COMP 3225

Natural Language Processing

Dependency Parsing

Stuart E. Middleton

sem03@soton.ac.uk

University of Southampton

Overview

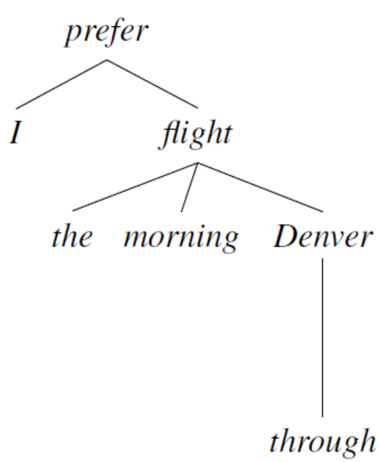
- Dependency Grammars
- Dependency Relations
- Dependency Formalisms
- Dependency Treebanks

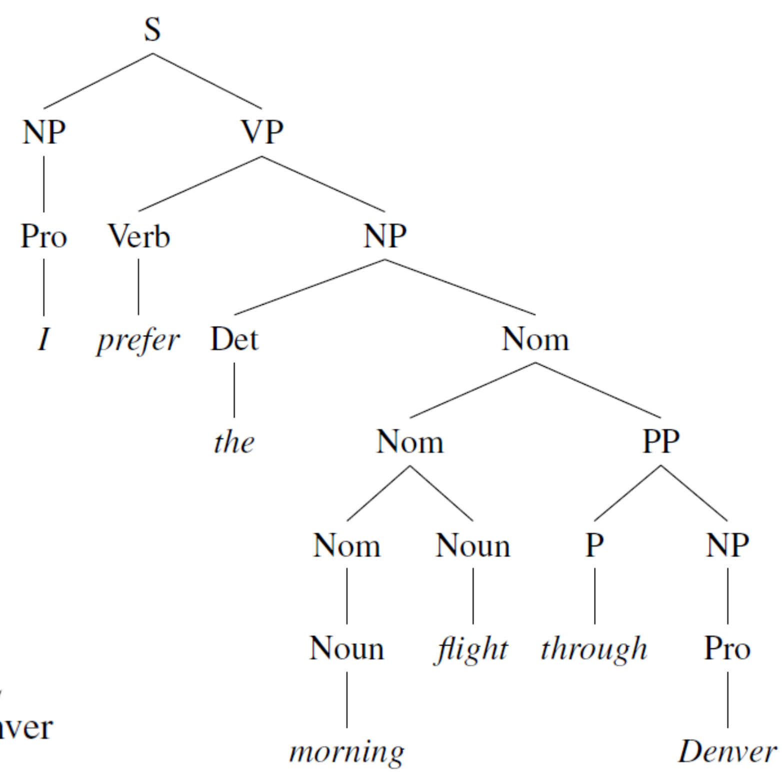
- <break - discussion point>

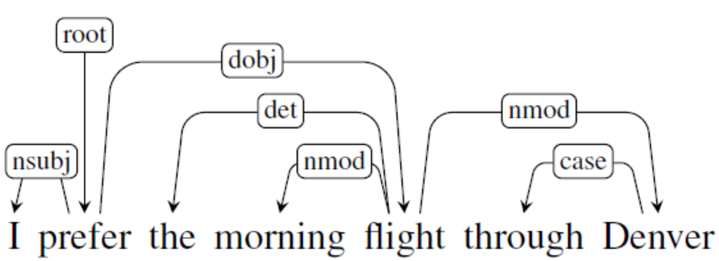
- Transition-based Dependency Parsing
- Graph-based Dependency Parsing
- Evaluation

Dependency Grammars

- Context free grammars provide a constituent-based structure
- Typed Dependency structures use grammatical relations
  - Grammatical relations defined by linguists
  - Labels for edges are grammatical relations
  - Nodes are words
- Dependency grammars have a free word order







Dependency Relations

- Grammatical relations connect head and dependent words
- The type of a grammatical relation is called its grammatical function

