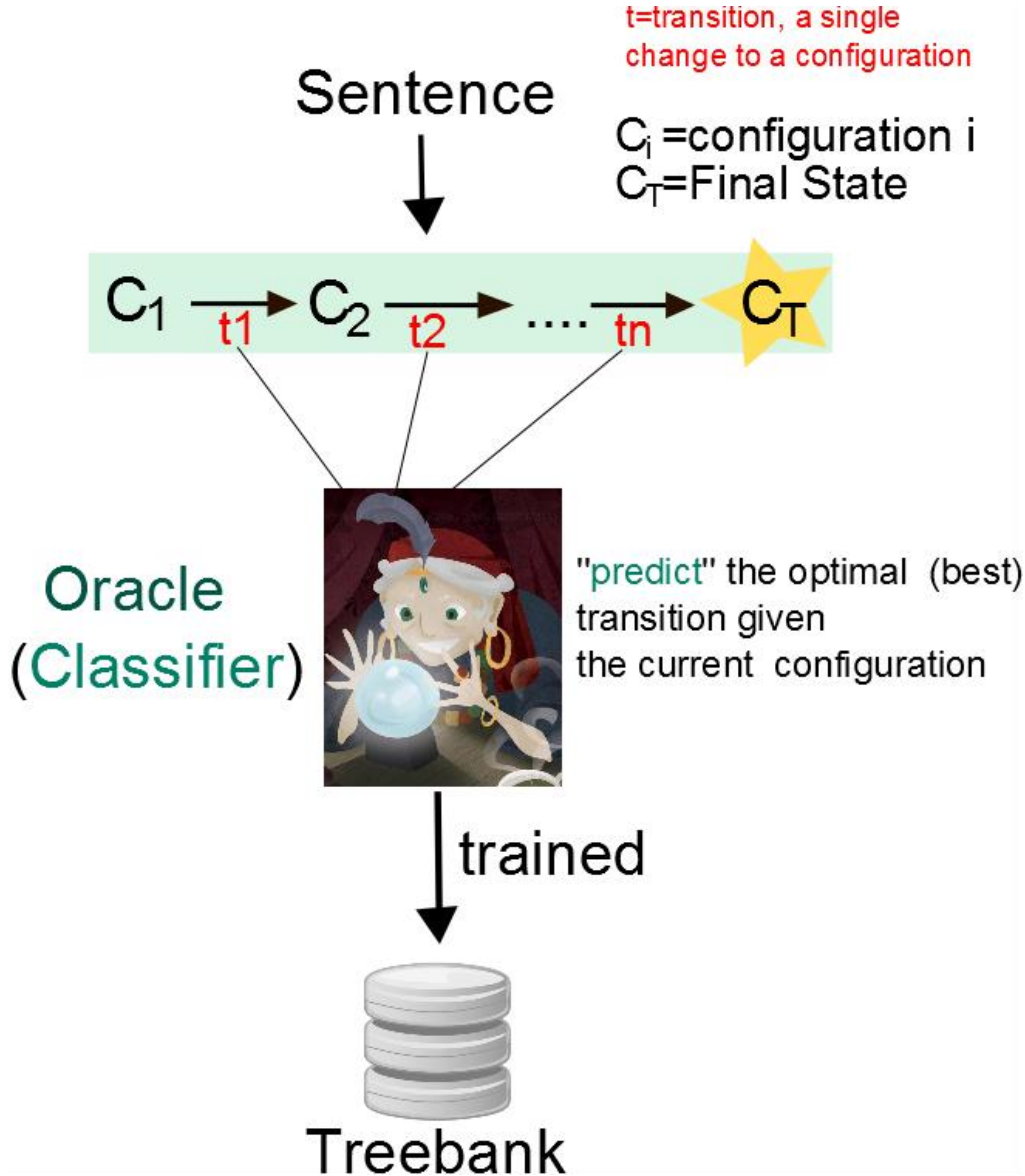
..

Buffer

W_i

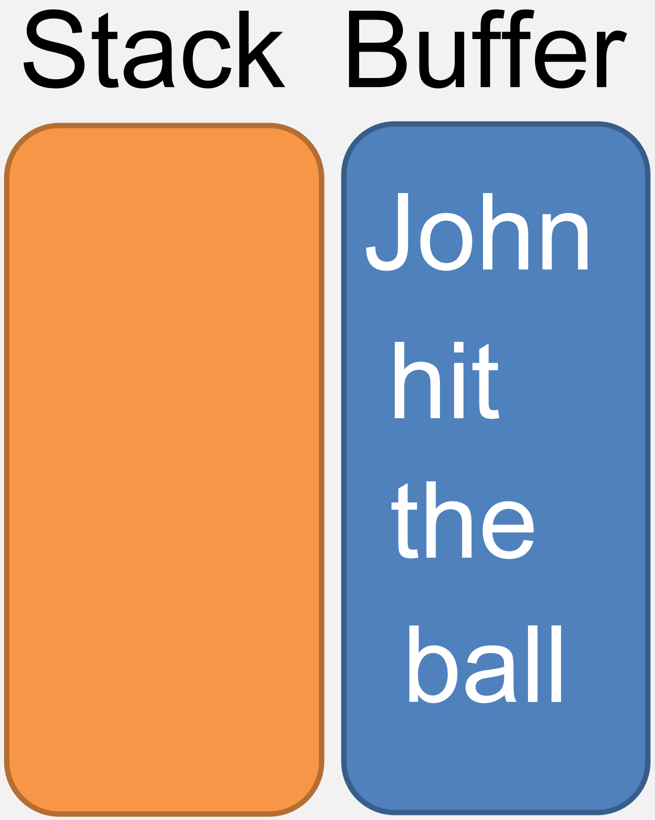
W_{i+1}

W_{i+2}


t=transition, a single
change to a configuration
Sentence
C_i =configuration i
C_T=Final State
C_1 $\xrightarrow{t1}$ C_2 $\xrightarrow{t2}$ .... $\xrightarrow{tn}$ C_T
Oracle
(Classifier)
"predict" the optimal (best)
transition given
the current configuration
trained
Treebank

