Dependency Parsing

Praveen
Joo Gek
Kay Pong
Roy
Su Wei
ZiKun
Junbin

Objectives

a) Syntactic Structure

b) Dependency Grammar

c) Transition-based dependency parsing

d) Neural dependency parsing

1. Syntactic structure - Constituency

Constituency = phrase structure grammar = context-free grammars (CFGs)

Definition of CFG:

$$G = (V, \Sigma, R, S)$$

V : set of nonterminals

Σ : set of terminals

R : relation from V to $(V \cup \Sigma)^*$

S : the start symbol

1. Syntactic structure - Constituency

An example for CFG:

$$S \rightarrow AA$$
$$A \rightarrow \alpha$$
$$A \rightarrow \beta$$

start symbol: S

nonterminal symbol: S A

terminal symbol: $\alpha \quad \beta$

Context-free:terminal symbols never appear on the left

1. Syntactic structure - Constituency

Basic unit: words

the, cat, cuddly, by, door
Det N Adj P N

Words combine into phrases

the cuddly cat, by the door
NP -> Det Adj N PP -> P NP

Phrases can combine into bigger phrases

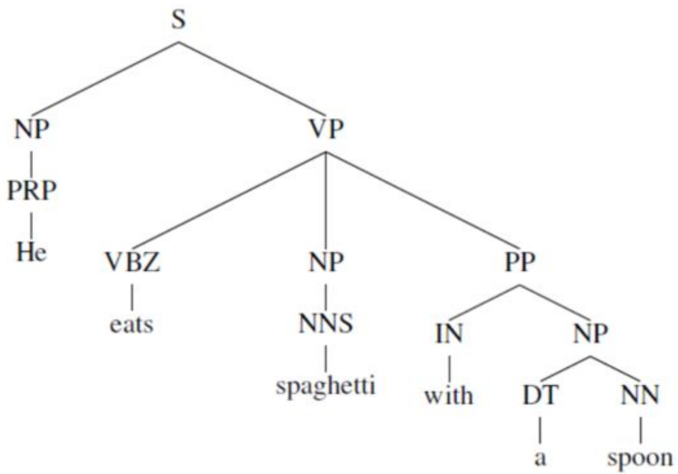
the cuddly cat by the door

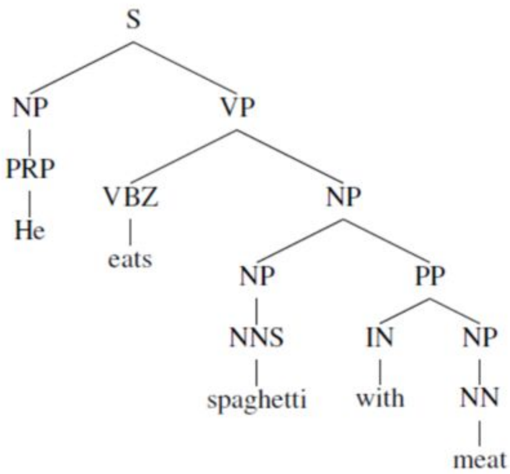
NP -> NP PP

$S \rightarrow NP$

1. Syntactic structure - Constituency

Constituency Trees:





VP→VBZ NP PP or VP→VBZ NP

1. Syntactic Structure – Dependency Grammar

- Differences compared to constituency parsing (CFG)

  Lacks phrasal categories, although acknowledges phrases

  Linguistic units, eg words, are connected by directed links

  Verb is taken to be the structural center

  Other syntactic units (words) directly or indirectly connected to verb

1. Syntactic Structure – Dependency Parsing

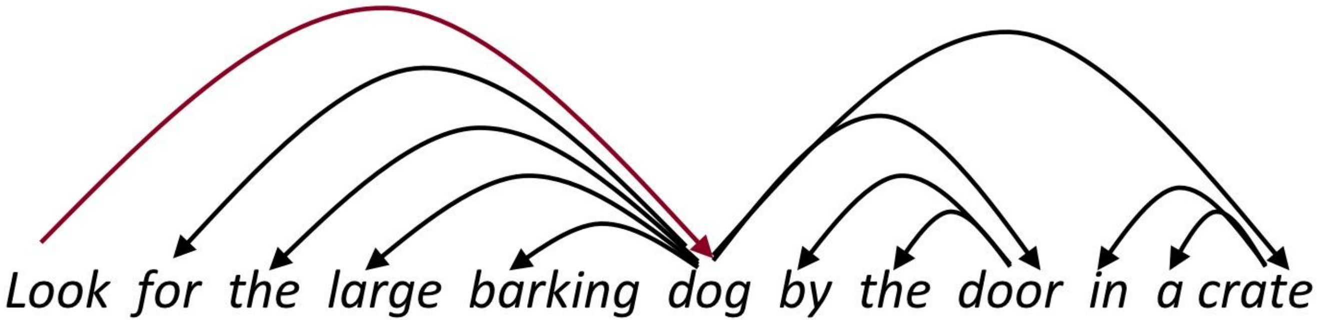
- Differences compared to constituency parsing (CFG)

