Overview of Natural Language Processing Part II & ACS L90

Lecture 5: Dependency Parsing

Andrew Caines

based on slides by Ann Copestake, Simone Teufel, Paula Buttery, Weiwei Sun

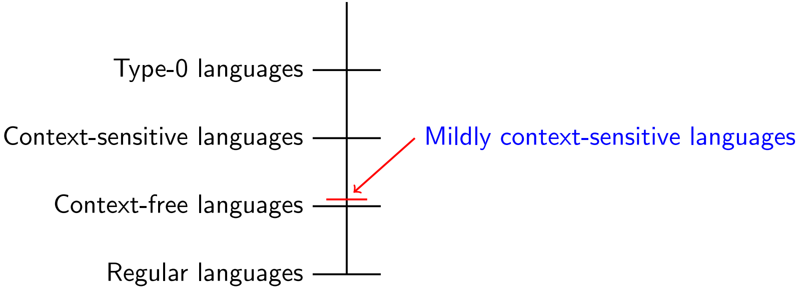
Department of Computer Science and Technology
University of Cambridge

Michaelmas 2021/22

Part 1: Background

Recall lecture 4: limitations of CFGs

Natural languages are provably non-context-free.
Natural languages as mildly context-sensitive languages



An alternative: model relations between word tokens

- rather than directly modelling phrase structure: model (labelled) relations between *word tokens*
- NB: why *word tokens*?

Aside: what's a word (token)?

- Well, what's a word?
- "Count the words in this sentence." =6 (orthography)
- "That's easy!" 2?
- tokenized →1:That 2:'s 3:easy 4:!
- phon. "Why you gotta be a time-waster?"
- →1:Why 2:you 3:got 4:ta 5:be 6:a 7:timewaster 8:?
- For syntactic parsing: morpho-syntax over phonology/orthography ... (plus semantics?)
- MWEs: "It's a dog eat dog world."
- →1:It 2:'s 3:a 4:dog_eat_dog 5:world 6:.
- Agglutination (root&affixes): Turkish *gelememiș* "apparently he couldn't come"
- Polysynthesis (diff.word.classes): Yupik (Alaska-Siberia) *tuntussuqatarniksaitengqiggtuq* "He had not yet said again that he was going to hunt reindeer"
- Tokenization as a fundamental task in NLP (En pre-trained tokenizers); nb complexity in Chinese, Japanese, Arabic, Hebrew, etc

An alternative: dependency grammar

- (un)Labelled relations between *word tokens*
- Binary directed relations from **heads** to **dependents**
- Dependency structure as a directed graph: $G = (V, A)$
- A set of vertices V (word tokens in a string: *written sentence, transcribed spoken utterance, tweet, etc*)
- A set of ordered pairs of vertices A, capturing relations between elements in V
- Different linguistic formalisms with different constraints, but for computational approaches, G should be acyclic and have:
  1. a single designated root node with no incoming arcs
  2. vertices (word tokens) with exactly 1 incoming arc each (except the root node which has 0)
  3. a unique path from the root node to each vertex

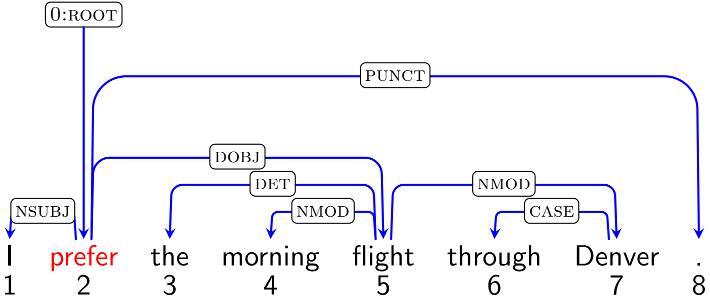
An alternative: dependency grammar

- First: tokenize string $x \to$ indexed word tokens $\to$ implicit **root** token at location 0
- Identify head–dependent token pairs
- (optionally/usually) With labels from a fixed inventory, e.g. `root`, `nsubj`, `dobj`, `det`, `nmod`
- Phrase structure not directly modelled, explicit structuring of nested constituents dropped (might infer it)
- Instead dependency parse to find relations between lexical items (& the root)

An example: dependency structure

I prefer the morning flight through Denver .

