CS584 Natural Language Processing

Dependency Parsing

Department of Computer Science
Yue Ning
yue.ning@stevens.edu

Late Submission Policy

- **10%** penalty for late submission within **<u>24 hours</u>**.
- **40%** penalty for late submissions within **<u>24-48 hours</u>**.
- After 48 hours, you get **NO** points on the assignment.

Today's plan

1. Syntactic structure
2. Dependency grammar
3. Transition-based dependency parsing
4. Neural dependency parsing

Two views of linguistic structure:

1. Constituency (phrase structure)

- Phrase structure organizes words into nested constituents.
- How do we know what is a constituent? (Not that linguists don't argue about some cases.)
- Distribution: a constituent behaves as a unit that can appear in different places:
  - John talked [to the children] [about drugs].
  - John talked [about drugs] [to the children].
  - *John talked drugs to the children about
- Substitution/expansion/pro-forms:
  - I sat [on the box/right on top of the box/there].
- Coordination, regular internal structure, no intrusion, fragments, semantics, …

Two views of linguistic structure:

1. Constituency (phrase structure)

- Phrase structure organizes words into nested constituents.
  - can represent the grammar with Context-Free Grammars (CFG) rules
- **Starting units**: words are given a category (part of speech = POS)

  the, cuddly, cat, by, the, door

  Det Adj N P Det N
- **Words combine into phrases** with categories

  the cuddly cat, by the door

  NP -> Det Adj N PP->P NP
- **Phrases can combine into bigger phrases** recursively

  the cuddly cat by the door

  NP -> NP PP

Headed phrase structure

- VP -> … VB* …
- NP -> .... NN* …
- ADJP (adjective phrase)-> ...JJ*(adjective)...
- ADVP (adverb phrase)-> ...RB* (adverb)...

- SBAR(Q) (subordinating conjunction) ->
  - S|SINV (Inverted declarative sentence)|SQ (Inverted yes/no question) ->
  - ...NP VP …

- Plus minor phrase types:
  - QP (quantifier phrase in NP), CONJP(multi word constructions: as well as), INTJ(interjections). etc

Two views of linguistic structure:
2. Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.

Why do we need sentence structure?

- We need to understand sentence structure in order to be able to interpret language correctly
- Humans communicate complex ideas by composing words together into bigger units to convey complex meanings
  - "*Inside the carriage, which is built on several levels, he sits in velveteen darkness, with nothing to smoke, feeling metal nearer and farther rub and connect, steam escaping in puffs, a vibration in the carriage frame, a poising, an uneasiness, all the others pressed in around, feeble ones, second sheep, all out of luck and time: drunks, old veterans still in shock from...*" - *Gravity's Rainbow*
- We need to know what is connected to what

Prepositional phrase attachment ambiguity

- ❑ “San Jose cops kill man with knife”
- ❑ “Scientists count whales from space”

√

PP attachment ambiguities multiply

- A key parsing decision is how we ‘attach’ various constituents
  - PPs, adverbial or participial phrases, infinitives, coordinations,

- *The board approved [its acquisition] [by Royal Trustco Ltd.] [of Toronto] [for $27 a share] [at its monthly meeting]*

Coordination scope ambiguity

THE NEWS-GAZETTE

NATION / WO

PRESIDENT'S FIRST PHYSICAL

Doctor: No heart, cognitive issues

But Trump needs to reduce his cholesterol, lose weight

By JILL COLVIN

6-foot-3 president weighed in at 239 pounds — three pounds heavier than he was in September 2016, the last time Trump revealed his weight to the public.

Trump's blood pressure was 122 over 74, and his

with no medical issues." Trump has no heart disease and no family history of it.

The 71-year-old president performed "exceedingly well" on cognitive screening, which is not

White H
reporte

Coordination scope ambiguity

Shuttle veteran and longtime NASA executive Fred Gregory appointed to board

Verb phrase (VP) attachment ambiguity

Dependency paths identify semantic relations e.g. for protein interaction

KaiC ←nsubj interacts nmod:with → SasA

KaiC ←nsubj interacts nmod:with → SasA conj:and→ KaiA

KaiC ←nsubj interacts prep_with→ SasA conj:and→ KaiB

[Erkan et al. EMNLP 07, Fundel et al. 2007, etc]

Dependency grammar and structure

- Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies

- The arrows are commonly typed with the name of grammatical relations (subject, prepositional object, apposition, etc.)