Example

John hit the ball



Example

Transition=Shift

John hit the ball

Stack Buffer

John

hit

the

ball

Example

Transition=left Arc

Stack Buffer

John

John hit the ball

Subj

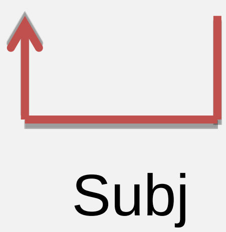
hit
the
ball

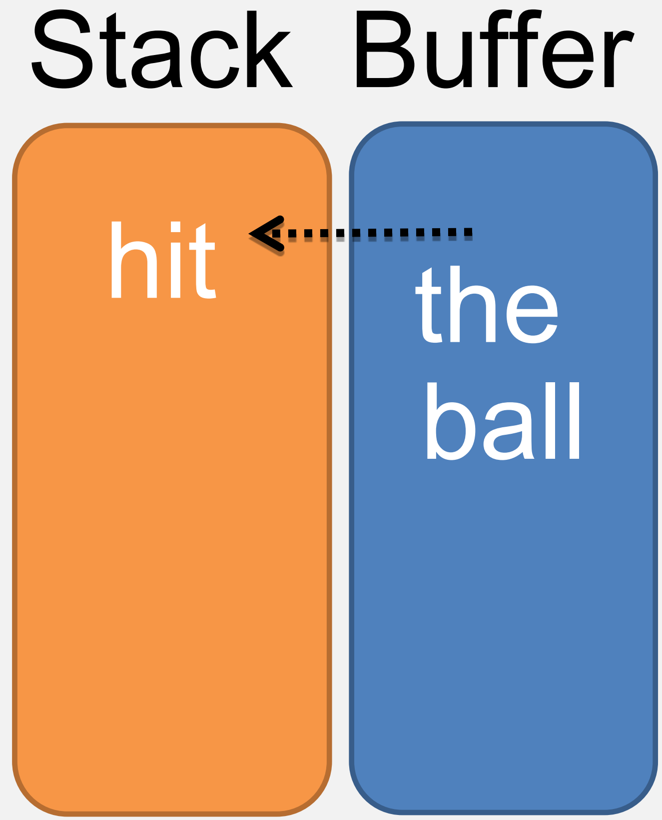
Only if $h(john) = 0$

Example

Transition=Shift

John hit the ball



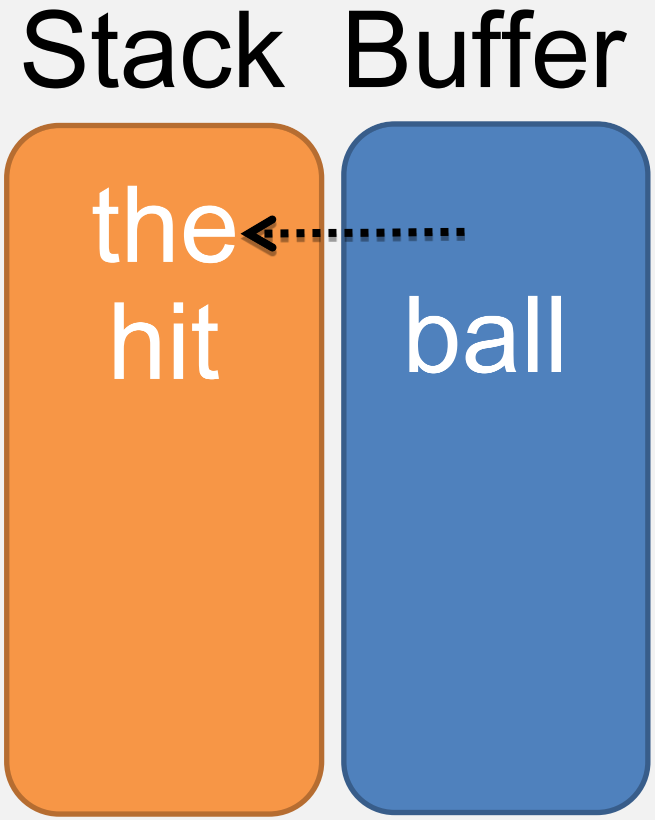


Example

Transition=Shift

John hit the ball

Subj



Example

Transition=left Arc

John hit the ball

Subj

Det

the

Stack

hit

Buffer

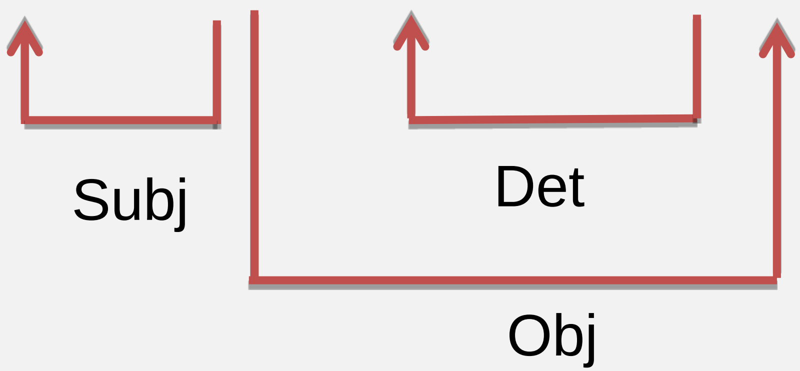
ball

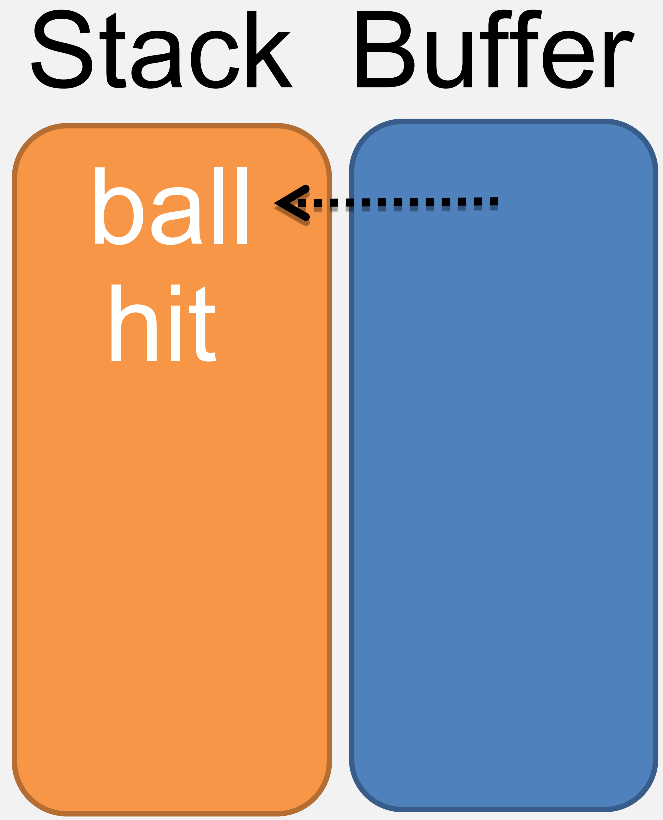
Only if $h(the) = 0$

Example

Transition=Right Arc

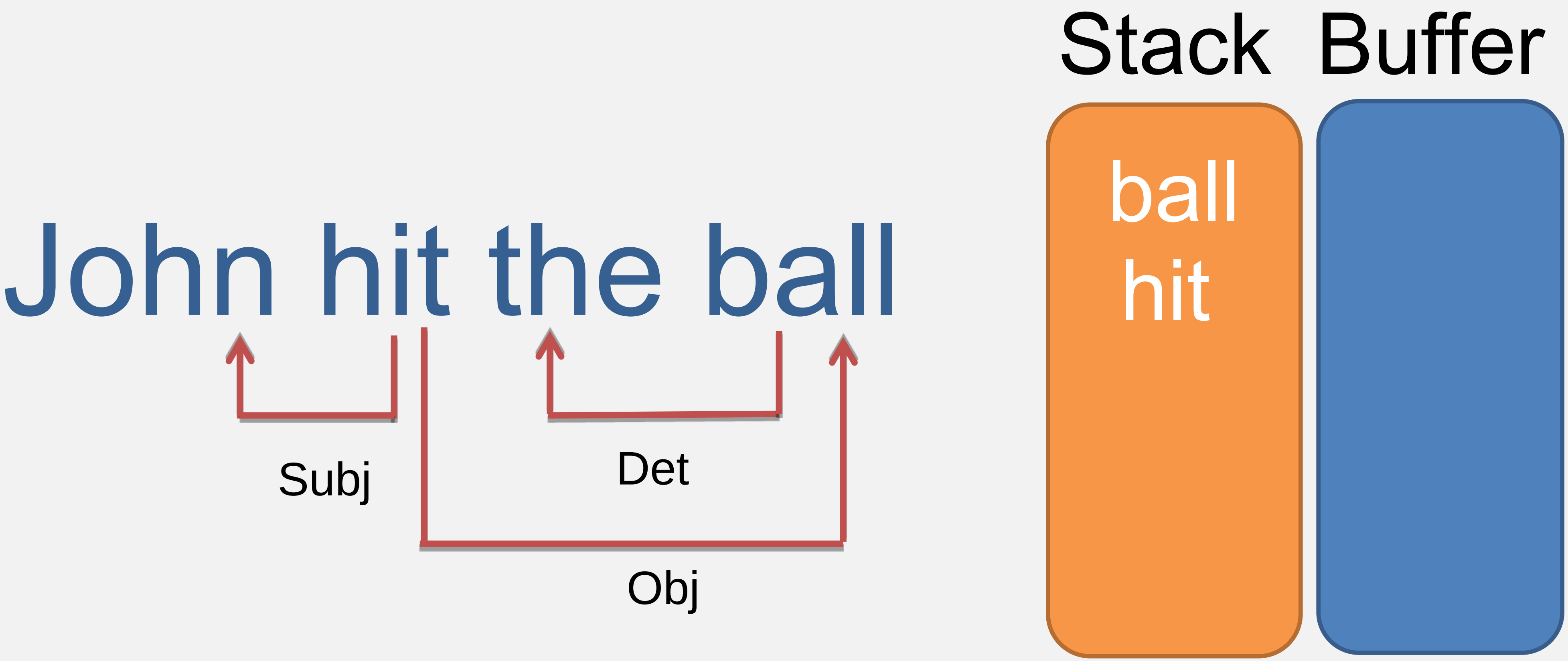
John hit the ball





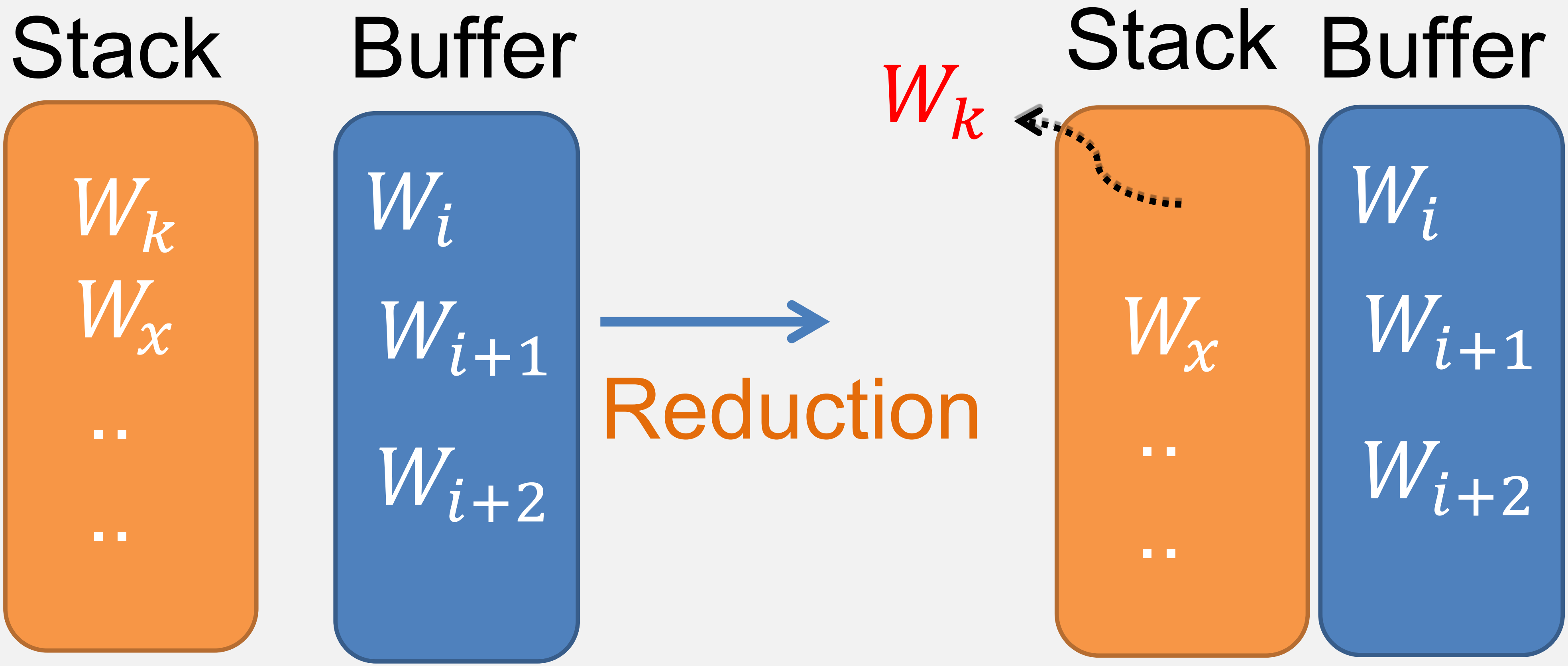
Only if $h(ball) = 0$

Example



Buffer is Empty= Terminal Configuration

Transition Based Parsing



$$Sentence: W_x\ W_k \ldots W_i\ W_{i+1}\ \ldots.$$

Only if $h(W_k) \neq 0$

Oracle

- Greedy Algorithm, choose a local optimal hoping it will lead to the global optimal
- It makes Transition Based Algorithm Deterministic.
  - Originally there might be more than one possible transition from one configuration to another
