Natural Language Processing

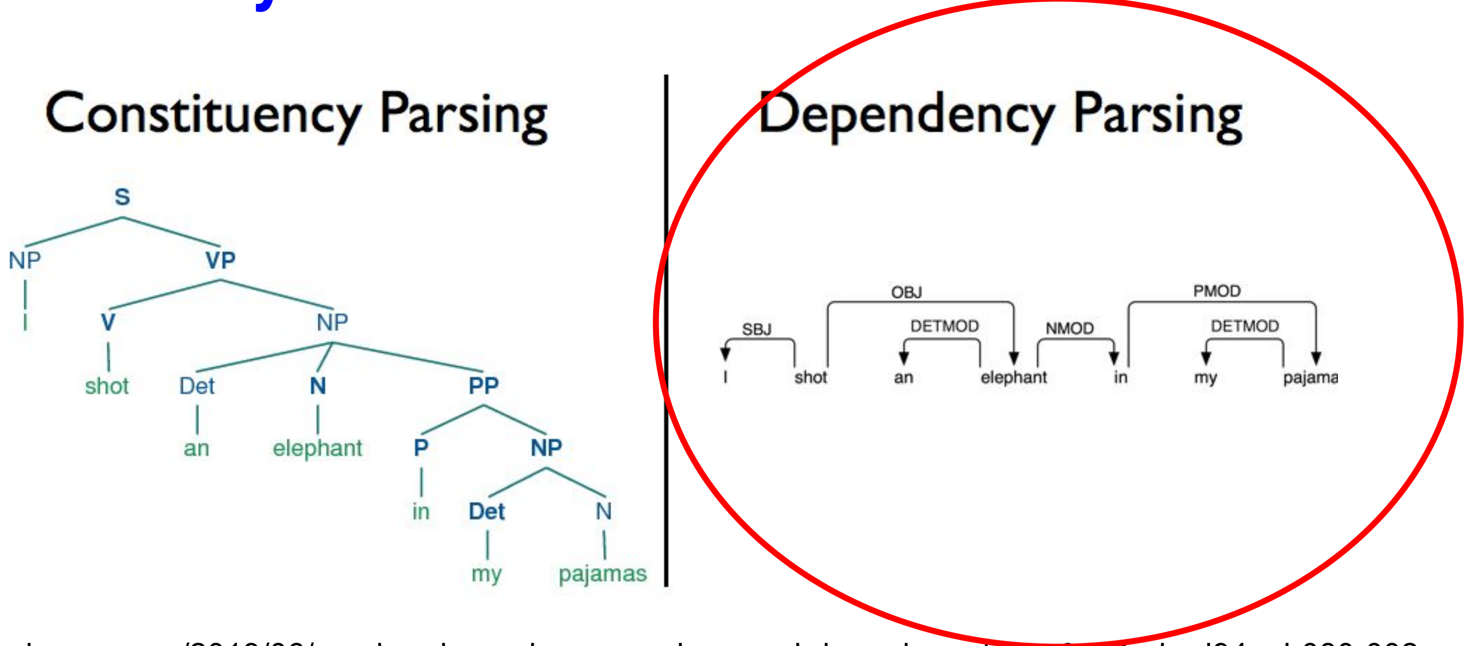
Nuri Cingillioglu
https://www.doc.ic.ac.uk/~nuric/

Many thanks to Lucia Specia

Dependency parsing

Definition

- Given a sequence of words (usually a sentence), generate its **syntactic structure**

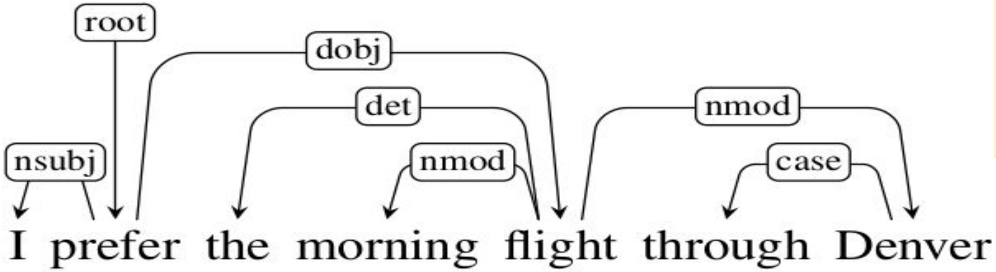


https://lee6boy.files.wordpress.com/2013/06/parsing-dependency-parsing-graph-based-parsingec9db4-ebad94eab080-002.png

