

CS 2731

Introduction to Natural Language Processing

Session 22: Dependency parsing

Michael Miller Yoder

November 10, 2025

Course logistics: project

- Project progress report **due this Thu Nov 13**
- Part 1: Task and dataset
 - Address the questions on basic dataset statistics, as well as how you will use your dataset to address your task
 - If you do not have a “traditional” dataset, present rough equivalents
- Part 2: Some kind of a result
 - Options: Baseline system evaluation on your dataset, a result from your own system, an example output from your system
- Part 3: Open questions and challenges
 - Need any help or additional resources?

Course logistics: homework

- Homework 4 is **due next Thu Nov 20**
 - More Hugging Face, this time BERT-based models for part-of-speech tagging

Review: What is the input and output format of sequence labeling tasks?

Overview: Dependency parsing

- What is syntax?
- Dependency grammar
 - Kinds of dependency in English
 - Dependencies and semantic roles
 - Dependency treebanks
- Dependency parsing
 - Transition-based dependency parsing
 - Projectivity
 - Evaluation
 - Tools and resources

What is syntax and why is it useful?

Syntax is Sentence and Phrase Structure

- Syntax concerns the patterns according to which words are combined to form phrases and sentences.
- It is distinct from morphology (how morphemes combine to form words) and semantics (what sentences mean) but is related to both.
- *Colorless green ideas sleep furiously.*

Syntax is the Door to Semantics

- To arrive at a semantic interpretation of a sentence, you have to know its syntax
- Parallel with programming languages
 - Semantics different from syntax (form versus function)
 - But semantics follows from syntax

Who Did What to Whom?

If you want to know **who** did **what** to **whom** with **what** thing having **what** properties, you must have access to syntax in some form.

Why do we need sentence structure (syntax)?

- Humans communicate complex ideas by composing words together into bigger units to convey complex meanings
- Human listeners need to work out what modifies (attaches to) what
- A model needs to understand sentence structure in order to be able to interpret language correctly
- Sometimes syntax can be ambiguous!

Ambiguity: prepositional phrase attachment



The image is a screenshot of the BBC News website. At the top, there is a navigation bar with the BBC logo on the left, a 'Sign in' button, and links to 'News', 'Sport', 'Weather', 'Shop', 'Reel', and 'Travel'. Below this is a large red banner with the word 'NEWS' in white capital letters. Under the banner is another navigation bar with links to 'Home', 'Video', 'World', 'US & Canada', 'UK', 'Business', 'Tech', 'Science', and 'Stories'. The 'Science' link is highlighted. Below the navigation bar, the text 'Science & Environment' is displayed and underlined. The main headline is 'Scientists count whales from space' in large, bold, black font. Below the headline, it says 'By Jonathan Amos' and 'BBC Science Correspondent'.

BBC Sign in News Sport Weather Shop Reel Travel

NEWS

Home Video World US & Canada UK Business Tech Science Stories

Science & Environment

Scientists count whales from space

By Jonathan Amos
BBC Science Correspondent

Ambiguity: prepositional phrase attachment

Scientists count whales from space



Ambiguity: coordination scope



Different perspectives on syntax

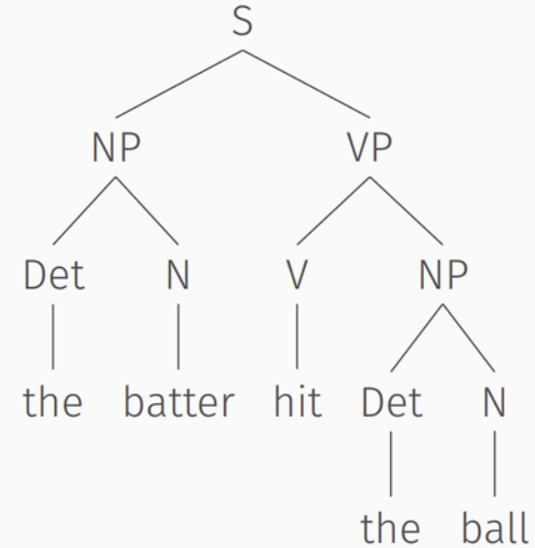
There are Two Major Approaches to Syntax in NLP