- Construct the Optimal Transition sequence for the Input Sentence
- How to Build the Oracle? Build a Classifier

Classifier

The Classifier

Transitions

Classes:

- Shift
- Left-arc
- Right-arc
- Reduction

Feature Vector (Features)

- POS of words in the Buffer and Stack
- Words themselves
- The First Word in the Stack
- The L World in the Buffer
- The current arcs in the Graph

Results of the MaltParser

- Evaluation Metrics:

  - **ASU (Unlabeled Attachment Score)**: Proportion of Tokens assigned the correct head

  - **ASL(Labeled Attachment Score):** Proportion of tokens assigned with the correct head and the correct dependency type

Results of the MaltParser

Danish

Dutch

English | Italian | Czech
Chinese | Swedish | Turkish

German

More Inflexible Word order,

'poor' Morphology

More flexible Word order

Rich Morphology

Goal ->

Evaluate if Maltparser can do reasonably accurate parsing for a wide variety of languages

Results of the MaltParser

Language	Asu	Asl
Czech	80.1	72.8
English	88.1	86.3
Italian	82.9	75.7
Chinese	81.1	79.2
Dutch	84.7	79.2
Swedish	86.3	82.0
German	88.1	83.4
Danish	85.6	79.5
Turkish	81.6	69.0

Results of the MaltParser

- Results:
  - Above 80% unlabeled dependency Accuracy (ASU) for all languages
