https://presemo.helsinki.fi/nlp2020

LECTURE 7: SYNTAX AND PARSING II

Mark Granroth-Wilding

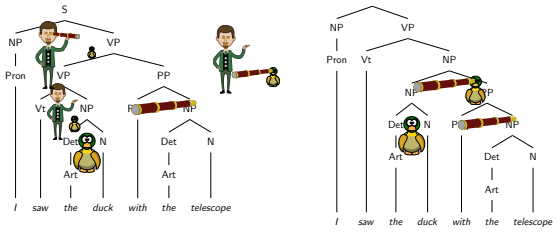
SYNTACTIC AMBIGUITY

- Parsing ambiguity a big problem
- **Correctness**: finding correct structure from *many* possible
- **Efficiency**: many structures to consider → parsing is slow

I saw the duck with the telescope

- Real ambiguities (perceived by humans) – want these
- *Overgeneration* by grammar – as few as possible

PARSING FOR SEMANTICS



- Represent meaning using **formal logic**
- Bits of sentence: bits of logic
  - Often with unfilled 'holes'
- Follow parse tree to combine bits: semantic **composition**

STATISTICAL PARSING

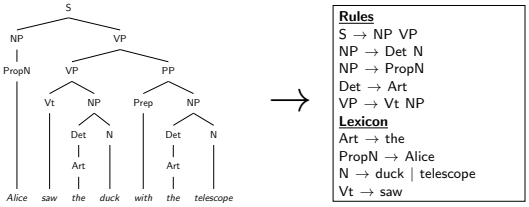
- CKY parses **exhaustively**
- Finds **all** possible trees
- But how do we know which is correct?
- Use **statistics** from corpus of parse trees
  - Compute scores for whole trees
  - Ignore unlikely-looking trees
- Another use: non-exhaustive parsing
  - Compute scores for partial trees
  - Throw away unlikely-looking *subtrees*
  - Keep high-scoring subtrees

STATISTICS FROM WHERE?

- Need a **corpus** of correct parse trees
  '**treebank**'
- Get linguists to **annotate** sentences
- Compute **probabilities** for subtrees from counts in corpus
- Combine to estimate probabilities of whole tree

TREEBANK GRAMMARS

- Can also derive **grammar** from treebank
- Don't write grammar: just annotate treebank
- Grammar ≡ expansions used in treebank



THE PENN TREEBANK

- First treebank: 1989
- Annotated syntactic trees (and other things)
- *Wall Street Journal* articles
- Other domains added later
- ∼40k sentences, ∼1M words
- Similar treebanks built for other languages, domains, annotation types

PROBABILISTIC CFGs (PCFGs)

- CFG with probability distributions associated with expansion
- **Generative** prob dist follows generative grammar rules

For rule $A \rightarrow B$ estimate conditional probability $p(B|A)$

- Finite possible Bs

$$\sum_X p(X|A) = 1$$

- Maximum likelihood estimate from treebank

$$p(B|A) = \frac{count(A \rightarrow B)}{\sum_X count(A \rightarrow X)}$$

PCFGs

- Problem: many rules, lots of low counts
- Use standard **smoothing** techniques with MLE
- Probability of tree is product of all expansions:

$$p(t) = \prod_{A \to B \in t} p(B|A)$$

- Exact extension of **Markov assumption** to CFGs

PCFG EXAMPLE

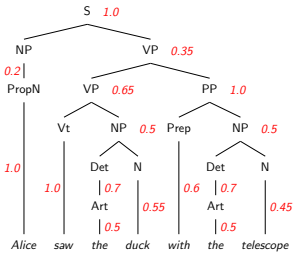
Rules

S → NP VP	1.0
NP → Det N	0.5
NP → NP PP	0.3
NP → PropN	0.2
Det → PosPro	0.3
Det → Art	0.7
VP → Vt NP	0.65
VP → VP PP	0.35
PP → Prep NP	1.0

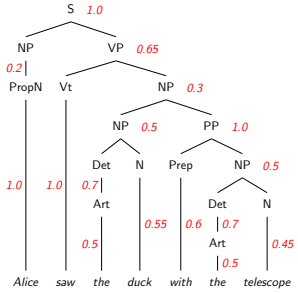
Lexicon

Art → the	0.5
Art → a	0.5
PropN → Alice	1.0
N → duck	0.55
N → telescope	0.45
Vt → saw	1.0
PosPro → my	0.45
PosPro → her	0.55
Prep → with	0.6
Prep → on	0.4

