11-411/11-611 Natural Language Processing

Syntax

David R. Mortensen and Lori Levin

March 28, 2023

Language Technologies Institute

Learning Objectives

- Describe dependency grammar and how it differs from phrase structure (or constituency) grammar.
- Identify several NLP applications for dependency parses.
- Describe the Universal Dependencies and be able to apply several common dependency relations that it defines.
- Implement transition-based parsing for dependencies.
- Be able to find appropriate pretrained models dependency parsing supporting a range of tasks.

Different perspectives on syntax

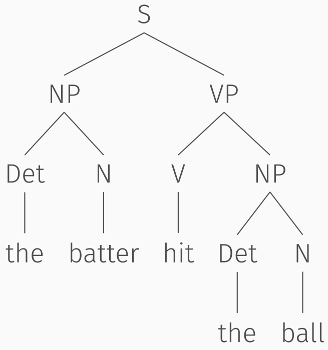
There are Two Major Approaches to Syntax in NLP

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”

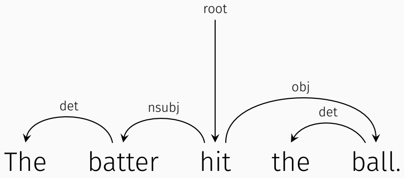
Phrase Structure Grammar is also Called Constituency Grammar

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



Dependency Grammar Is Based On Bilexical Dependencies

- The second approach is called DEPENDENCY GRAMMAR.
- Basic element — BILEXICAL DEPENDENCY



Introduction to Dependency Grammar

Words Relate to Other Words

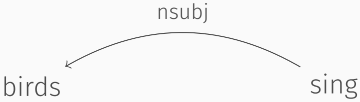
Words relate to other words:

- 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 represents these relations (subject, object, modifier, etc.)

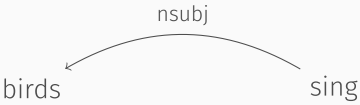
The Bilexical Dependency is the Basic Unit of Dependency Grammar

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



The Bilexical Dependency is the Basic Unit of Dependency Grammar

In this tree for the sentence *Birds sing*, *sing* is the head because the sentence refers to a singing event. *Sing* is portrayed as a predicate that takes one argument, *birds*. The arrow points from the head (*sing*) to the dependent (*birds*). The label on the arc indicates the type of dependency. In this case, *birds* is the ***`nsubj`*** (noun subject) of *sing*.



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

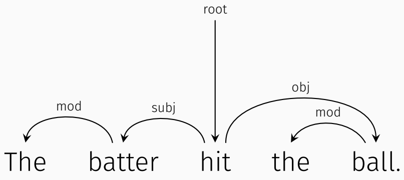
What is a head?

Each phrase (noun phrase, verb phrase, prepositional phrase) has a word that makes it the kind of phrase it is.

- A noun is the head of a noun phrase (the tall **student** in the back row)
- A verb is the head of a verb phrase (quickly **ate** a lot of doughnuts)
- A verb is also the head of a sentence (I **ate** a lot of doughnuts.)
- A preposition is the head of a prepositional phrase (**in** the classroom)
- An adjective is the head of an adjective phrase (so very **proud** of my son)

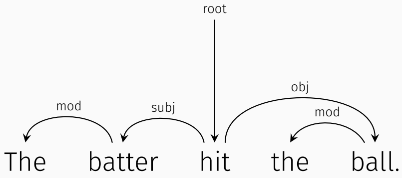
Dependencies Form a Tree

We will focus on dependency representations where all the words are connected, each dependent depends on exactly one head, and the arcs don't cross, although, we will see later that not all sentences can be represented this way.