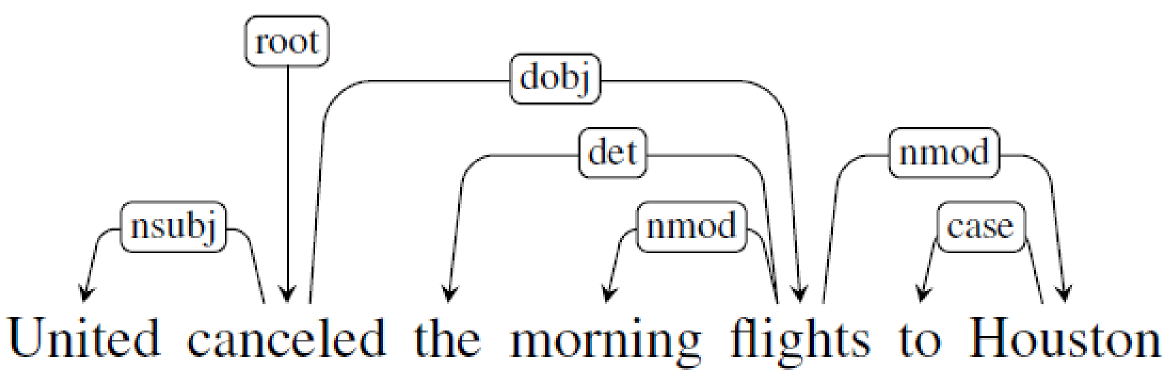
Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET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

Cite: Nivre, J. et. al. Universal Dependencies v1: A multilingual treebank collection. LREC (2016)

Dependency Relations

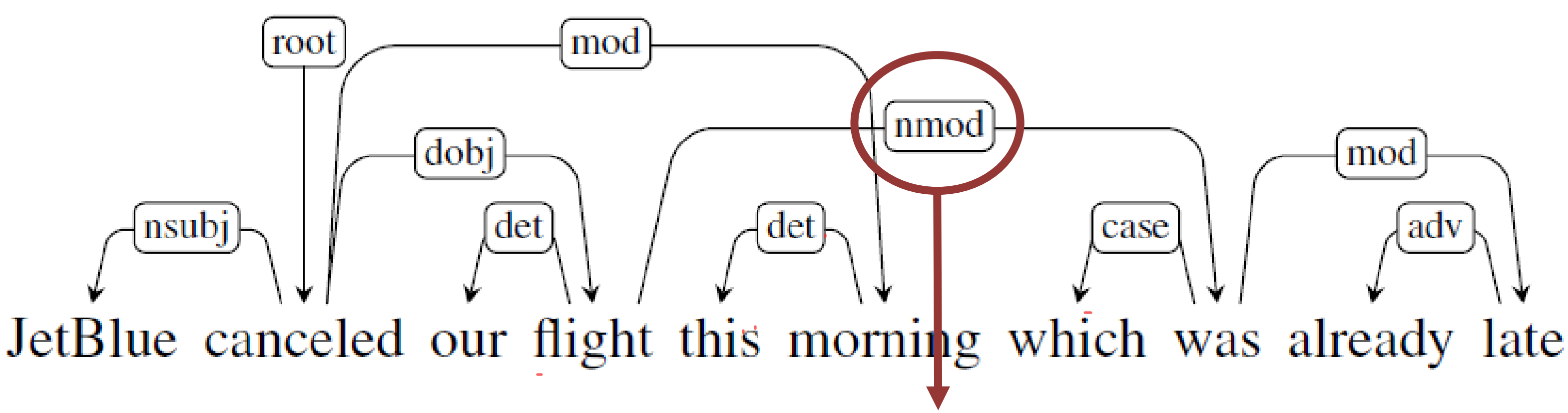
- Grammatical relations connect head and dependent words
- The type of a grammatical relation is called its grammatical function

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.
	rove to Steamboat.
	Houston.



