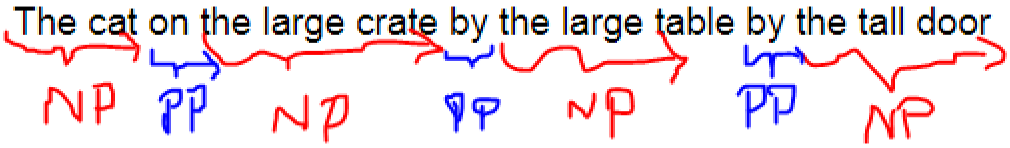
Linguistic structures:

1.Phrase structure Grammar:

This method includes interpreting sentences as a collection of phrases.
Example:
There are different phrase structures for different languages.
In english:

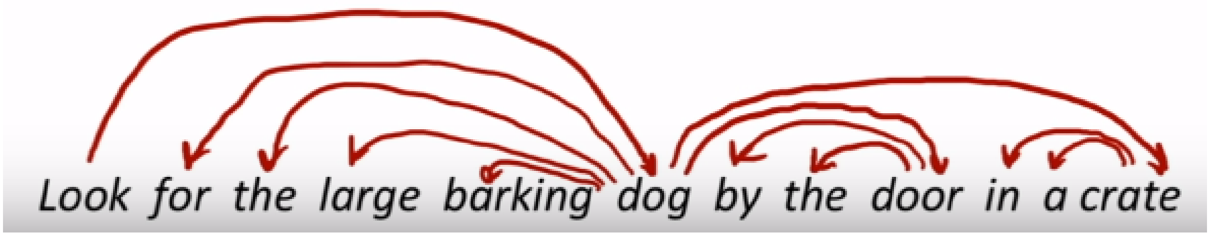
1. Noun phrase: NP: Det. - (Adj) - N - PP
2. Preposition phrase: PP: Pre - NP
3. Verb phrase: VP: V - NP
4.

Where: (Adj) means optional adjective,
Det are determiners (The, That)
Prepositions (on, over, under, etc.)


The cat on the large crate by the large table by the tall door
NP
PP
NP
PP
NP
PP
NP

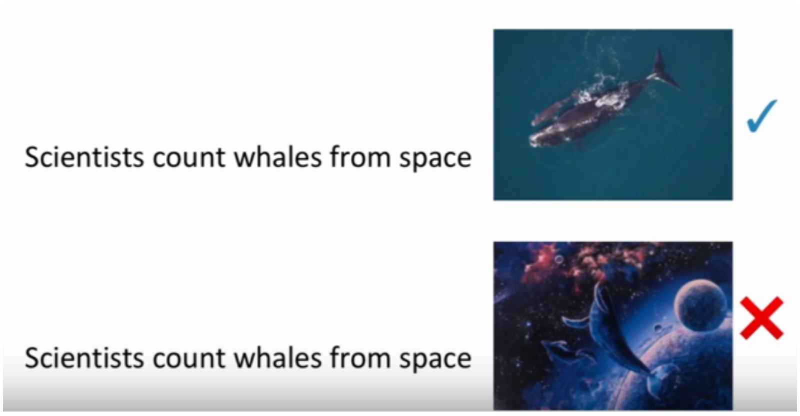
2. Dependency structure:

This deals with how words depend on (are either modifiers or dependencies) of other words.



Here look is the root word and all the other words are its modifiers or modifiers of modifiers.

The ambiguity attached with PP:

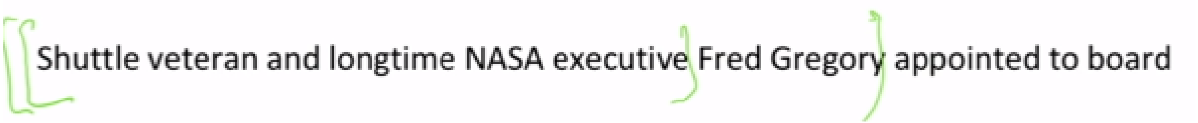


Scientists count whales from space ✓

Scientists count whales from space ✗

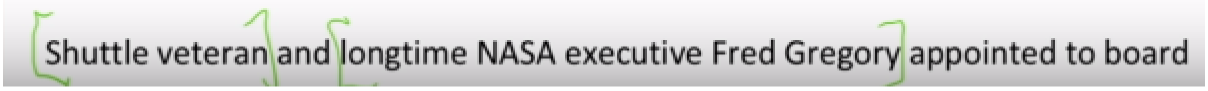
This type of ambiguity where we don't know what a modifier is modifying increases exponentially with the length of the sentence.

