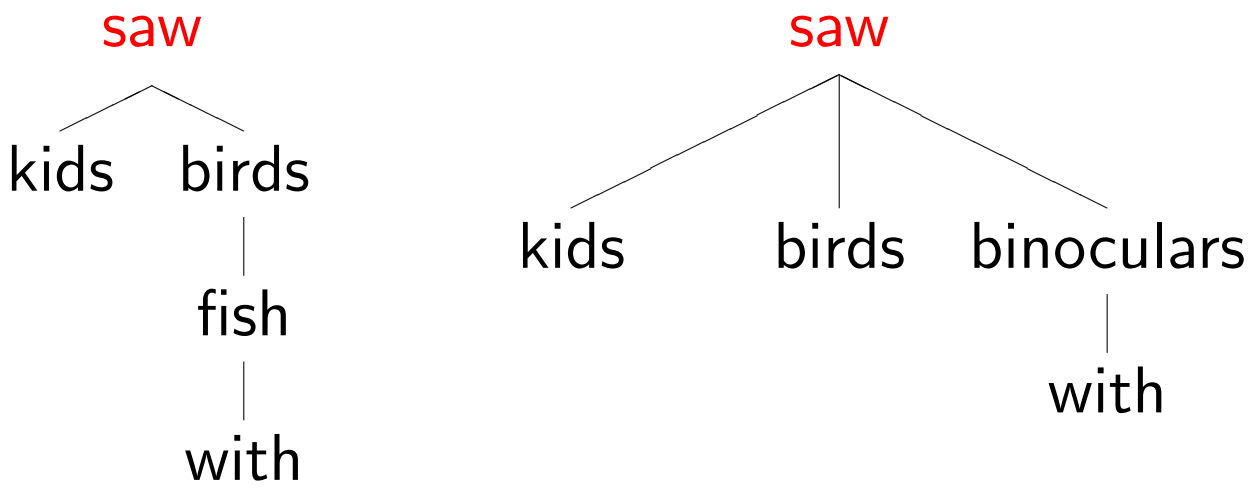
Dependency Parses/Parsing

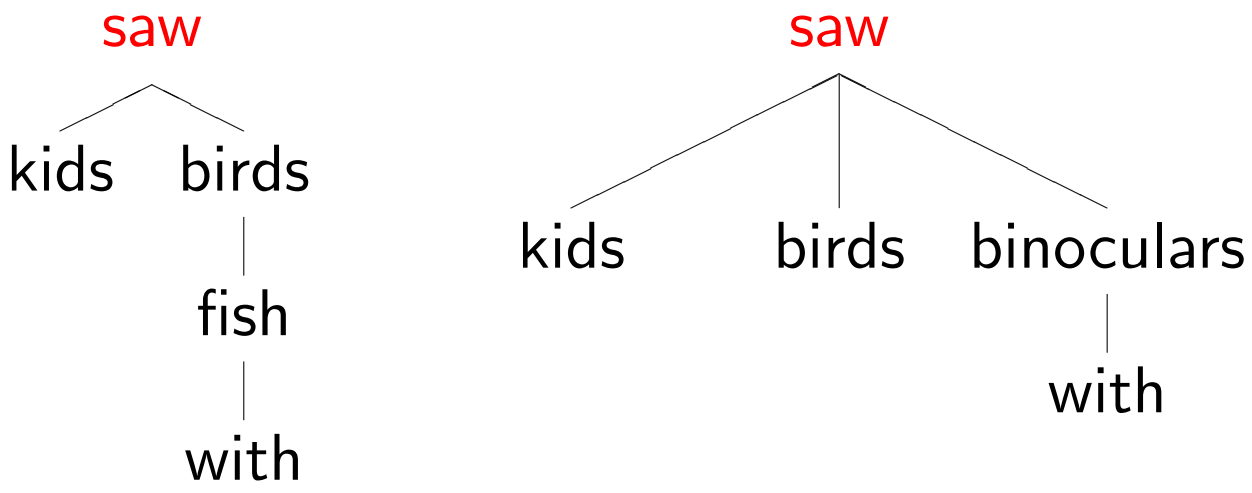
Dependency Parse



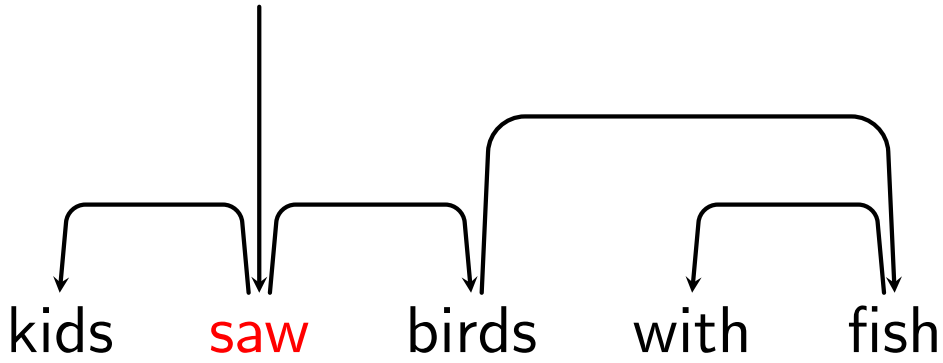
Linguists have long observed that the meanings of words within a sentence depend on one another, mostly in *asymmetric*, *binary* relations.