PCFG DISAMBIGUATION



$p(t) = 2.07 \times 10^{-4}$

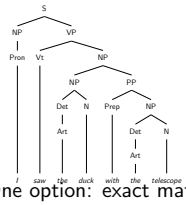


$p(t) = 2.96 \times 10^{-4}$

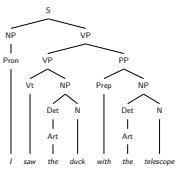
EVALUATING A PARSER

- How good is our parser?
- Evaluate by comparing to *gold standard*: treebank

Parser output



Treebank (GS) tree

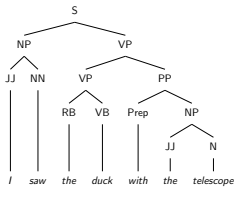


- One option: exact match
- Is tree identical to GS?
- Accuracy over many sentences

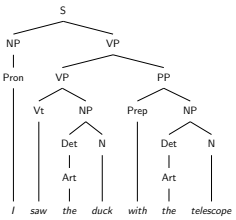
EXERCISE

In small groups (3-4)

Parser output



Treebank (GS) tree



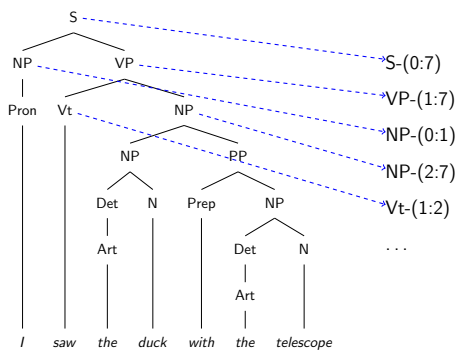
- This tree is wrong! How wrong?
- What types of error has it made?
- How might you measure the errors?

EVALUATING A PARSER

- Problem: no credit for partial tree
- Often, most trees only partially right
- Another option: recovery of (labelled) bracketing
- Standard metric: *PARSEVAL*

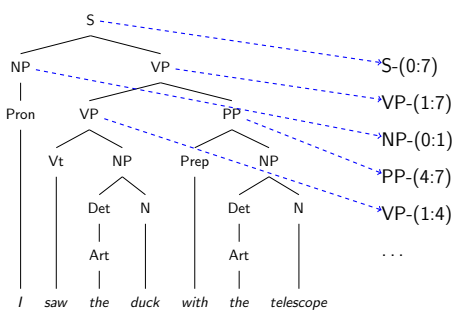
PARSEVAL

Parser output



PARSEVAL

Treebank (GS) tree



PARSEVAL

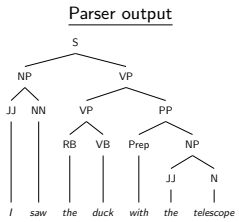
Parser	Treebank
S-(0:7)	S-(0:7)
VP-(1:7)	VP-(1:7)
NP-(0:1)	NP-(0:1)
NP-(2:7)	PP-(4:7)
Vt-(1:2)	VP-(1:4)
Pron-(0:1)	Pron-(0:1)
NP-(2:4)	Vt-(1:2)
PP-(4:7)	NP-(2:4)
...	...

- Compute precision, recall, f-score for labelled spans
- Problems:
  - Some mistakes worse than others: no weighting
  - No score for *almost* correct results: e.g. slightly wrong span

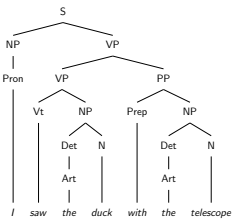
EXERCISE

In small groups

Parser output



Treebank (GS) tree



- How many matching spans does this tree get? (P, R & F if you can...)
- Does this match your earlier ideas about evaluation?
- If not, how could we measure them better?

PARSING EVALUATION

- What else can we do?
- Don't evaluate tree: just important structures
  - Semantics
  - Recovered word-word **dependencies**
  - Extrinsic evaluating: e.g. information extraction
- We'll return to this later...

SYNTACTIC LANGUAGE MODEL

- Derivation probability actually

$$p(t, W|S)$$

  W: word sequence (sentence)
- Compute $p(W)$ by summing over all possible trees

$$\sum_{t \in T} p(t, W|S)$$

- **Language model**
- Takes account of syntactic structures to compute probability

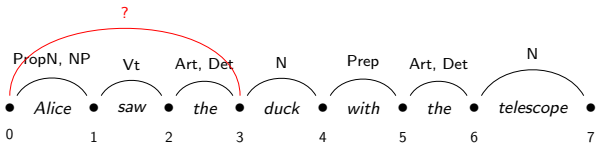
USING PROBABILITIES TO PARSE

- Treebank grammars are *huge*!
- Exhaustive parsing of *shorter* PTB sentences:
  average 1M items in chart![1]
- Most full parses: low probability
- Can we avoid generating many parses that will never be used?
  - Compute scores for partial trees
  - Throw away unlikely-looking *subtrees*
  - Keep high-scoring subtrees

[1] Charniak, Goldwater & Johnson (1998)