Dependency Formalisms

- Dependency tree is a directed graph
  - G = (V,A)
  - V = Vertices = nodes = words or sometimes stems/affixes
  - A = Arcs = grammatical function relationships
  - Root node has no incoming A
  - Each V has one incoming A
  - Root node has a path to connect to every V
- Projectivity
  - An arc is 'projective' if for a pair(head, dependent) there is a path from the head to all words in-between the head and dependent word.



nmod(flight, was) >> in-between words this morning not connected to flight >> not projective

Dependency Formalisms

- Dependency tree is a directed graph
  - G = (V,A)
  - V = Vertices = nodes = words or sometimes stems/affixes
  - A = Arcs = grammatical function relationships
  - Root node has no incoming A
  - Each V has one incoming A
  - Root node has a path to connect to every V
- Projectivity
  - An arc is 'projective' if for a pair(head, dependent) there is a path from the head to all words in-between the head and dependent word.
- Older parsing algorithms assume projective trees
  - English phrase-structure derived Treebanks guaranteed to be projective
  - Other languages often have hand annotated graphs so can include non-projective trees

Dependency Treebanks

- Treebanks
  - Annotated datasets by linguists for particular linguistic tasks
  - Penn Treebank >> Wall Street Journal news articles
  - Ontonotes >> Speech transcripts, weblogs, newsgroups, broadcasts
- Translation of parse-structures to dependency structures
  - Identify head-dependent structures (head rules)
  - Connect children to heads using a dependency relation
  - see Xia 2001 for more details
- Problems with translation
  - Cannot represent non-projective structures (as there are no examples in phrase-structured treebanks)
  - Lack of structure in flat noun phrases
- Non-English treebanks usually annotated manually

Cite: Xia, F. and Palmer, M. (2001). Converting dependency structures to phrase structures. HLT

Break

- Panopto Quiz - discussion point

- When would you prefer a dependency graph over a constituent-based structure?

For training named entity recognition models (e.g. noun phrase tagging)
For training relational extraction models (e.g. clause argument detection)
For machine translation models (e.g. Russian to English translation)