- Though some constructions don't cleanly fit this pattern: e.g., coordination, relative clauses.

Dependency Parse



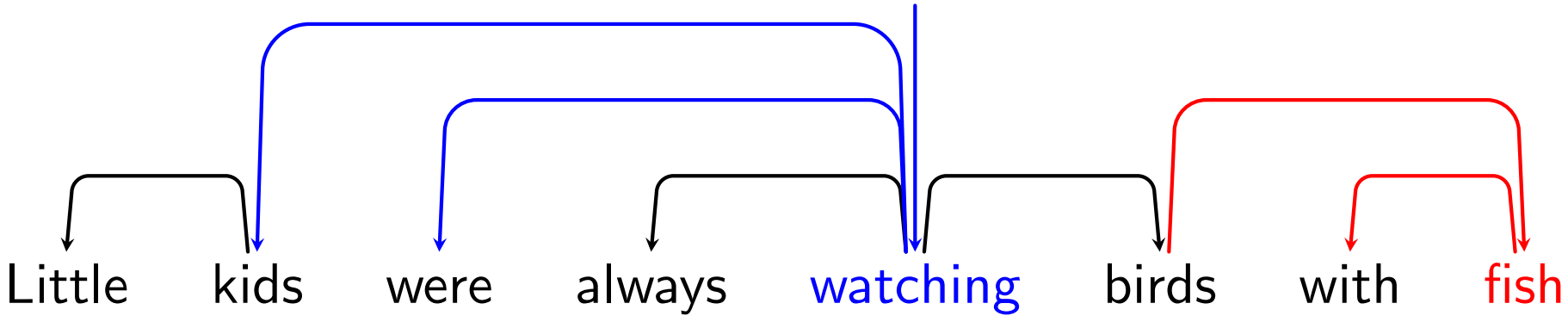
Equivalently, but showing word order (head → modifier):



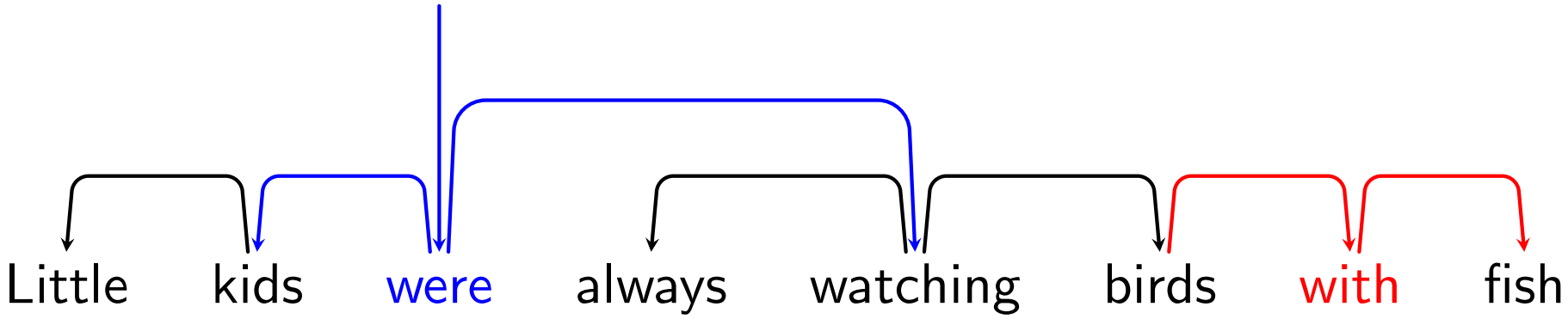
Because it is a tree, every word has exactly one parent.

