COMP90042

Workshop Week 06

- Homework 3
  - Due: Sunday, April 15

- Workshop solutions released (week 2/3/4)

Syllabus

1	Introduction and Preprocessing	Text classification
2	Lexical semantics	Distributional semantics
3	Part of Speech Tagging	Hidden Markov Models
4	Unsupervised Hidden Markov Models	Context-Free Grammars
5	Probabilistic Parsing	Dependency parsing
	Easter holiday break	
6	N-gram language modelling	Deep learning for language models and tagging
7	Information Extraction	Question Answering
8	Topic Models	*ANZAC day holiday*
9	Information Retrieval -- Boolean search and the vector space model	Indexing and querying in the vector space model, evaluation
10	Index and vocabulary compression	Efficient query processing
11	The Web as a Graph: Page-rank & HITS	Machine Translation (word based)
12	Machine translation (phrase based) and neural encoder-decoder	Subject review

Outline

- Unsupervised Learning (HMMs)
  - Training set, extra data, and test set
- Probabilistic parsing
  - The PCYK algorithm
- Dependency parsing

Datasets

- Training set
  - $tagged\ D_{train}$
- Extra data without labels/tags
  - D_{extra}
    - A potential training set, but can't be used directly
- Test set
  - $tagged\ D_{test}$

- A baseline
  - $tagged\ D_{train} \rightarrow HMM$
  - $D_{test} \xrightarrow{HMM} tagged\ D_{test}$
- Problems to solve (modify pi/A/B properly)
  - Unseen words
  - Different tagsets
    - Some tags are the same but have different names
    - Special tags for test set

- A baseline
  - $tagged\ D_{train} \rightarrow HMM$
  - $D_{test} \xrightarrow{HMM} tagged\ D_{test}$
- With untagged extra data
  - $tagged\ D_{train} \rightarrow HMM$
  - $D_{extra} \xrightarrow{HMM} tagged\ D_{extra}$
  - $tagged\ D_{train}$ & $tagged\ D_{extra} \rightarrow HMM\_new$
  - $D_{test} \xrightarrow{HMM\_new} tagged\ D_{test}$

- A baseline
  - $tagged\ D_{train} \rightarrow HMM$
  - $D_{test} \xrightarrow{HMM} tagged\ D_{test}$
- Hard-EM
  - $tagged\ D_{train} \rightarrow HMM^{(0)}$
  - for i in range(n):
    - $D_{extra} \xrightarrow{HMM^{(i)}} tagged\ D_{extra}$
    - $tagged\ D\_train$ & $tagged\ D_{extra} \rightarrow HMM^{(i+1)}$
  - $D_{test} \xrightarrow{HMM^{(n)}} tagged\ D_{test}$

Outline

- Unsupervised Learning (HMMs)
  - Training set, extra data, and test set
- Probabilistic parsing
  - The PCYK algorithm
- Dependency parsing