Dependency parsing

Figure from Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018, ch 15

- Connect words in sentence to indicate dependencies between them - much older linguistic theory
- Build around notion of having **heads** and **dependents**
- Arrows can be annotated by different types of dependencies
  - **Head** (governor), also called **argument**: origin
  - **Dependent**, also called **modifier**: destiny



From head to dependent

Dependency parsing

- There are versions without **typed dependencies**, just arcs

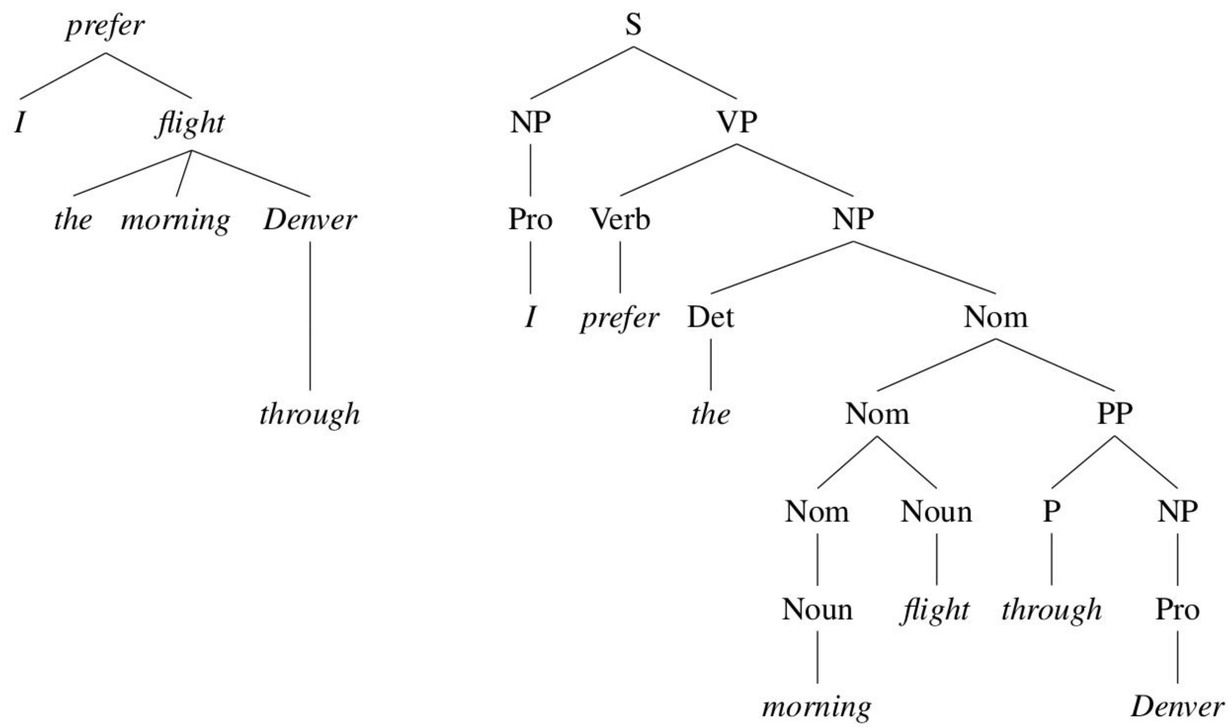
  - Untyped variant is simpler to build but less informative

- Can also represent as a tree

Dependency parsing

Figure from Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018, ch 15

- Comparison to constituency parsing



Dependency parsing

- Main **differences** to constituency parsing
  - No nodes corresponding to phrasal constituents or lexical categories
  - The internal structure consists only of directed relations between **pairs** of lexical items
  - These relationships allow directly encoding important information, e.g.:
    - The arguments of the verb *prefer* are directly linked to it in the dependency structure

Dependency parsing

- Main **advantages** in dependency parsing
  - Ability to deal with languages that are morphologically rich and have a relatively free word order. E.g. Czech location adverbs may occur before or after object:
    I caught the fish **here** vs I caught **here** the fish
  - Would have to represent two rules for each possible place of the adverb for constituency
  - Dependency approach: only one link; abstracts away from word order

Dependency parsing

- Main **advantages** in dependency parsing
  - Head-dependent relations provide approximation to semantic relationship between predicates and arguments
  - Can be directly used to solve problems such as co-reference resolution, question answering, etc.