Content vs. Functional Heads

Some treebanks prefer **content heads**:

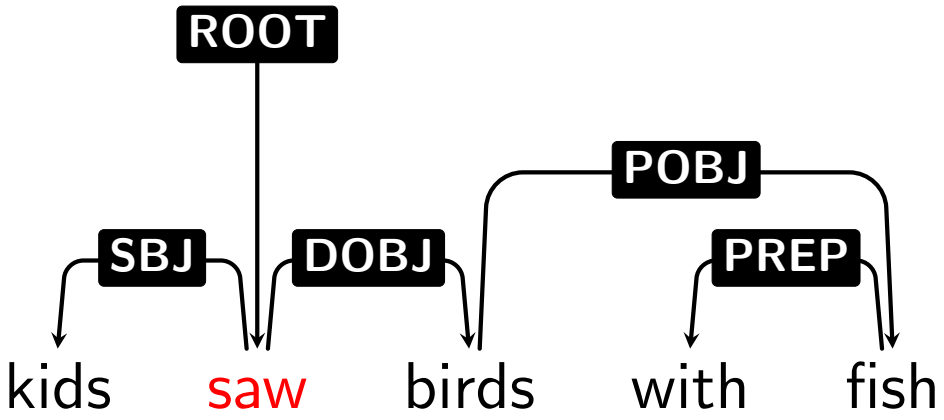


Others prefer **functional heads**:



Edge Labels

It is often useful to distinguish different kinds of head → modifier **relations**, by labeling edges:

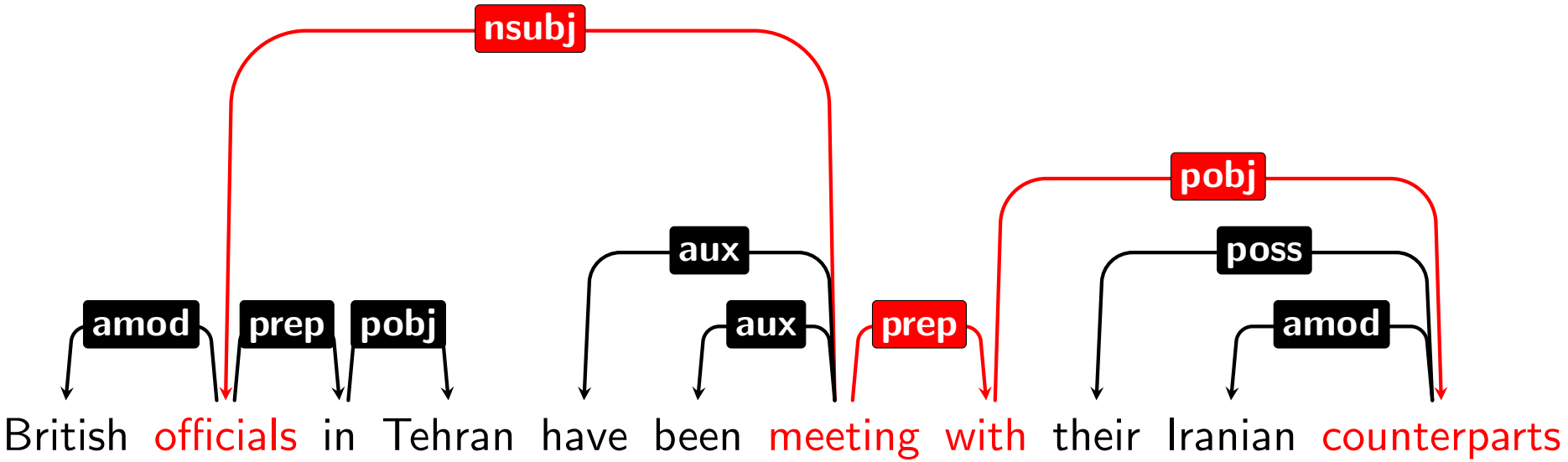


Important relations for English include subject, direct object, determiner, adjective modifier, adverbial modifier, etc. (Different treebanks use somewhat different label sets.)

