CS447: Natural Language Processing
http://courses.engr.illinois.edu/cs447

Lecture 11: Penn Treebank Parsing; Dependency Grammars

Julia Hockenmaier
juliahmr@illinois.edu
3324 Siebel Center

Class Admin

Midterm Exam: Friday, Oct 12

The midterm will be during class.

Closed book exam:

- You are not allowed to use any cheat sheets, computers, calculators, phones etc.(you shouldn't have to anyway)
- The exam will cover the material from the lectures

Format: Short answer questions

Review session: Wednesday, Oct 10 in class.

Review the material *before* that class, so that we can clear up any confusions

Conflict Exam or DRES accommodations:
Email me (juliahmr@illinois.edu) asap

Exam Question types

***Define* X:**
Provide a mathematical/formal definition of X

***Explain* X; *Explain* what X is/does:**
Use plain English to define X and say what X is/does

***Compute* X:**
Return X; Show the steps required to calculate it

***Draw* X:**
Draw a figure of X

***Show* that X is true/is the case/…:**
This may require a (typically very simple) proof.

***Discuss/Argue* whether …**

Use your knowledge (of X,Y,Z) to argue your point

4th Credit Hour

Either a **research project** (alone or with one other student)
or a **literature survey** (alone)

Upcoming deadlines:

Fri, Oct 19: Proposal due
Fri, Nov 9: Progress report due (Is your paper on track?)
Thu, Dec 13: Final report due (Summary of papers)

Good places to find NLP papers:

- **ACL anthology** http://aclweb.org/anthology
  covers almost everything published in NLP
- **JNLE** http://journals.cambridge.org/action/displayJournal?jid=NLE
  is another big NLP journal that is not part of the ACL
- Standard machine learning/AI conferences (**NIPS, ICML, IJCAI, AAAI**)
  and journals (**JMLR, JAIR** etc.) are okay as well.
- Other venues: check with me that this is actually NLP

4th Credit hour: Proposal

Upload a **one-page PDF** to Compass by Oct 19

- written in **LaTeX** (not MS Word)
- with **full bibliography** of the papers you want to read
  or base your project on
  (ideally with **links to online versions**; add url-field to your bibtex file)
- include a **motivation** of why you have chosen those papers
- for a research project: tell me whether you have the **data** you need, what **existing software** you will be using, what you will have to **implement yourself.**
- mention any **questions/concerns** that you may have.

Today's lecture

Penn Treebank Parsing

Dependency Grammars
Dependency Treebanks
Dependency Parsing

Penn Treebank Parsing

The Penn Treebank

The first publicly available syntactically annotated corpus

Wall Street Journal (50,000 sentences, 1 million words)
also Switchboard, Brown corpus, ATIS

The annotation:

– POS-tagged (Ratnaparkhi's MXPOST)
– Manually annotated with phrase-structure trees
– Richer than standard CFG: *Traces* and other *null elements* used to represent non-local dependencies (designed to allow extraction of predicate-argument structure) [more on this later in the semester]

Standard data set for English parsers

The Treebank label set

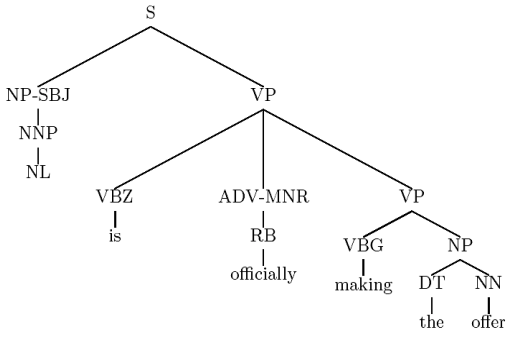
48 preterminals (tags):

– 36 POS tags, 12 other symbols (punctuation etc.)
– Simplified version of Brown tagset (87 tags)
(cf. Lancaster-Oslo/Bergen (LOB) tag set: 126 tags)

14 nonterminals:

standard inventory (S, NP, VP,...)