Dependency grammar and structure

- Dependency syntax postulates that syntactic structure consists of lexical items linked by binary asymmetric relations ("arrows") called dependencies

- The arrow connects a head (governor, superior, regent) with a dependent (modifier, inferior, subordinate)

- Usually, dependencies form a tree (connected, acyclic, single-head)

Relation between phrase structure and dependency structure

- A dependency grammar has a notion of a head. Officially, CFGs don't.
- But modern linguistic theory and all modern statistical parsers (Charniak, Collins, Stanford, …) do, via hand-written phrasal "head rules":
  - The head of a Noun Phrase is a noun/number/adj/…
  - The head of a Verb Phrase is a verb/modal/….
- The head rules can be used to extract a dependency parse from a CFG parse

- The closure of dependencies give constituency from a dependency tree
- But the dependents of a word must be at the same level (i.e., "flat")

Dependency grammar/parsing history

- The idea of dependency structure goes back a long way
  - To Pāṇini's grammar (c.5th century BCE)
  - Basic approach of 1st millennium Arabic grammarians

- Constituency/context-free grammars is a new-fangled invention
  - 20th century invention (R.S. Wells,1947;then Chomsky)

- Modern dependency work often sourced to L. Tesnière(1959)
  - Was dominant approach in "East" in 20th Century (Russia, China, ...)
    - Good for free-er word order languages

- Among the earliest kinds of parsers in NLP, even in the US:
  - David Hays, one of the founders of U.S. computational linguistics, built early (first?) dependency parser (Hays 1962)

Pre 1990 ("Classical") NLP parsing

- Wrote symbolic grammar (CFG or often richer) and lexicon

S -> NP VP	NN -> *interest*
NP -> (DT) NN	NNS -> *rates*
NP -> NN NNS	NNS -> *raises*
NP -> NNP	VBP -> *interest*
VP -> V NP	VBZ -> *rates*

- Used grammar/proof systems to prove parses from words
- This scaled very badly and didn't give coverage. For sentence:

Fed raises interest rates 0.5% in effort to control inflation

  - Minimal grammar: 36 parses
  - Simple 10 rule grammar: 592 parses
  - Real-size broad-coverage grammar: millions of parses

Classical NLP parsing

- Categorical constraints can be added to grammars to limit unlikely/weird parses for sentences
  - But the attempt makes the grammars not robust
    - In traditional systems, commonly 30% of sentences in even an edited text would have *no* parse.
- A less constrained grammar can parse more sentences
  - But simple sentences end up with ever more parses with no way to choose between them
- We need mechanisms that allow us to find the most likely parse(s) for a sentence
  - Statistical parsing lets us work with very loose grammars that admit millions of parses for sentences but still quickly find the best parse

The rise of annotated data:

The Penn Treebank

```
( (S
        (NP-SBJ (DT The) (NN move))
        (VP (VBD followed)
        (NP
        (NP (DT a) (NN round))
        (PP (IN of)
        (NP
        (NP (JJ similar) (NNS increases))
        (PP (IN by)
        (NP (JJ other) (NNS lenders)))
        (PP (IN against)
        (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
        (, ,)
        (S-ADV
        (NP-SBJ (-NONE- *))
        (VP (VBG reflecting)
        (NP
        (NP (DT a) (VBG continuing) (NN decline))
        (PP-LOC (IN in)
        (NP (DT that) (NN market)))))))
  (. .)))
```

Marcus et al. 1993, Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics. https://aclweb.org/anthology/J93-2004

The rise of annotated data

- Starting off, building a treebank seems a lot slower and less useful than building a grammar
- But a treebank gives us many things
  - Reusability of the labor
    - Many parsers, POS taggers, etc. can be built on it
    - Valuable resource for linguistics
  - Broad coverage
  - Frequencies and distributional information
  - A way to evaluate systems

Marcus et al. 1993, Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics. https://aclweb.org/anthology/J93-2004

Dependency conditioning preferences

- What are the sources of information for dependency parsing?
  a. Bilexical affinities [discussion->issues] is plausible
  b. Dependency distance mostly with nearby words
  c. Intervening material
     - ❑ Dependencies rarely span intervening verbs or punctuation
  d. Valency of heads
     - ❑ How many dependents on which side are usual for a head?