Two approaches:

- Syntax means taking sentences, dividing them into phrases and dividing those phrases into smaller phases until you arrive at individual words, yielding a tree of “constitutents”
- Syntax means taking sentences and characterizing the relationships between pairs of words in the sentence, yielding a tree or graph of “heads” and “dependents”

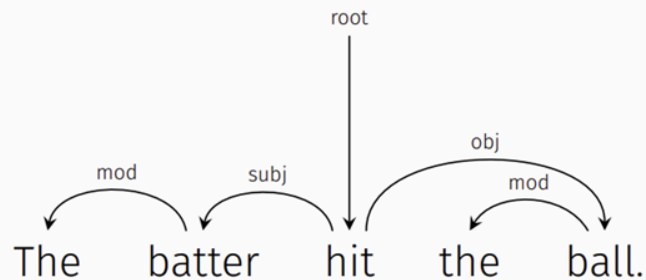
Phrase Structure Grammar is also Called Constituency Grammar

- The first approach is called PHRASE STRUCTURE GRAMMAR or CONSTITUENCY GRAMMAR.
- Basic unit — constituent
- Used by the parsers in the interpreters/compiler of most programming languages



Dependency Grammar Is Based On Bilexical Dependencies

- The second approach is called DEPENDENCY GRAMMAR.
- Basic element — BILEXICAL DEPENDENCY
- Especially useful for many contemporary NLP tasks



Dependency grammar

Words Relate to Other Words

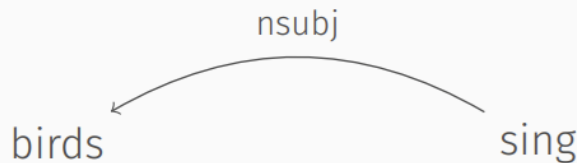
Words relate to other words:

- Nouns can be subjects or objects of verbs
- Adjectives can be modifiers of nouns
- Adverbs can be modifiers of verbs, adjectives, and other adverbs

Dependency grammar seeks to capture these relations (subject, object, modifier, etc.)

The Bilexical Dependency is the Basic Unit of Dependency Grammar

The basic unit in dependency grammar is a bilexical dependency, a “link” between two words: a head (governor) and a dependent



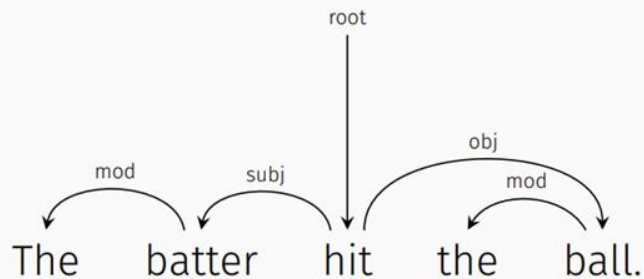
Dependents Contribute to the Meaning of Heads

Head provides the basic content (meaning, grammatical content)

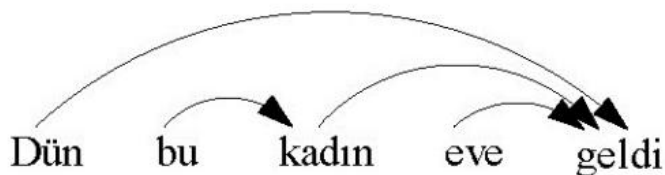
Dependent modifies or serves as an argument of the head

Dependencies Form a Tree

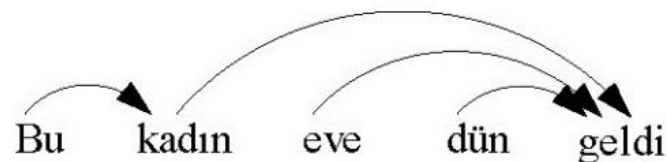
- Typically, the head of a sentence is a verb
- Every word is the dependent of one head
- The head verb is a dependent of ROOT



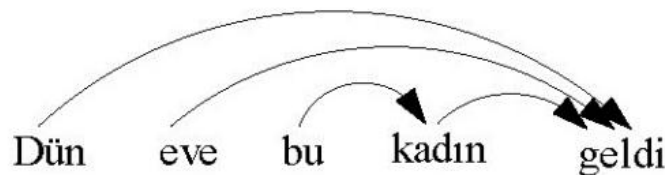
Dependencies are useful for languages with free word order