A simple example



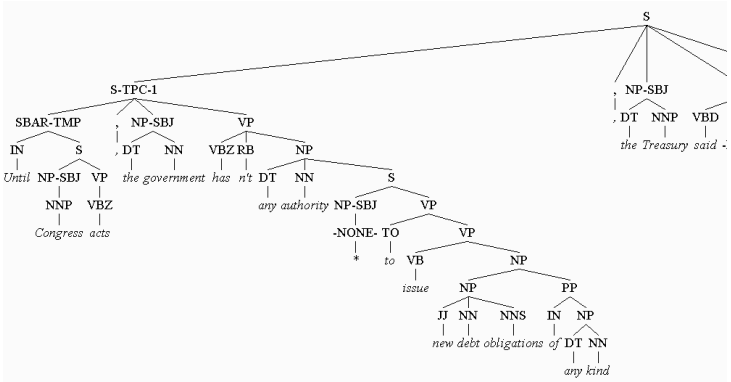
Relatively flat structures:

– There is no noun level
– VP arguments and adjuncts appear at the same level

Function tags, e.g. -SBJ (subject), -MNR (manner)

A more realistic (partial) example

Until Congress acts, the government hasn't any authority to issue new debt obligations of any kind, the Treasury said .... .



The Penn Treebank CFG

The Penn Treebank uses very flat rules, e.g.:

```
NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBG NN NNP NNP FW NNP
NP → NP JJ , JJ `` SBAR '' NNS
```

– Many of these rules appear only once.
– Many of these rules are very similar.
– Can we pool these counts?

PCFGs in practice:

Charniak (1996) *Tree-bank grammars*

How well do PCFGs work on the Penn Treebank?

– Split Treebank into test set (30K words) and training set (300K words).
– Estimate a PCFG from training set.
– Parse test set (with correct POS tags).
– Evaluate unlabeled precision and recall

Sentence Lengths	Average Length	Precision	Recall
2-12	8.7	88.6	91.7
2-16	11.4	85.0	87.7
2-20	13.8	83.5	86.2
2-25	16.3	82.0	84.0
2-30	18.7	80.6	82.5
2-40	21.9	78.8	80.4

Two ways to improve performance

… change the (internal) grammar:

- **Parent annotation/state splits:**
  Not all NPs/VPs/DTs/… are the same.
  It matters where they are in the tree

… change the probability model:

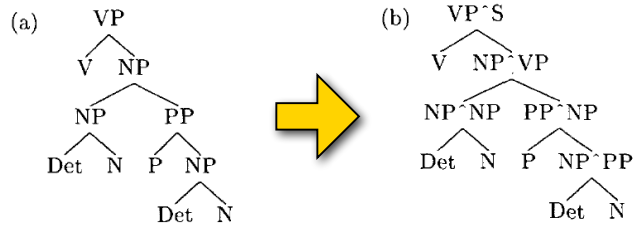
- **Lexicalization:**
  Words matter!
- **Markovization:**
  Generalizing the rules

The parent transformation

PCFGs assume the expansion of any nonterminal is independent of its parent.

But this is not true: NP subjects more likely to be modified than objects.

We can change the grammar by adding the name of the parent node to each nonterminal



Markov PCFGs (Collins parser)

The RHS of each CFG rule consists of:
one head H_X, *n* left sisters L_i and *m* right sisters R_i:

$$X \rightarrow \underbrace{L_n \ldots L_1}_{\text{left sisters}} \; H_X \; \underbrace{R_1 \ldots R_m}_{\text{right sisters}}$$

Replace rule probabilities with a generative process:
For each nonterminal X

- generate its head H_X (nonterminal or terminal)
- then generate its left sisters $L_{1..n}$ and a STOP symbol conditioned on H_X
- then generate its right sisters $R_{1...n}$ and a STOP symbol conditioned on H_X

Lexicalization

PCFGs can't distinguish between
"eat sushi with chopsticks" and *"eat sushi with tuna".*

We need to take words into account!

$P(VP_{eat} \rightarrow VP\ PP_{with\ chopsticks} \mid VP_{eat})$
vs. $P(VP_{eat} \rightarrow VP\ PP_{with\ tuna} \mid VP_{eat})$

Problem: sparse data ($PP_{with\ fattylwhitel...\ tuna....}$)
Solution: only take **head words** into account!

Assumption: each constituent has one head word.