Dependency parsing

- A sentence is parsed by choosing for each word what other word (including ROOT) is it a dependent of
- Usually some constraints:
  - Only one word is a dependent of ROOT
  - Don't want cycles A->B,B->A
- This makes the dependencies a tree
- Final issue is whether arrows can cross (non-projective) or not

Projectivity

- Defn: There are no crossing dependency arcs when the words are laid out in their linear order, with all arcs above the words
- Dependencies parallel to a CFG tree must be projective
  - Forming dependencies by taking 1 child of each category as head
- But dependency theory normally does allow non-projective structures to account for displaced constituents
  - You can't easily get the semantics of certain constructions right without these non projective dependencies

Methods of dependency parsing

1. **Dynamic programming**
   a. Eisner (1996) gives a clever algorithm with complexity $O(n^3)$, by producing parse items with heads at the ends rather than in the middle
2. **Graph algorithms**
   a. Create a Minimum Spanning Tree for a sentence
   b. McDonald et al.'s (2005) MSTParser scores dependencies independently using an ML classifier (he uses Margin-infused relaxed algorithm MIRA, for online learning, but it can be something else)
3. **Constraint Satisfaction**
   a. Edges are eliminated that don't satisfy hard constraints. Karlsson (1990), etc.
4. **"Transition-based parsing" or "deterministic dependency parsing"**
   a. Greedy choice of attachments guided by good machine learning classifiers MaltParser (Nivre et al. 2006). Has proven highly effective.

Greedy transition-based parsing

[Nivre 2003]

- A simple form of greedy discriminative dependency parser
- The parser does a sequence of bottom up actions
  - Roughly like “shift” or “reduce” in a shift-reduce parser, but the "reduce" actions are specialized to create dependencies with head on left or right
- The parser has:
  - a stack σ, written with top to the right
    - which starts with the ROOT symbol
  - a buffer β, written with top to the left
    - which starts with the input sentence
- a set of dependency arcs A
  - which starts off empty
- a set of actions

Basic transition-based dependency parser

- Start σ = [ROOT], $\beta = w_1,\ldots,w_n$, $A = \emptyset$

1. Shift $\sigma, w_i \mid \beta, A \rightarrow \sigma \mid w_i, \beta, A$
2. Left-Arc$_r$ $\sigma \mid w_i \mid w_j, \beta, A \rightarrow \sigma \mid w_i, \beta, A \cup \{ r(w_j, w_i)\}$
3. Right-Arc$_r$ $\sigma \mid w_i \mid w_j, \beta, A \rightarrow \sigma \mid w_i, \beta, A \cup \{ r(w_i, w_j)\}$

- Finish: σ = [ROOT], $\beta = \emptyset$

Arc-standard transition-based parser

Analysis of "I ate fish"

Start: $\sigma = [ROOT]$, $\beta = w_1, \ldots, w_n$, $A = \emptyset$

1. Shift $\sigma, w_i|\beta, A \rightarrow \sigma|w_i, \beta, A$
2. Left-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_j, \beta, A \cup \{r(w_j, w_i)\}$
3. Right-Arc$_r$ $\sigma|w_i|w_j, \beta, A \rightarrow \sigma|w_i, \beta, A \cup \{r(w_i, w_j)\}$

Finish: $\beta = \emptyset$

Arc-standard transition-based parser

Analysis of “I ate fish”

MaltParser

[Nivre and Hall 2005]

- We have left to explain how we choose the next action
  - Answer: Stand back,I know machine learning!
- Each action is predicted by a discriminative classifier (e.g., softmax classifier) over each legal move
  - Max of 3 untyped choices; max of |R|*2+1 when typed (R: # arc actions)
  - Features: top of stack word, POS; first in buffer word,POS;etc.
- There is NO search (in the simplest form)
  - But you can profitably do a beam search if you wish (slower but better): You keep k good parse prefixes at each time step
- The model's accuracy is *fractionally* below the state of the art in dependency parsing, but
- It provides very fast linear time parsing, with great performance

Evaluation of dependency parsing

(labeled) dependency accuracy

$$\text{Acc} = \frac{\#\text{ correct deps}}{\#\text{ of deps}}$$

UAS = 4 / 5 = 80%

LAS = 2 / 5 = 40%

Gold			
1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	obj

Parsed			
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

UAS: unlabeled attachment score
LAS: labeled attachment score

Conventional feature representation

binary, sparse
dim $=10^6 \sim 10^7$

0	0	0	1	0	0	1	0	...	0	0	1	0

Feature templates: usually a combination of 1 ~ 3 elements from the configuration.