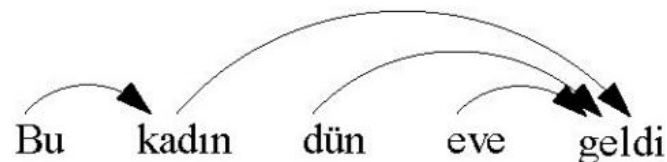
Yesterday, this woman came home



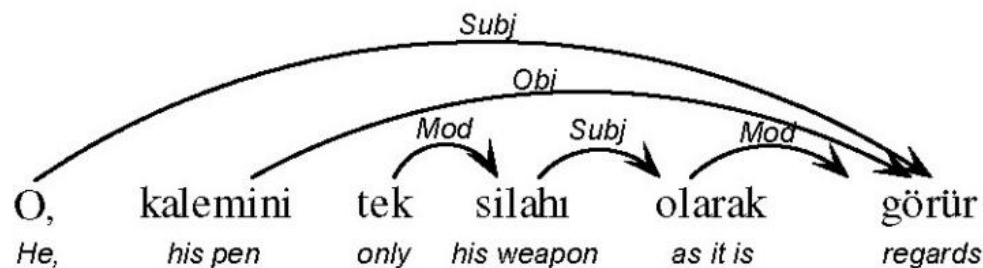
This woman came home yesterday



It was this woman who came home yesterday



This woman came home yesterday (as opposed to going elsewhere)



Kinds of dependency in English

Dependency Relations for Universal Dependencies

	Nominals	Clauses	Modifier words	Function Words
Core arguments	<u>nsubj</u> <u>obj</u> <u>iobj</u>	<u>csbj</u> <u>ccomp</u> <u>xcomp</u>		
Non-core dependents	<u>obl</u> <u>vocative</u> <u>expl</u> <u>dislocated</u>	<u>advcl</u>	<u>advmod</u> * <u>discourse</u>	<u>aux</u> <u>cop</u> <u>mark</u>
Nominal dependents	<u>nmod</u> <u>appos</u> <u>nummod</u>	<u>acl</u>	<u>amod</u>	<u>det</u> <u>clf</u> <u>case</u>
Coordination	MWE	Loose	Special	Other
<u>conj</u> <u>cc</u>	<u>fixed</u> <u>flat</u> <u>compound</u>	<u>list</u> <u>parataxis</u>	<u>orphan</u> <u>goeswith</u> <u>reparandum</u>	<u>punct</u> <u>root</u> <u>dep</u>

Six Dependency Relations Common in English

nsubj the subject noun of a verb

obj the object of a verb

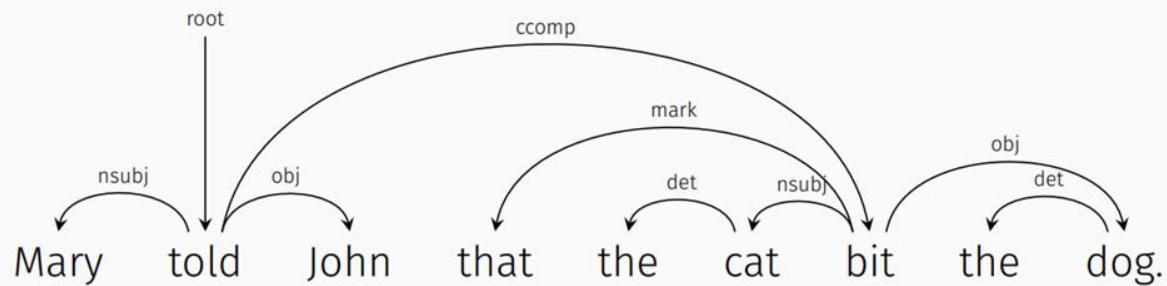
ccomp the complement of a verb

amod the adjectival modifier of a noun

det the determiner of a noun

mark a word marking a clause as subordinate

An Illustration of Six UD Relations



Dependencies and who did what to whom?

Semantic Roles Are Important to NLP

Often, in NLP, we want to know the semantic roles (thematic roles) of the noun phrases in a sentence.