CFGs

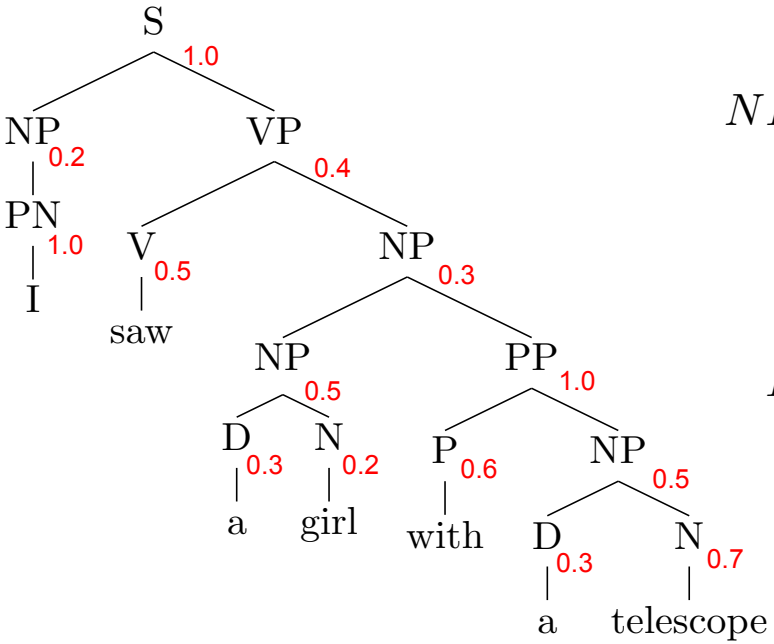
$S \rightarrow NP\ VP$ 1.0

$VP \rightarrow V$ 0.2
$VP \rightarrow V\ NP$ 0.4
$VP \rightarrow VP\ PP$ 0.4

$NP \rightarrow NP\ PP$ 0.3
$NP \rightarrow D\ N$ 0.5
$NP \rightarrow PN$ 0.2

$PP \rightarrow P\ NP$ 1.0

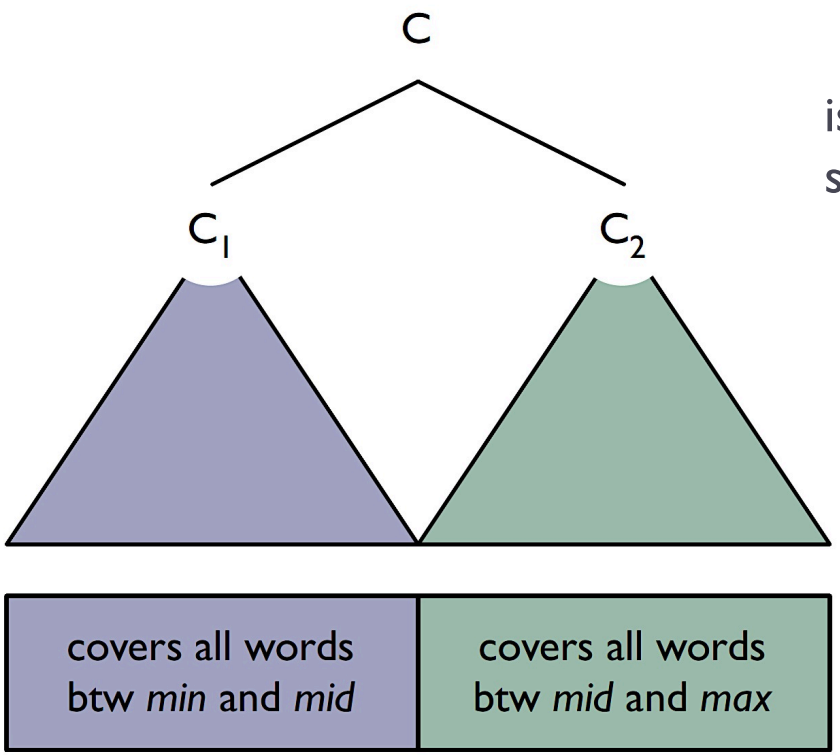
$N \rightarrow girl$ 0.2
$N \rightarrow telescope$ 0.7
$N \rightarrow sandwich$ 0.1
$PN \rightarrow I$ 1.0
$V \rightarrow saw$ 0.5
$V \rightarrow ate$ 0.5
$P \rightarrow with$ 0.6
$P \rightarrow in$ 0.4
$D \rightarrow a$ 0.3
$D \rightarrow the$ 0.7



$$p(T) = 1.0 \times 0.2 \times 1.0 \times 0.4 \times 0.5 \times 0.3 \times$$
$$0.5 \times 0.3 \times 0.2 \times 1.0 \times 0.6 \times 0.5 \times 0.3 \times 0.7$$
$$= 2.26 \times 10^{-5}$$

Intuition

$$C \rightarrow C_1 \ C_2$$



For every C choose C_1, C_2 and mid such that

$$P(T_1) \times P(T_2) \times P(C \rightarrow C_1 C_2)$$

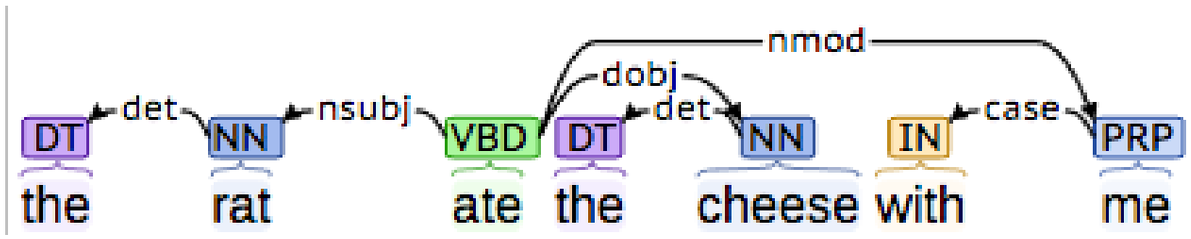
is maximal, where T_1 and T_2 are left and right subtrees.