- How would you identify the subject in a constituency parse?

Dependency Paths

For **information extraction** tasks involving real-world relationships between entities, chains of dependencies can provide good features:

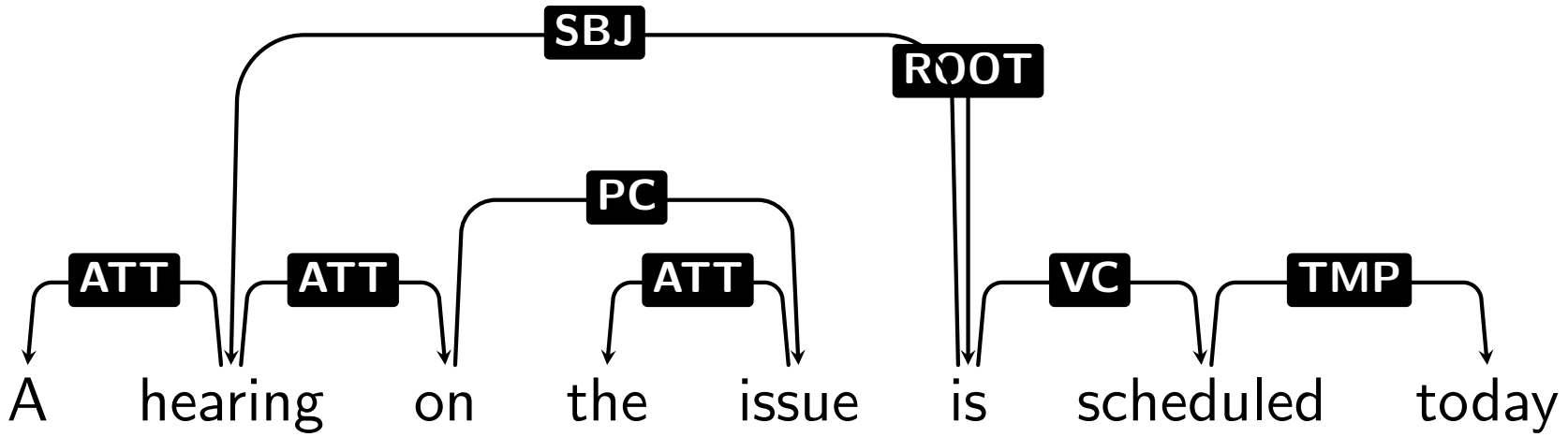


(example from Brendan O'Connor)

Projectivity

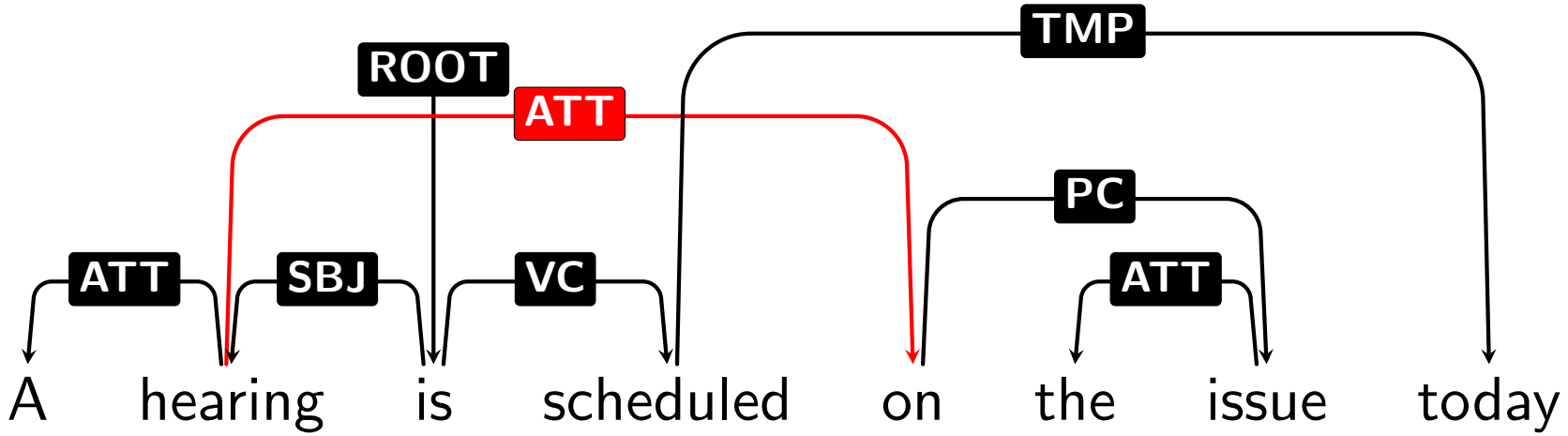
- A sentence's dependency parse is said to be **projective** if every subtree (node and all its descendants) occupies a *contiguous span* of the sentence.