Agent the doer of an action

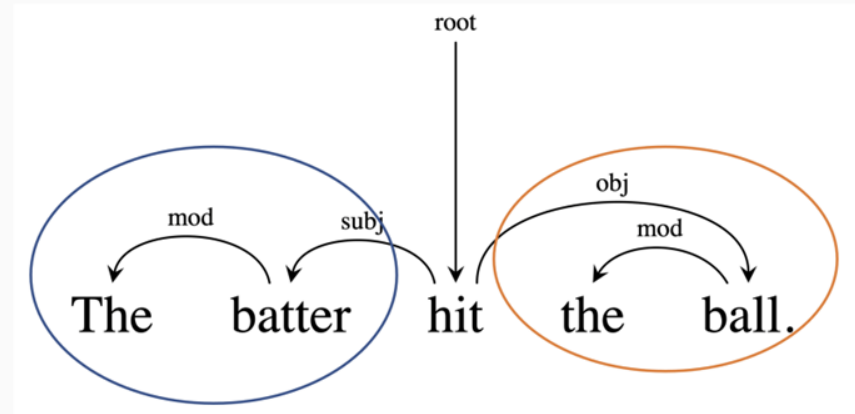
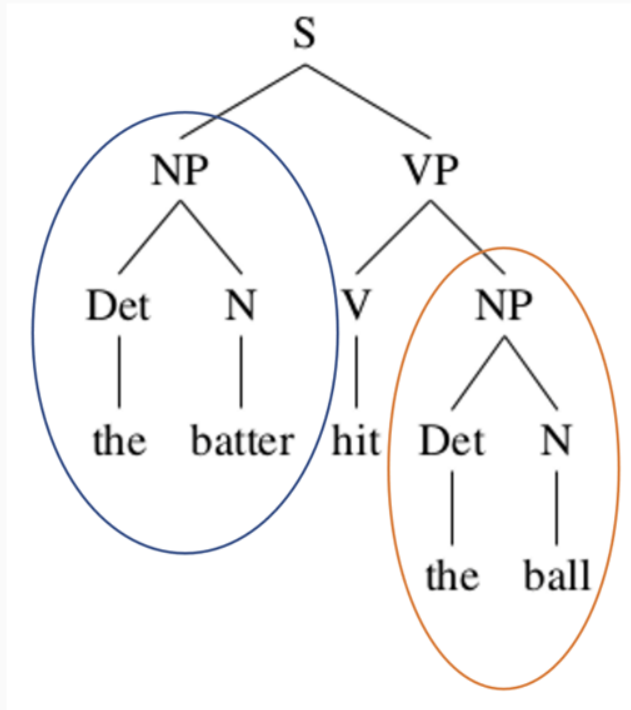
Patient the one to whom an action is done

Instrument that with which an action is done
etc.

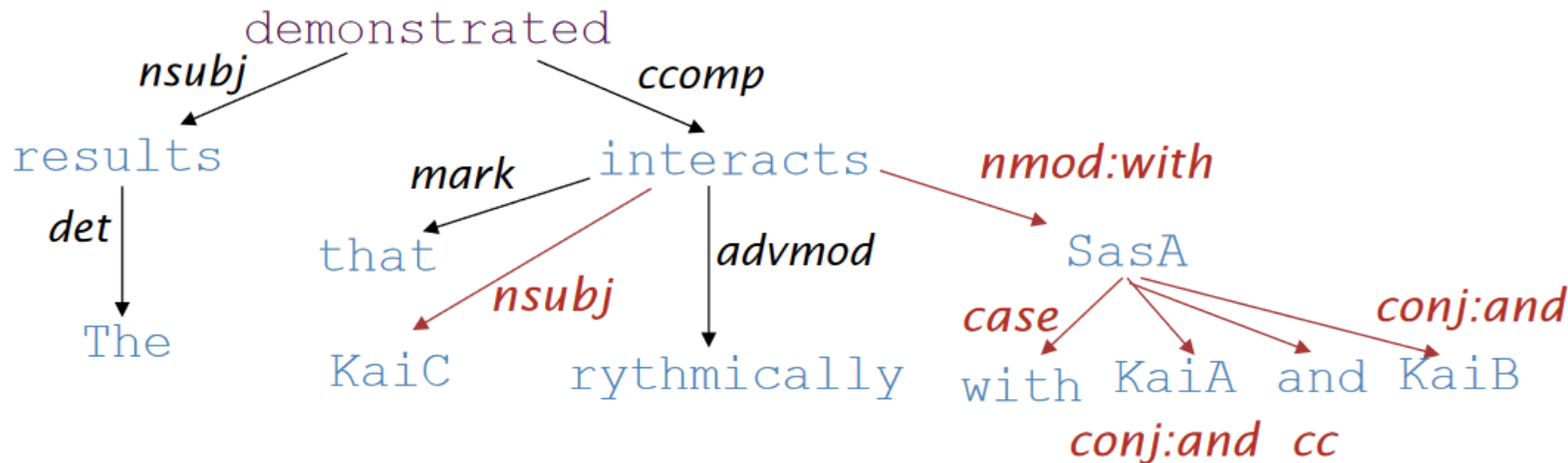
Semantic Roles are Related to but not Identical to, Grammatical Relation