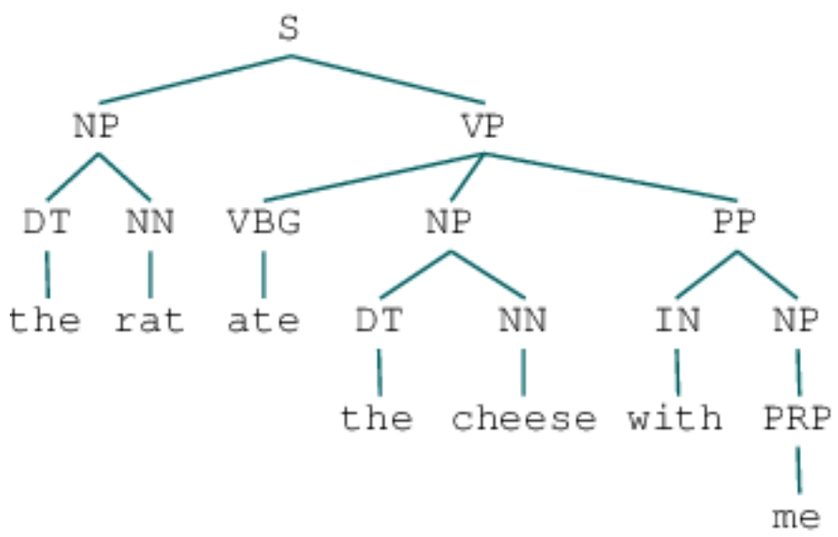
Outline

- Unsupervised Learning (HMMs)
  - Training set, extra data, and test set
- Probabilistic parsing
  - The PCYK algorithm
- Dependency parsing

Why dependencies?

- Dependency tree more directly represents the core of the sentence: *who did what to whom?*
  - captured by the links incident on verb nodes, e.g., NSUBJ, DOBJ etc; easier to answer questions like:
    - what was the main thing being expressed in the sentence (*eating*)