An example: dependency structure



Remember:

1. a single designated root node with no incoming arcs
2. vertices (word tokens) with exactly 1 incoming arc each (except the root node)
3. a unique path from the root node to each vertex

Dependency relations

- ROOT
- NSUBJ: nominal subject
- (D)OBJ: direct object
- DET: determiner
- NMOD: nominal modifier
- CASE: case marker
- PUNCT
- And more, and alternatives

Dependency structure & phrase structure side-by-side

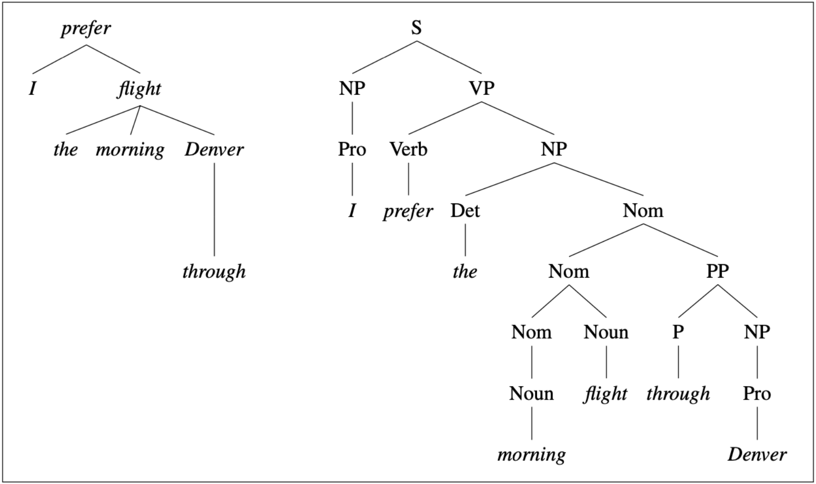
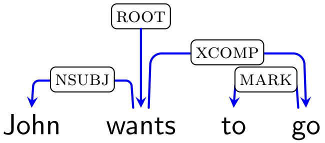


Figure 14.1 Dependency and constituent analyses for *I prefer the morning flight through Denver.*

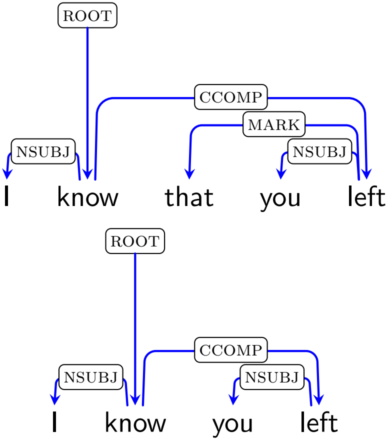
(Formal Models of Language: weak equivalence between dep.gramms & CFGs, can derive same set of strings, systematic mapping b/w trees)

Dependency structure (contd)



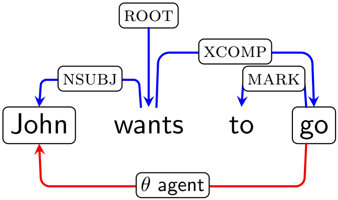
- XCOMP: open clausal complement (without its own subject; subject usually the object or subject of the higher clause)
- MARK: marker to indicate subordinate clauses, e.g. *that, if, whether, to* (might be omitted)

Dependency structure (contd)



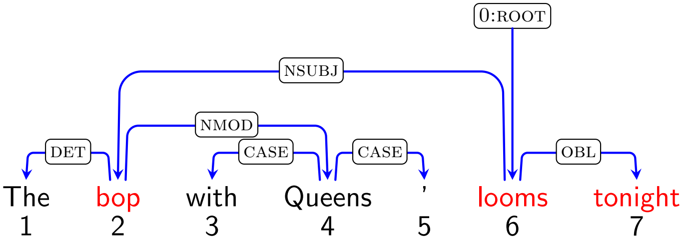
- CCOMP: clausal complement
- MARK: marker (might be omitted)

Dependency structure (contd)



