

Dependency parsing

Data Structures and Algorithms for Computational Linguistics III
(ISCL-BA-07)

Çağrı Çöltekin
ccoltekin@ifa.uni-tuebingen.de

University of Tübingen
Seminar für Sprachwissenschaft

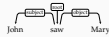
Winter Semester 2021/22

version: 03/2021, 002221-01-26

Dependency grammars

introduction

- Dependency grammars gained popularity in linguistics (particularly in CL) rather recently
- They are old: roots can be traced back to Pāṇini (approx. 5th century BCE)
- Modern dependency grammars are often attributed to **tesnière1959**
- The main idea is capturing the relations between words, rather than grouping them into (abstract) constituents



© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 1 / 26

Dependency grammars



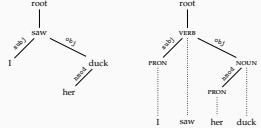
- No constituents, units of syntactic structure are words
- The structure of the sentence is represented by *asymmetric, binary* relations between syntactic units
- Each relation defines one of the words as the **head** and the other as **dependent**
- Typically, the links (relations) have labels (dependency types)
- Often an artificial *root* node is used for computational convenience

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 2 / 26

Dependency grammars Dependency parsing Transition-based parsing MST for dependency parsing Evaluation/alternatives/improvements

Dependency grammars: alternative notation(s)



© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 3 / 26

Dependency grammars: common assumptions

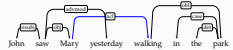
- Every word has a single head
- The dependency graphs are acyclic
- The graph is connected
- With these assumptions, the representation is a tree
- Note that these assumptions are not universal but common for dependency parsing

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 4 / 26

Dependency grammars Dependency parsing Transition-based parsing MST for dependency parsing Evaluation/alternatives/improvements

Dependency grammars: projectivity



- If a dependency graph has no crossing edges, it is said to be *projective*, otherwise *non-projective*
- Non-projectivity stems from long-distance dependencies and free word order
- Projective dependency trees can be represented with context-free grammars
- In general, projective dependencies are parseable more efficiently

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 5 / 26

Dependency grammars

Advantages and disadvantages

- + Close relation to semantics
- + Easier for flexible/free word order
- + Lots, lots of (multi-lingual) computational work, resources
- + Often much useful in downstream tasks
- + More efficient parsing algorithms
- No distinction between modification of head or the whole 'constituent'
- Some structures are difficult to annotate, e.g., coordination

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 6 / 26

Dependency grammars Dependency parsing Transition-based parsing MST for dependency parsing Evaluation/alternatives/improvements

Dependency parsing

- Dependency parsing has many similarities with context-free parsing (e.g., trees)
- It also has some differences (e.g., number of edges and depth of trees are limited)
- Dependency parsing can be
 - grammar-driven (hand crafted rules or constraints)
 - data-driven (rules/model is learned from a treebank)
- There are two main approaches:
 - Graph-based similar to context-free parsing, search for the best tree structure
 - Transition-based similar to shift-reduce parsing (used for programming language parsing), but using greedy search for the best transition sequence

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 7 / 26

Grammar-driven dependency parsing

- Grammar-driven dependency parsers typically based on
 - lexicalized CF parsing
 - constraint satisfaction problem
 - start from fully connected graph, eliminate trees that do not satisfy the constraints
 - exact solution is intractable, often employ heuristics, approximate methods
 - sometimes 'soft', or weighted, constraints are used
 - Practical implementations exist
- Our focus will be on data-driven methods

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 8 / 26

Dependency grammars Dependency parsing Transition-based parsing MST for dependency parsing Evaluation/alternatives/improvements

Data-driven dependency parsing

common methods for data-driven parsing

- Almost any modern/practical dependency parser is statistical
- The 'grammar', and the (soft) constraints are learned from a *treebank*
- There are two main approaches:
 - Graph-based search for the best tree structure, for example
 - find minimum spanning tree (MST)
 - adaptations of CF chart parser (e.g., CKY)(in general, computationally more expensive)
 - Transition-based similar to shift-reduce (LR(k)) parsing
 - Single pass over the sentence, determine an operation (shift or reduce) at each step
 - Linear time complexity
 - We need an approximate method to determine the best operation

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 9 / 26

Shift-Reduce parsing

a refresher through an example

General

Parser data structure

$S \rightarrow P \mid S + P \mid S - P$
 $P \rightarrow \text{Num} \mid P \times \text{Num} \mid P / \text{Num}$

Stack	Input buffer	Action
	2 + 3 × 4	shift
2	+ 3 × 4	reduce ($P \rightarrow \text{Num}$)
P	+ 3 × 4	reduce ($S \rightarrow P$)
S	+ 3 × 4	shift
S +	3 × 4	shift
S + 3	× 4	reduce ($P \rightarrow \text{Num}$)
S + P	× 4	shift
S + P ×	4	shift
S + P × 4		reduce ($P \rightarrow P \times \text{Num}$)
S + P		reduce ($S \rightarrow S + P$)
S		accept

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 10 / 26

Dependency grammars Dependency parsing Transition-based parsing MST for dependency parsing Evaluation/alternatives/improvements

Transition-based parsing

differences from shift-reduce parsing

- The shift-reduce (LR) parsers for formal languages are deterministic, actions are determined by a table lookup
- Natural language sentences are ambiguous, a dependency parser's actions cannot be made deterministic
- Operations are (somewhat) different: instead of reduce (using phrase-structure rules) we use arc operations connecting two words with a labeled arc
- More operations may be defined (e.g., to deal with non-projectivity)