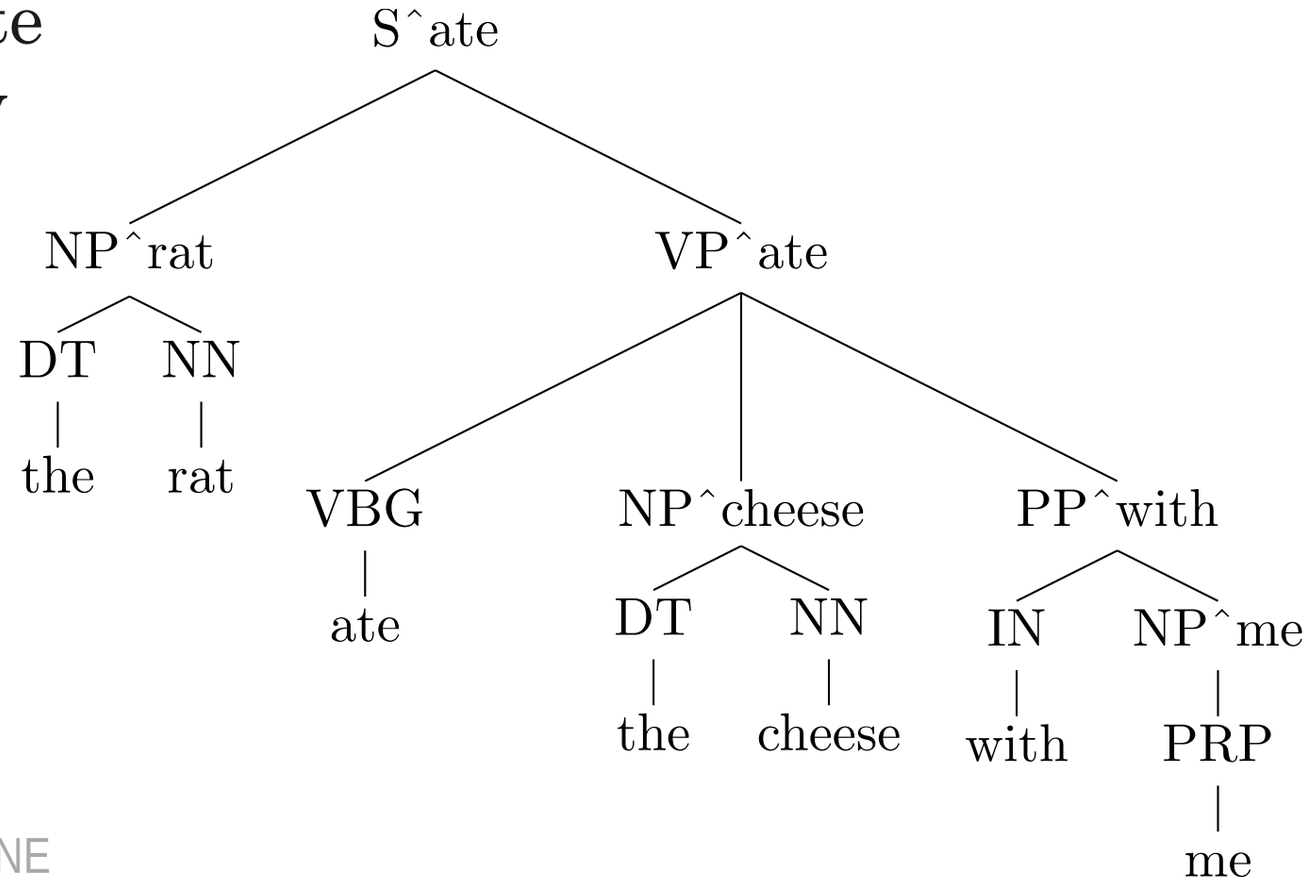




  - more minor details are buried deeper in the tree (e.g., adjectives, determiners etc)

dependency vs head

- Close similarity with ‘head’ in phrase-structure grammars
  - the ‘head’ of an XP is (mostly) an X,
    i.e., noun in a NP, verb in a VP etc.
    see https://en.wikipedia.org/wiki/Head_(linguistics)
  - main dependency edges captured in rewrite rules
    - S^ate -> NP^rat VP^ate
      captures dependency
      rat ← ate



(Non-)projectivity

- A tree is *projective* if, for all arcs from head to dependent
  - there is a path from the head to every word that lies between the head and the dependent
- More simply, the tree can be drawn on a plane without any arcs crossing
- Most sentences are projective, however exceptions exist (fairly common in other languages)

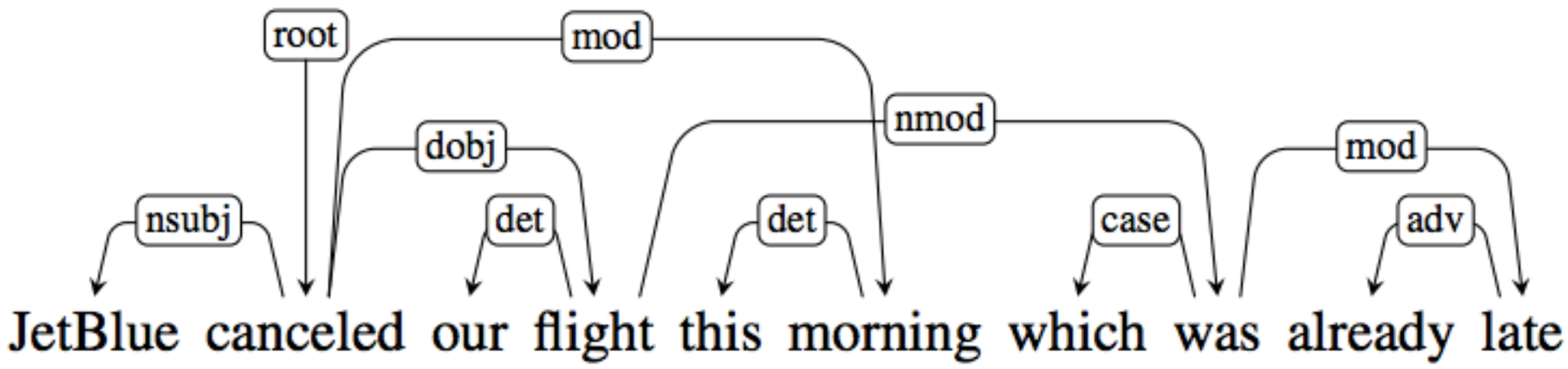
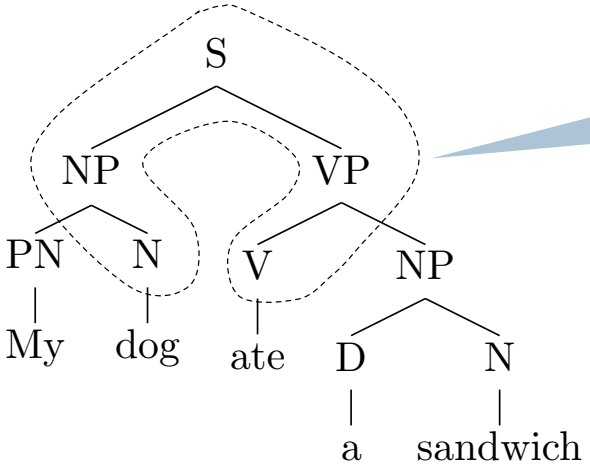