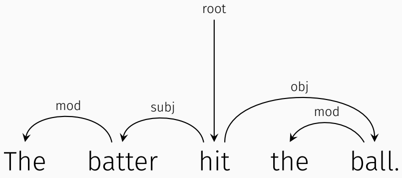


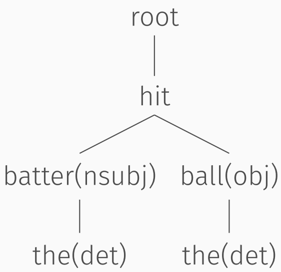
Dependencies Form a Tree

- 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



Three notations for dependency trees





1	the	2	det
2	batter	3	nsubj
3	hit	root	
4	the	5	det
5	ball	3	obj

UD Treebank Example

```
# sent_id = GUM_academic_discrimination-4
# s_prominence = 4
# s_type = decl
# transition = null
# text = Personal experiences of discrimination and bias have been the focus of much social science research.
# newpar
# newpar_block = p (6 s)
1	Personal	personal	ADJ	JJ	Degree=Pos	2	amod	2:amod	Discourse=cont
2	experiences	experience	NOUN	NNS	Number=Plur	10	nsubj	10:nsubj	_
3	of	of	ADP	IN	_	4	case	4:case	_
4	discrimination	discrimination	NOUN	NN	Number=Sing	2	nmod	2:nmod:of	Entity
5	and	and	CCONJ	CC	_	6	cc	6:cc	_
6	bias	bias	NOUN	NN	Number=Sing	4	conj	2:nmod:of|4:conj:and	Entity=(10-abs
7	have	have	AUX	VBP	Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin	10	aux
8	been	be	AUX	VBN	Tense=Past|VerbForm=Part	10	cop	10:cop	_
9	the	the	DET	DT	Definite=Def|PronType=Art	10	det	10:det	Entity=(11-abs
10	focus	focus	NOUN	NN	Number=Sing	0	root	0:root	_
11	of	of	ADP	IN	_	15	case	15:case	_
12	much	much	ADJ	JJ	Degree=Pos	15	dep	15:dep	Entity=(12-abstract-new-cf5-4-
13	social	social	ADJ	JJ	Degree=Pos	14	amod	14:amod	Entity=(13-abstract-new-cf8-2-
14	science	science	NOUN	NN	Number=Sing	15	compound	15:compound	Entity=13)
15	research	research	NOUN	NN	Number=Sing	10	nmod	10:nmod:of	Entity
16	.	.	PUNCT	.	_	10	punct	10:punct	_
```

About dependency trees

- Each word is dependent on one other word
- To build a tree, you have to decide for each word, which other word it is a dependent of and what its relationship is

1	the	2	det
2	batter	3	nsubj
3	hit	root	
4	the	5	det
5	ball	3	obj

Build a dependency tree without labels

1	Mary	
2	told	root
3	John	
4	that	
5	the	
6	cat	
7	bit	
8	the	
9	dog	

Build a dependency tree without labels

1	Mary	2
2	told	root
3	John	2
4	that	7
5	the	
6	cat	
7	bit	
8	the	
9	dog	

Build a dependency tree without labels

1	Mary	2
2	told	root
3	John	2
4	that	7
5	the	6
6	cat	
7	bit	
8	the	9
9	dog	

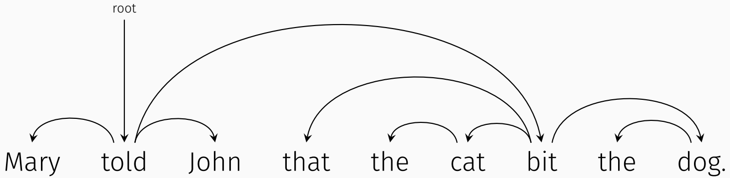
Build a dependency tree without labels

1	Mary	2
2	told	root
3	John	2
4	that	7
5	the	6
6	cat	7
7	bit	
8	the	9
9	dog	7

Build a dependency tree without labels

1	Mary	2
2	told	root
3	John	2
4	that	7
5	the	6
6	cat	7
7	bit	2
8	the	9
9	dog	7

The tree (without labels)



What is a treebank?

A treebank is a corpus of sentences where each sentence has been parsed by humans or parsed with a parser and corrected by humans.

- There are phrase structure treebanks such as the Penn Treebanks.
- There are dependency treebanks such as the Universal Dependency Treebanks and the Prague Dependency Treebanks.

What is Universal Dependency

Universal Dependency (UD) started in 2013 with the goal of having treebanks for many languages using the same annotation guidelines. There are now hundreds of UD treebanks.