© Çöltekin, ISL / University of Tübingen

Winter Semester 2021/22 11 / 26

Transition based parsing

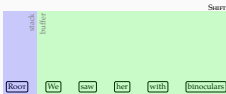
- Use a stack and a buffer of unprocessed words
- Parsing as predicting a sequence of transitions like
 - LEFT-ARC: mark current word as the head of the word on top of the stack
 - RIGHT-ARC: mark current word as a dependent of the word on top of the stack
 - SHIFT: push the current word on to the stack
- Algorithm terminates when all words in the input are processed
- The transitions are not naturally deterministic, best transition is predicted using a machine learning method

A typical transition system



- LEFT-ARC: $(\sigma | w_i, w_j | \beta, A) \rightarrow (\sigma, w_j | \beta, A \cup \{(w_i, \tau, w_j)\})$
- pop w_i
 - add arc (w_i, τ, w_j) to A (keep w_j in the buffer)
- RIGHT-ARC: $(\sigma | w_i, w_j | \beta, A) \rightarrow (\sigma, w_i | \beta, A \cup \{(w_j, \tau, w_i)\})$
- pop w_i
 - add arc (w_j, τ, w_i) to A
 - move w_j to the buffer
- SHIFT: $(\sigma, w_j | \beta, A) \rightarrow (\sigma | w_j, \beta, A)$
- push w_j to the stack
 - remove it from the buffer

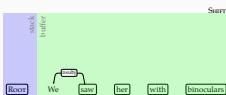
Transition based parsing: example



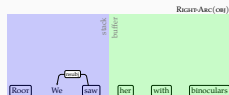
Transition based parsing: example



Transition based parsing: example



Transition based parsing: example



Note: We need Shift for NP attachment.

Transition based parsing: example



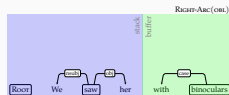
Transition based parsing: example



Transition based parsing: example



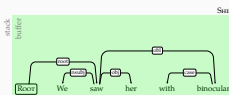
Transition based parsing: example



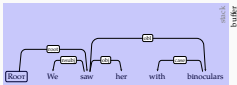
Transition based parsing: example



Transition based parsing: example



Transition based parsing: example



Making transition decisions

- Unlike deterministic parsing (for formal languages), we cannot build a table to determinize the parser actions
- The typical method is to train a (discriminative) classifier
- Almost any machine learning (classification) method is applicable
- The features used for prediction is extracted from the states of the parser:
 - Top-k words on the stack
 - Next-m words in the buffer
 - Transition decisions made so far (the arcs)
- Given these objects, one can extract and use arbitrary features:
 - Words as categorical variables
 - POS tags
 - Embeddings
 - ...

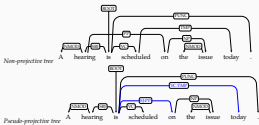
The training data

- The features for transition-based parsing have to be from *parser configurations*
- The data (treebanks) need to be preprocessed for obtaining the training data
- The general idea is to construct a transition sequence by performing a 'mock' parsing using treebank annotations as an 'oracle'
- There may be multiple sequences that yield the same dependency tree, this procedure defines a 'canonical' transition sequence
- For example,
 - LEFT-ARC_r if $(\beta[\beta], r, \sigma[\beta]) \in A$
 - RIGHT-ARC_r if $(\sigma[\beta], r, \beta[\beta]) \in A$
 - and all dependents of $\beta[\beta]$ are attached
 - SHIFT otherwise

Non-projective parsing

- The transition-based parsing we defined so far works only for projective dependencies
- One way to achieve (limited) non-projective parsing is to add special operations:
 - SWAP operation that swaps tokens in the stack and the buffer
 - LEFT-ARC and RIGHT-ARC transitions to/from non-top words from the stack
- Another method is pseudo-projective parsing:
 - preprocessing to 'projectivize' the trees before training
 - The idea is to attach the dependents to a higher level head that preserves projectivity, while marking the operation on the new dependency label
 - post-processing for restoring the projectivity after parsing
 - Re-introduce projectivity for the marked dependencies

Pseudo-projective parsing



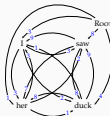
Transition based parsing: summary/notes

- Linear time, greedy, projective parsing
- Can be extended to non-projective dependencies
- We need some extra work for generating gold-standard transition sequences from treebanks
- Early errors propagate, transition-based parsers make more mistakes on long-distance dependencies
- The greedy algorithm can be extended to beam search for better accuracy (still linear time complexity)

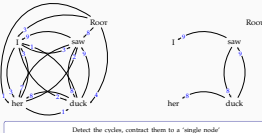
MST algorithm for dependency parsing

- 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

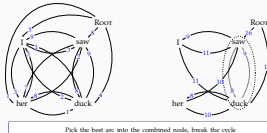
MST example



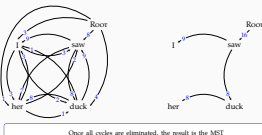
MST example



MST example



MST example



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

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

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

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

23 / 36

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

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 ($\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$)

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

38 / 36

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

Evaluation example

Gold standard

I

saw

her

duck

NP

NP

VP

NP

NP

NP

VP

NP

NP

NP

VP

NP

I

saw

her

duck

NP

NP

VP

NP

NP

NP

VP

NP

NP

NP

VP

NP

UAS

100%

LAS

50%

Precision_{n_{subj}}

50%

Recall_{n_{subj}}

100%

Precision_{obj}

0%

Recall_{obj}

0%

(assumed)

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

29 / 36

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

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)
- Reading suggestion: jurafsky2009 kubler2009

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

36 / 36

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

Acknowledgments, references, additional reading material

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.1

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.2

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.3

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.4

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.5

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.6

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.7

Dependency grammars

Dependency parsing

Transition-based parsing

MST for dependency parsing

Evaluation/alternatives/improvements

C. Gillekin

NR | University of Tübingen

Winter Semester 2021/22

A.8