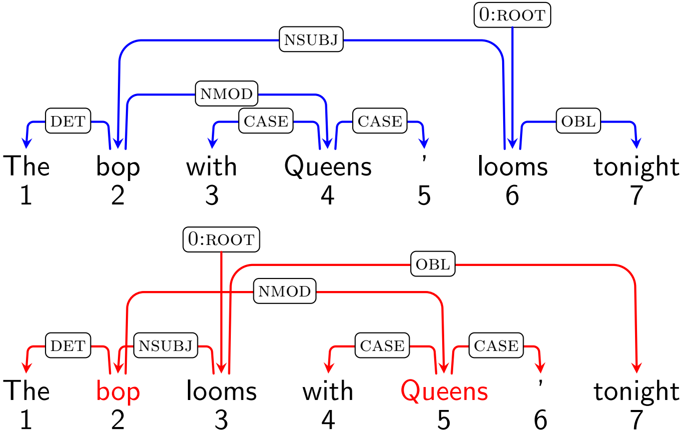
- But semantics: *John* is also the AGENT of *go* (control) – *not captured by dependency analysis* because no phrase structure. And this kind of relation is systematic.
- *He* **wants** *to sleep in class.*
- *He* **promises** *her not to sleep in class.*
- Also raising: *She seems to be reading a book.*

Projectivity



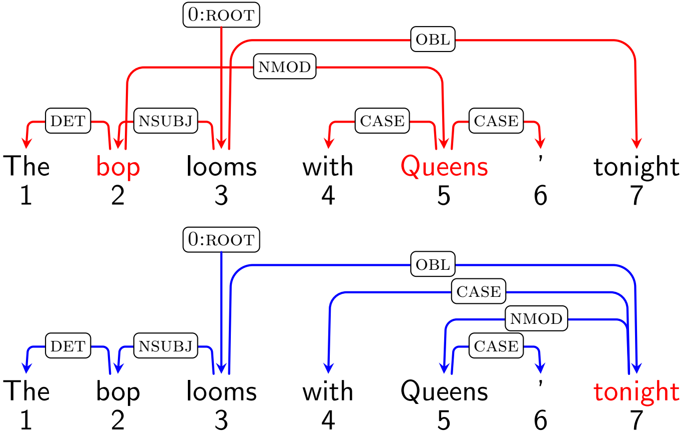
- Dependency trees can be projective or non-projective: tree is projective if every arc is projective
- Arc from head to dependent is projective if there is a path from the head to every token between the head and dependent in the string (can check this visually)
- OBL: oblique nominal ('tonight' treated as a temporal nominal; cf. 'later' an adverbial with ADVMOD)

Projectivity



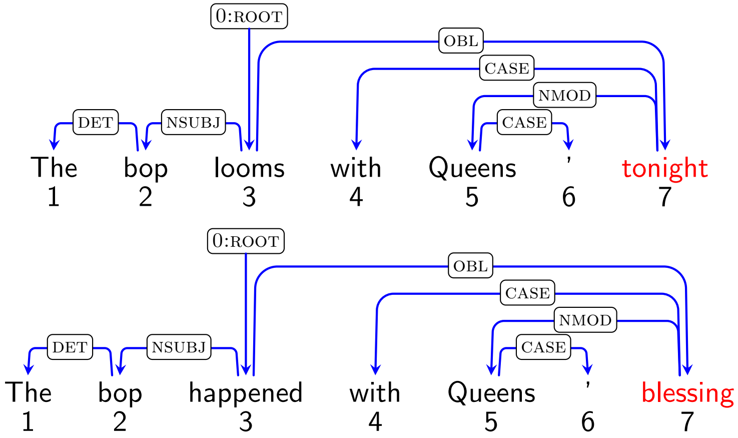
- Dependency trees can be projective or *non-projective*
- Arc from head to dependent is projective if there is a path from the head to every token between the head and dependent in the string

Projectivity



- Dependency trees can be projective or *non-projective*
- But in practice: most widely used dependency parsers produce projective trees only (for others see J&M 3rd edn §14.5)