UD Dependency Labels

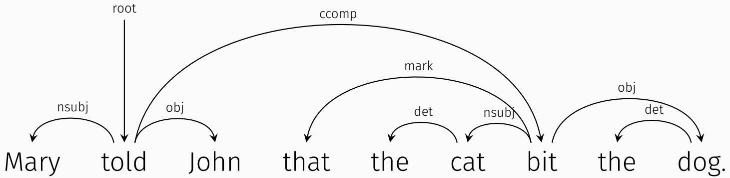
37 Dependency labels for Universal Dependencies

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

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

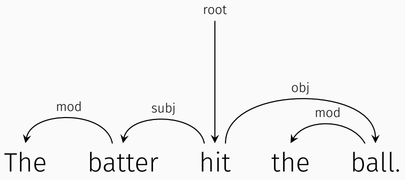


Some problems with dependency trees

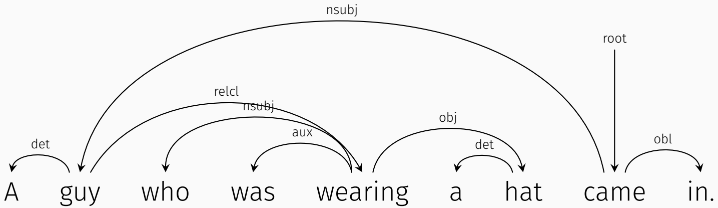
Three issues concerning dependency trees

- Non-projectivity
- Dependents connected to more than one head
- What to do with *and*

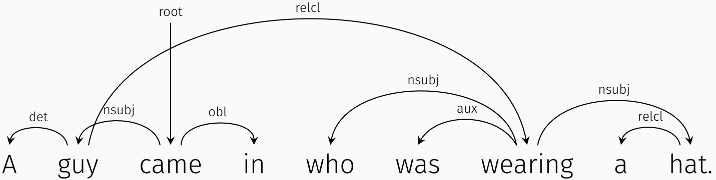
Projective dependency trees: the arcs don't cross



Projective: the arcs don't cross

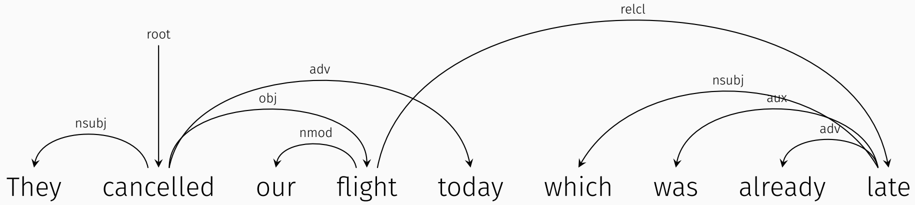


Non-projective: crossing arcs



The relcl arc crosses the root arc. **The relcl arc from “guy” to “wearing” is non-projective because there is not a path from “guy” to every word between “guy” and “wearing”.** In particular, there is not a path from “guy” to “came” and “in” following arrows only in the direction they are pointing.

Non-projective: crossing arcs

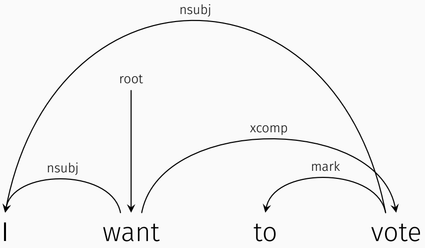


The arc from “flight” to “late” is non-projective because there is not a path from “flight” to every word between “flight” and “late”. In particular, there is no path from “flight” to “today”.

Graphs vs Trees

- In a tree, each node has one parent.
- In a graph, a node can have more than one parent.

This is a graph

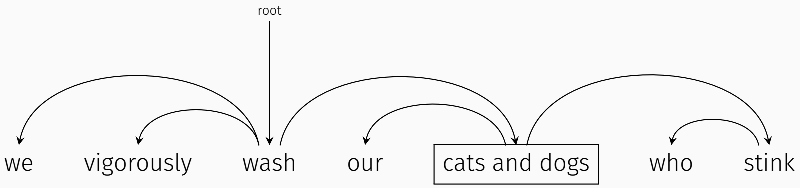


This graph indicates that “I” is the subject of “want” and “I” is also the subject of “vote”. This is important for semantic interpretation. We need to know the arguments of each verb.

Why are non-projectivity and graphs important?

Graphs and non-projective trees are more linguistically expressive than projective trees, but they require different algorithms than projective trees. So, you have to weigh the pros and cons of allowing graphs and non-projectivity.

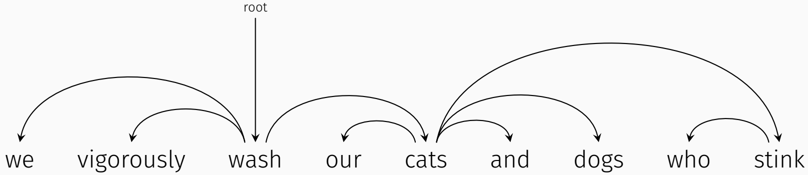
It Is not Obvious How to Model Coordination in Dependency Grammars



Most likely the most important problem UD.