Lexicalized PCFGs

At the root (start symbol S), generate the head word of the sentence, w_s, with $P(w_s)$

Lexicalized rule probabilities:
Every nonterminal is lexicalized: X_{w_x}
Condition rules $X_{w_x} \rightarrow \alpha Y \beta$ on the lexicalized LHS X_{w_x}
$P(X_{w_x} \rightarrow \alpha Y \beta \mid X_{w_x})$

Word-word dependencies:
For each nonterminal Y in RHS of a rule $X_{w_x} \rightarrow \alpha Y \beta$,
condition w_Y (the head word of Y) on X and w_x:
$P(w_Y \mid Y, X, w_X)$

Dealing with unknown words

A lexicalized PCFG assigns zero probability
to any word that does not appear in the training data.

Solution:

Training: Replace rare words in training data
with a token 'UNKNOWN'.

Testing: Replace unseen words with 'UNKNOWN'

Refining the set of categories

Unlexicalized Parsing (Klein & Manning '03)
Unlexicalized PCFGs with various transformations of the training data and the model, e.g.:
– Parent annotation (of terminals and nonterminals):
distinguish preposition IN from subordinating conjunction IN etc.
– Add head tag to nonterminals
(e.g. distinguish finite from infinite VPs)
– Add distance features
Accuracy: 86.3 Precision and 85.1 Recall

The Berkeley parser (Petrov et al. '06, '07)
Automatically learns refinements of the nonterminals
Accuracy: 90.2 Precision, 89.9 Recall

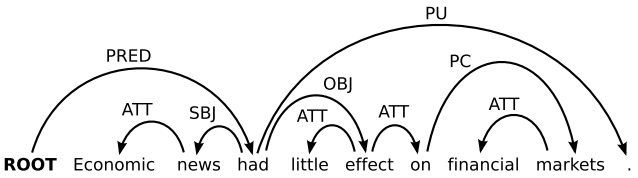
Summary

The Penn Treebank has a large number of very flat rules.
Accurate parsing requires modifications to the basic PCFG model: refining the nonterminals, relaxing the independence assumptions by including grandparent information, modeling word-word dependencies, etc.

How much of this transfers to other treebanks or languages?

Dependency Grammar

A dependency parse



Dependencies are (labeled) asymmetrical binary relations between two lexical items (words).

Dependency grammar

Word-word dependencies are a component of many (most/all?) grammar formalisms.

Dependency grammar assumes that syntactic structure consists *only* of dependencies.

Many variants. Modern DG began with Tesniere (1959).

DG is often used for **free word order languages**.

DG is **purely descriptive** (not a generative system like CFGs etc.), but some formal equivalences are known.

Different kinds of dependencies

Head-argument: *eat sushi*

Arguments may be obligatory, but can only occur once.
The head alone cannot necessarily replace the construction.

Head-modifier: *fresh sushi*

Modifiers are optional, and can occur more than once.
The head alone can replace the entire construction.

Head-specifier: *the sushi*

Between function words (e.g. prepositions, determiners) and their arguments. Syntactic head ≠ semantic head

Coordination: *sushi and sashimi*

Unclear where the head is.

Dependency structures

Dependencies form a graph over the words in a sentence.

This graph is **connected** (every word is a node) and (typically) **acyclic** (no loops).

Single-head constraint:
Every node has at most one incoming edge.
This implies that the graph is a **rooted tree**.

From CFGs to dependencies

Assume each CFG rule has **one head child** (bolded)
The other children are **dependents** of the head.

S	→ NP **VP**	**VP** is head, NP is a dependent
VP	→ **V** NP NP	
NP	→ DT **NOUN**	
NOUN	→ ADJ **N**	

The **headword** of a constituent is the terminal that is reached by recursively following the head child.

(here, V is the head word of S, and N is the head word of NP).

If in rule XP → **X** Y, X is head child and Y dependent, the headword of Y depends on the headword of X.

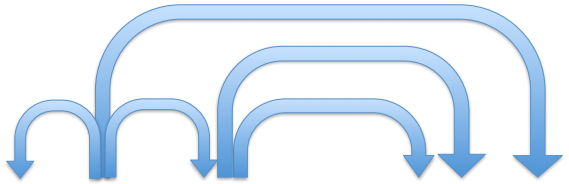
The **maximal projection** of a terminal *w* is the highest nonterminal in the tree that *w* is headword of.
Here, Y is a maximal projection.