Break

- Panopto Quiz - discussion point

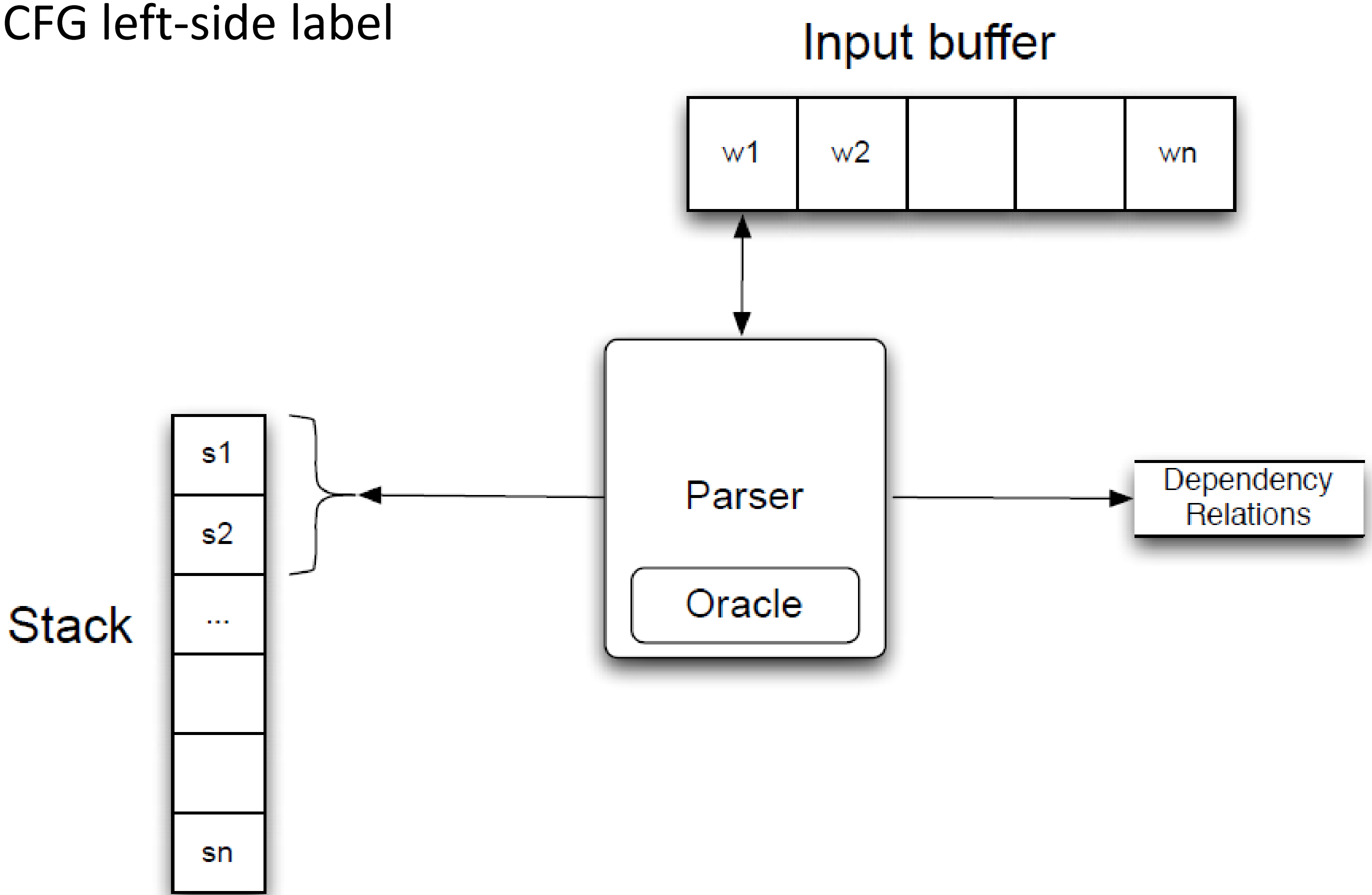
- When would you prefer a dependency graph over a constituent-based structure?

For training named entity recognition models (e.g. noun phrase tagging)
For training relational extraction models (e.g. clause argument detection)
For machine translation models (e.g. Russian to English translation)

Why? Clauses need subject/object grammar. For other apps dep graph will not hurt, but information around sequential syntactic features are likely to offer more information value.

Transition-based Dependency Parsing

- Classic shift-reduce parsing
  - Context Free Grammar + stack + list of words
  - Shift words to stack >> match word pairs to CFG >> replace word pair with CFG left-side label



Cite: Aho, A. V. and Ullman, J. D. (1972). The Theory of Parsing, Translation, and Compiling, Vol. 1. Prentice Hall

Transition-based Dependency Parsing

- Basic Transition-based Parser
  - s = Stack = graph so far = root node at start
  - b = Input buffer = words to try = words in sentence at start
  - Set of relations = dep tree = empty set at start
- Arc Standard approach >> simple and effective
  - Oracle >> provides correct transition operator given a configuration state
  - Transition operators

    >> LeftArc - connect 1st word of stack (head) with 2nd word (dep)

    >> RightArc - connect 2nd word of stack (dep) with 1st word (head)

    >> Shift - Move next word in buffer into the stack

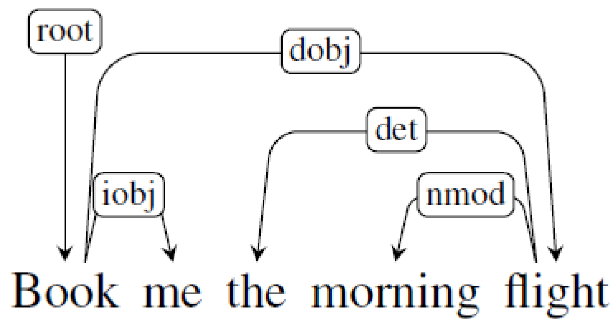
    >> Done - Finished

function DEPENDENCYPARSE(*words*) **returns** dependency tree

state ← {[root], [*words*], [] } ; initial configuration
while *state* **not final**
  t ← ORACLE(*state*) ; choose a transition operator to apply
  state ← APPLY(*t*, *state*) ; apply it, creating a new state
return *state*