Dependency relations (e.g. from universal dep.)

Figure from Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018, ch 15

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Dependency relations (e.g. from universal dep.)

Figure from Jurafsky, D and Martin, J, “Speech and Language Processing,” 2018, ch 15

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Dependency formalisms - general case

- A dependency structure is a **directed graph** $G = (V, A)$
  - V = set of vertices (words, punctuation, ROOT)
  - A = set of ordered pairs of vertices (i.e. arcs)

- A dependency tree (directed graph):
  - Has a single ROOT node that has no incoming arcs
  - Each vertex has exactly one incoming arc (except for ROOT)
  - There's a unique path from ROOT to each vertex
  - There are no cycles $A \rightarrow B$, $B \rightarrow A$

Dependency formalisms – general case

- **This ensures the following properties**:
  - Dependency structure becomes a tree
  - Each word has a single head
  - The dependency tree is connected
  - There is a single ROOT from which a unique directed path follows to each word in sentence

Dependency parsing –sources of info

- Distance between head and dependent
  - Mostly nearby words
- Intervening material
  - Dependencies don’t cross over verbs or punctuation
- Valency of verbs
  - For a typical word, what kind of dependency it generally takes? E.g. A noun takes dependencies on the left (DET, JJ) but not on the right

Dependency parsing – two approaches

- Dynamic programming (cubic time, not very accurate)
- Shift-reduce (transition-based)
  - Predict from left-to-right
  - Fast (linear), but slightly less accurate
  - MaltParser
- Spanning tree (graph-based, constraint satisfaction)
  - Calculate full tree at once
  - Slightly more accurate, slower
  - MSTParser

Dependency parsing – transition-based

- Deterministic parsing, shift-reduce (MALT parser)
  - Greedy choice of attachment for each word in order, guided by ML classifier
  - Works very well in practice
  - **Linear time parsing!**

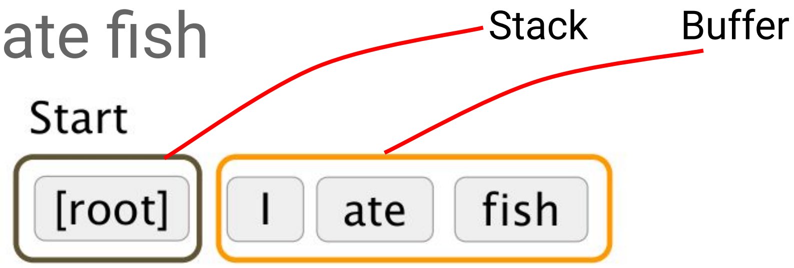
Dependency parsing –MALT parser

- Reads sentence word by word, left to right
- Greedy decision as to how to attach each word as it is read
- Sequence of actions bottom-up
- Formally, 3 data structures:
  - σ = **Stack**, which starts with ROOT
  - β = **Buffer**, which starts with all words in sentence
  - **A** = **Set of arcs**, which starts empty
- Set of actions
  - **Shift** / **left arc** / **right arc**
  - Optionally, set of **dep. labels** for left and right arc actions (~40)