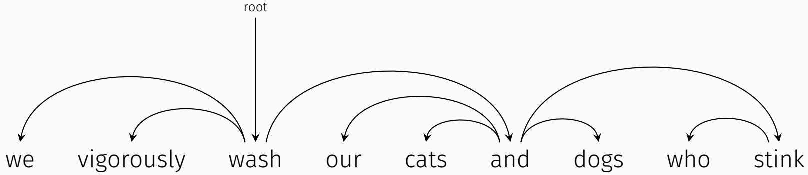
Terminology: the part of speech of *and* is *conjunction*. *Cats and dogs* is a *conjoined* noun phrase. *Cats* and *dogs* are *conjuncts*. A conjoined phrase is also called a *coordinate structure*, an instance of the phonomenon of *coordination*.

Proposal 1: The First Conjunct is the Head



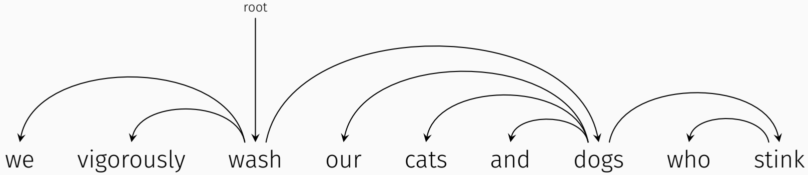
Make the first conjunct head?

Proposal 2: The Conjunction Is the Head



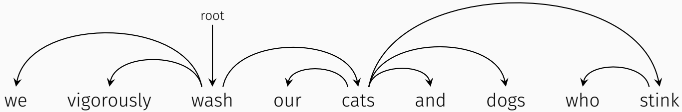
Make the coordinating conjunction the head?

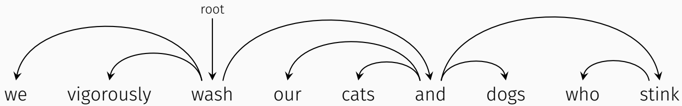
Proposal 3: the Second Conjunct Is the Head

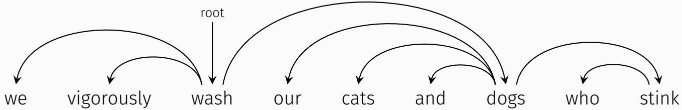


Make the second conjunct the head?

The Three Proposals for Coordination, Compared







What is a common property among these trees?

Dependency labels are grammatical relations, not semantic roles

Semantic Roles Are Important to NLP

Often, in NLP, we want to know the semantic 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.

The	student	smacked	the	moth	with	a	swatter.
	AGENT			PATIENT			INSTRUMENT

What are grammatical relations?

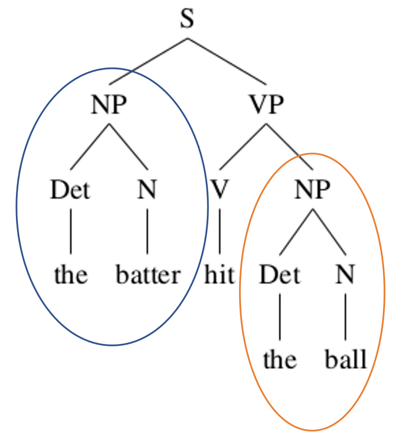
Grammatical relations affect the order of words and the presence of morphemes.

- Subject (in English)
    - before the verb
    - agrees with the verb (She runs/We run)
    - can be I, me, he, she, we, they
- Object (in English)
    - immediately after the verb
    - can be me, him, her, us, them
- Oblique (in English)
    - preceded by a preposition (at, on, by, over)
    - can be me, him, her, us, them

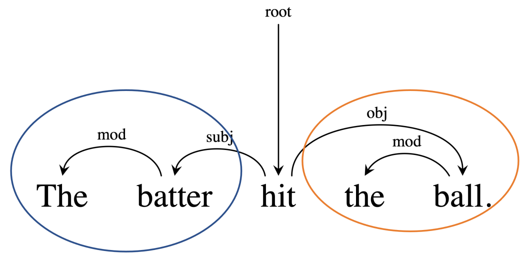
Semantic roles are not identical to grammatical relations

- Subject is agent
  - The student opened the door with a key.
- Subject is patient
  - The door was opened with a key.
  - The door opens with a key.
- Subject is instrument
  - The key opened the door.
- There is an agent, but it isn't the subject.
  - The door was opened by the student.