  - morphological richness and word order are the cause of variation across languages

  In General lower accuracy for languages like Czech and Turkish.
    - There are more non-projective structures in those languages
  - It is difficult to do Cross-Language Comparison:
    - Big difference in the amount of annotated data
    - existence of accurate POS Taggers..

  State of the art for Italian, Swedish, Danish, Turkish

Graph Based VS Transition Based

Graph Based

- Search for Optimal Graph (Highest Scoring Graph)
- Globally Trained(Global Optimal)
- Limited History of Parsing Desitions
- Less rich feature representation

Transition Based

- Search for Optimal Graph by finding the best transition between two states. (Local Optimal Desitions)
- Locally Trained (configurations)
- Rich History of Parsing Desitions
- More rich feature but Error Propagation (Greedy Alg.)

Graph Based vs Transition Based

Graph Based (MST)

- Better for Long Dependencies
- More accurate for dependents that are :
  - Verbs
  - Adjectives
  - Adverbs

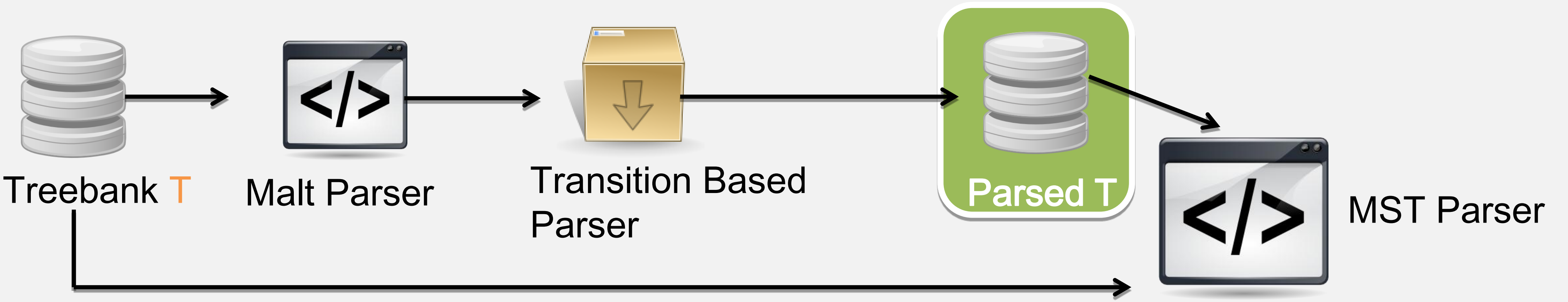
Transition Based(Malt)

- Better for Short dependencies
- More accurate for dependents that are:
  - Nouns
  - Pronouns