Dependency parsing –MALT parser

Example by Chris Manning

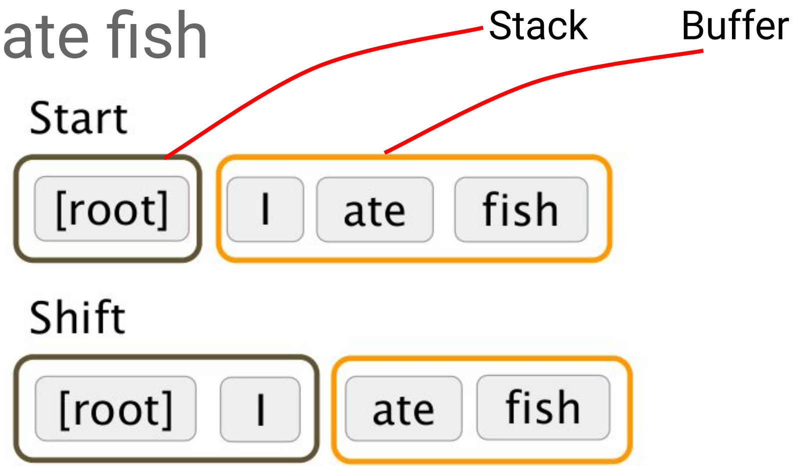
I ate fish



Dependency parsing –MALT parser

Example by Chris Manning

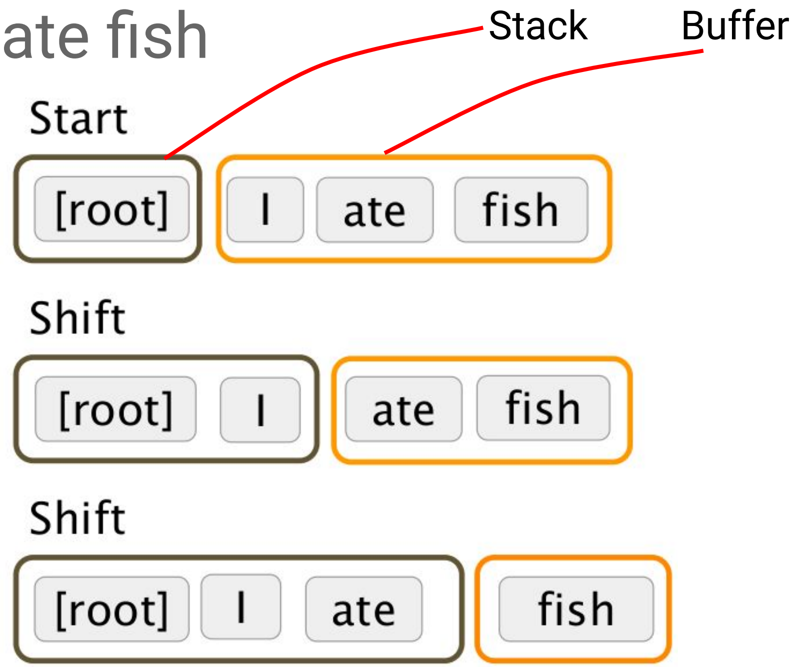
I ate fish



Dependency parsing –MALT parser

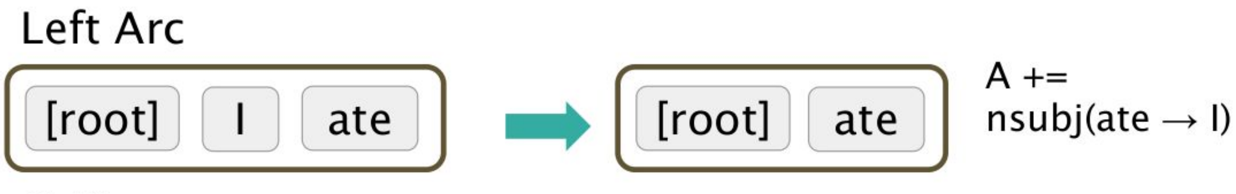
Example by Chris Manning

I ate fish



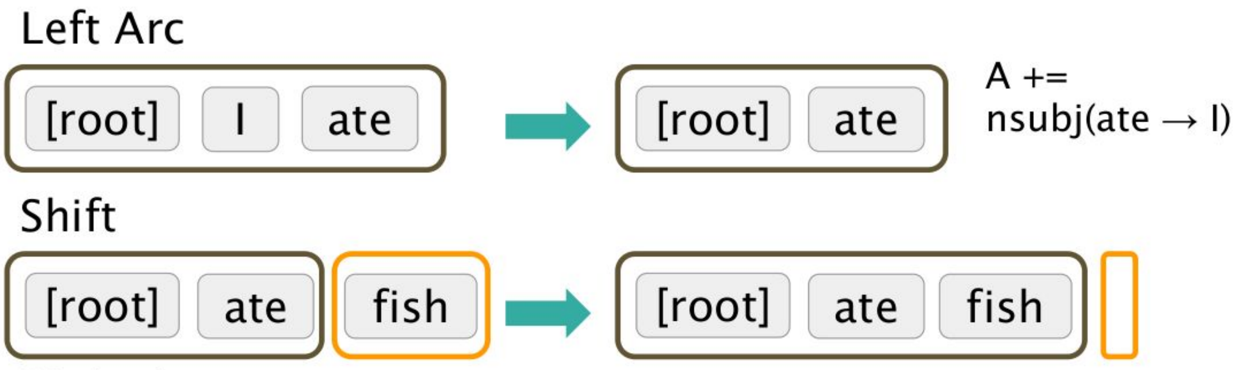
Dependency parsing –MALT parser

Example by Chris Manning



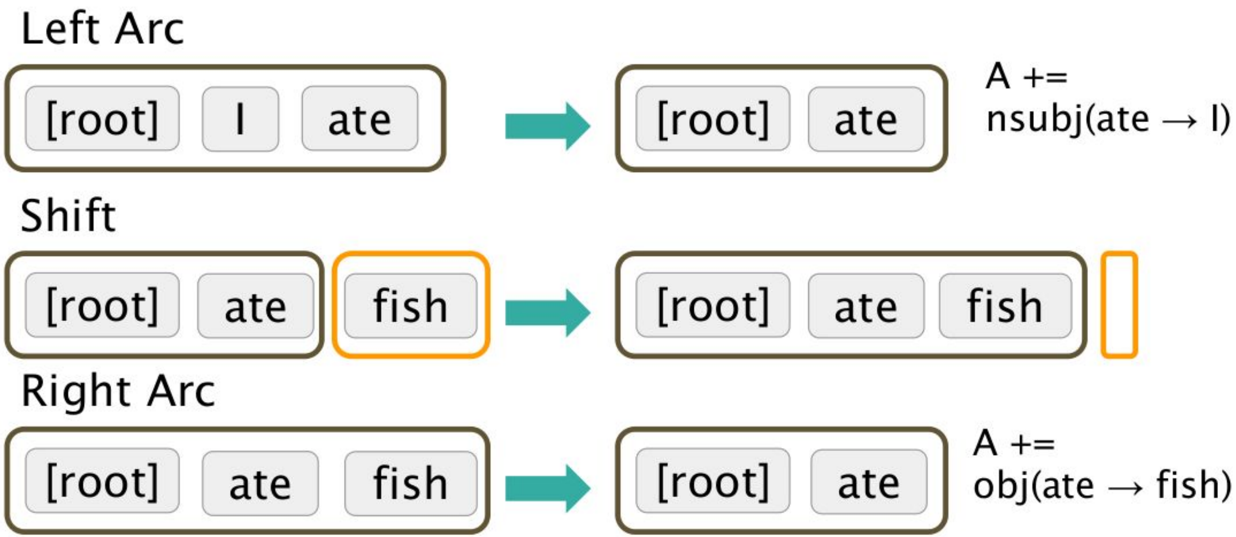
Dependency parsing –MALT parser

Example by Chris Manning



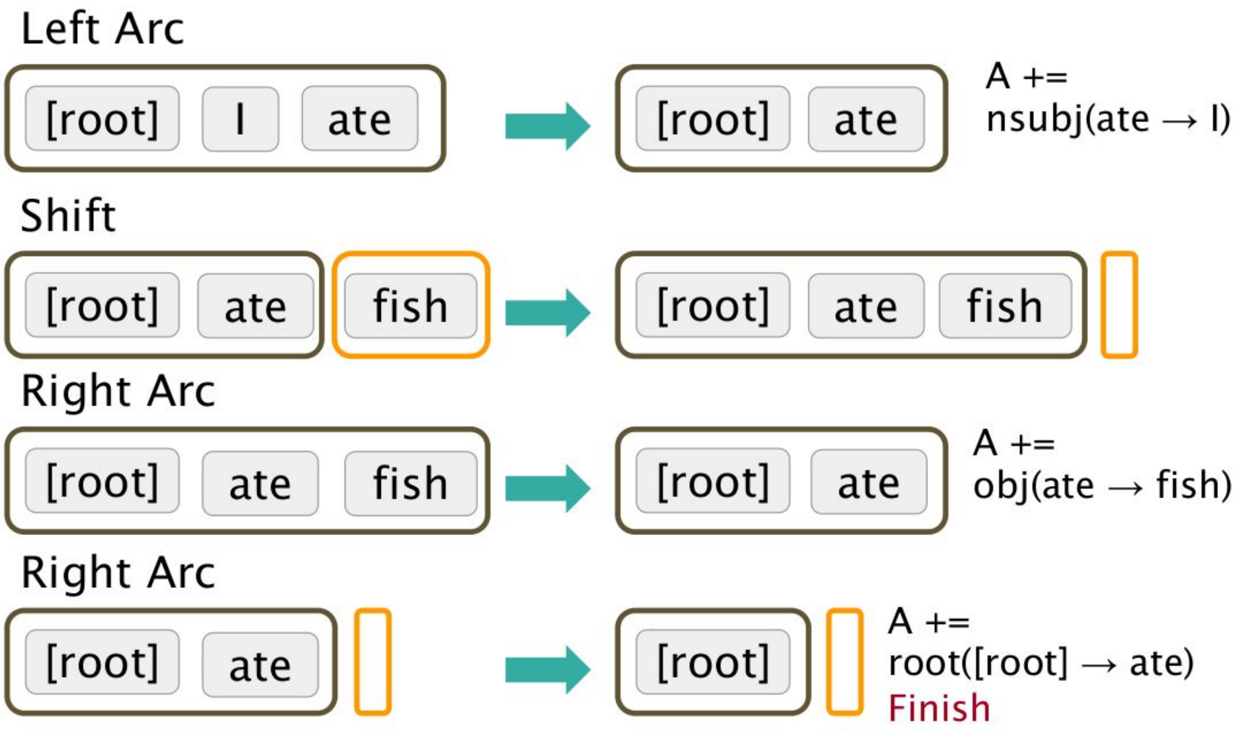
Dependency parsing –MALT parser

Example by Chris Manning