    Lacks phrasal categories

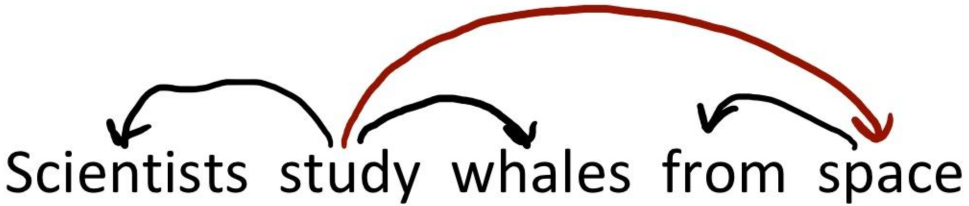
    Might be simpler than CFG based parse-tree (less layers)

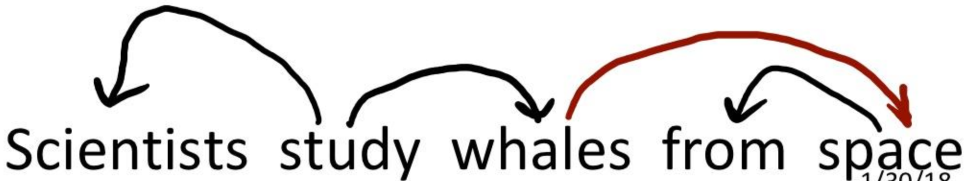
Two views of linguistic structure: Dependency structure

- Dependency structure shows which words depend on (modify or are arguments of) which other words.
  - Determiners, adjectives, and (sometimes) verbs modify nouns
  - We will also treat prepositions as modifying nouns
  - The prepositional phrases are modifying the main noun phrase
  - The main noun phrase is an argument of “look”



PP attachment ambiguities in dependency structure





Attachment ambiguities

- 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]

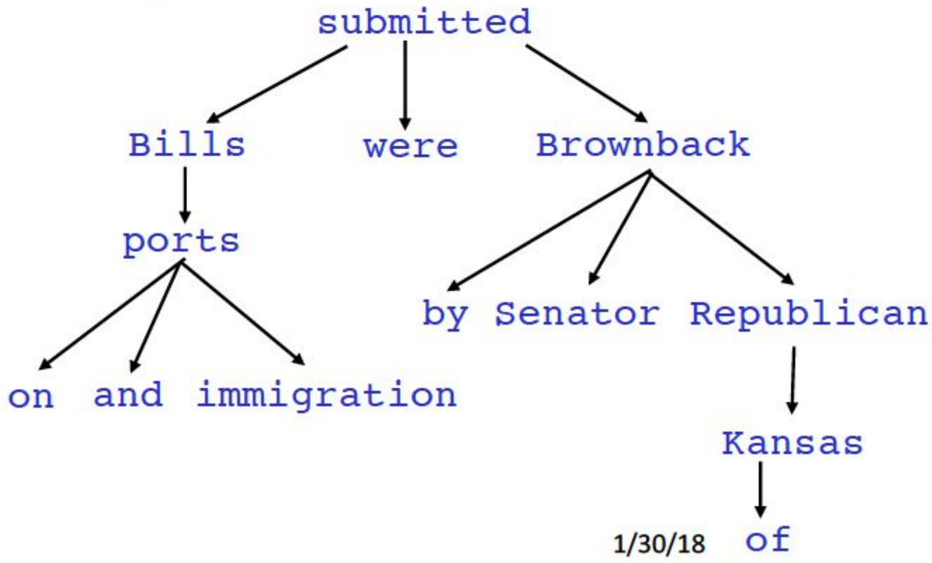
[for $27 a share]

[at its monthly meeting].

- Catalan numbers: $C_n = (2n)!/[(n+1)!n!]$
- An exponentially growing series, which arises in many tree-like contexts
- But normally, we assume nesting.