Transition-based Dependency Parsing

- Basic Transition-based Parser
  - Stack = graph so far = root node at start
  - Input buffer = words to try = words in sentence at start
  - Set of relations = dep tree = empty set at start



Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Transition-based Dependency Parsing

- Problems with Arc Standard approach
  - Greedy algorithm >> there may be other parse trees that are viable also
  - Assumes Oracle is 100% perfect >> not true in practice
  - Action set is needed for each relation (nsubj, dobj, ...) >> large action space to process
- Alternatives
  - Arc Eager >> get words attached to heads earlier (than other dependants)
  - Beam Search >> consider alternative parse trees in parallel

Transition-based Dependency Parsing

- Creating an Oracle
- Supervised ML trained on treebanks
- Compile training examples from Treebank reference phrase structures
  - Training examples = (configuration, transition) x N
  - configuration = [stack] + [word list] + [existing relation set]
  - transition = action type e.g. leftarc(nsubj)
- Feature templates (like we used in NER lecture for CRF models)
  - Use front of stack/buffer only >> Reduce sparsity of feature space
  - Features can be lemma, POS, wordforms, word embeddings, dep relation ...

Source	Feature templates		
One word	$s_1.w$	$s_1.t$	$s_1.wt$
	$s_2.w$	$s_2.t$	$s_2.wt$
	$b_1.w$	$b_1.w$	$b_0.wt$
Two word	$s_1.w \circ s_2.w$	$s_1.t \circ s_2.t$	$s_1.t \circ b_1.w$
	$s_1.t \circ s_2.wt$	$s_1.w \circ s_2.w \circ s_2.t$	$s_1.w \circ s_1.t \circ s_2.t$
	$s_1.w \circ s_1.t \circ s_2.t$	$s_1.w \circ s_1.t$	

Transition-based Dependency Parsing

- Apply feature template to training example to produce X
  - Feature template = K feature types
  - X = value set for K feature types
  - Y = action
  - Now it is a standard supervised learning problem formulation
    >> Multinomial logistic regression, SVM, deep learning models

Graph-based Dependency Parsing

- Encode possible trees as directed graphs
  - Graph-based approaches can produce non-projective graphs and they avoid long-distant dependency issues by scoring entire trees
- Edge-factored approach to graph-based dependency parsing
  - Score for a tree is sum of the score for all its edges
  - S = sentence
  - t = possible parse tree
  - T(S) = best parse tree

$$\hat{T}(S) = \underset{t \in \mathcal{G}_S}{\operatorname{argmax}}\, score(t, S)$$