Ambiguity with coordination scope:



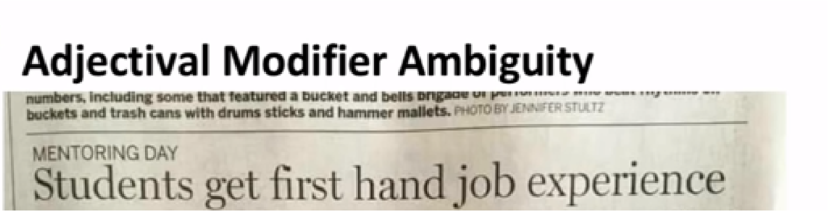
[[Shuttle veteran and longtime NASA executive] Fred Gregory] appointed to board

Or,



[Shuttle veteran] and [longtime NASA executive Fred Gregory] appointed to board

Lol

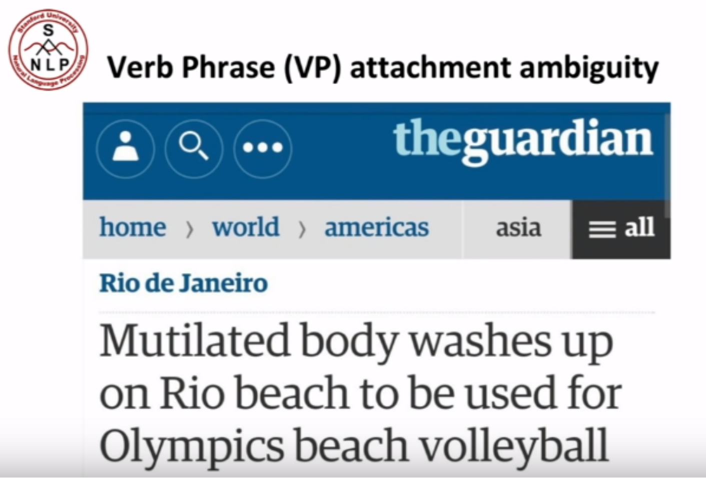


Adjectival Modifier Ambiguity

numbers, including some that featured a bucket and bells brigade of performers who beat rhythms on buckets and trash cans with drums sticks and hammer mallets. PHOTO BY JENNIFER STULTZ