Dependency Grammar and Dependency Structure

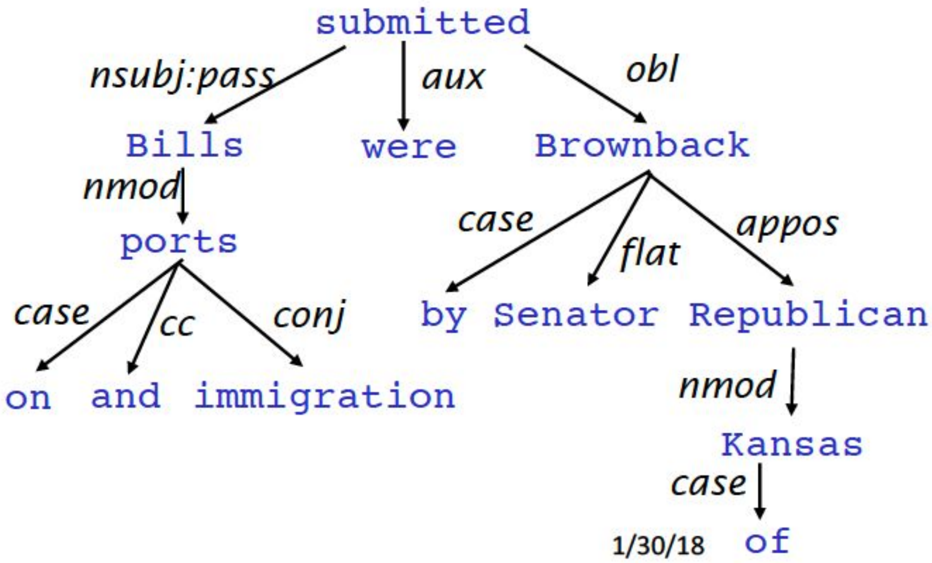
Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations ("arrows") called dependencies



Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally binary asymmetric relations (“arrows”) called dependencies

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

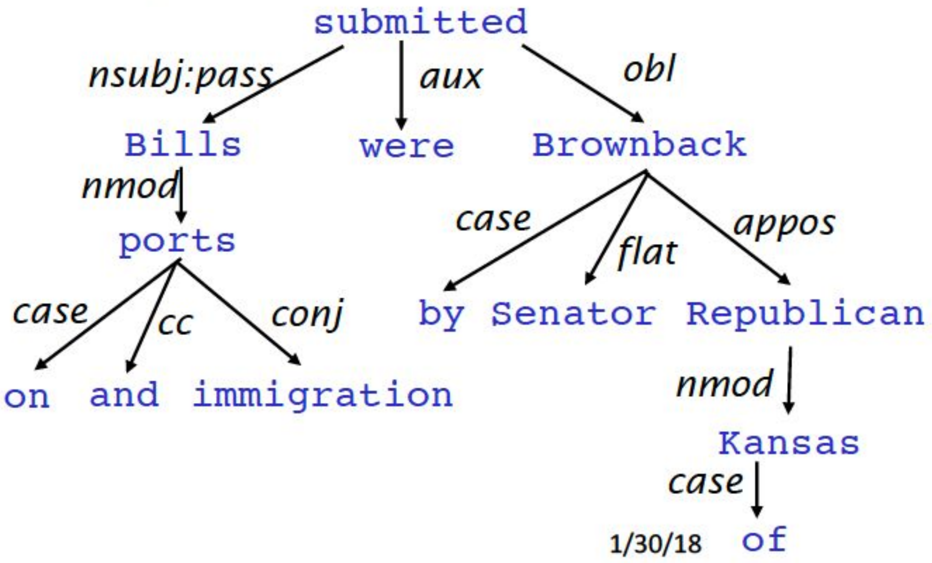


Dependency Grammar and Dependency Structure

Dependency syntax postulates that syntactic structure consists of relations between lexical items, normally 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)



Dependency Relations

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)
https://web.stanford.edu/~jurafsky/slp3/14.pdf

Dependency Grammar and Dependency Structure

- Some people draw the arrows one way; some the other way!
  - Tesnière had them point from head to dependent…
  - Ours will point from head to dependent
- Usually add a fake ROOT so every word is a dependent of precisely 1 other node

Dependency Conditioning Preferences

What are the sources of information for dependency parsing?