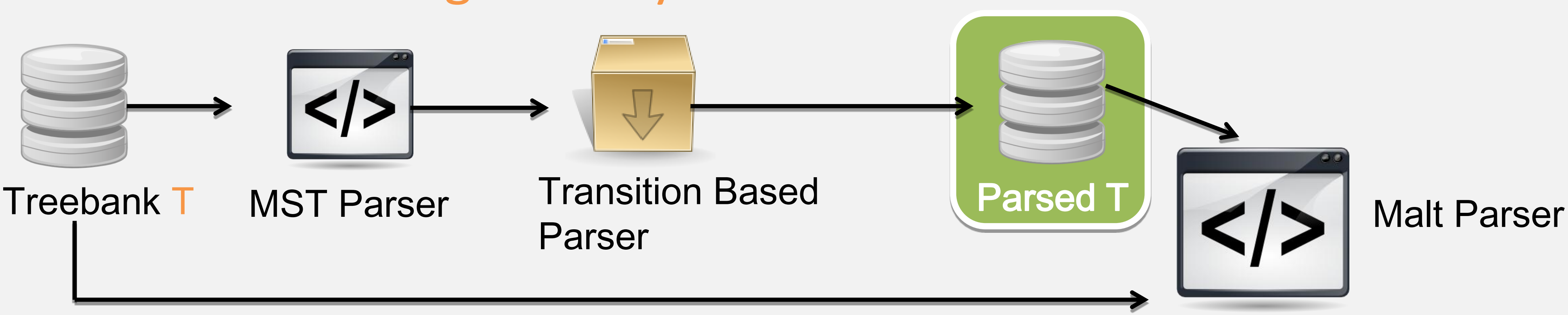
Integrate Both Approaches

Integrating Graph and Transition Based

- Integrate both approaches at learning time.
  - Base MSTParser guided by Malt



- Base MALTParser guided by MLT



Features used in the Integration

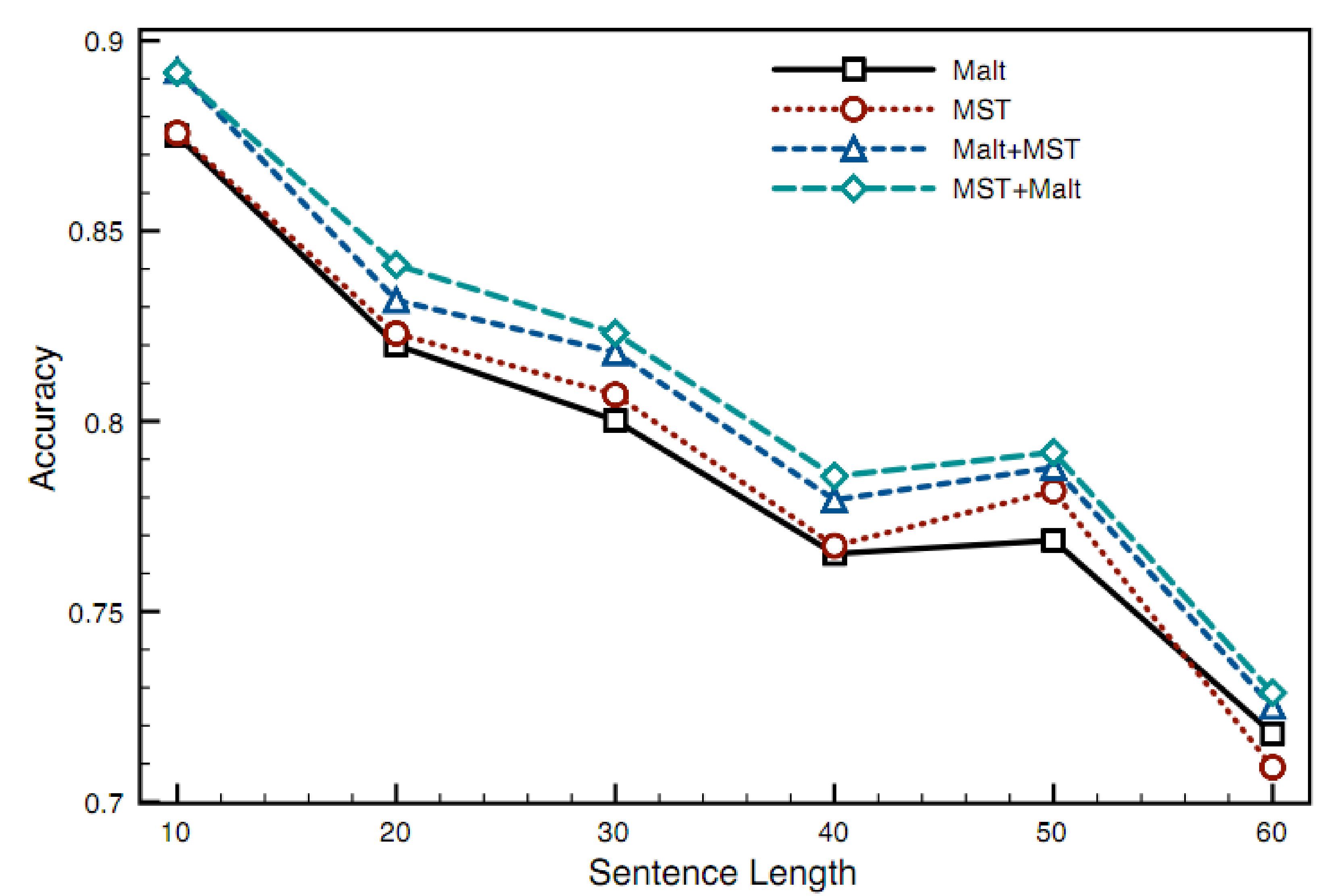
- MSTParser guided by Malt
  - Is arc $(i, j, *)$ in G_{malt}
  - Is arc (i, j, l) in G_{malt}
  - Is arc $(i, j, *) not$ in G_{malt}
  - Identity of l' such that (i, j, l') is in G_{malt}
  - ..

MaltParser guided by MST

- Is arc $(S^0, B^0, *)$ in G_{mst}
- Is arc $(B^0, S^0, *)$ in G_{mst}
- Head direction of B^0 in G_{mst} (left,right,root..)
- Identity of l' such that $(*, B^0, l')$ is in G_{mst}