MENTORING DAY

Students get first hand job experience



Verb Phrase (VP) attachment ambiguity

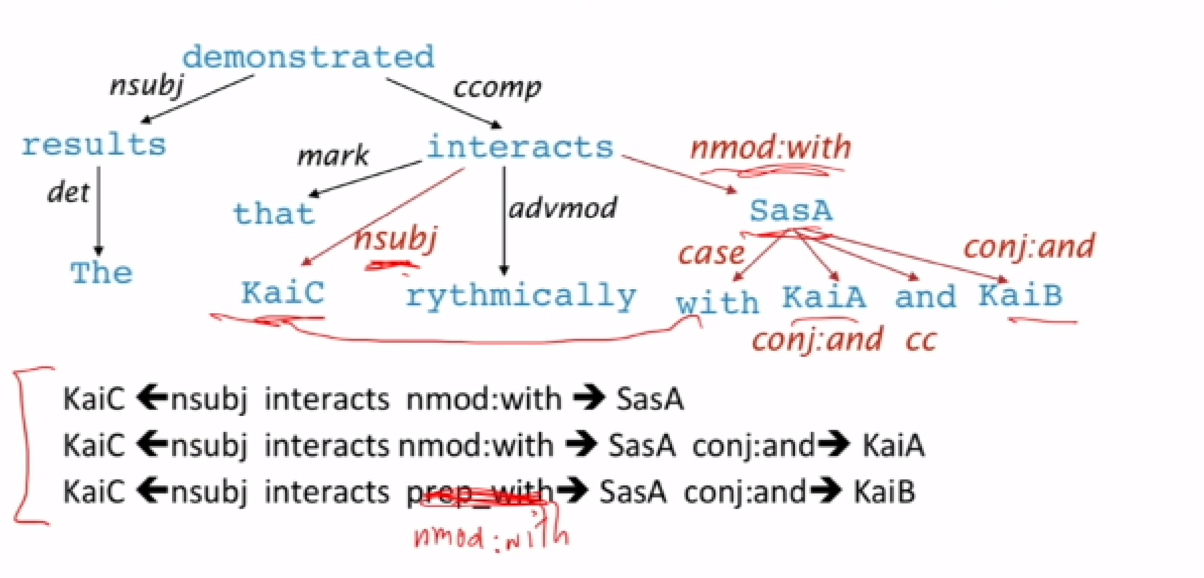
theguardian

home › world › americas asia ≡ all

Rio de Janeiro

Mutilated body washes up on Rio beach to be used for Olympics beach volleyball

Dependency paths



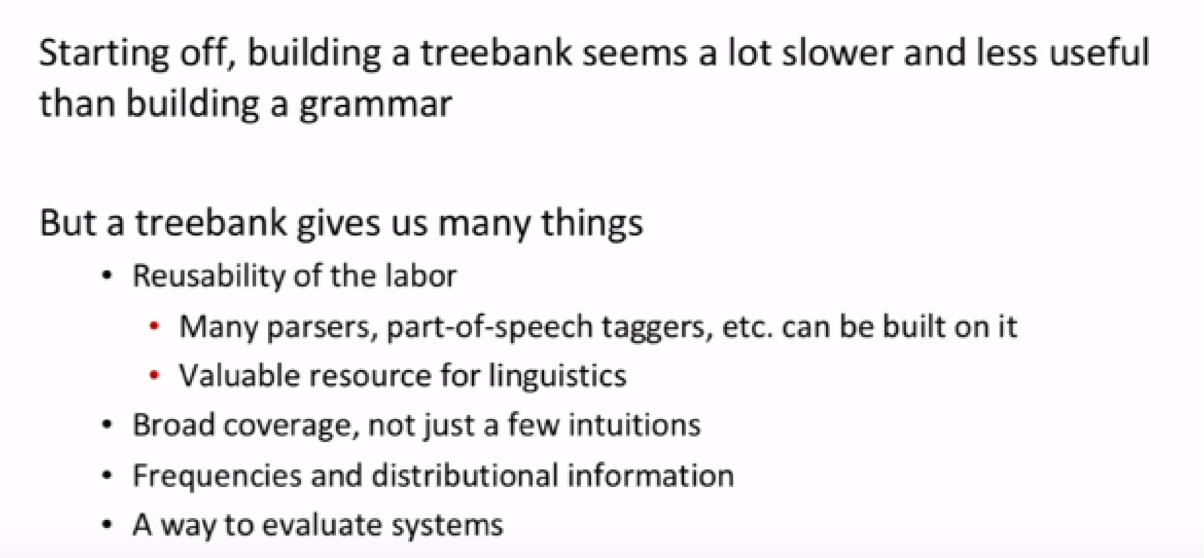
KaiC ←nsubj interacts nmod:with → SasA

KaiC ←nsubj interacts nmod:with → SasA conj:and→ KaiA

KaiC ←nsubj interacts ~~prep_with~~→ SasA conj:and→ KaiB

nmod:with

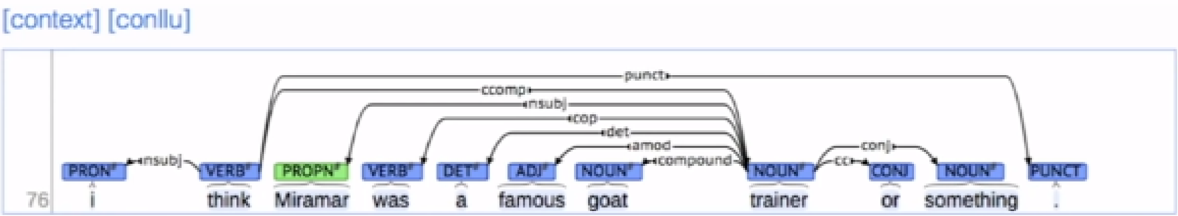
Annotated data:



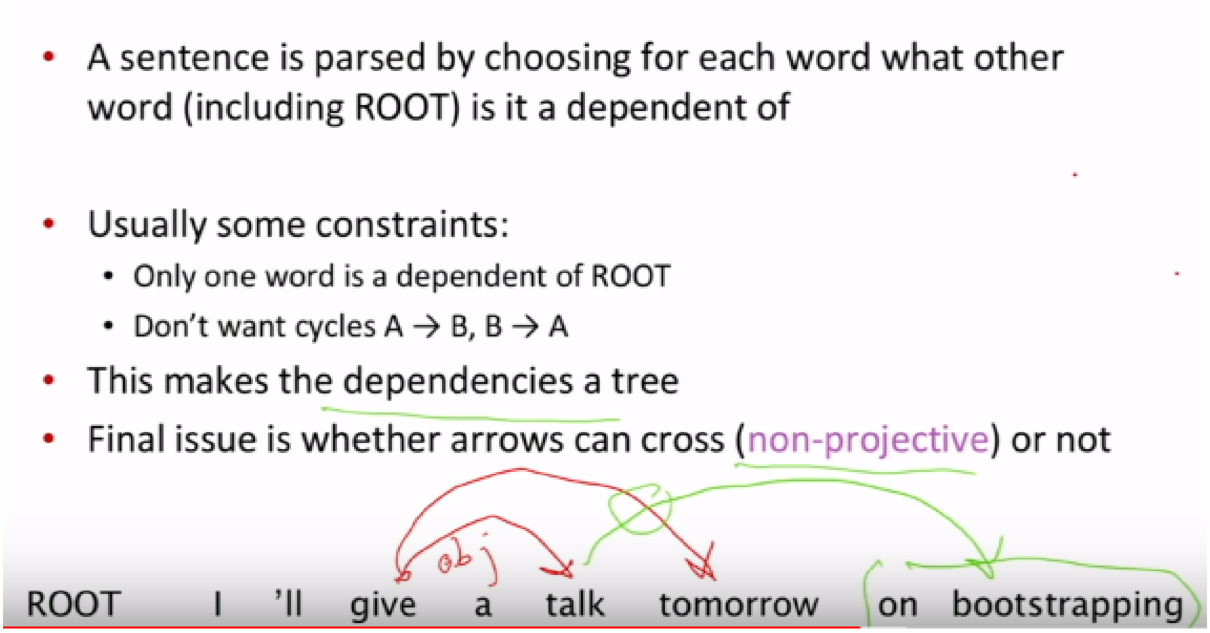
Starting off, building a treebank seems a lot slower and less useful than building a grammar

But a treebank gives us many things

- Reusability of the labor
  - Many parsers, part-of-speech taggers, etc. can be built on it
  - Valuable resource for linguistics
- Broad coverage, not just a few intuitions
- Frequencies and distributional information
- A way to evaluate systems