Figure JM3, Ch 14

Constituent and dependency representations

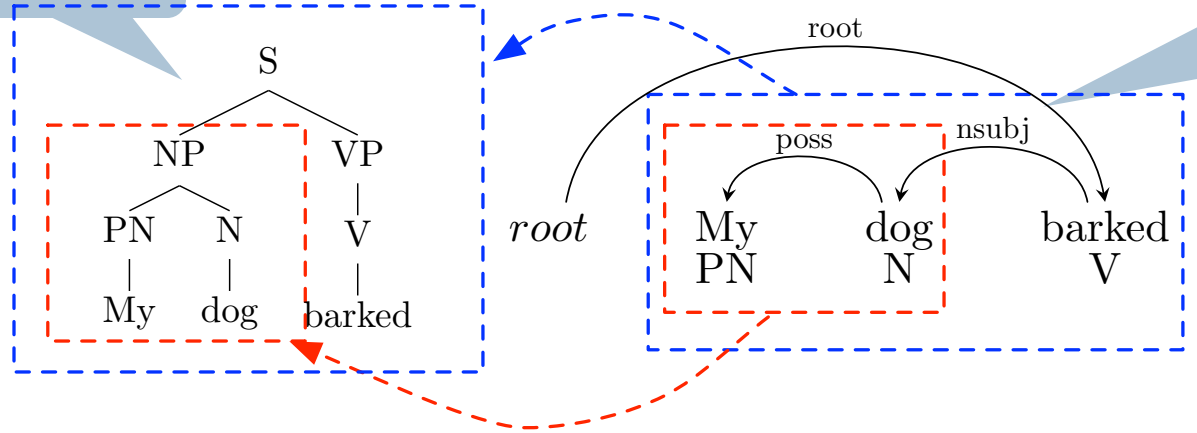
- Constituent trees can (potentially) be converted to dependency trees



One potential rule to extract nsubj dependency

- Dependency trees can (potentially) be converted to constituent trees

Recovering labels is harder



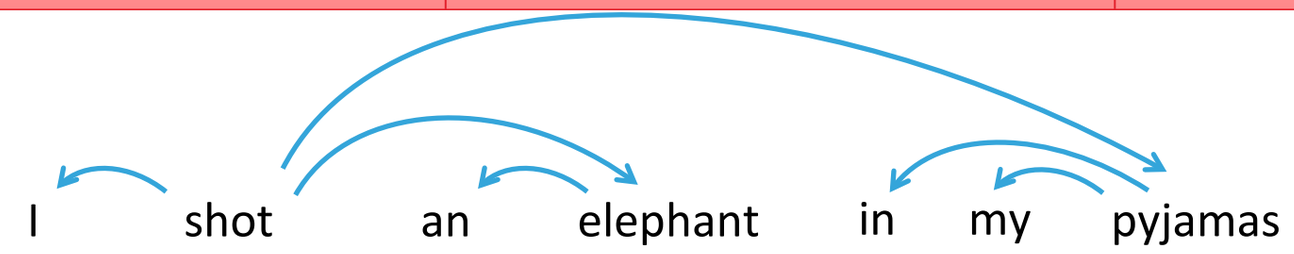
Roughly: every word along with all its dependents corresponds to a phrase = to an inner node in the constituent tree

Example

- I shot an elephant in my pyjamas

Buffer	Stack	Action
I shot an elephant in my pyjamas		Shift
shot an elephant in my pyjamas	I	Shift
an elephant in my pyjamas	I, shot	Arc-left
an elephant in my pyjamas	shot	Shift
elephant in my pyjamas	shot, an	Shift
in my pyjamas	shot, an, elephant	Arc-left
in my pyjamas	shot, elephant	Arc-right
in my pyjamas	shot	Shift
…	…	…
	shot	<done>

Generated parse:



References

- Slides from Ivan Titov
  - http://ivan-titov.org/teaching/nlp1-15/index.html
  - http://ivan-titov.org/teaching/nlmi-15/node2.html