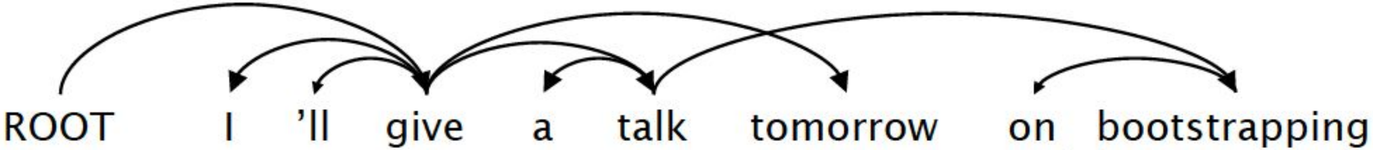
1. Bilexical affinities [discussion → issues] is plausible
2. Dependency distance mostly with nearby words
3. Intervening material

   Dependencies rarely span intervening verbs or punctuation
4. 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) it is a dependent of
  - i.e., find the right outgoing arrow from each word
- 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



Methods of Dependency Parsing

1. Dynamic programming
2. Graph algorithms

   You create a Minimum Spanning Tree for a sentence

   McDonald et al.'s (2005) MSTParser scores dependencies independently using an ML classifier (he uses MIRA, for online learning, but it can be something else)
3. Constraint Satisfaction

   Edges are eliminated that don't satisfy hard constraints. Karlsson (1990), etc.
4. "Transition-based parsing" or "deterministic dependency parsing"

   Greedy choice of attachments guided by good machine learning classifiers

   MaltParser (Nivre et al. 2008). Has proven highly effective.

1. Transition-based dependency parsing

How to extract the word dependencies

The Parser :

- 3 components : Stack , Buffer and Dependency Arc Stack
- Parse actions (Transitions) : Shift ,Left- Arc, Right- Arc
- Operations

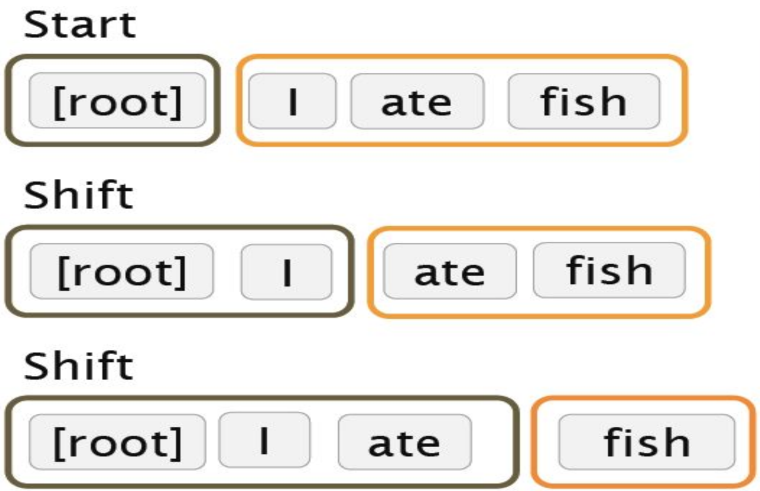
1. Start: Stack-root, Buffer – Input sentence (all words), empty Arc Stack
2. A series of actions
3. End : Stack-root, empty Buffer, Arc Stack- extracted dependencies

1. Transition-based dependency parsing
Examples

Arc-standard transition-based parser

(there are other transition schemes …)

Analysis of "I ate fish"