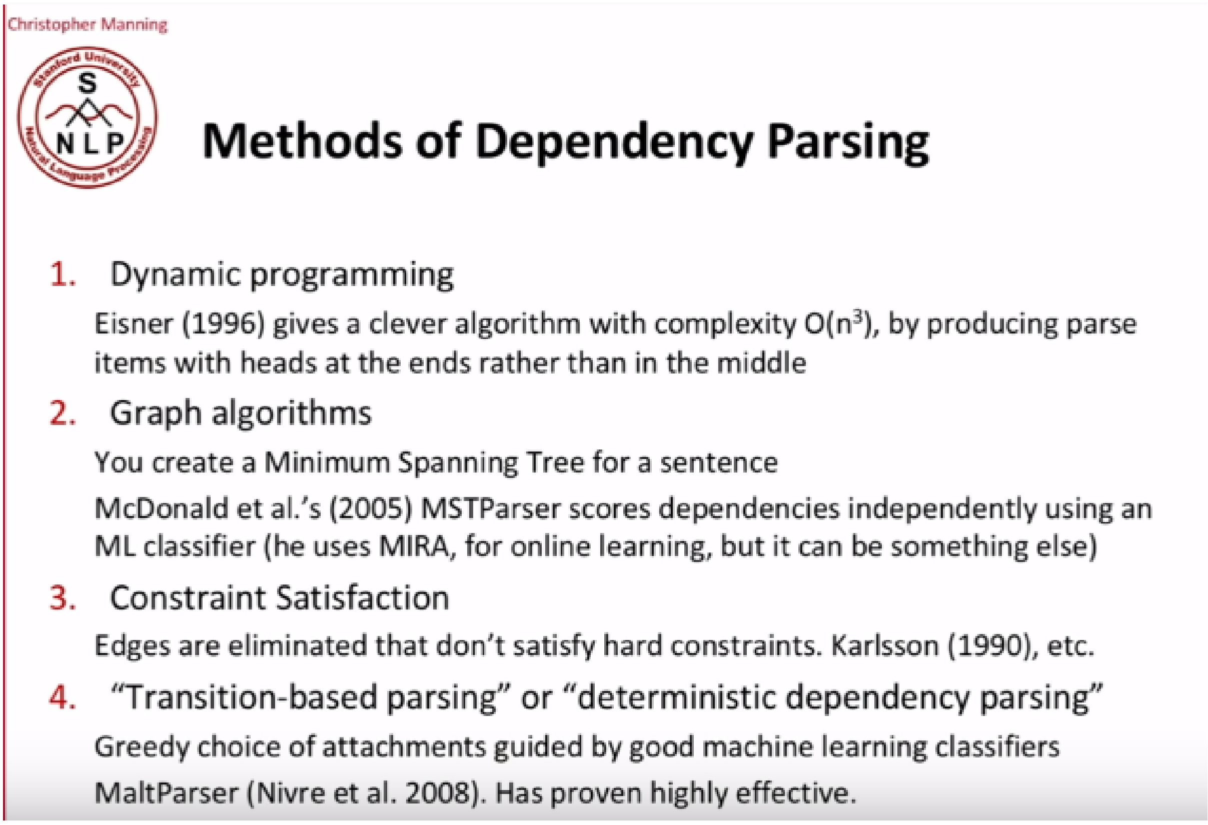
Dependency parsing:



- A sentence is parsed by choosing for each word what other word (including ROOT) is it a dependent of
- Usually some constraints:
  - Only one word is a dependent of ROOT
  - Don’t want cycles A → B, B → A
- This makes the dependencies a tree
- Final issue is whether arrows can cross (non-projective) or not

ROOT I ’ll give a talk tomorrow on bootstrapping

This type of crossing happens with delayed modifiers. We could have just said I’ll give a talk on bootstrapping tomorrow.