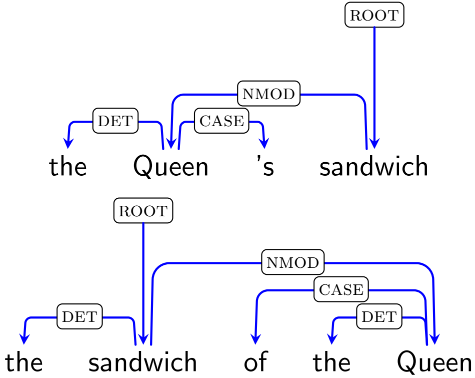
Projectivity



- Dependency trees can be projective or *non-projective*
- But in practice: parsers tend to be projective only (!linguistic alarm)

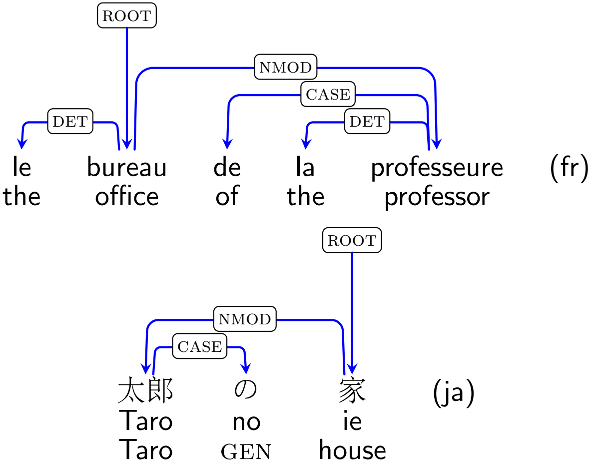
The CASE relation

- The CASE relation: prepositions, postpositions, clitic markers
- Parallel analyses for possessive alternation:



The CASE relation

- The CASE relation: prepositions, postpositions, clitic markers
- Cross-linguistic consistency:



Word order

- English word order: usually subject verb object (SVO)
- 'who did what to whom': *The dog bites the man* $\neq$ *The man bites the dog* (word order matters)
- Passives are still SVO (morpho-syntactically, cf. semantics): *The man was bitten by the dog*
- Can 'front' the object in topicalization: *The man, the dog bites* ... *He's having a rough day*
- Other languages may be more flexible with word order

Word order

- e.g. German: *Der Hund beißt den Mann* = *Den Mann beißt der Hund*.
  'The dog bites the man'
- e.g. Russian (example from Emily Bender, 2013):

  Chelovek ukusil sobaku

  man.NOM.SG.M bite.PAST.PFV.SG.M dog.ACC.SG.F

  'the man bit the dog'

  All word orders possible with same meaning (in different discourse contexts):

  Chelovek ukusil sobaku

  Chelovek sobaku ukusil

  Ukusil chelovek sobaku

  Ukusil sobaku chelovek

  Sobaku chelovek ukusil

  Sobaku ukusil chelovek

Word order

- Due to WO variability: rules like `S → NP VP` do not work in all languages (rule explosion)
- Could allow discontinuous constituency: e.g. Sobaku chelovek ukusil ('dog man bit') with a split VP
- But parsing discontinuities is not straightforward
- $\rightarrow$ dependency structures
- Binary relations between words

(questions?) $\rightarrow$Part 2: Parsing

Transition-based dependency parsing

A transition system for parsing is a quadruple $S = (C, T, c_s, C_t)$, where

1. C is a set of configurations, each of which represents a parser state.
2. T is a set of transitions, each of which represents a parsing action,
3. c_s initializes S by mapping a sentence x to a particular configuration,
4. $C_t \subseteq C$ is a set of terminal configurations.

Transition-based dependency parsing

A stack-based configuration (parser state) for a string $x = w_0, w_1, \ldots, w_n$ (V) as $c = (\sigma, \beta, \mathcal{A})$, where

1. σ is a stack of tokens with which to build the parse;
2. β is a buffer of tokens yet to be parsed;
3. $\mathcal{A}$ is a set of relations arcs representing a dependency tree

Transition-based dependency parsing

$\textsc{DepParse}(x = (w_0, w_1, \ldots, w_n))$
$c \leftarrow \{[\textsc{root}], [w_0, w_1, \ldots, w_n], []\}$
while $c \notin C_t$
  $t \leftarrow \textsc{Oracle}(c)$; choose transition operator
  $c \leftarrow \textsc{Apply}(t, c)$; apply it, creating a new state
return c