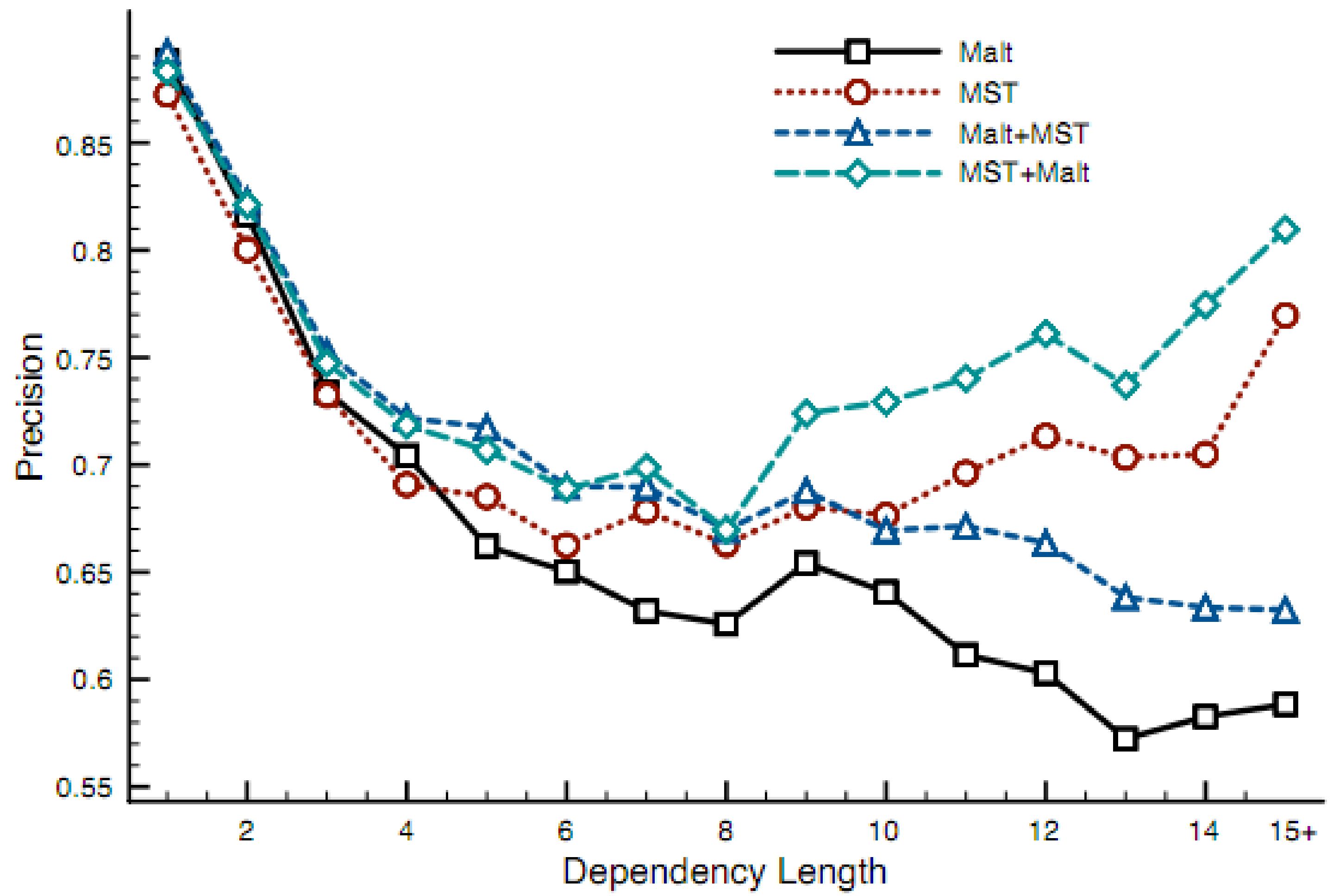
S^0=fist element of the Stack, B^0 =First element of the Buffer

Results of Integration



Asl(Correct head And Correct Label)

Results of Integration



Asl(Correct head And Correct Label)

Results of Integration

Language	MST	MST_{Malt}	Malt	Malt_{MST}
Arabic	66.91	**68.64** (+1.73)	66.71	67.80 (+1.09)
Bulgarian	87.57	**89.05** (+1.48)	87.41	88.59 (+1.18)
Chinese	85.90	**88.43** (+2.53)	86.92	87.44 (+0.52)
Czech	80.18	**82.26** (+2.08)	78.42	81.18 (+2.76)
Danish	84.79	**86.67** (+1.88)	84.77	85.43 (+0.66)
Dutch	79.19	**81.63** (+2.44)	78.59	79.91 (+1.32)
German	87.34	**88.46** (+1.12)	85.82	87.66 (+1.84)
Japanese	90.71	91.43 (+0.72)	91.65	**92.20** (+0.55)
Portuguese	86.82	87.50 (+0.68)	87.60	**88.64** (+1.04)
Slovene	73.44	**75.94** (+2.50)	70.30	74.24 (+3.94)
Spanish	82.25	**83.99** (+1.74)	81.29	82.41 (+1.12)
Swedish	82.55	**84.66** (+2.11)	84.58	84.31 (–0.27)
Turkish	63.19	64.29 (+1.10)	65.58	**66.28** (+0.70)
Average	80.83	**82.53** (+1.70)	80.74	82.01 (+1.27)

Asl(Correct head And Correct Label)

Results of Integration

- Graph-based models predict better long arcs
- Each model learn streghts from the others
- The integration actually improves accuracy
- Trying to do more chaining of systems do not gain better accuracy

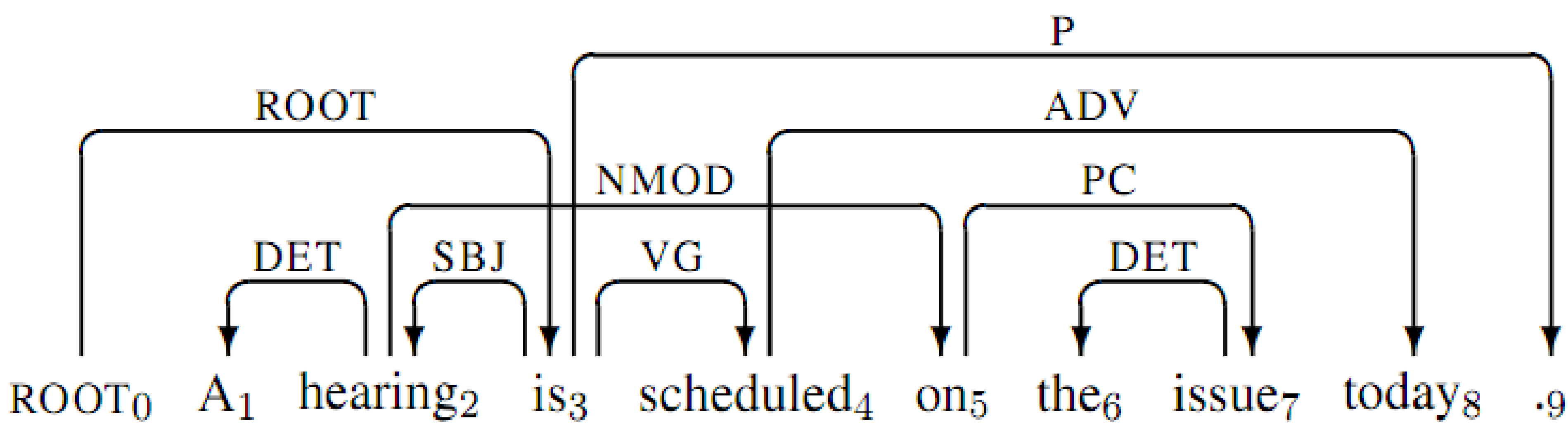
Non-Projectivity

- Some Sentences have long distance dependencies which cannot be parsed with this algorithm
  - Cause it only consider relations between neighbors words
- 25% or more of the sentences in some languages are non-projective
- Useful for some languages with less constraints on word order
- Harder Problem, There could be relations over unbounded distances.