Dependency Labels are Grammatical Relations



Constituency trees represent grammatical relations indirectly. Subject is the NP under S. Object is the NP under VP.



Dependency labels represent grammatical relations explicitly.

Later in the semester we will learn about representations of semantic roles

The syntax of English questions in dependency grammar

Types of questions

- Yes-no questions
  - Can I play now?
  - Do you want some ice cream?
- Wh-questions
  - What is your name?
  - Who are you?
  - Why are you doing that?
- Alternative questions
  - Is she a student or a teacher?
- Tag questions
  - You’re going, aren’t you?
  - They haven’t left, have they?

English Auxiliary Verbs

Have	not always an auxiliary verb
Be	copula
Will	modal
Would	modal
Can	modal
Could	modal
Shall	modal
Should	modal
May	modal
Might	modal
Must	modal

Sequences of English auxiliary verbs

The order of auxiliary verbs in English is modal-have-be-be. They are all optional, but when they are present, they have to be in that order. The longest sequence of auxiliary verbs you can have is one modal auxiliary (will, would, can, could, shall, should, may, might, must) followed by "have" followed by two kinds of "be".

They **must have been being** arrested.

Auxiliary verbs in yes-no questions: Part 1

Put the first auxiliary verb before the subject. The auxiliary verb will not necessarily be at the beginning of the sentence.

Yesterday she **was** swimming.	Yesterday **was** she swimming?
When they got home, they **could** have been drunk.	When they got home, **could** they have been drunk?

Auxiliary verbs in yes-no questions: Part 2

If there is no auxiliary verb, put “do”, “does”, or “did” before the subject.

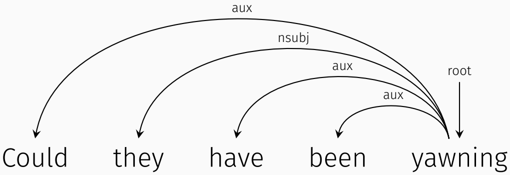
They ate potatoes.	**Did** they eat potatoes?
They eat potatoes.	**Do** they eat potatoes?
He laughs a lot.	**Does** he laugh a lot?
You yawn during lectures.	**Do** you yawn during lectures?

Wh-questions

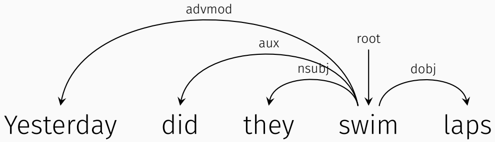
To make a sentence into a wh-question, you have to remove something from the sentence and put a corresponding wh-word at the front of the sentence. You also have to put an auxiliary verb before the subject (unless you have removed the subject).

I saw **Alex**	**Who did** I see?
Alex saw me	**Who** saw me?
I aced it **by studying**.	**How did** you ace it?
I am studying **to get an A**.	**Why** are you studying?