- Initial configuration c_s: $\textsc{root}$ in stack, tokens of x in buffer, empty set of relations in A
- Actions based on current *configuration*: {stack, buffer, relations so far} and consulting an *oracle* for correct operation, which is applied to produce a new configuration
- $\rightarrow$ End goal: buffer empty and only $\textsc{root}$ on stack, A as a set of relations representing the parse for input string x, such that $G = (V, A)$ is a dependency graph for x

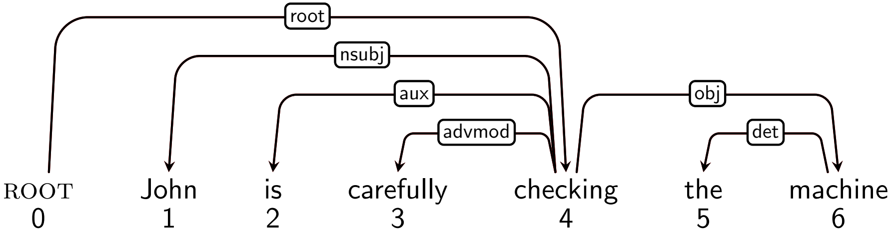
Transition-based dependency parsing

- The **arc standard** approach (Covington 2001, Nivre 2003)
- Available transitions/actions: SHIFT, LEFT-ARC, RIGHT-ARC
- At each step: either SHIFT or link word tokens with LEFT-ARC, RIGHT-ARC (analogies with intuitive tree drawing)
- SHIFT: pop word token from front of buffer to top of stack (≈ postpone dealing with the current token, store it for later on)
- LEFT-ARC: draw head-dependent arc between token at the top of the stack and the second item on the stack (cannot be ROOT); remove second item from stack (≈ assign current token as head of a previously seen token)
- RIGHT-ARC: draw head-dependent arc from second token on the stack to the first one; remove top item from stack (≈ assign previously seen token as head of current token)
- Can parameterize the actions to produce labels: LEFT-ARC$_{subj}$, etc

Transition-based dependency parsing

- Metaphor from shift-reduce parsing (analysis of programming languages): left-arc and right-arc are REDUCE operations, combining elements on the stack
- *Greedy*: Oracle provides a single choice at each step, parser proceeds, no further exploration or opportunity to back-track, single parse at the end
- Complexity is linear to length of the sentence: a single left-to-right pass through x
- Parsing as a "sequence of transitions through the space of possible configurations" (J&M)
- (more complex algorithms in J&M 3rd edn ch.14, and L95)
- $\rightarrow$ walk-through...

Parsing example



Shift | Left-Arc$_{nsubj}$ | Left-Arc$_{advmod}$

Left-Arc$_{aux}$ | Left-Arc$_{det}$ | Right-Arc$_{obj}$

Stack

[ROOT] [ROOT, John] [ROOT, John, is] [ROOT, is] [ROOT, is, carefully] [ROOT, is, carefully, checking] [ROOT, is, checking] [ROOT,

Buffer

[John, is, ..., the, machine] [is, carefully, checking, the, machine] [carefully, checking, the, machine] [carefully, checking, the, machine]

Transition-based dependency parsing

- How to learn to take the correct actions for each parse configuration?
- Feature-based classification: surrounding words, lemmas, PoS-tags, morph features, relations already assigned (and concatenations) – associated with the correct action to take
- e.g. s_0^w, s_1^w, b_0^w, s_0^l, s_1^l, s_0^t, b_0^t, s_0^m, b_0^m, r_0, r_1, etc
- In training: choose LEFT-ARC if it produces a correct head-dependent relation given the reference parse and current configuration
- Or choose RIGHT-ARC if it produces a correct head-dependent relation given the reference parse and dependents have already been assigned (*prefer* the *flight* through *Denver*)
- Else choose SHIFT
- Train with a classifier such as multinomial logistic regression, support vector machines, etc
- More recently: word (token) embeddings and neural networks (next lecture)