Non-Projectivity

A dependency Tree T is Projective:

if for every $Arc\ (W_i, W_j, rel)$ there is a path from W_i to W_k , if W_k is between W_i and W_j



From 'Scheduled' W_2 there is an arc to W_5 however there is no way to get to W_4, W_3 from W_2

Non-Projectivity

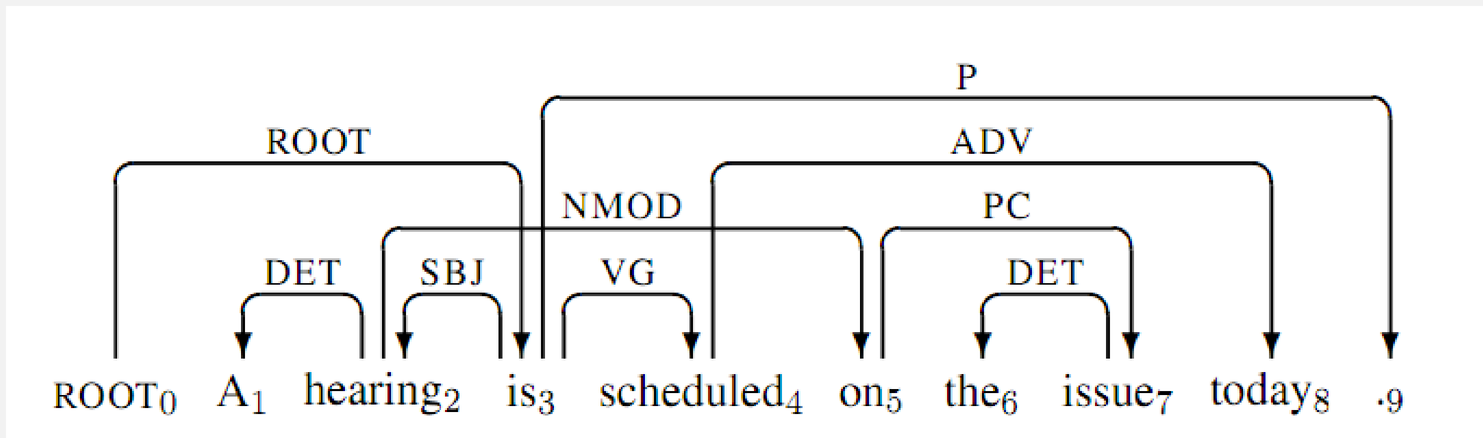
- Why the previous transition algorithm would not be able to generate this tree?