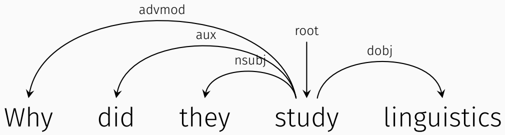
Dependency parses for questions



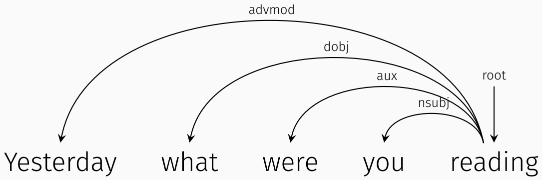
Dependency parses for questions



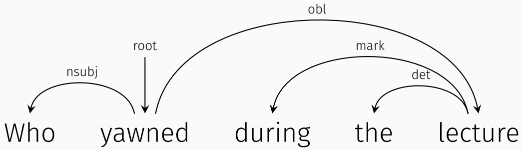
Dependency parses for questions



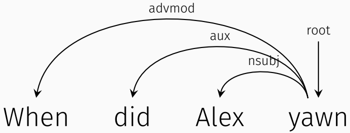
Dependency parses for questions



Dependency parses for questions

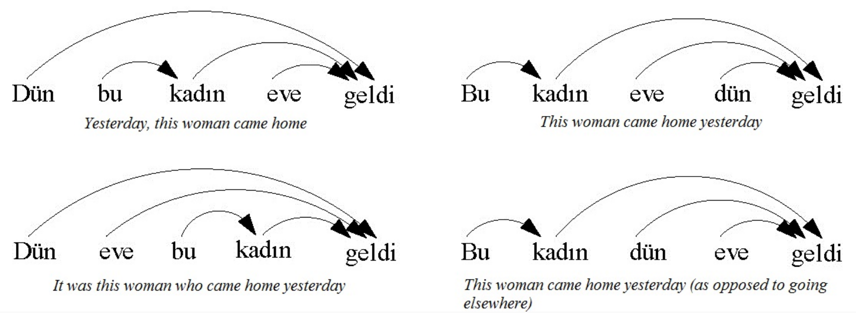


Dependency parses for questions



Examples of Dependency Trees from Other Languages

Dependency Examples from Turkish: free word order

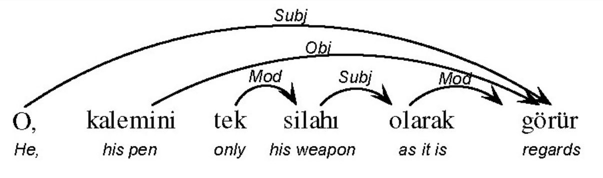


Note that in these examples, the arrows point from dependent to head.

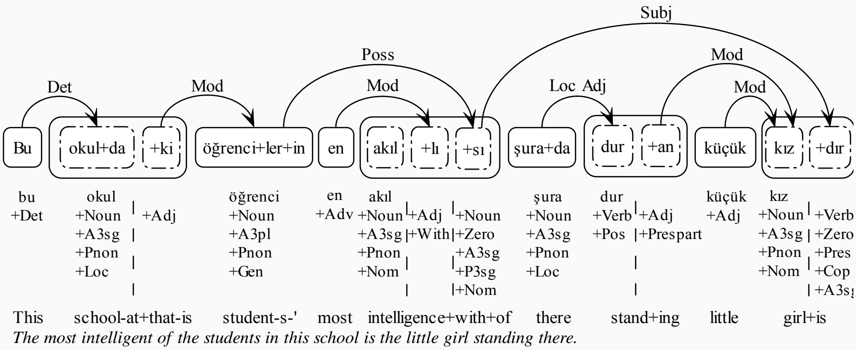
Turkish Lexicon:

dün	(yesterday)	bu	(this)
geldi	(came)	kadin	(woman)
eve	(to home)		

Translate this sentence into English

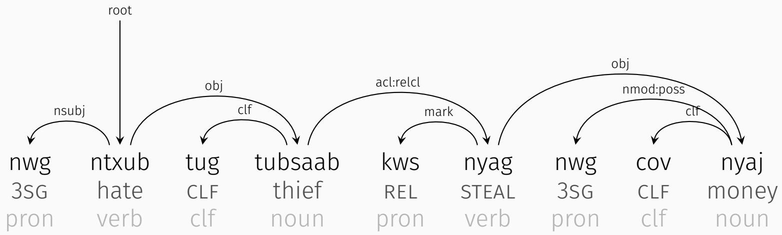


Dependency Examples from Other Languages



A Dependency Tree from Mong Leng

A sentence from Mong Leng (also called Green Hmong or 苗语)



Dependency Treebanks

Dependency Treebanking Is Easy (or Easier)

Creating dependency treebanks seems to be easier than constituency treebanks:

- Simpler data structure
- More intuitive tools
- Requires less expertise
  - Deciding which word to connect to and what label to use is easier to learn than testing for constituents.

As a result, there are now many dependency treebanks.

Universal Dependencies Treebanks

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

Tools and resources for dependency parsing

Dependency Parsers: UD

- UDPipe
  - Widely used
  - Provides parsing, morphological analysis, etc.
  - A little harder to use that Stanza
- Stanza
  - Newer than UDPipe
  - Performs better (in my experience)
  - Pure Python—easier to integrate into NLP systems

Dependency Parsers: Other

- Stanford Parser
  - High quality and widely used
  - Pretrained models for English and Chinese
  - Java (using efficiently from Python is not hard, but not easy)
- SpaCy (English)
  - Convenient Python library
  - Performs many other NLP tasks (in addition to parsing)
  - For the most part, English only

Dependency Parsing

Dependency Tree: Definition

Let $\mathbf{x} = [x_1, \ldots, x_n]$ 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

- $p \in \{0, \ldots, n\}$ is the index of a *parent*.
- $c \in \{1, \ldots, n\}$ 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).