Context-free grammars

CFGs capture only **nested** dependencies

The dependency graph is a **tree**

The dependencies **do not cross**

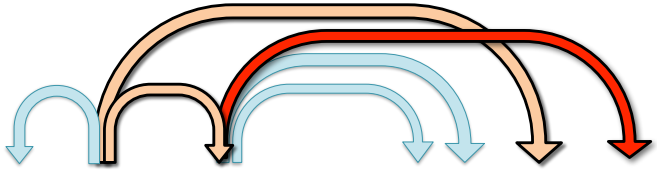


Beyond CFGs: Nonprojective dependencies

Dependencies: tree with crossing branches

Arise in the following constructions

- (Non-local) **scrambling** (free word order languages)
  Die Pizza hat Klaus versprochen zu bringen
- **Extraposition** (*The guy is coming who is wearing a hat*)
- **Topicalization** (*Cheeseburgers, I thought he likes*)



Dependency Treebanks

Dependency treebanks exist for many languages:

Czech
Arabic
Turkish
Danish
Portuguese
Estonian
....

Phrase-structure treebanks (e.g. the Penn Treebank) can also be translated into dependency trees (although there might be noise in the translation)

The Prague Dependency Treebank

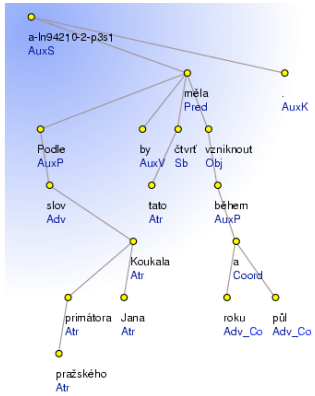
Three levels of annotation:

morphological: [<2M tokens]
Lemma (dictionary form) + detailed analysis
(15 categories with many possible values = 4,257 tags)

surface-syntactic ("analytical"): [1.5M tokens]
Labeled dependency tree encoding grammatical functions (subject, object, conjunct, etc.)

semantic ("tectogrammatical"): [0.8M tokens]
Labeled dependency tree for predicate-argument structure, information structure, coreference (not all words included)
(39 labels: agent, patient, origin, effect, manner, etc....)

Examples: analytical level



METU-Sabanci Turkish Treebank

Turkish is an agglutinative language with free word order.

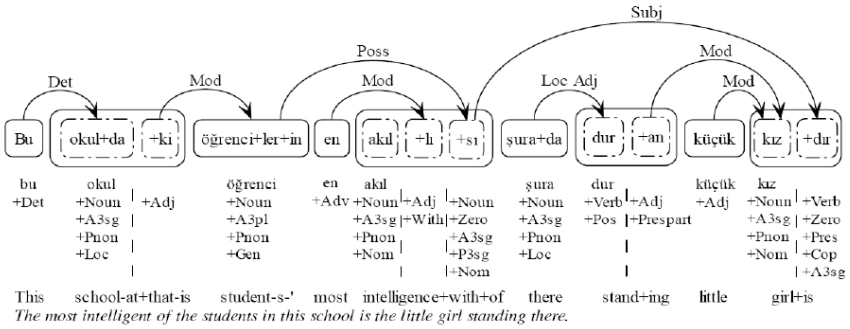
Rich morphological annotations

Dependencies (next slide) are at the morpheme level

- iyileştiriliyorken
  - (literally) while it is being caused to become good
  - while it is being improved
- iyi+Adj ^DB+Verb+Become^DB+Verb+Caus ^DB+Verb+Pass+Pos+Pres^DB+Adverb+While

Very small -- about 5000 sentences

METU-Sabanci Turkish Treebank



[this and prev. example from Kemal Oflazer's talk at Rochester, April 2007]

Universal Dependencies

37 syntactic relations, intended to be applicable to all languages ("universal"), with slight modifications for each specific language, if necessary.

http://universaldependencies.org

Universal Dependency Relations

Nominal core arguments: `nsubj` (nominal subject), `obj` (direct object), `iobj` (indirect object)
Clausal core arguments: `csubj` (clausal subject), `ccomp` (clausal object ["complement"])
Non-core dependents: `advcl` (adverbial clause modifier), `aux` (auxiliary verb),
Nominal dependents: `nmod` (nominal modifier), `amod` (adjectival modifier),
Coordination: `cc` (coordinating conjunction), `conj` (conjunct)

and many more…

Parsing algorithms for DG

'Transition-based' parsers:

learn a sequence of actions to parse sentences

Models:

State = stack of partially processed items
+ queue/buffer of remaining tokens
+ set of dependency arcs that have been found already

Transitions (actions) = add dependency arcs; stack/queue operations

'Graph-based' parsers:

learn a model over dependency graphs

Models:

a function (typically sum) of local attachment scores

For dependency trees, you can use a minimum spanning tree algorithm

Transition-based parsing (Nivre et al.)