Stack

is
hearing

Buffer

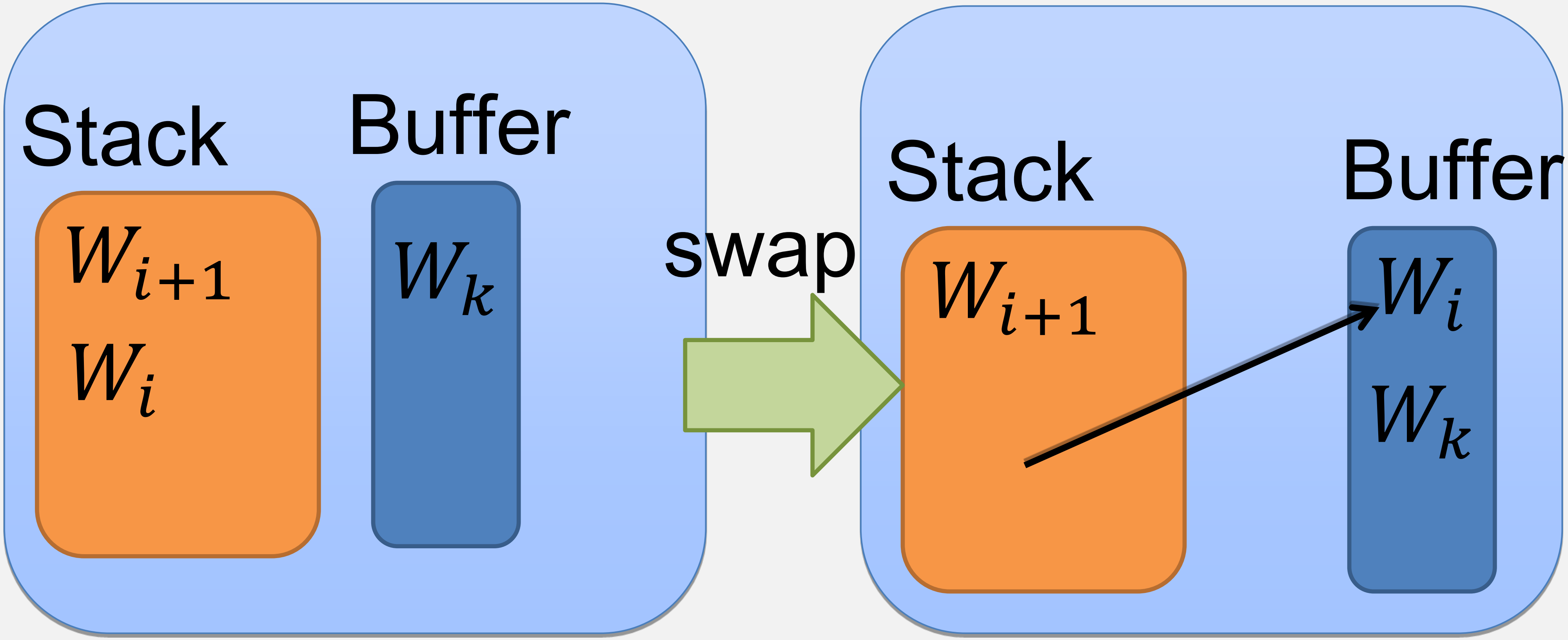
On
…
…



'is' can never be reduced
'hearing' and 'on' will never get an arc

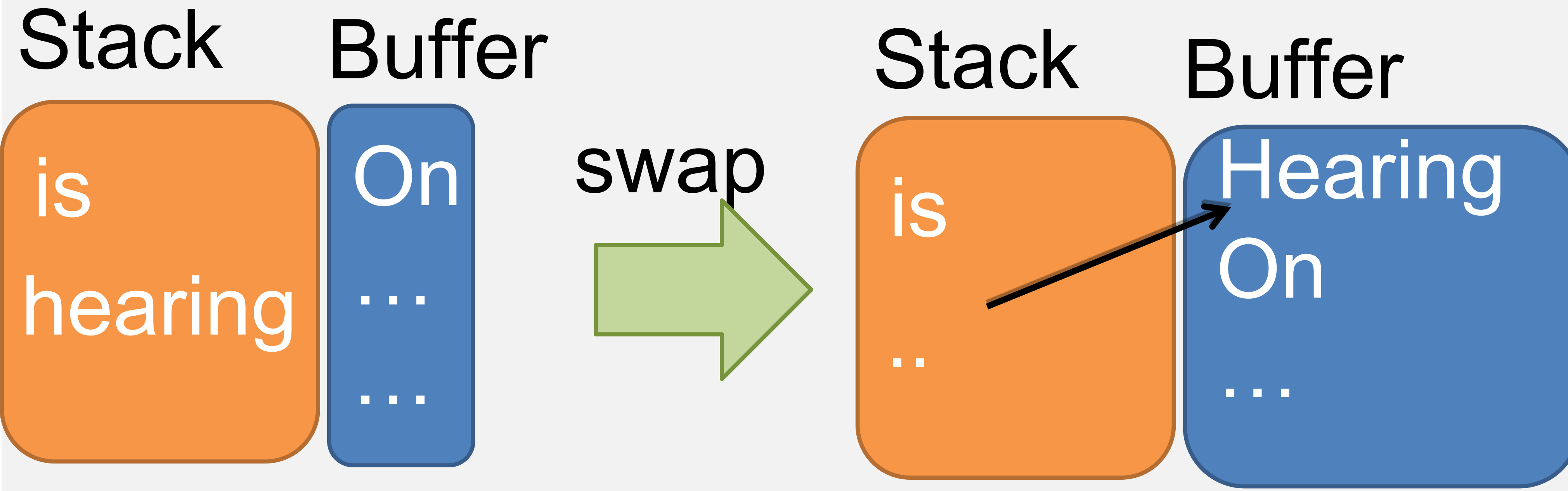
Handling Non-Projectivity

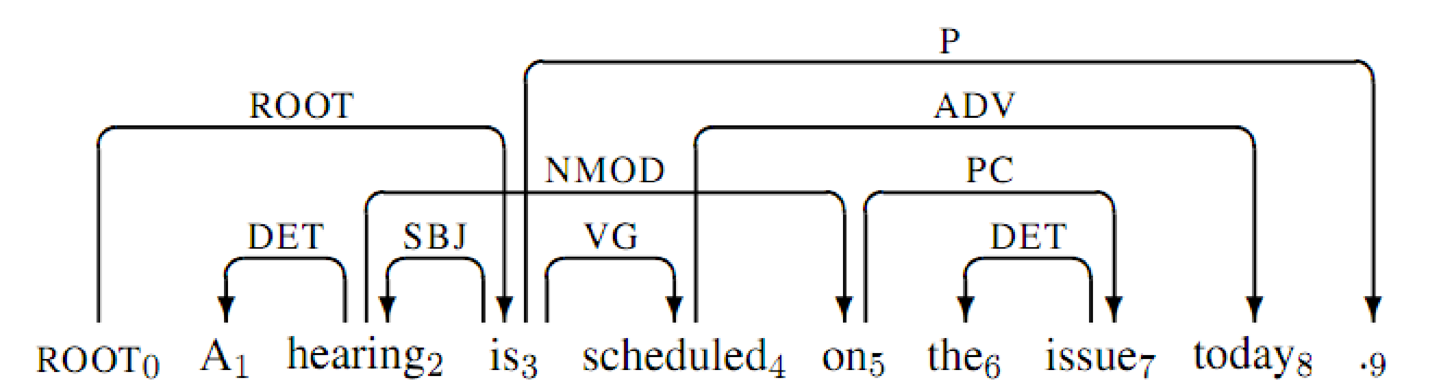
- Add a new Transition – ”Swap”



- Re-Order the initial Input Sentance

Non-Projectivity





Non-Projective Dependency Parsing

- Useful for some languages with less constraints on word order

---

Theoretically

- Best case $O(N)$, , that is: no swaps
- Worst Case $O(N^2)$,

Results
Non-Projective Dependency Parsing

Running Time

- Test on 5 languages(Danish, Arabic, Czech, Slovene, Turkish)
- In practice the running time is $O(N)$.

Parsing Accuracy

- Criteria
  - **Attachment Score:** Percentage of tokens with correct head and dependency label
  - **Exact match:** completely correct labeled dependency tree

Results
Non-Projective Dependency Parsing

- Systems Compared
  - S_u **= allowing Non Projective**
  - S_p **=Just Projective**
  - S_{pp} **=Handling non-Projectivity as a pos-processing**
  - **AS:** Percentage of tokens with correct head and dependency label
  - **EM:** completely correct labeled dependency tree

System	Arabic		Czech		Danish		Slovene		Turkish	
	AS	EM	AS	EM	AS	EM	AS	EM	AS	EM
S_u	67.1 (9.1)	**11.6**	**82.4** (**73.8**)	**35.3**	84.2 (22.5)	26.7	75.2 (23.0)	**29.9**	64.9 (**11.8**)	**21.5**
S_p	67.3 (**18.2**)	**11.6**	80.9 (3.7)	31.2	84.6 (0.0)	27.0	74.2 (3.4)	**29.9**	65.3 (6.6)	21.0
S_{pp}	67.2 (**18.2**)	**11.6**	82.1 (60.7)	34.0	84.7 (22.5)	28.9	74.8 (20.7)	26.9	65.5 (**11.8**)	20.7
Malt-06	66.7 (**18.2**)	11.0	78.4 (57.9)	27.4	84.8 (27.5)	26.7	70.3 (20.7)	19.7	65.7 (9.2)	19.3
MST-06	66.9 (0.0)	10.3	80.2 (61.7)	29.9	84.8 (**62.5**)	25.5	73.4 (26.4)	20.9	63.2 (**11.8**)	20.2
MST_{Malt}	**68.6** (9.4)	11.0	82.3 (69.2)	31.2	**86.7** (60.0)	**29.8**	**75.9** (**27.6**)	26.6	**66.3** (9.2)	18.6

Results
Non-Projective Dependency Parsing

- **AS**
  - Performance of S_u is better for for:
    - Czech and Slovene → more non-porjective arcs in this languages.
  - In AS S_u is lower than S_p, however the drop is not really significant
  - For Arabic the results are not meaningful since there are only 11 non-projective arcs in the whole set
- **ME**
  - S_u outperforms all other parsers.
  - The positive effect of S_u is dependent on the non-projectivity arcs in the language


happy
holidays

References

- Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: a language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41, 2007.
- Joakim Nivre and Ryan McDonald. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June 2008.
- Joakim Nivre. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, 2009.
- Sandra Kübler, Ryan McDonald, Joakim Nivre. Dependency Parsing, Morgan & Claypool Publishers, 2009