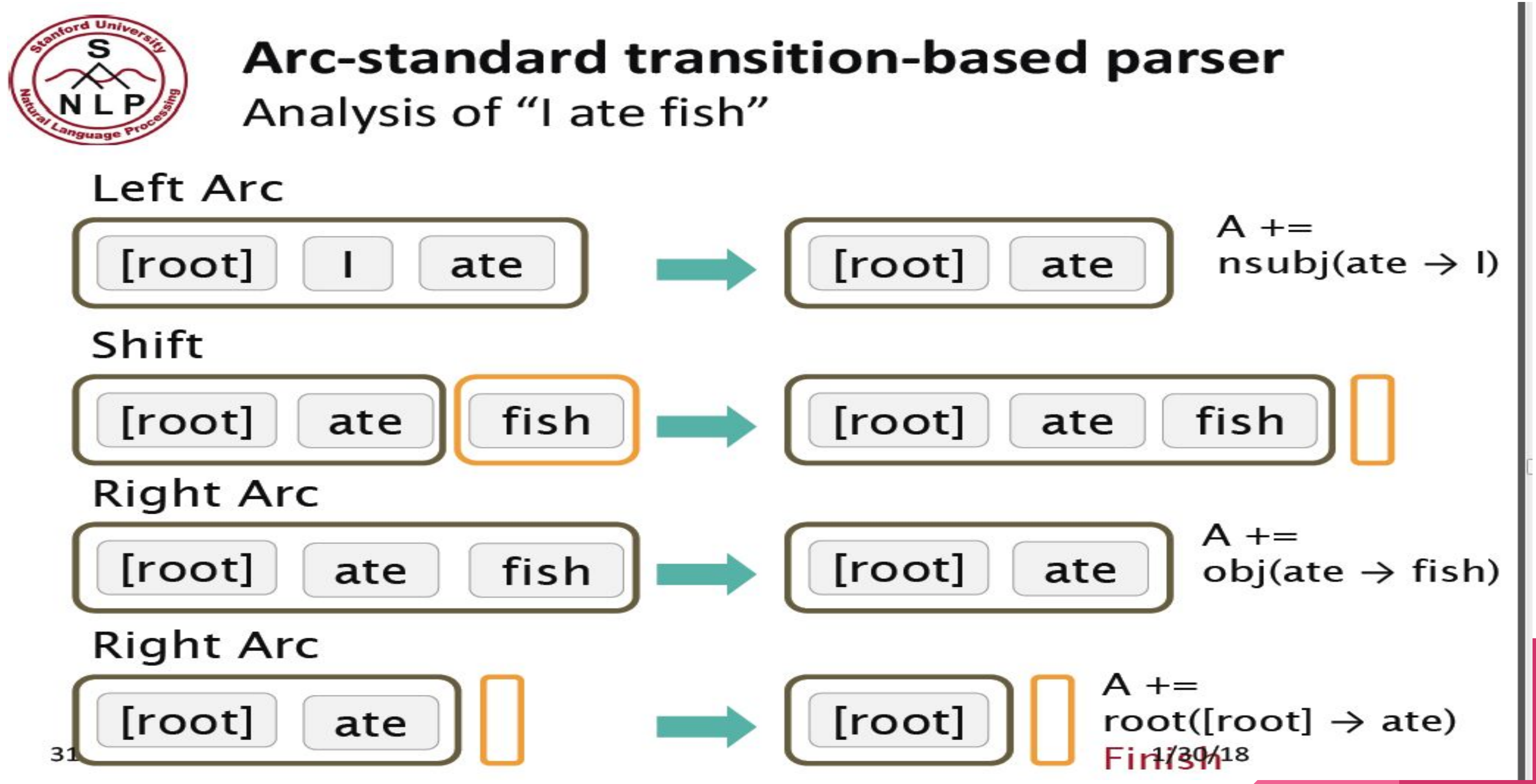


Start: $\sigma = [\text{ROOT}]$, $\beta = w_1, \ldots, w_n$, $A = \varnothing$

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

Finish: $\beta = \varnothing$

1. Transition-based dependency parsing
Examples



1. Transition-based dependency parsing
Algorithm

Basic transition-based dependency parser

Start: $\sigma = [\text{ROOT}]$, $\beta = w_1, \ldots, w_n$, $A = \varnothing$

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

Finish: $\sigma = [w]$, $\beta = \varnothing$

Transition-based dependency parsing

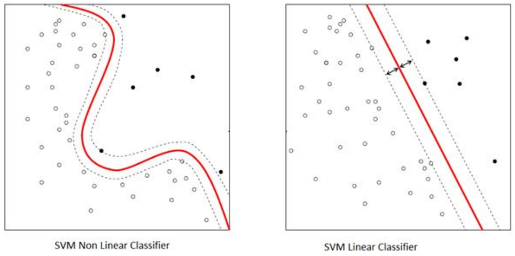
How to choose actions

-Classifier predicts next actions based on the word features

Example:
Support Vector Machine (SVM)

- **Find the hyperplane that separate 2 different classes.**

- **Maximize gap between classes and decision boundary**



Ref: https://en.wikipedia.org/wiki/Support_vector_machine

Transition-based dependency parsing

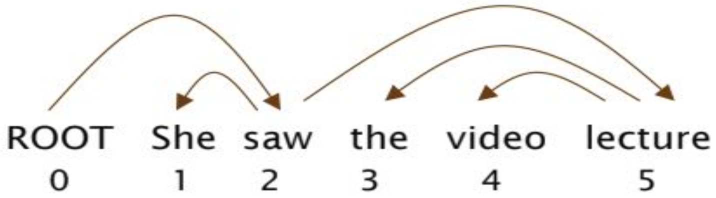
Examples of features or parse configuration

- POS of word in Stack, POS of word in buffer;
- Length of the dependency Arc;
- Meaning and distance between the parsed words etc
- Examples : The head is a verb, the dependent is a noun

  then predict Right-Arc, to go from left to right

Transition-based dependency parsing

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

Transition-based dependency parsing

- Basic and deterministic parsing
  - No Searching and predict actions based on words in buffer and Stack
  - - Simple , Fast and efficient
  - - Good accuracy

- Challenges
  - Ambiguities
  - - Part of speech labelling ; Multiple sentence attachments or words order

Projectivity

Projectivity is a principle of tree structures.

A tree structure is said to be *projective* if there are **NO** crossing of dependency edges. Any occurrence of such crossing (a.k.a. discontinuity or projectivity violation), the structure will be considered as *non-projective*.