- These are not the same as subject, object, etc.
- However, there is a function from grammatical relations like subject and object to thematic roles
- Syntax \Rightarrow grammatical relations \Rightarrow semantic/thematic roles

Dependency Trees Encode Grammatical Relations Directly



Practical example: extracting protein-protein interaction



KaiC \leftarrow nsubj interacts nmod:with \rightarrow SasA

KaiC \leftarrow nsubj interacts nmod:with \rightarrow SasA conj:and \rightarrow KaiA

KaiC \leftarrow nsubj interacts nmod:with \rightarrow SasA conj:and \rightarrow KaiB

[Erkan et al. EMNLP 07, Fundel et al. 2007, etc.]

Dependency treebanks

Proper Ambivalence toward Treebanks



Why you should have great
respect for treebanks



Why you should be cautious
around treebanks

Why You Should Respect Treebanks

Treebanks require great skill

- Expert linguists make thousands of decisions
- Many annotators must remember all of the decisions and use them consistently, including knowing which decision to use
- The “coding manual” containing all of the decisions is hundreds of pages long

Treebanks take many years to make

- Writing the coding manual, training coders, building user-interface tools, etc., all take a lot of time
- So does the actual coding of the data and quality assurance

Treebanks are expensive

Somebody has to secure funding for these projects

You Should Be Cautious around Treebanks

- They are **too big to fail**
- They are **produced under pressure** of time and funding
- Although most of the decisions are made by experts, **most of the coding is done by non-experts**

Universal Dependencies Treebanks

- Over 200 treebanks in almost 100 languages
- UD annotation scheme
- Standard, easy to process, U-CONLL file format
- Despite attempts at standardization, considerable variation in conventions, quality

Dependency parsing

Dependency Tree: Definition

Let $\mathbf{x} = [x_1, \dots, x_n]$ be a sentence. We add a special ROOT symbol as “ x_0 ”.

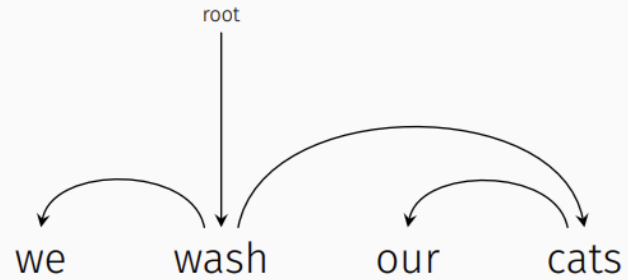
A dependency tree consists of a set of tuples $[p, c, \ell]$ where

- $p \in \{0, \dots, n\}$ is the index of a *parent*.
- $c \in \{1, \dots, n\}$ is the index of a *child*.
- $\ell \in \mathcal{L}$ is a label.