Training data

- Training data: hand-labelled treebanks (a corpus sub-type)
- Laborious and costly to put together (or, automatically derived from phrase-structure annotation, can be lossy)
- Best-known are the annotated *WSJ* texts in the Penn Treebank
- and *Universal Dependencies*: a long-running, multi-institution project
- Collecting and releasing treebanks for many languages (v2.9, gt. 100 at present)
- Concretely proposed in publications by Ryan McDonald et al (ACL 2013) then Marie-Catherine de Marneffe et al (LREC 2014); the ideas pre-date this, have evolved since
- Website with more info and open datasets: universaldependencies.org
- 5th Universal Dependencies Workshop 2021 (ACL Anthology)
- Often with a shared task: cross-lingual parsing

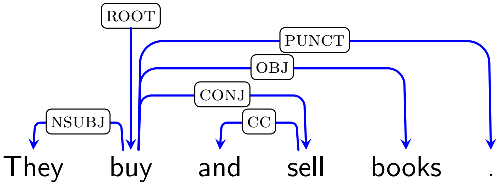
CoNLL-X treebank format

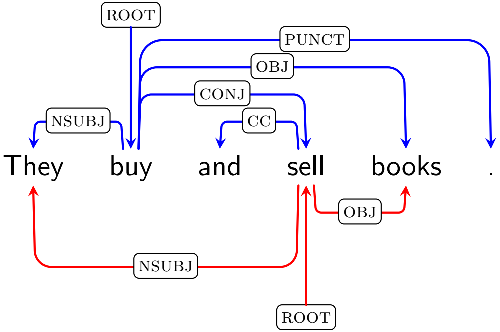
```
# sent_id = 1
# text = They buy and sell books.
1   They    they    PRON    PRP   Case=Nom|Number=Plur               2   nsubj   2:nsubj|4:nsubj   _
2   buy     buy     VERB    VBP   Number=Plur|Person=3|Tense=Pres    0   root    0:root            _
3   and     and     CONJ    CC    _                                  4   cc      4:cc              _
4   sell    sell    VERB    VBP   Number=Plur|Person=3|Tense=Pres    2   conj    0:root|2:conj     _
5   books   book    NOUN    NNS   Number=Plur                        2   obj     2:obj|4:obj       SpaceAfter=No
6   .       .       PUNCT   .     _                                  2   punct   2:punct           _

# sent_id = 2
# text = I have no clue.
1   I      I      PRON    PRP   Case=Nom|Number=Sing|Person=1      2   nsubj   _   _
2   have   have   VERB    VBP   Number=Sing|Person=1|Tense=Pres    0   root    _   _
3   no     no     DET     DT    PronType=Neg                       4   det     _   _
4   clue   clue   NOUN    NN    Number=Sing                        2   obj     _   SpaceAfter=No
5   .      .      PUNCT   .     _                                  2   punct   _   _

# sent_id = panc0.s4
# text = तत् यथानुश्रूयते।
# translit = tat yathānuśrūyate.
# text_fr = Voilà ce qui nous est parvenu par la tradition orale.
# text_en = This is what is heard.
1     तत्         तद्       DET     _   Case=Nom|…|PronType=Dem   3   nsubj    _   Translit=tat|LTranslit=tad|Gloss=it
2-3   यथानुश्रूयते   _        _       _   _                         _   _        _   SpaceAfter=No
2     यथा        यथा      ADV     _   PronType=Rel              3   advmod   _   Translit=yathā|LTranslit=yathā|Gloss=how
3     अनुश्रूयते    अनु-श्रु    VERB    _   Mood=Ind|…|Voice=Pass     0   root     _   Translit=anuśrūyate|LTranslit=anu-śru|Gloss=
4     ।          ।        PUNCT   _   _                         3   punct    _   Translit=.|LTranslit=.|Gloss=.
```

ID, form, lemma, UPOS, XPOS, morph, head, dep-rel, enhanced deps, misc.
and # commented info

Enhanced dependencies

- As well as 'basic' UDs: intended for shallow analyses and downstream NLP tasks
- 'Enhanced' UDs: for optional, deeper syntactic analyses