Christopher Manning

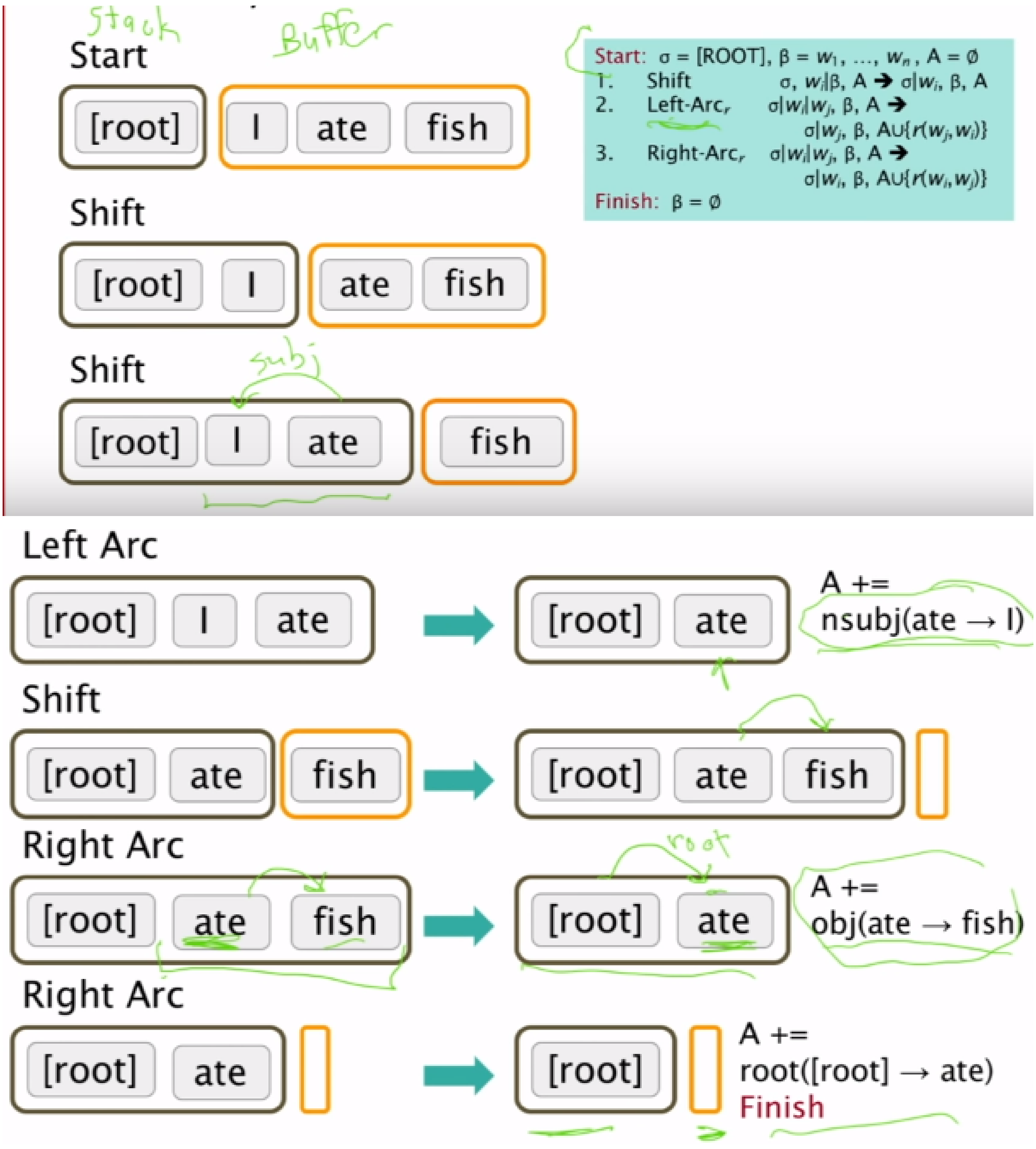
Methods of Dependency Parsing

1. Dynamic programming
   Eisner (1996) gives a clever algorithm with complexity O(n^3), by producing parse items with heads at the ends rather than in the middle
2. Graph algorithms
   You create a Minimum Spanning Tree for a sentence
   McDonald et al.’s (2005) MSTParser scores dependencies independently using an ML classifier (he uses MIRA, for online learning, but it can be something else)
3. Constraint Satisfaction
   Edges are eliminated that don’t satisfy hard constraints. Karlsson (1990), etc.
4. “Transition-based parsing” or “deterministic dependency parsing”
   Greedy choice of attachments guided by good machine learning classifiers
   MaltParser (Nivre et al. 2008). Has proven highly effective.

Arc standard transition based parser:

In this we basically have a stack and a buffer and in every step we have an option to pull a word from buffer into stack, draw a left arc: word left of the second word (which is the first word) is a dependent of the second word, or draw a right arc.

Example process:



Start: σ = [ROOT], $\beta = w_1, \ldots, w_n$, A = Ø
1. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A\cup\{r(w_j,w_i)\}$
3. Right-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A\cup\{r(w_i,w_j)\}$

Finish: β = Ø

A += nsubj(ate → I)

A += obj(ate → fish)

A += root([root] → ate)

Finish

This is where we use machine learning. We can just use a classifier algo.