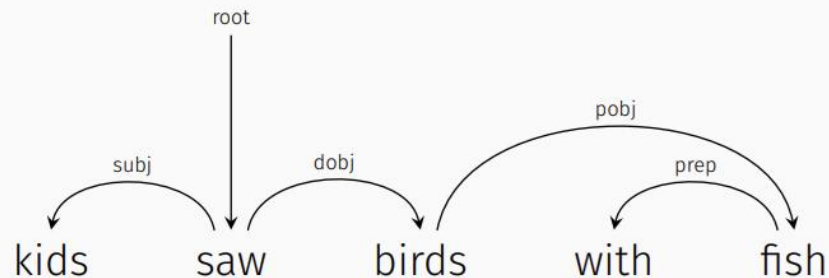
Different annotation schemes define different label sets \mathcal{L} , and different constraints on the set of tuples. Most commonly:

- The tuple is represented as a directed edge from x_p to x_c with label ℓ .
- The directed edges form an directed tree with x_0 as the root (sometimes denoted as ROOT).

Example



"Bare bones" dependency tree.



Key dependency relations captured in the labels include:

- Subject (arrow from predicate/main verb to subject)
- Direct Object (arrow from verb to object)
- Indirect Object (arrow from verb to object)
- Preposition Object (arrow from main noun to object of the preposition)
- Adjectival Modifier (arrow from noun to modifying adjective)
- Adverbial Modifier (arrow from noun to modifying adverb)

Practice: parse these sentences

1. Enraged cow injures farmer with ax.
2. Hospitals are sued by seven foot doctors.
3. The woman saw the man with the telescope.

Key dependency relations captured in the labels include:

- Subject
- Direct Object
- Indirect Object
- Preposition Object
- Adjectival Modifier
- Adverbial Modifier

Two approaches to dependency parsing

Transition-based parsing

- Proceed through a sequence of actions, building up a representation step by step
- The representation, and any step, depends on the representations that came before

Graph-based parsing

- Start with probabilities for each edge
- Apply some sort of dynamic programming

Transition-based dependency parsing

Transition-based dependency parsing

- Process input from left-to-right once, making a sequence of greedy parsing decisions
- Represents the current state/configuration of the parse:
 - Stack
 - Buffer
 - Current set of relations
- In arc-standard parsing, possible actions are:
 - SHIFT: move first word in the buffer to the stack
 - LEFT-ARC: draw an arc from word in the top of the stack to second word in the stack; remove dependent word (second word)
 - RIGHT-ARC: draw an arc from second word in the stack to the top of the stack; remove dependent word (top of the stack)

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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	
3			SHIFT	
4			SHIFT	
5			SHIFT	
6			LEFTARC	
7			LEFTARC	
8			RIGHTARC	
9			RIGHTARC	
10			Done	

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

Example of transition-based parsing

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

Figure 18.6 Trace of a transition-based parse.

How does the parser know which step to take next?

- This is a three-way **classification** problem (or, for parsing labeled dependencies, more)
- Various classifiers have been used
 - Traditional classifiers
 - Feed-forward neural nets
 - etc.
- What features? Stay tuned!

The Core of Transition-based Parsing

- At each iteration, choose among {SHIFT, RIGHT-ARC, LEFT-ARC}.
 - Actually, among all \mathcal{L} -labeled variants of RIGHT- and LEFT-ARC.
- Training data: Dependency treebank trees converted into “oracle” transition sequence.
 - These transition sequences give the right tree,
 - $2 \cdot n$ pairs: $\langle state, correcttransition \rangle$.
 - Each word gets SHIFTed **once** and participates as a child in **one** ARC.

Features for Transition Parsing Come from the Configuration

Where do the features for making parsing decisions come from?

- The words in the buffer
- The words in the stack (e.g. the roots of the trees)
- The children of these roots
- The POS tags of the words
- History of actions

Feature combinations are important:

- When parsing English, suppose that the second word in S is a verb and the first is a noun.
- The model should probably choose LEFT-ARC