Dependency parsing –MALT parser

Example by Chris Manning



Dependency parsing –MALT parser

Pseudocode by Chris Manning

Start: σ = [ROOT], $\beta = w_1, \ldots, w_n$, A = $\emptyset$

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$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i,w_j)\}$

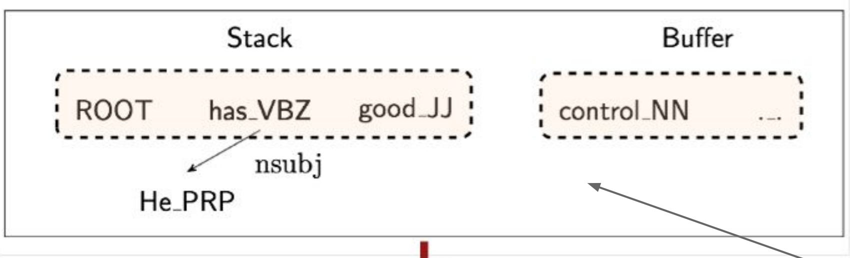
Finish: $\sigma = [w]$, $\beta = \emptyset$

Dependency parsing –MALT parser

- How do we make the shift/reduce left/right decisions?
  - ML classifier
  - Each action is predicted by a **discriminative classifier** over each move
  - 3 classes for untyped dependencies: shift, left or right
  - 2*categories + 1 for typed dependencies
  - **Features**: top of stack word, its POS; first in buffer word, its POS; etc.
  - No beam-search in original version

Dependency parsing –MALT parser

Example by Chris Manning



top of stack word, its POS; first in buffer word, its POS; etc.

encode

binary, sparse
dim $=10^6 \sim 10^7$

0	0	0	1	0	0	1	0	...	0	0	1	0

Feature templates: usually a combination of 1 ~ 3 elements from the configuration.

Indicator features

$$s1.w = \text{good} \wedge s1.t = \text{JJ}$$
$$s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$$
$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$
$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