Transition-based parsing: assumptions

This algorithm works for projective dependency trees.

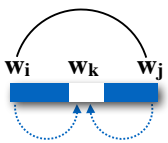
Dependency tree:

Each word has a single parent
(Each word is a dependent of [is attached to] one other word)

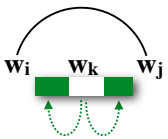
Projective dependencies:

There are no crossing dependencies.
For any i, j, k with $i < k < j$: if there is a dependency between w_i and w_j,
the parent of w_k is a word w_l between (possibly including) i and j: $i \le l \le j$,
while any child w_m of w_k has to occur between (excluding) i and j: $i<m<j$

the parent of w_k:
one of $w_i \ldots w_j$



any child of w_k:
one of $w_{i+1} \ldots w_{j-1}$



Transition-based parsing

Transition-based shift-reduce parsing processes the sentence $S = w_0 w_1 ... w_n$ from left to right.
Unlike CKY, it constructs a **single tree**.

Notation:

- w_0 is a special ROOT token.
- $V_S = \{w_0, w_1, ..., w_n\}$ is the vocabulary of the sentence
- R is a set of dependency relations

The parser uses three data structures:

- σ: a **stack** of partially processed words $w_i \in V_S$
- β: a **buffer** of remaining input words $w_i \in V_S$
- A: a **set of dependency arcs** $(w_i, r, w_j) \in V_S \times R \times V_S$

Parser configurations (σ, β, A)

The **stack σ** is a list of partially processed words

- We push and pop words onto/off of **σ**.
- **σ|w** : **w** is on top of the stack.
- Words on the stack are not (yet) attached to any other words.
- Once we attach **w**, **w** can't be put back onto the stack again.

The **buffer β** is the remaining input words

- We read words from **β** (left-to-right) and push them onto σ
- **w|β** : **w** is on top of the buffer.

The **set of arcs A** defines the current tree.

- We can add new arcs to **A** by attaching the word on top of the stack to the word on top of the buffer, or vice versa.

Parser configurations (σ, β, A)

We start in the **initial configuration** ($[w_0]$, $[w_1, ..., w_n]$, {})

(w_0, Input Sentence, Empty tree)

We can attach the first word (w_1) to the root token w_0,
or we can push w_1 onto the stack.
(w_0 is the only token that can't get attached to any other word)

We want to end in the **terminal configuration** ([], [], A)

(Empty stack, Empty buffer, Complete tree)

Success!
We have read all of the input words (empty buffer) and have attached all input words to some other word (empty stack)

Transition-based parsing

We process the sentence $S = w_0 w_1 ... w_n$ from left to right ("incremental parsing")

In the parser configuration $(\sigma|w_i, w_j|\beta, A)$:

- w_i is on top of the stack. w_i may have some children
- w_j is on top of the buffer. w_j may have some children
- w_i precedes w_j ($i < j$)

We have to either attach w_i to w_j, attach w_j to w_i, or decide that there is no dependency between w_i and w_j

If we reach $(\sigma|w_i, w_j|\beta, A)$, all words w_k with $i < k < j$ have already been attached to a parent w_m with $i \leq m \leq j$

Parser actions

(σ, β, A): Parser configuration with stack **σ**, buffer **β**, set of arcs **A**
(w, r, w'): Dependency with head **w**, relation **r** and dependent **w'**

SHIFT: Push the next input word w_i from the buffer β onto the stack σ

$$(\sigma, w_i|\beta, A) \Rightarrow (\sigma|w_i, \beta, A)$$

LEFT-ARC$_r$: … w_i … w_j … (dependent precedes the head)

Attach dependent w_i (top of stack σ) to head w_j (top of buffer β) with relation **r** from w_j to w_i. Pop w_i off the stack.