Example of Features: Feed-Forward Neural Transition Parser

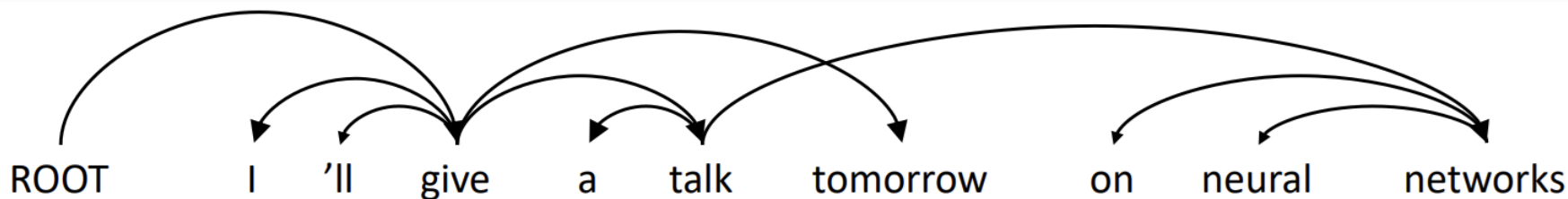
Here are the features extracted by Chen and Manning's (2014) feed-forward neural model for shift-reduce parsing:

- The top three words on S and B (6 features)
 $s_1, s_2, s_3, b_1, b_2, b_3$
- The two leftmost/rightmost children of the top two words on S (8 features)
 $lc_1(s_i), lc_2(s_i), rc_1(s_i), rc_2(s_i) \ i = 1, 2$
- The leftmost and rightmost grandchildren (4 features)
 $lc_1(lc_1(s_i)), rc_1(rc_1(s_i)) \ i = 1, 2$
- POS tags for all words invoked above (18 features)
- Arc labels of all children/grandchildren invoked above (12 features)

Projectivity

Projectivity

- Definition of a **projective parse**: There are no crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words
- Most syntactic structure is projective like this, but dependency theory normally does allow non-projective structures to account for displaced constituents
 - You can't easily get the semantics of certain constructions right without these nonprojective dependencies



Handling non-projectivity

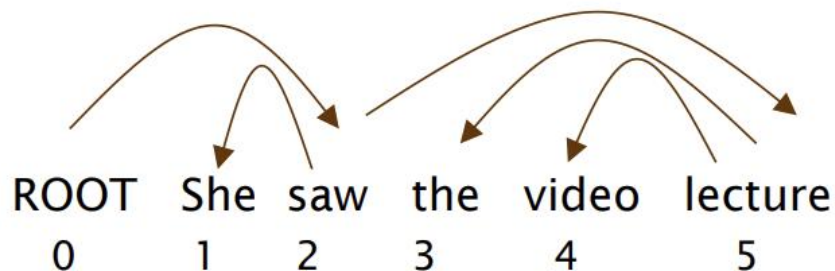
- The arc-standard algorithm we just presented only builds projective dependency trees
- Possible directions to head:
 1. Just declare defeat on nonprojective arcs 🤖
 2. Use a postprocessor to a projective dependency parsing algorithm to identify and resolve nonprojective links
 3. Add extra transitions that can model at least most non-projective structures (e.g., add an extra SWAP transition will allow any non-projectivity)
 4. Move to a parsing mechanism that does not use or require any constraints on projectivity (e.g., the graph-based MSTParser or Dozat and Manning (2017))

Evaluation

Dependency Parsing Evaluation

- **Unlabeled attachment score (UAS):** Did you identify the head and the dependent correctly?
- **Labeled attachment score (LAS):** Did you identify the head and the dependent AND the label correctly?

Evaluation: an example



$$\text{Acc} = \frac{\text{\# correct deps}}{\text{\# of deps}}$$

$$\text{UAS} = 4 / 5 = 80\%$$

$$\text{LAS} = 2 / 5 = 40\%$$

Gold

1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

Parsed

1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

Tools and resources for dependency parsing

Dependency parsers

- UDPipe
 - Widely used
 - Provides parsing, morphological analysis, etc
 - A little harder to use than Stanza
- Stanza
 - New version of the classic Stanford Parser (which was in Java)
 - Pure Python
- spaCy (English)
 - Convenient Python library
 - Performs many other NLP tasks in addition to parsing
 - For the most part, is English-only

Wrapping up

- Syntax concerns rules for grouping and ordering words into meaningful phrases and sentences
- Constituencies and dependencies are two high-level formalisms for syntax
- The dependency grammar formalism models syntactic head-dependent relationships between words
- Dependency relationships are key to understanding who did what to whom (semantic roles)
- Key families of algorithms for dependency parsing include transition-based and graph-based parsers

Questions?