But non-projective structures are commonly seen in natural languages, especially non-English.

- You can't easily get the semantics of certain constructions right without these nonprojective dependencies

Handling non-projectivity

Many parsing algorithms are restricted to projective dependency graphs.

Two Main Approaches to Handle This Problem

1. Algorithms for non-projective dependency parsing:
   - Constraint satisfaction methods (Foth et al., 2004)
   - McDonald’s spanning tree algorithm (McDonald et al., 2005)
   - Covington’s algorithm (Nivre, 2006)

2. Post-processing of projective dependency graphs:
   - Pseudo-projective parsing (Nivre and Nilsson, 2005)
   - Corrective modeling (Hall and Nov´ak, 2005)
   - Approximate non-projective parsing (McDonald and Pereira, 2006)

Neural Dependency Parser

Problems

- Sparsity - too many features contributing too little
- Incompleteness - arc from s1 to s2???
- Expensive feature computation - millions of features -> slow lookup

[Chen and Manning. 2014]

Sparse Feature representation

Combination of 1~3 elements from the configuration

Words, word-pair, three-word conjunctions

Dimension - 10^6 - 10^7 - sparse !!

This is because words are represented as discrete symbols and not dense vectors.

$$s1.w = \text{good} \wedge s1.t = \text{JJ}$$
$$s2.w = \text{has} \wedge s2.t = \text{VBZ} \wedge s1.w = \text{good}$$
$$lc(s_2).t = \text{PRP} \wedge s_2.t = \text{VBZ} \wedge s_1.t = \text{JJ}$$
$$lc(s_2).w = \text{He} \wedge lc(s_2).l = \text{nsubj} \wedge s_2.w = \text{has}$$

Single-word features (9)
$s_1.w$; $s_1.t$; $s_1.wt$; $s_2.w$; $s_2.t$; $s_2.wt$; $b_1.w$; $b_1.t$; $b_1.wt$
Word-pair features (8)
$s_1.wt \circ s_2.wt$; $s_1.wt \circ s_2.w$; $s_1.wts_2.t$; $s_1.w \circ s_2.wt$; $s_1.t \circ s_2.wt$; $s_1.w \circ s_2.w$ $s_1.t \circ s_2.t$; $s_1.t \circ b_1.t$
Three-word feaures (8)
$s_2.t \circ s_1.t \circ b_1.t$; $s_2.t \circ s_1.t \circ lc_1(s_1).t$; $s_2.t \circ s_1.t \circ rc_1(s_1).t$; $s_2.t \circ s_1.t \circ lc_1(s_2).t$; $s_2.t \circ s_1.t \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ rc_1(s_2).t$; $s_2.t \circ s_1.w \circ lc_1(s_1).t$; $s_2.t \circ s_1.w \circ b_1.t$

Table 1: The feature templates used for analysis. $lc_1(s_i)$ and $rc_1(s_i)$ denote the leftmost and rightmost children of s_i, w denotes word, t denotes POS tag.

[Chen and Manning. 2014]

Incomplete Feature representation

Eg conjunction of s1, b2 is omitted

You don't have features for some combination of the configurations.

[Chen and Manning. 2014]

Expensive Computation

95% time spent computing feature and looking up either weights.

Lookups are slow because there are too many features.

[Chen and Manning. 2014]

Neural Network Based Parser - Model

Softmax layer:

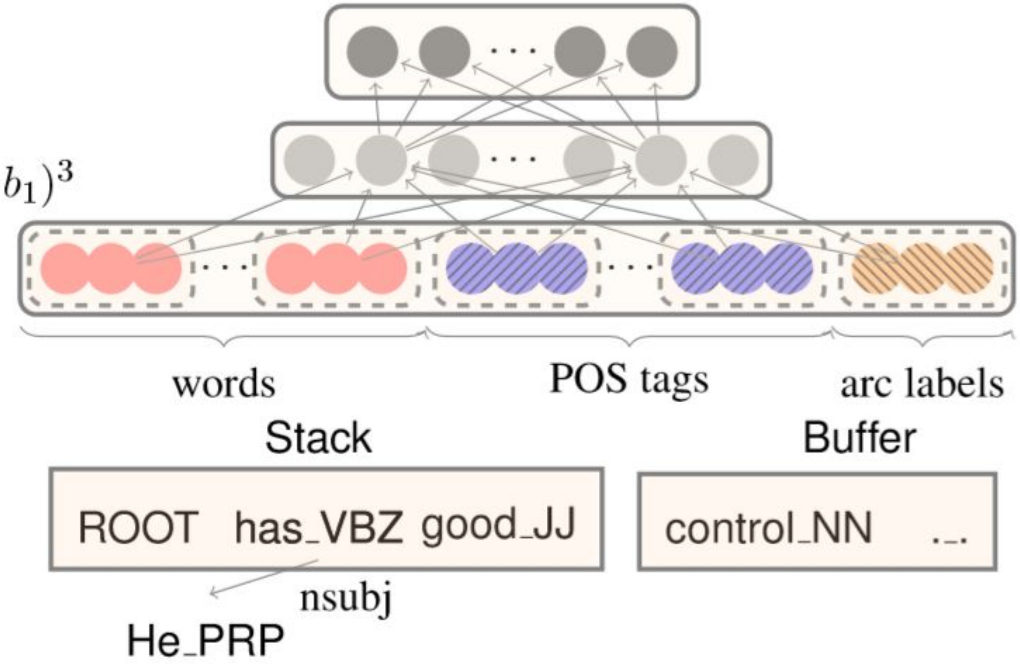
$$p = \mathtt{softmax}(W_2 h)$$

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



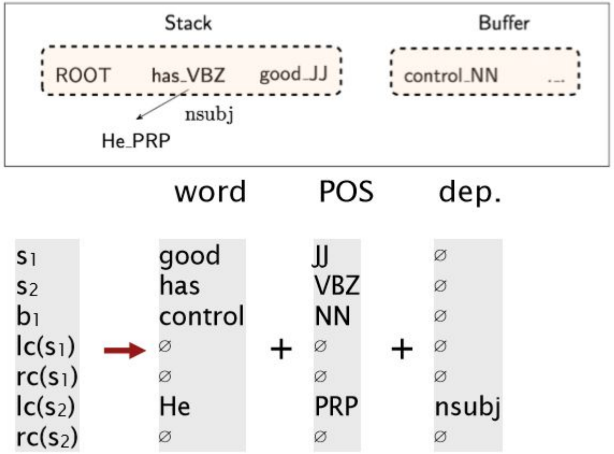
[Chen and Manning. 2014]

Distributed Representation

- Instead of using discrete representation for word we use word vectors
- Reduced dimensionality.
- We are using the semantics of the words instead of treating them as unrelated symbols.
- POS and dependency labels(from the partially generated graph) can also be represented as dense vectors. (to preserve their similarities)
- Can train word vectores while training Model weights.

[Chen and Manning. 2014]

Features - Input Layer



- Top 3 words from stack and buffer. s1, s2, s3, b1, b2, b3.
- The first and second leftmost / rightmost children of the top two words on the stack: lc1(si), rc1(si), lc2(si), rc2(si), i = 1, 2.
- The leftmost of leftmost / rightmost of rightmost children of the top two words on the stack: lc1(lc1(si)), rc1(rc1(si)), i = 1, 2.
- POS tags for all those words.
- Arc labels for all those words (except for 6 words of stack/buffer).

[Chen and Manning. 2014]

Output

The last layer of the network is a softmax layer to convert the value of the inner layers to probability.

The nodes are LEFT-ARC, RIGHT-ARG along with the label and SHIFT.

The value with the max probability is chosen to the next action for the configuration.

[Chen and Manning. 2014]

Training

Cost Function : Cross Entropy error with L2 regularization.

Parameters : {Ww 1 , Wt 1 , Wl 1 , b1, W2, Ew, Et , El}

Initialization : Ew - word vectors; Et , El - radom (-0.01.0.01)

Optimization Algorithm : mini-batch AdaGrad with drop out

Hyperparameters: h, α , λ, d

[Chen and Manning. 2014]

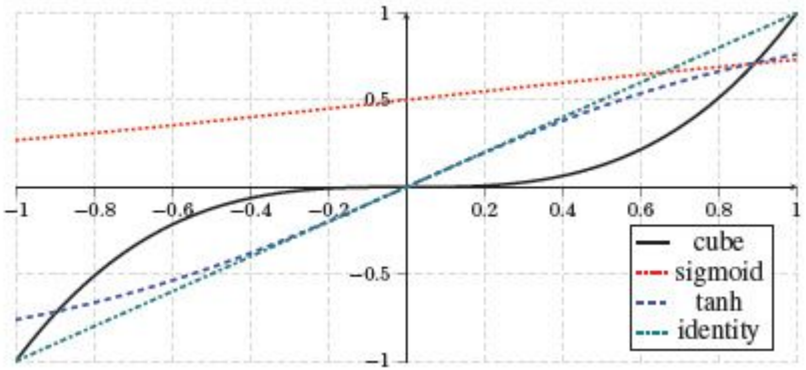
Activation function

Why does the activation function needs to be non-linear?