- = The dependency parse can be drawn on top of the sentence without any crossing edges.



Nonprojectivity

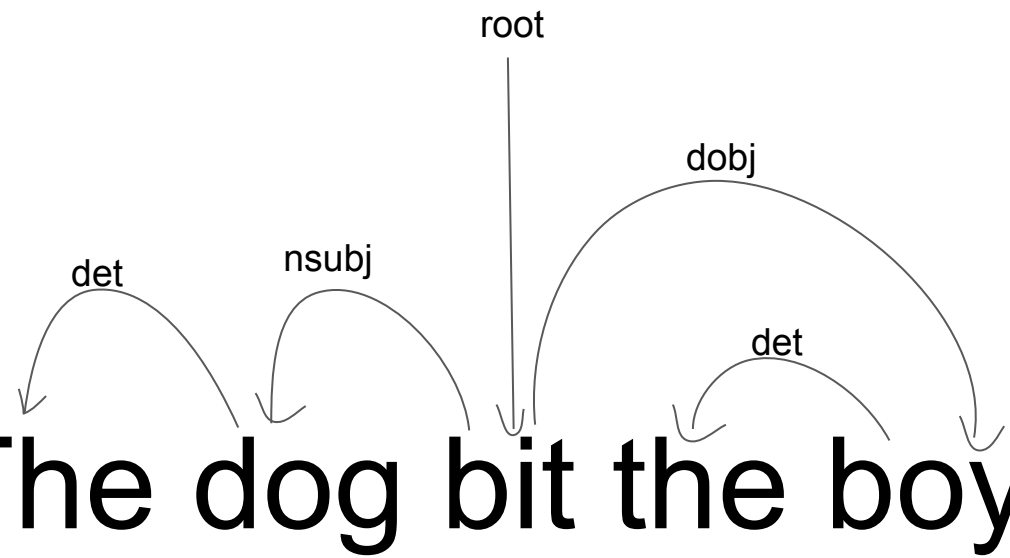
- Other sentences are **nonprojective**:



- Nonprojectivity is rare in English, but quite common in many languages.

A quick dependency parse:

The dog bit the boy


root
dobj
nsubj
det
det
The dog bit the boy

Why is this useful?

Why is this useful?

- Conveys some level of semantic meaning
- Good for languages with freer word order

Transition Based Dependency Parsing

- High level idea
  - Process words from left to right

Transition Based Dependency Parsing

- High level idea
  - Process words from left to right
  - At each stage, decide if two words should be attached

Transition Based Dependency Parsing

- Similar to shift-reduce parsing for programming languages
- 3 components
  - Input buffer (the words of the sentence)
  - Stack (where the words are moved to and manipulated)
  - Dependency relations (the list of relations between words that becomes the dependency parse)
- **Configuration**: some state of the 3 components
- Parsing consists of a sequence of transitions between configurations until all the words have been accounted for
  - The available transitions define the type of approach