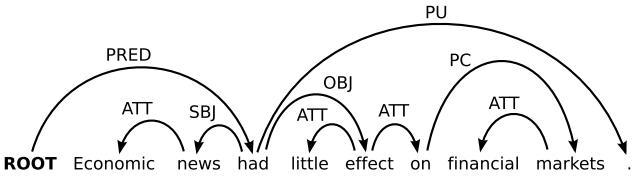
$$(\sigma|w_i, w_j|\beta, A) \Rightarrow (\sigma, w_j|\beta, A \cup \{(w_j, r, w_i)\})$$

RIGHT-ARC$_r$: … w_i … w_j … (dependent follows the head)

Attach dependent w_j (top of buffer β) to head w_i (top of stack σ) with relation **r** from w_i to w_j. Move w_i back to the buffer

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

An example sentence & parse



Economic news had little effect on financial markets .

Economic news had little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)

Economic news had little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)

Economic news had little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)

Economic news had little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(\text{news, ATT, Economic})\}$)

Economic **news had** little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(\text{news, ATT, Economic})\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)

Economic **news had** little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(\text{news, ATT, Economic})\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(\text{had, SBJ, news})\}$)

Economic news **had little** effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(\text{news, ATT, Economic})\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(\text{had, SBJ, news})\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)

Economic news had **little effect** on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(\text{news, ATT, Economic})\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(\text{had, SBJ, news})\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)

Economic news **had little effect** on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(\text{news, ATT, Economic})\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(\text{had, SBJ, news})\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(\text{effect, ATT, little})\}$)

Economic news had little **effect** **on** financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)

Economic news had little effect **on** **financial** markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)

Economic news had little effect on **financial** **markets** .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)

Economic news had little effect **on** **financial** **markets** .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)

Economic news had little **effect** **on** financial **markets** .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)

Economic news **had** little **effect** **on** financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)
RA_{ATT} ⇒	([ROOT, had],	[effect, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$)

Economic news **had** little **effect** on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)
RA_{ATT} ⇒	([ROOT, had],	[effect, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$)
RA_{OBJ} ⇒	([ROOT],	[had, .],	$A_7 = A_6 \cup \{(had, OBJ, effect)\}$)

Economic news **had** little effect on financial markets **.**

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)
RA_{ATT} ⇒	([ROOT, had],	[effect, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$)
RA_{OBJ} ⇒	([ROOT],	[had, .],	$A_7 = A_6 \cup \{(had, OBJ, effect)\}$)
SH ⇒	([ROOT, had],	[.],	A_7)

Economic news **had** little effect on financial markets **.**

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)
RA_{ATT} ⇒	([ROOT, had],	[effect, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$)
RA_{OBJ} ⇒	([ROOT],	[had, .],	$A_7 = A_6 \cup \{(had, OBJ, effect)\}$)
SH ⇒	([ROOT, had],	[.],	A_7)
RA_{PU} ⇒	([ROOT],	[had],	$A_8 = A_7 \cup \{(had, PU, .)\}$)

Economic news **had** little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)
RA_{ATT} ⇒	([ROOT, had],	[effect, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$)
RA_{OBJ} ⇒	([ROOT],	[had, .],	$A_7 = A_6 \cup \{(had, OBJ, effect)\}$)
SH ⇒	([ROOT, had],	[.],	A_7)
RA_{PU} ⇒	([ROOT],	[had],	$A_8 = A_7 \cup \{(had, PU, .)\}$)
RA_{PRED} ⇒	([ ],	[ROOT],	$A_9 = A_8 \cup \{(ROOT, PRED, had)\}$)

Economic news had little effect on financial markets .

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	∅)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	∅)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)
RA_{ATT} ⇒	([ROOT, had],	[effect, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$)
RA_{OBJ} ⇒	([ROOT],	[had, .],	$A_7 = A_6 \cup \{(had, OBJ, effect)\}$)
SH ⇒	([ROOT, had],	[.],	A_7)
RA_{PU} ⇒	([ROOT],	[had],	$A_8 = A_7 \cup \{(had, PU, .)\}$)
RA_{PRED} ⇒	([ ],	[ROOT],	$A_9 = A_8 \cup \{(ROOT, PRED, had)\}$)
SH ⇒	([ROOT],	[ ],	A_9)

Transition-based parsing in practice

Which action should the parser take under the current configuration?

We also need a parsing model that assigns a score to each possible action given a current configuration.

- Possible actions:
  SHIFT, and for any relation r: LEFT-ARC$_r$, or RIGHT-ARC$_r$
- Possible features of the current configuration:
  The top {1,2,3} words on the buffer and on the stack, their POS tags, etc.

We can learn this model from a dependency treebank.