Dependency parsing – neural parser

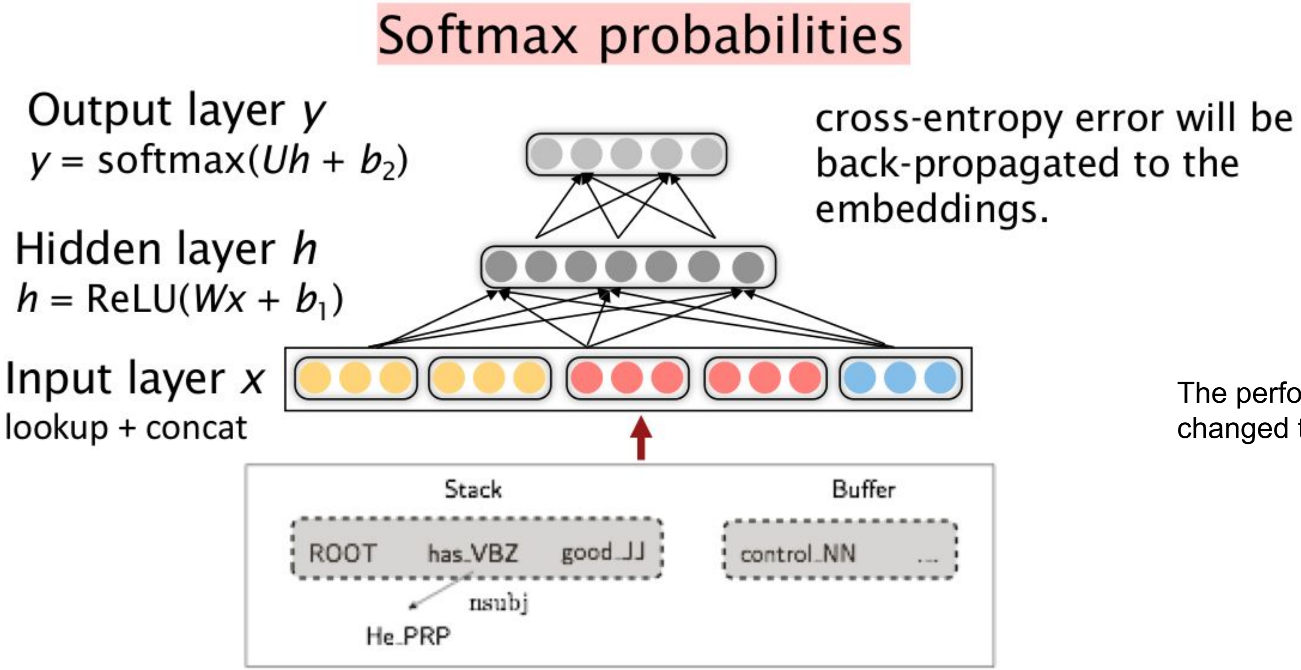
- Follow-up work (Chen and Manning, 2014)
  - Replace binary features by **embeddings** (words & POS)
  - Concatenate these embeddings
  - Train a FNN as classifier with cross-entropy loss
  - Superior performance!
    - UAS = untyped
    - LAS = typed dependencies

Parser	UAS	LAS	sent. / s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	**92.3**	89.6	8
C & M 2014	92.0	**89.7**	**654**

Dependency parsing – neural parser

Example by Chris Manning

- Follow-up work (Chen and Manning, 2014)



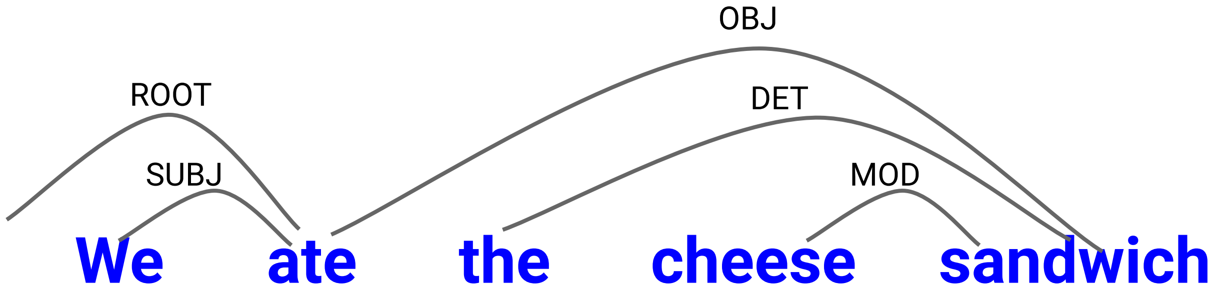
The performance hasn't changed too much since.

Dependency parsing – treebanks

- Treebanks exist in the same was as for constituency parsing for training these classifiers
  - From converting constituency treebanks
  - Annotated from scratch, esp. For morphologically rich languages such as Czech, Hindi and Finnish

Dependency parsing – evaluation

- Accuracy, precision or recall:
    - Count identical dependencies: span (non-typed parsing) or span and type of dependency (typed parsing). E.g.:



Dependency parsing – evaluation

- Accuracy, precision or recall:
  - Count identical dependencies: span (non-typed parsing) or span and type of dependency (typed parsing). E.g.:

Reference		**Hypothesis**	
(1,2) We	SUBJ	(1,2) We	SUBJ
(2,0) eat	ROOT	(2,0) eat	ROOT
(3,5) the	DET	(3,4) the	DET
(4,5) cheese	MOD	(4,2) cheese	OBJ
(5,2) sandwich	OBJ	(5,2) sandwich	PRED

Dependency parsing – evaluation

- Accuracy, precision or recall:
  - Count identical dependencies: span (non-typed parsing) or span and type of dependency (typed parsing). E.g.:

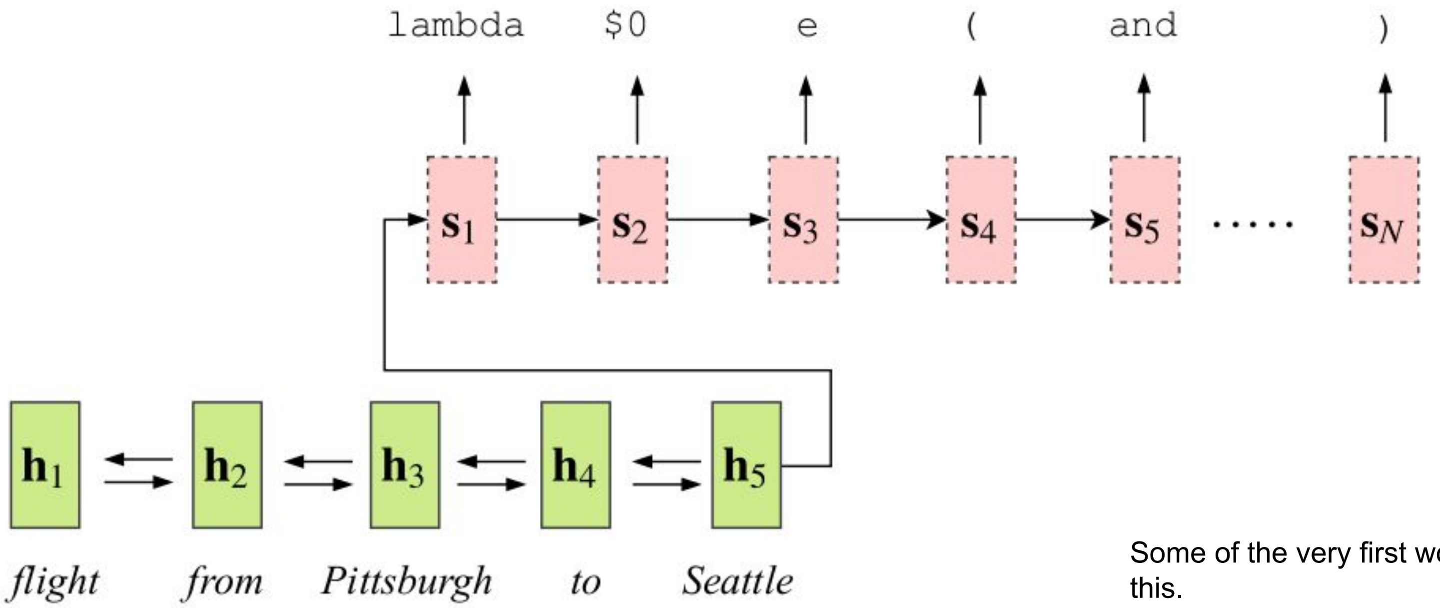
Reference		**Hypothesis**	
(1,2) We	SUBJ	(1,2) We	SUBJ
(2,0) eat	ROOT	(2,0) eat	ROOT
(3,5) the	DET	(3,4) the	DET
(4,5) cheese	MOD	(4,2) cheese	OBJ
(5,2) sandwich	OBJ	(5,2) sandwich	PRED

Accuracy = ⅖ = 40%

Neural parsing

Neural parsing – simple approach

(Jia and Liang, 2016; Dong and Lapata, 2016)



Some of the very first works doing this.

Neural parsing – simple approach

- Parsing as translation
  - Linearise grammar from treebank: convert tree to bracketed representation, all in one line
  - Extract sentences from bracketed representation
  - Pair sentences and their linearised trees
  - No need to compute/represent probabilities - **learning**

Neural parsing – simple approach

- Train sequence to sequence model to *translate* from
  - Sentences to linearised tree; **brackets are tokens**!
  - Use, e.g. LSTM, transformers etc…
  - Attention helps
  - Train with cross-entropy
  - Evaluate as a translation (BLEU) or parsing tasks (parseval)

Neural parsing – advanced approaches

Table by Chris Manning

- Graph-based methods

Method	UAS	LAS (PTB WSJ SD 3.3
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79
Dozat & Manning 2017	**95.74**	**94.08**

- UAS: Unlabeled attachment score
- LAS: Labeled attachment score

Not much more progress from there

Discussion

- Parsing is an important step for many applications
- Statistical models such as PCFGs allow for resolution of ambiguities
  - Lexicalisation and non-terminal splitting are required to effectively better resolve many ambiguities
- Current statistical/neural parsers are quite accurate
  - ~95% dependency; 97% constituency
  - Human-expert agreement: ~98%