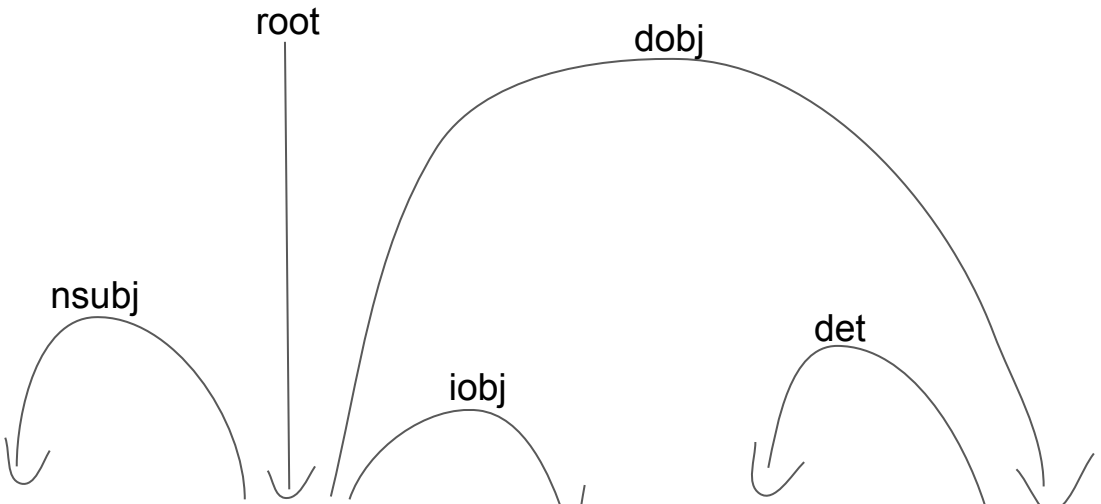
The Arc-Standard Approach

- **LEFTARC**: Assert a head-dependent relation between the word at the top of the stack and the word directly beneath it; remove the lower word from the stack
- **RIGHTARC**: Assert a head-dependent relation between the second word on the stack and the word at the top; remove the word at the top of the stack
- **SHIFT**: Remove the word from the front of the input buffer and push it onto the stack

Restrictions

- LEFTARC cannot be applied when the root is the second element of the stack (the root cannot be a dependent)
- LEFTARC & RIGHTARC can only be applied if there are 2 or more elements on the stack.

She gave me the book



She gave me the book

STACK
[root]

WORD LIST
[She, gave, me, the, book]

RELATIONS

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	

Operation: SHIFT

STACK

[root]

[root, She]

[root, She, gave]

WORD LIST

[She, gave, me, the, book]

[gave, me, the, book]

[me, the, book]

RELATIONS

Operation: SHIFT

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	(She ← gave)
[root, gave]	[me, the, book]	

Operation: LEFTARC

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	(She ← gave)
[root, gave]	[me, the, book]	(root → gave)
[root]	[me, the, book]	

Operation: RIGHTARC?

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	(She ← gave)
[root, gave]	[me, the, book]	
[root, gave, me]	[the, book]	

Operation: SHIFT!

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	(She ← gave)
[root, gave]	[me, the, book]	
[root, gave, me]	[the, book]	(gave → me)
[root, gave]	[the, book]	

Operation: RIGHTARC

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	(She ← gave)
[root, gave]	[me, the, book]	
[root, gave, me]	[the, book]	(gave → me)
[root, gave]	[the, book]	
[root, gave, the]	[book]	

Operation: SHIFT

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	(She ← gave)
[root, gave]	[me, the, book]	
[root, gave, me]	[the, book]	(gave → me)
[root, gave]	[the, book]	
[root, gave, the]	[book]	
[root, gave, the, book]	[]	

Operation: SHIFT

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	
[root, gave]	[me, the, book]	(She ← gave)
[root, gave, me]	[the, book]	
[root, gave]	[the, book]	(gave → me)
[root, gave, the]	[book]	
[root, gave, the, book]	[]	
[root, gave, book]	[]	(the ← book)

Operation: LEFTARC

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	
[root, gave]	[me, the, book]	(She ← gave)
[root, gave, me]	[the, book]	
[root, gave]	[the, book]	(gave → me)
[root, gave, the]	[book]	
[root, gave, the, book]	[]	
[root, gave, book]	[]	(the ← book)
[root, gave]	[]	(gave → book)

Operation: RIGHTARC

STACK	WORD LIST	RELATIONS
[root]	[She, gave, me, the, book]	
[root, She]	[gave, me, the, book]	
[root, She, gave]	[me, the, book]	
[root, gave]	[me, the, book]	(She ← gave)
[root, gave, me]	[the, book]	
[root, gave]	[the, book]	(gave → me)
[root, gave, the]	[book]	
[root, gave, the, book]	[]	
[root, gave, book]	[]	(the ← book)
[root, gave]	[]	(gave → book)
[root]	[]	(root → gave)

Operation: RIGHTARC

Run time

Run time

- Linear in the size of the sentence

Run time

- Linear in the size of the sentence
- A head decision for each word uniquely defines a tree

How to decide what to do at each step?