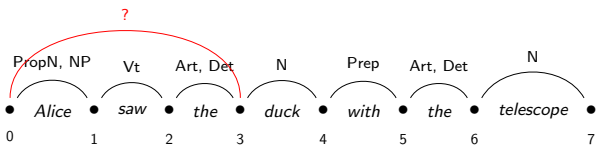
BEAM SEARCH

- Keep just most probable constituent for each span
  → might not result in full parse
- Others (less probable) might have led to S
- Throw away *very* improbable constituents
  1. Don't look promising
  2. If produce full parse, likely to have low score



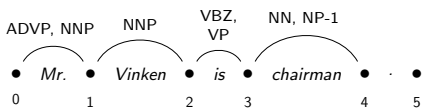
BEAM SEARCH

- Throw away *very* improbable constituents
- On each edge, store top K subtrees
- Higher K:
  - less likely to miss things
  - slower parsing



BEAM SEARCH EXAMPLE

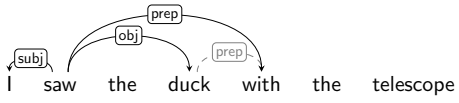
- PCFG induced from Penn Treebank (see assignment)



- Many NTs on most edges
- Keep K with highest probability sub-tree

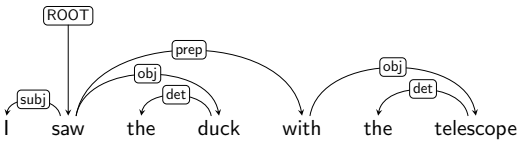
DEPENDENCIES

- **Dependencies**: syntactic *connections* between words
- Parse trees capture
  - syntactic dependencies
  - constituency (phrase categories)
- Can draw dependencies explicitly:

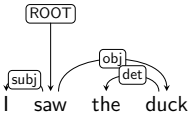


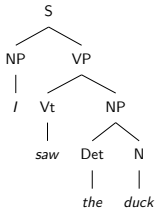
DEPENDENCIES

- Different way to represent syntax
- Complete **tree** of dependency relations
- Covers whole sentence
- Every **node** associated with a **word**



DEPENDENCY STRUCTURES AND TREES

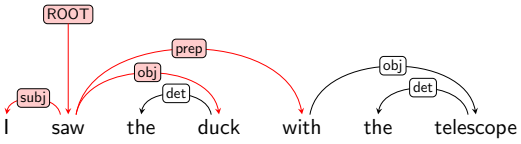




- Deps: no notion of **constituency**
- In some sense, closer to **meaning** / **semantics**
- More natural way to handle **word order** variation
  - Easy to adjust formalism for **free word-order** languages

DEPENDENCY GRAPHS

- Every word **dependent** on exactly one other
- One word dependent on unique **root**
- A word may have multiple dependent words



EXERCISE: DEPENDENCIES

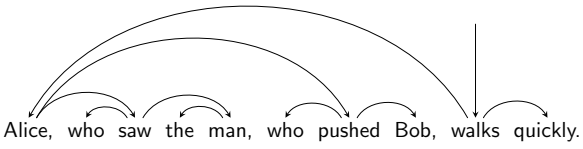
In small groups

Alice, who saw the man, who pushed Bob, walks quickly.

- Same sentence as last lecture (almost)
- Now we know more about dependencies
  - Every word **dependent** on exactly one other
  - One word dependent on unique **root**
  - A word may have multiple dependent words
- Draw **dependency** edges (no labels)

EXERCISE: DEPENDENCIES

My attempt



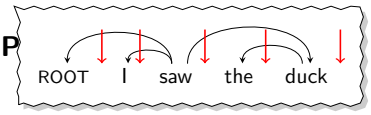
DEPENDENCY PARSING

- Various algorithms
- Here: **transition-based** parsing
- Incremental: word by word, left to right
- Same algorithm *can* be applied to CFG parsing
  - But we didn't do that
  - *Chart parsing*

TRANSITION-BASED PARSING

- Single pass through words, left to right
- Maintain a **stack**
- In each step, either
  - SHIFT new word onto stack; or
  - link top two items, using LEFTARC or RIGHTARC
- Just keep head word on stack after linking
- Continue until
  - No words left
  - Just ROOT on stack
- **Oracle** component choose actions:
  SHIFT, LEFTARC, RIGHTARC

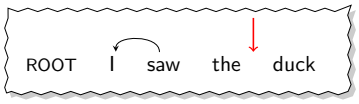
PARSING EXAMP



Input: *I saw the duck*

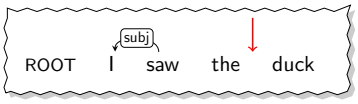
Stack	Words	Action
ROOT	I, saw, the, duck	SHIFT
ROOT, I	saw, the, duck	SHIFT
ROOT, I, saw	the, duck	LEFTARC: I ← saw
ROOT, saw	the, duck	SHIFT
ROOT, saw, the	duck	SHIFT
ROOT, saw, the, duck		LEFTARC: the ← duck
ROOT, saw, duck		RIGHTARC: saw → duck
ROOT, saw		RIGHTARC: ROOT → saw
ROOT		*Done*

ORACLE



- **Oracle**: chooses action at each step
- Type of **classification**
- Input: stack, word list
- Classes: SHIFT, LEFTARC, RIGHTARC
- **Supervised** ML: train on dependency *treebank*
- Any classifier: MaxEnt, SVM, neural network, ...
- Many possible features: words seen, previous actions, POSs, morphology, ...