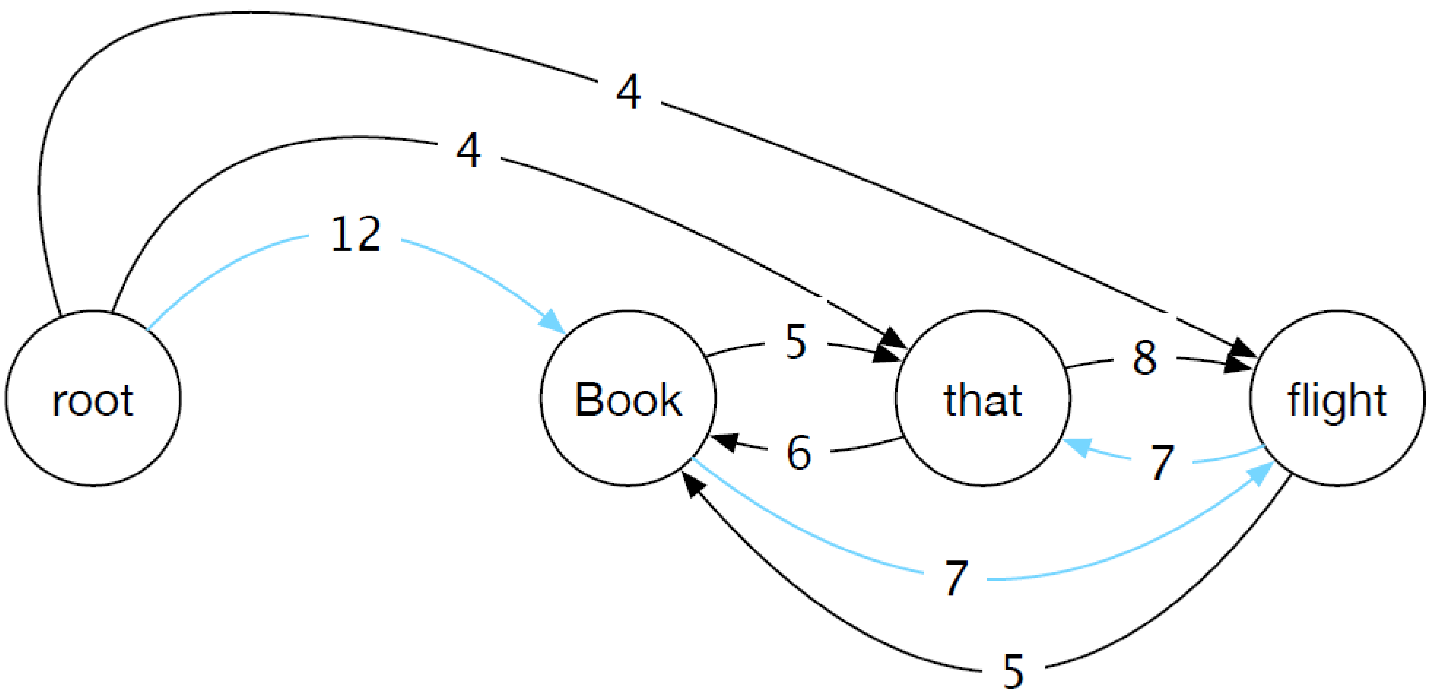
$$score(t, S) = \sum_{e \in t} score(e)$$

- Maximum spanning tree
  - Every vertex (node) has one incoming edge (parent)

Graph-based Dependency Parsing

- Parsing
  - Greedy selection >> choose most probable edge assignment
  - Cleanup >> avoid cycles by adjusting weights using heuristics