Which one to choose from? Sigmoid, tanh, hard tanh, ReLU

https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

Should be part-wise differentiable

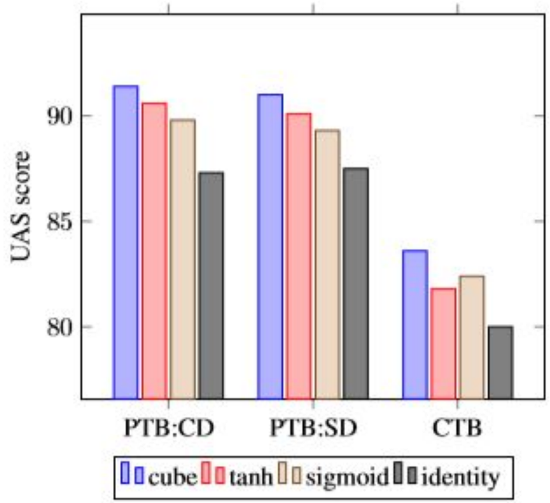


[Chen and Manning. 2014]

What works - Model Analysis

Cube Activation Function

Better captures higher order interactions of selected features



Input interpretation:

expand	$(a+b+c)^3$

Result:

$$a^3 + 3a^2 b + 3a^2 c + 3ab^2 + 6abc + 3ac^2 + b^3 + 3b^2 c + 3bc^2 + c^3$$

(10 terms)

$$g(w_1x_1 + \ldots + w_mx_m + b) = \sum_{i,j,k}(w_iw_jw_k)x_ix_jx_k + \sum_{i,j} b(w_iw_j)x_ix_j \ldots$$

[Chen and Manning. 2014]

What works - Model Analysis

POS tags and arc labels dense representations

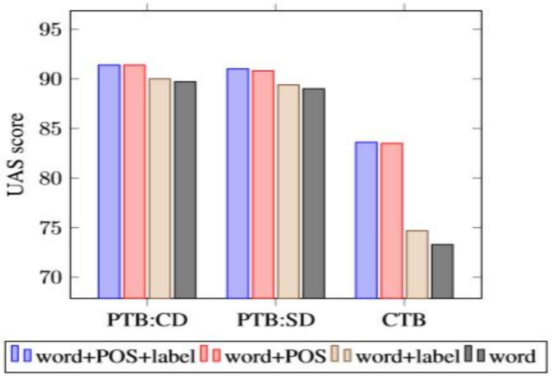
Captures semantic similarities between POS tags (”apple” should be close to “apples”)

Automatically finds out dominant features instead of hand made features

– Feature 1: s1.t, s2.t, lc(s1).t.

– Feature 2: rc(s1).t, s1.t, b1.t.

– Feature 3: s1.t, s1.w, lc(s1).t, lc(s1).l.

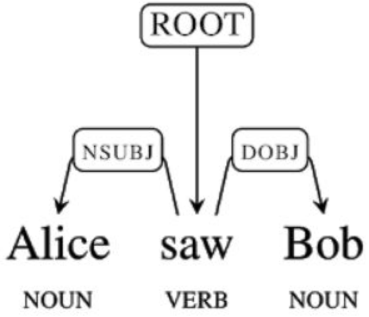


[Chen and Manning. 2014]

About POS tags

(Part of Speech,POS) is a category of words which play similar roles within the grammatical structure of sentences.

Commonly listed English parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, determiner and etc.



Simple example: He (pronoun) eats (verb) apples (noun)

https://en.wikipedia.org/wiki/Part-of-speech_tagging

SyntaxNet (For discussion)

https://ai.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html

“it is very important to use *beam search* in the model. Instead of simply taking the first-best decision at each point, multiple partial hypotheses are kept at each step, with hypotheses only being discarded when there are several other higher-ranked hypotheses under consideration.”

- Bigger, deeper networks with better tuned hyperparameters.
- Beam search.
- CRF-style inference on the decision sequence.

Conclusions

a) Syntactic Structure (Constituency)

b) Dependency Grammar

   Introduction of dependency grammar & structure

c) Transition-based dependency parsing

   Arc-standards: parser action & arc-label (dependency label)

d) Neural dependency parsing

   Motivation+3 features+neural network model with cubic activation function.

Reference

Deep Learning for NLP Lecture 7.

http://web.stanford.edu/class/cs224n/lectures/lecture7.pdf

A Fast and Accurate Dependency Parser using Neural Networks

[Chen and Manning 2014]

https://cs.stanford.edu/~danqi/papers/emnlp2014.pdf

http://www.wolframalpha.com/

Wikipedia article on dependency grammar and Context free grammar

Thank you