DEPENDENCY PARSING WITH LABELS



- Same algorithm
- Oracle must now choose **label** as well as action
- Still a classifier: more classes

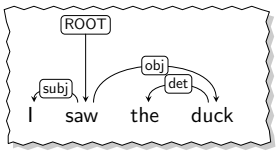
LEFTARC ⇒ LASUBJ, LAOBJ, LAPREP, ...
etc...

DEPENDENCY PARSING

- Can be extremely fast
- Transition-based algorithm is **greedy**
- If it makes a mistake, can't backtrack
- No grammar!
  - All actions are allowed
  - Effectively learn grammar from *treebank*
  - Soft grammar: learned by classifier
- Extremely flexible in what it captures
  - Just add **features** to classifier
  - Agreement, case, ...

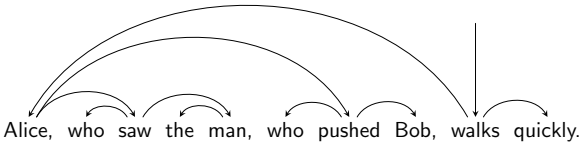
DEPENDENCY PARSING EVALUATION

- Typical measure: **attachment accuracy**
- Every word dependent on one other ('*head*')
- How often is correct head chosen?
- Accuracy: *Unlabelled Attachment Score* (UAS)



EXERCISE: DEPENDENCY EVALUATION

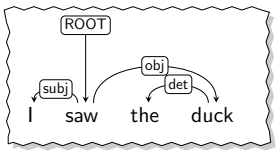
- You parsed this earlier
- Treat this as gold standard:



- Every word dependent on one other ('*head*')
- How often is correct head chosen?
  Unlabelled Attachment Score

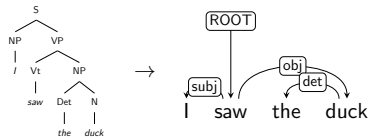
DEPENDENCY PARSING EVALUATION

- We also care about the **label**
- How often is correct head *and label* chosen?
  - Accuracy: *Labelled Attachment Score* (LAS)
- Can also measure *just* label accuracy, ignoring head choice
  - *Label accuracy score* (LS)
- 3 metrics: UAS, LAS, LS
- Capture different aspects of performance: often all reported



CONSTIUENCY PARSING EVALUATION

- Earlier: evaluation of *constituency trees* (CFG derivations)
  - Exact match
  - PARSEVAL
- Another way to evaluate: extract implicit *depencency relations*
- Evaluate in same way as dependency parser evaluation
  - Perhaps more focussed on important aspects of structure
  - Less affected by small errors in boundaries

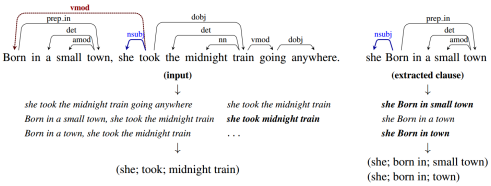


INCREMENTALITY

- Transition-based algorithm is **incremental**
- Processes **left to right**
- Desirable for parsing: *it's what humans do!* (mostly)
- Practical: partial syntax available halfway through
- Other formalisms permit incrementality
  - E.g. CKY for CFGs
  - Adjust **grammar** to make more incremental
  - Other **algorithms** more incremental
- Some formalisms more focussed on this:
  e.g. *Combinatory Categorial Grammar* (CCG)

PARSING APPLICATION: OpenIE

- OpenIE: extract **relation tuples** from plain text
- Uses **syntactic dependencies**



- Triples extracted according to dependency patterns
- Distant supervision from some known relations

SUMMARY

- **Statistical parsing**
  - Treebanks: grammars, models
  - PCFGs
- Parsing evaluation
- Efficient parsing with statistical model: **beam search**
- **Dependency** structures
  - Transition-based **dependency parsing**
  - Parsing evaluation

READING MATERIAL

- Statistical parsing, PCFGs:
  J&M3 ch 12, *Eisenstein 10.3-4*
- Dependency parsing:
  J&M3 ch 13, *Eisenstein ch 11*

More on OpenIE