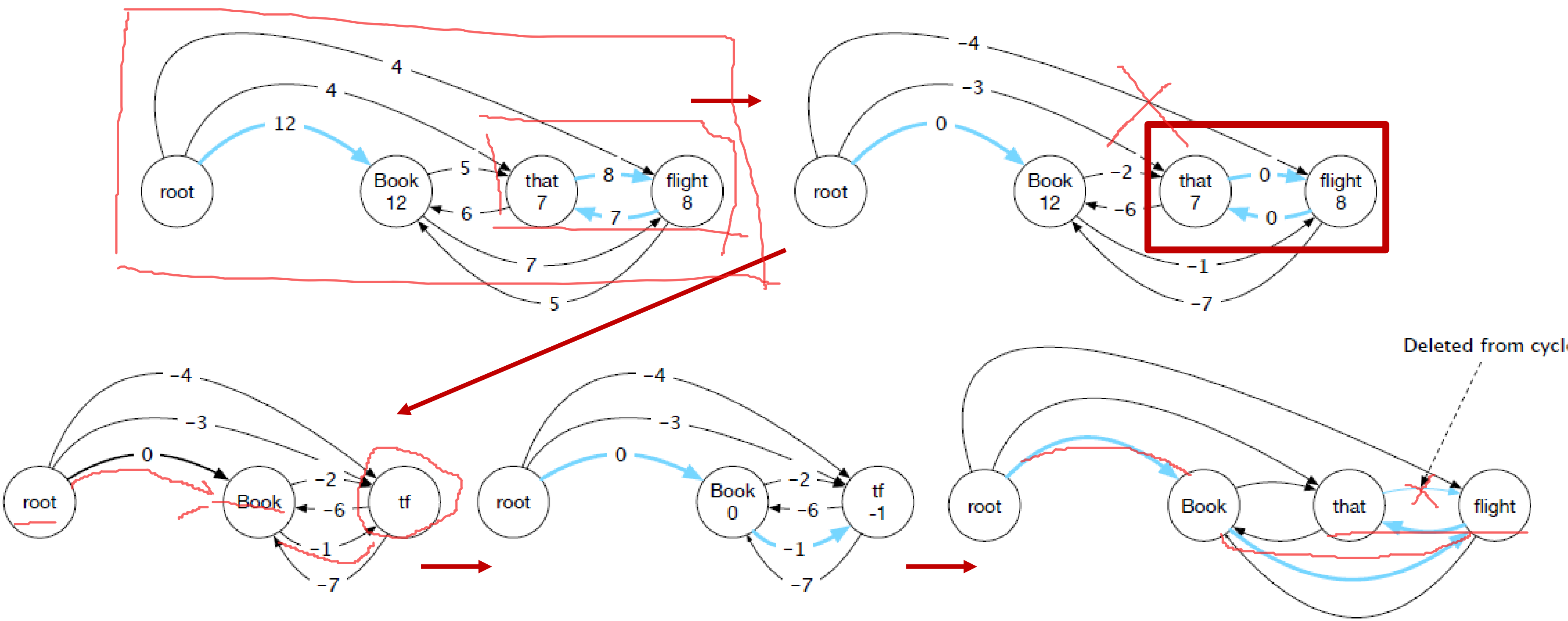
    subtract max edge weight (incoming) from all edge weights (incoming)
    collapse nodes in a cycle into a single node
    repeat process (recursively)



Graph-based Dependency Parsing

- Parsing
  - Greedy selection >> choose most probable edge assignment
  - Cleanup >> avoid cycles by adjusting weights using heuristics

    subtract max edge weight (incoming) from all edge weights (incoming)
    collapse nodes in a cycle into a single node
    repeat process (recursively)



Graph-based Dependency Parsing

- Training a supervised ML parser
- Apply feature template to training example to produce X
  - Feature template = K feature types
  - X = value set for K feature types
  - Y = edge probability/prediction
  - A reference parse tree from a Treebank is used for ground truth labels
  - This is now a standard supervised learning problem formulation
- Example
  - Dozat 2017's LSTM model using features from (word, POS and character) embeddings. Character embeddings help handle rare words not in training vocabulary.

Cite: Dozat, T., Qi, P., and Manning, C. D. (2017). Stanford’s graph-based neural dependency parser at the CoNLL 2017 shared task. Proceedings of the CoNLL 2017 Shared Task

Evaluation

- Metrics for evaluating dependency graph parsers
  - Exact Match
    >> very conservative, minor errors will cause most sentences to fail
  - Labelled Attachment Score (LAS)
    >> TP = correct assignment of [word -> head + dep relation]
  - Unlabelled Attachment Score (UAS)
    >> TP = correct assignment of [word -> head]
  - Label Accuracy Score (LS)
    >> percentage of words with correct edge label (ignoring where it came from)

Required Reading

- Dependency Parsing
  - Jurafsky and Martin, Speech and Language Processing, 3rd edition (online) >> chapter 14

Questions

- Panopto Quiz - 1 minute brainstorm for interactive questions

  Please write down in Panopto quiz in **1 minute** two or three questions that you would like to have answered at the next interactive session.

  Do it **right now** while its fresh.

  Take a screen shot of your questions and **bring them with you** at the interactive session so you have something to ask.