MST (and more on dependency parsing)

Parsing
ISCL-BA-06

Çağrı Çöltekin
ccoltekin@sfs.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

Winter Semester 2020/21

version: 2202193 @2021-02-12

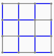
Graph-based parsing: preliminaries

- Enumerate all possible dependency trees
- Pick the best scoring tree
- Features are based on limited parse history (like PCFG parsing)
- Two well-known flavors:
  - Maximum (weight/probability) spanning tree (MST)
  - Chart-parsing based methods

MST parsing: preliminaries

Spanning tree of a graph

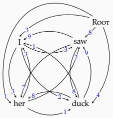
- Spanning tree of a connected graph is a sub-graph which is a tree and traverses all the nodes
- For fully-connected graphs, the number of spanning trees are exponential in the size of the graph
- The problem is well studied
- There are efficient algorithms for enumerating and finding the optimum spanning tree on weighted graphs



MST algorithm for dependency parsing

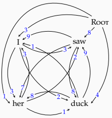
- For directed graphs, there is a polynomial time algorithm that finds the minimum/maximum spanning tree (MST) of a fully connected graph (Chu-Liu-Edmonds algorithm)
- The algorithm starts with a dense/fully connected graph
- Removes edges until the resulting graph is a tree

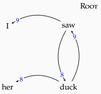
MST example



For each node select the incoming arc with highest weight

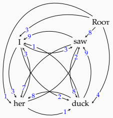
MST example

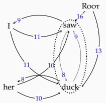




Detect the cycles, contract them to a 'single node'

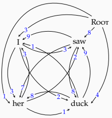
MST example

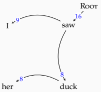




Pick the best arc into the combined node, break the cycle

MST example





Once all cycles are eliminated, the result is the MST

Properties of the MST parser

- The MST parser is non-projective
- There is an algorithm with $O(n^2)$ time complexity
- The time complexity increases with typed dependencies (but still close to quadratic)
- The weights/parameters are associated with edges (often called 'arc-factored')
- We can learn the arc weights directly from a treebank
- However, it is difficult to incorporate non-local features

Non-local features

- The graph-based dependency parsers use edge-based features
- This limits the use of more global features
- Some extensions for using 'more' global features are possible
- This often leads non-projective parsing to become intractable
- Another option is using beam search, and re-ranking based on different/global features

CKY for dependency parsing

- The CKY algorithm can be adapted to projective dependency parsing
- For a naive implementation the complexity increases drastically $O(n^6)$
  - Any of the words within the span can be the head
  - Inner loop has to consider all possible splits
- For projective parsing, the observation that the left and right dependents of a head are independently generated reduces the complexity to $O(n^3)$

External features

- For both type of parsers, one can obtain features that are based on unsupervised methods such as
  - clustering
  - dense vector representations (embeddings)
  - alignment/transfer from bilingual corpora/treebanks

Errors from different parsers

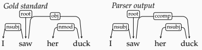
- Different parsers make different errors
  - Transition based parsers do well on local arcs, worse on long-distance arcs
  - Graph based parsers tend to do better on long-distance dependencies
- Parser combination is a good way to combine the powers of different models. Two common methods
  - Majority voting: train parsers separately, use the weighted combination of their results
  - Stacking: use the output of a parser as features for another

Evaluation metrics for dependency parsers

- Like CF parsing, exact match is often too strict
- *Attachment score* is the ratio of words whose heads are identified correctly.
  - *Labeled attachment score* (LAS) requires the dependency type to match
  - *Unlabeled attachment score* (UAS) disregards the dependency type
- *Precision/recall/F-measure* often used for quantifying success on identifying a particular dependency type

precision is the ratio of correctly identified dependencies (of a certain type)
recall is the ratio of dependencies in the gold standard that parser predicted correctly
f-measure is the harmonic mean of precision and recall $\left(\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}\right)$

Evaluation example



UAS	100%	
LAS	50%	
$\text{Precision}_{\text{nsubj}}$	50%	
$\text{Recall}_{\text{nsubj}}$	100%	
$\text{Precision}_{\text{obj}}$	0%	(assumed)
$\text{Recall}_{\text{obj}}$	0%	

Averaging evaluation scores

- Average scores can be

macro-averaged over sentences
micro-averaged over words

- Consider a two-sentence test set with

	words	correct
sentence 1	30	10
sentence 2	10	10

  - word-based average attachment score: 50% (20/40)
  - sentence-based average attachment score: 66% ($(1 + 1/3)/2$)

Dependency parsing: summary

- Dependency relations are often semantically easier to interpret
- It is also claimed that dependency parsers are more suitable for parsing free-word-order languages
- Dependency relations are between words, no phrases or other abstract nodes are postulated
- Two general methods:

transition based greedy search, non-local features, fast, less accurate
graph based exact search, local features, slower, accurate (within model limitations)

- Combination of different methods often result in better performance
- Non-projective parsing is more difficult
- Most of the recent parsing research has focused on better machine learning methods (mainly using neural networks)

Acknowledgments, references, additional reading material

Kübler, Sandra, Ryan McDonald, and Joakim Nivre (2009). *Dependency Parsing*. Synthesis lectures on human language technologies. Morgan & Claypool. ISBN: 9781598295962.