Indicator features

$$s1.w = \text{good} \wedge s1.t = \text{JJ}$$

$$s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$$

$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$

$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

Motivation of a neural dependency parser [Chen and Manning 2014]

- Problem 1: sparse
- problem 2: incomplete
- problem 3: expensive computation
- More than 95% of parsing time is consumed by feature computation

binary, sparse
dim $=10^6 \sim 10^7$

0	0	0	1	0	0	1	0	…	0	0	1	0

Feature templates: usually a combination of 1 ~ 3 elements from the configuration.

Indicator features

$$s1.w = \text{good} \wedge s1.t = \text{JJ}$$

$$s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$$

$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$

$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

Motivation of a neural dependency parser [Chen and Manning 2014]

- Proposed approach:
  - learn a dense and compact feature representation
- English parsing to Stanford dependencies:
  - UAS = head
  - LAS = head and label

Parser	UAS	LAS	sent. / s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	**92.3**	89.6	8
C & M 2014	92.0	**89.7**	**654**

Distributed representations

- Represent each word as a d-dimensional dense vector (i.e., word embedding)
  - Similar words are expected to have close vectors.
- Meanwhile, part-of-speech tags (POS) and dependency labels are also represented as d-dimensional vectors.
  - The smaller discrete sets also exhibit many semantical similarities.

- ❑ NNS (plural noun) should be close to NN (singular noun)
- ❑ num (numerical modifier) should be close to amod (adjective modifier)

Extracting Tokens and then vector representations from configuration

- Extract a set of tokens based on the stack/buffer positions

	word		POS		dep.
s_1	good		JJ		∅
s_2	has		VBZ		∅
b_1	control		NN		∅
$lc(s_1)$ →	∅	+	∅	+	∅
$rc(s_1)$	∅		∅		∅
$lc(s_2)$	He		PRP		nsubj
$rc(s_2)$	∅		∅		∅

- We convert them to vector embeddings and concatenate them

Model architecture

Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$

Configuration

Chen and Manning. EMNLP 2014. https://aclweb.org/anthology/papers/D/D14/D14-1082/

Dependency parsing for sentence structure

Neural networks can accurately determine the structure of sentences, supporting interpretation

- [Chen and Manning 2014] was the first simple, successful neural dependency parser
- The dense representation let it outperform other greedy parsers in both accuracy and speed

Chen and Manning. EMNLP 2014. https://aclweb.org/anthology/papers/D/D14/D14-1082/

Further developments in transition-based neural dependency parsing

This work was further developed and improved by others

- ❑ Bigger, deeper networks with better tuned hyperparameters
- ❑ Beam search
- ❑ Global, conditional random field (CRF)-style inference over the decision sequence

Leading to SyntaxNet and the Parsey McParseFace model

Method	UAS	LAS (PTB WSJ SD 3.3)
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79

Dependency parsing progress: http://nlpprogress.com/english/dependency_parsing.html
https://ai.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html

Graph-based dependency parsers

Compute a score for every possible dependency for each edge

e.g., picking the head for “big”

Graph-based dependency parsers

- Compute a score for every possible dependency for each edge
  - Then add an edge from each word to its highest-scoring candidate head
  - And repeat the same process for each other word

e.g., picking the head for “big”

A neural graph-based dependency parser

[Dozat and Manning 2017; Dozat, Qi, and Manning 2017]

- Revived graph-based dependency parsing in a neural world
  - Design a biaffine scoring model for neural dependency parsing
    - Also using a neural sequence model
- Really great results!
  - But slower than simple neural transition-based parsers
    - There are n^2 possible dependencies in a sentence of length n

Method	UAS	LAS (PTB WSJ SD 3.3
Chen & Manning 2014	92.0	89.7
Weiss et al. 2015	93.99	92.05
Andor et al. 2016	94.61	92.79
Dozat & Manning 2017	**95.74**	**94.08**

Summary

- ❑ Dependency parsing
- ❑ Transition-based dependency parsing
- ❑ Neural-based dependency parsing

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

stevens.edu

Thank You