Two Approaches to Dependency Parsing

1. **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
    - **Compare**: HMMs for POS tagging
2. **Graph-based parsing**
    - Start with probabilities for each edge (in phrase structure parsing, each constituent)
    - Apply some sort of dynamic programming
    - **Compare**: CRF for POS tagging

- Process $\mathbf{x}$ once, from left to right, making a sequence of greedy parsing decisions.

- Process $\mathbf{x}$ once, from left to right, making a sequence of greedy parsing decisions.
- Formally, the parser is a **state machine** (*not* a finite-state machine) whose state is represented by a stack S and a buffer B (which you can also view as a queue).

Transition-Based Parsing is a Simple Game with a Simple Board and Simple Moves

- Board
  - Buffer (of words) (B)—like a Python `list`
  - Stack (of `Tree`s) (S)—like a Python `deque`
- Moves
  - SHIFT

    ```
    S.push(Tree(B.popleft()))
    ```
  - RIGHT-ARC

    ```
    u=S.pop(); v=S.pop(); S.push(v.right_arc_to(u))
    ```
  - LEFT-ARC

    ```
    u=S.pop(); v=S.pop(); S.push(u.left_arc_to(v))
    ```

Transition-based Parsing

- Initialize the buffer to contain $\mathbf{x}$ and the stack to contain the ROOT symbol.

Stack S	Buffer B
ROOT	we vigorously wash our cats who stink

- We can take one of three actions:
  - SHIFT the word at the front of the buffer B onto the stack S.
  - RIGHT-ARC: $u = \boldsymbol{pop}(S); v = \boldsymbol{pop}(S); \boldsymbol{push}(S, v \rightarrow u)$.
  - LEFT-ARC: $u = \boldsymbol{pop}(S); v = \boldsymbol{pop}(S); \boldsymbol{push}(S, v \leftarrow u)$.

    (for labeled parsing, add labels to the LEFT-ARC and RIGHT-ARC transitions.)

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!

Transition-based Parsing Example

Buffer B

we
vigorously
wash
our
cats
who
stink

Stack S

ROOT

Actions:

Transition-based Parsing Example

Buffer B

vigorously
wash
our
cats
who
stink

Stack S

we
ROOT

Actions: SHIFT

Transition-based Parsing Example

Buffer B

Stack S

Stack S	Buffer B
vigorously	wash
we	our
ROOT	cats
	who
	stink

Actions: SHIFT SHIFT

Transition-based Parsing Example

Stack S	Buffer B
wash	our
vigorously	cats
we	who
ROOT	stink

Actions: SHIFT SHIFT SHIFT

Transition-based Parsing Example

Stack S

vigorously wash

we

ROOT

Buffer B

our

cats

who

stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC

Transition-based Parsing Example

Stack S

we vigorously wash

ROOT

Buffer B

our
cats
who
stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC

Transition-based Parsing Example

Stack S

our

we vigorously wash

ROOT

Buffer B

cats
who
stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT

Transition-based Parsing Example

Stack S

cats
our
we vigorously wash
ROOT

Buffer B

who
stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT

Transition-based Parsing Example

Stack *S*

our cats

we vigorously wash

ROOT

Buffer *B*

who
stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC

Transition-based Parsing Example

Stack S

who

our cats

we vigorously wash

ROOT

Buffer B

stink

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT

Transition-based Parsing Example

Stack S

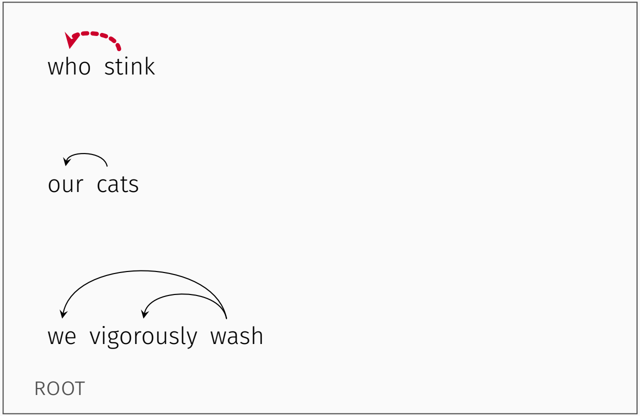
stink
who
our cats
we vigorously wash
ROOT

Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT

Transition-based Parsing Example

Stack S

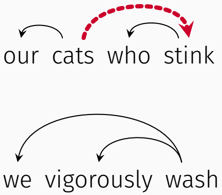


Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT LEFT-ARC

Transition-based Parsing Example

Stack S



ROOT

Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT LEFT-ARC RIGHT-ARC

Transition-based Parsing Example

Stack S

we vigorously wash our cats who stink

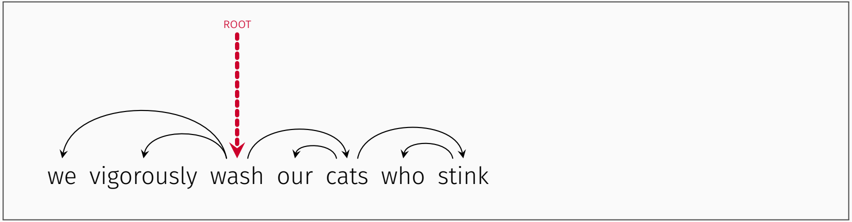
ROOT

Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT LEFT-ARC RIGHT-ARC RIGHT-ARC

Transition-based Parsing Example

Stack S



Buffer B

Actions: SHIFT SHIFT SHIFT LEFT-ARC LEFT-ARC SHIFT SHIFT LEFT-ARC SHIFT SHIFT LEFT-ARC RIGHT-ARC RIGHT-ARC

The Core of Transition-based Parsing

- At each iteration, choose among {SHIFT, RIGHT-ARC, LEFT-ARC}.

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.

- 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.

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,

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: $\langle state, correcttransition \rangle$.

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

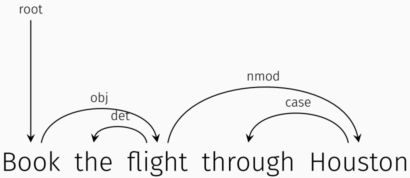
Here are the features extracted by Chen and Manning's (2014) feed-forward neural model for shift-reduce parsing:

- 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)

But Wait, What about Training?

Problem: Transition-based parsing is carried out through a sequence of atomic operations (SHIFT, LEFT-ARC, RIGHT-ARC). Treebanks have whole trees.
Solution: Create a TRAINING ORACLE to train the oracle (classifier). "derive appropriate training instances consisting of configuration-transition pairs from a treebank by **simulating the operation of a parser** in the context of a reference dependency tree. **We can deterministically record correct parser actions at each step** as we progress through each training example, thereby creating the training set we require."

Creating Training Oracle: An Example



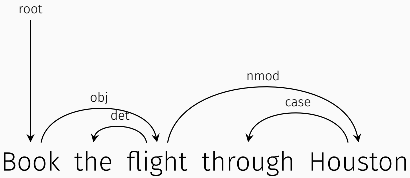
Step 1

1. LEFT-ARC is not applicable since *root* ← *book* is not in the reference tree.
2. RIGHT-ARC is not applicable since *root* → *book* is not in the reference tree.
3. The only option is to SHIFT *the* onto the stack.

Step 2

1. LEFT-ARC is not applicable since *book* ← *the* is not in the reference tree.
2. RIGHT-ARC is not applicable since *book* → *the* is not in the reference tree.
3. The only option is to SHIFT *flight* onto the stack.

Creating Training Oracle: An Example



Step 1

1. LEFT-ARC is not applicable since *root* $\leftarrow$ *book* is not in the reference
2. RIGHT-ARC is not applicable since *root* $\rightarrow$ *book* is not in the reference
3. SHIFT is the only option

Step 2: Same conditions hold. SHIFT.

Step 3: Same conditions hold. SHIFT.

Step 5: Can we add arc *book* $\rightarrow$ *flight*? It's present in the reference but would prevent the attachment of *Houston* since *flight* would have been removed from the stack. The preconditions for RIGHT-ARC are not satisfied and SHIFT is the only option.

Transition-based Parsing: Remarks

- Can also be applied to phrase-structure parsing. Keyword: “shift-reduce” parsing.
- **The algorithm for making decisions doesn’t need to be greedy; can maintain multiple hypotheses.**
    - e.g., beam search
- **Potential flaw**: the classifier is typically trained under the assumption that previous classification decisions were all correct. As yet, there is no principled solution to this problem, but there are approximations based on “dynamic oracles”.

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?

Questions?