How to decide what to do at each step?

- Build an oracle

How to decide what to do at each step?

- Build an oracle with machine learning!
- Need something that maps configurations to transitions

How to decide what to do at each step?

- Build an oracle with machine learning!
- Need something that maps configurations to transitions
- Data comes from Treebanks

How to decide what to do at each step?

- Build an oracle with machine learning!
- Need something that maps configurations to transitions
- Data comes from Treebanks
  - Corpora annotated with gold trees
  - http://universaldependencies.org/

How to decide what to do at each step?

- Build an oracle with machine learning!
- Need something that maps configurations to transitions
- Data comes from Treebanks
  - Corpora annotated with gold trees
  - http://universaldependencies.org/
- Best results have historically come from multinomial logistic regression and SVM models.

How to decide what to do at each step?

- Build an oracle with machine learning!
- Need something that maps configurations to transitions
- Data comes from Treebanks
  - Corpora annotated with gold trees
  - http://universaldependencies.org/
- Best results have historically come from multinomial logistic regression and SVM models.
- Recently, Neural Networks have been performing well

How to decide what to do at each step?

- Build an oracle with machine learning!
- Need something that maps configurations to transitions
- Data comes from Treebanks
  - Corpora annotated with gold trees
  - http://universaldependencies.org/
- Best results have historically come from multinomial logistic regression and SVM models.
- Recently, Neural Networks have been performing well.
  - Naturally lend themselves to the task
    - Forms analysis before reading in the whole sentence
    - Neural networks model a sequence of decisions, which is exactly how the parsing operates

Possible features?

Possible features?

- Some obvious ones, the word currently at the top of the stack, etc.

Possible features?

- Some obvious ones, the word currently at the top of the stack, etc.
- POS tags are also very useful

Possible features?

- Some obvious ones, the word currently at the top of the stack, etc.
- POS tags are also very useful
  - Usually a POS tagged is run and used as input to the dependency parser

Edge Labels

- The example only dealt with connections
- Can modify the oracle to learn and output the transition, as well as the arc label at each step (if RIGHTARC or LEFTARC is called)

Possible Weaknesses?

Possible Weaknesses?

- Can only produce projective parses

Weaknesses of Dependency Parses

Weakness of Dependency Parses

- Head-modifier relation doesn’t always work neatly
- Coordination
  - “Cats **and** dogs ran.”
- Auxiliaries
  - “**Do** you want coffee?”
- Relative clauses
  - “I met the girl **who** started this year”
- Prepositional phrases:
  - “I saw a cow **in** the barn”

Advanced Methods

- Arc Eager transition system

Advanced Methods

- Arc Eager transition system
  - We couldn’t add the arc between root and gave because gave still needed to point to other words
  - In general, the longer a word has to wait to get assigned its head the more opportunities there are for something to go awry

Advanced Methods

- Arc Eager transition system
  - We couldn’t add the arc between root and gave because gave still needed to point to other words
  - In general, the longer a word has to wait to get assigned its head the more opportunities there are for something to go awry
  - Solution: Change the set of operators

New Operators

- LEFTARC: Assert a head-dependent relation between the word at the front of the input buffer and the word at the top of the stack; pop the stack.
- RIGHTARC: Assert a head-dependent relation between the word on the top of the stack and the word at the front of the input buffer; shift the word at the front of the input buffer to the stack.
- SHIFT: Remove the word from the front of the input buffer and push it onto the stack (stays the same).
- REDUCE: Pop the stack.

Advanced Methods

- Arc Eager transition system
  - We couldn’t add the arc between root and gave because gave still needed to point to other words
  - In general, the longer a word has to wait to get assigned its head the more opportunities there are for something to go awry
- Graph based methods
  - Can think of dependency parses as a directed graph with arc labels
  - Other methods use graph based algorithms to find the best dependency parse