Enhanced dependencies

- For optional, deeper syntactic analyses
- To make implicit relations explicit (e.g. coordination example), and make finer-grained distinctions within large catch-all classes (e.g. nominal modifiers NMOD, oblique nominals OBL)
- Other examples include:
  - control/raising: from embedded verb to subj (‘*Jo* wants to *buy* a book’, ‘*Jo* seems to be *reading*’; !non-proj)
  - relative clauses: rel.pronouns attached to antecedents as well as the main predicate in the relative clause (‘the *boy* ← *who* flew’; !cyclic)
  - case information: to disambiguate semantic roles (‘they went *to* dinner *after* work’, OBL:TO, ADVCL:AFTER)
- Enhanced UDs were a feature of the UD shared task this year: you could look at the published papers to see how participants adapted to them

More transition-based dependency parsing

- Can adapt or introduce other actions: SHIFT, LEFT-ARC, RIGHT-ARC
- REDUCE in arc-eager: pop the stack (left & right-arcs b/w stack and buffer)
- Can assign rightward relations sooner, rather than holding them on the stack, reduces late-assignment errors
- EDIT: to delete previously proposed arcs
- e.g. can deal with spoken disfluencies: false starts and repetition
- Training on the annotated treebank in the Switchboard Corpus of telephone conversations
- As with the Redshift non-monotonic (revise previous decisions), arc-eager (see J&M) parser by Honnibal & Johnson (CoNLL 2013, TACL 2014)
- Honnibal $\rightarrow$ spaCy: industry-ready dependency parsing
- Previously Redshift, and now neural

Transition-based dependency parsing in the wild

DEPENDENCY PARSING SYSTEM	UAS	LAS
spaCy RoBERTa (2020)	95.1	93.7
Mrini et al. (2019)	97.4	96.3
Zhou and Zhao (2019)	97.2	95.7

- Penn Treebank test set (*WSJ*)
- UAS: unlabelled attachment scores
- LAS: labelled attachment scores

Multilingual state-of-the-art dependency parsing

Model	LAS	MLAS	BLEX	Paper / Source
Stanford (Qi et al.)	74.16	62.08	65.28	Universal Dependency Parsing from Scratch
UDPipe Future (Straka)	73.11	61.25	64.49	UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task
HIT-SCIR (Che et al.)	75.84	59.78	65.33	Towards Better UD Parsing: Deep Contextualized Word Embeddings, Ensemble, and Treebank Concatenation

- from: NLP Progress `http://nlpprogress.com`
- CoNLL 2018 Shared Task: 82 test sets, 57 languages
- MLAS: morphology-aware labeled attachment score (incl PoS-tags & morph feats)
- BLEX: bi-lexical dependency score (dep.rels & lemmas)

Summary

- Pros include: argument relations to the fore, enables cross-lingual work (use of morph feats, flex word order, UDs), fast and practical
- Cons include: linguistically shallow, some semantic relations not captured, catch-all classes, parses of convenience (because projectivity constraint)
- Greedy algorithm can go wrong, but usually reasonable accuracy (cp human language processing?)
- No notion of grammaticality (so robust to typos)
- Fast (linear time) and practical: downstream applications such as question answering, coreference resolution, information extraction – approximate semantic relations between words via morpho-syntax
- What's your motivation? (NLP / CL)
- More on dependency parsing in Part IB/II Formal Models of Language & ACS L95

NLP lectures

1. Overview lecture
2. Morphology & FSTs
3. PoS-tagging & log-linear models
4. Phrase structure & structured prediction
5. Dependency parsing

   (Now could be a good time to consolidate: morpho-syntactic foundations)
6. Neural Networks

Further reading

- Ann's lecture notes.
  https://www.cl.cam.ac.uk/teaching/2122/NLP/materials.html
- Chapter 14 'Dependency Parsing' by Jurafsky & Martin *Speech and Language Processing* 3rd edition (in prep).
  https://web.stanford.edu/~jurafsky/slp3/14.pdf
- de Marneffe et al, 'Universal Dependencies', in *Computational Linguistics* https://doi.org/10.1162/coli_a_00402
- Universal Dependencies events
  https://universaldependencies.org/events.html
- IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies https://universaldependencies.org/iwpt21
  and Proceedings https://aclanthology.org/volumes/2021.iwpt-1