Natural Language Processing 1

Lecture 4: Syntactic parsing (continued). Lexical semantics

Katia Shutova

ILLC
University of Amsterdam

6 November 2019

Outline.

Syntactic parsing

Introduction to semantics & lexical semantics

A simple CFG for a fragment of English

rules

```
S -> NP VP
VP -> VP PP
VP -> V
VP -> V NP
VP -> V VP
NP -> NP PP
PP -> P NP
```

lexicon

```
V -> can
V -> fish
NP -> fish
NP -> rivers
NP -> pools
NP -> December
NP -> Scotland
NP -> it
NP -> they
P -> in
```

Chart parsing

chart store partial results of parsing in a vector

edge representation of a rule application

Edge data structure:

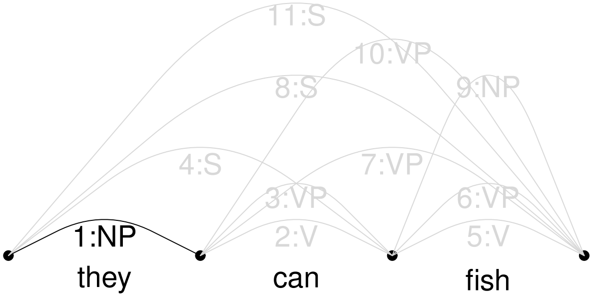
[*id*,*left_vtx*, *right_vtx*,*mother_category*, *dtrs*]

```
. they . can . fish .
0        1      2        3
```

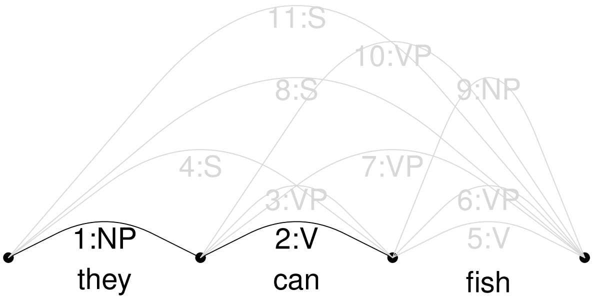
Fragment of chart:

id	left	right	mother	daughters
1	0	1	NP	(they)
2	1	2	V	(can)
3	1	2	VP	(2)
4	0	2	S	(1 3)

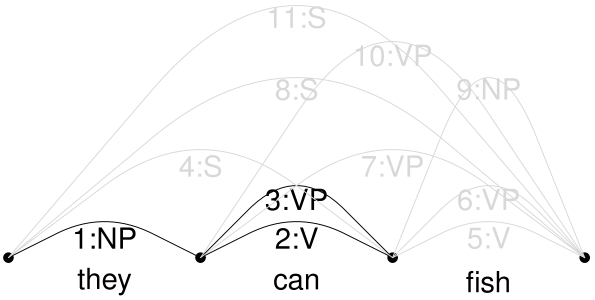
Bottom up parsing: example



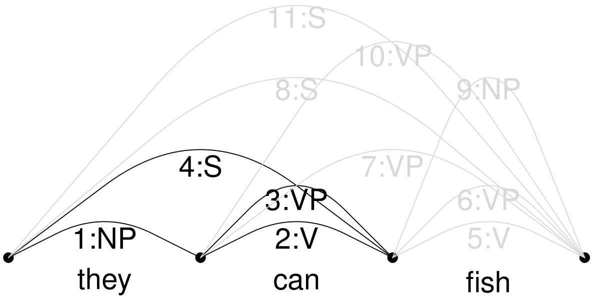
Bottom up parsing: example



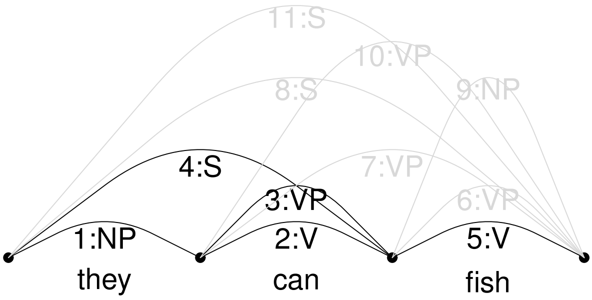
Bottom up parsing: example



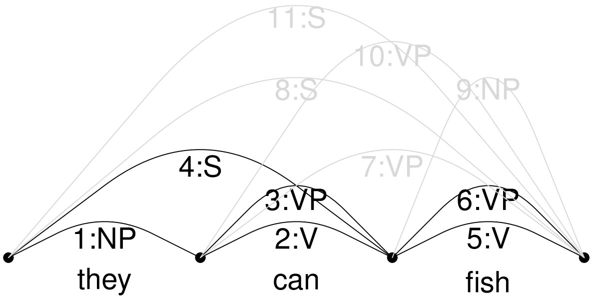
Bottom up parsing: example



Bottom up parsing: example

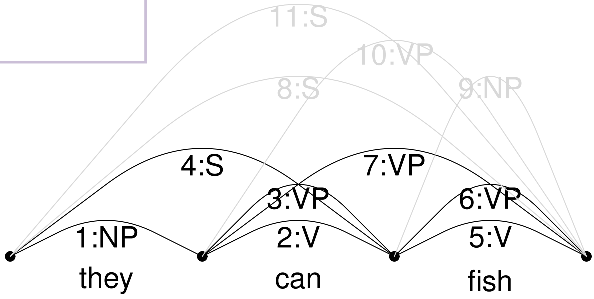


Bottom up parsing: example

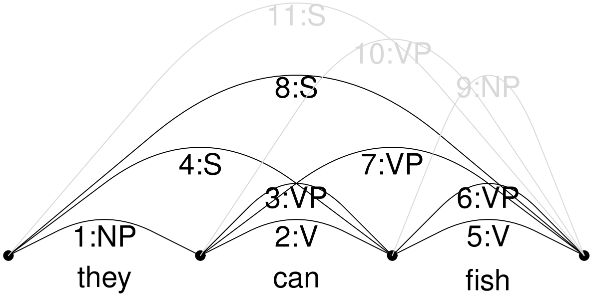


Bottom up parsing: example

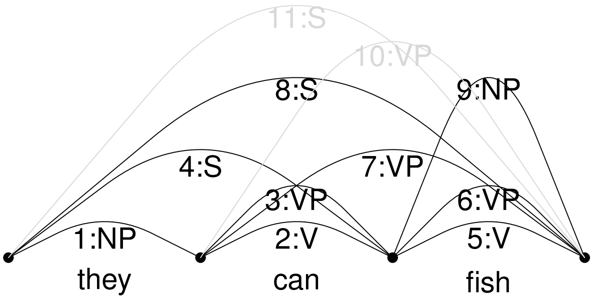
```
S -> NP VP
VP -> VP PP
VP -> V
VP -> V NP
VP -> V VP
NP -> NP PP
PP -> P NP
```



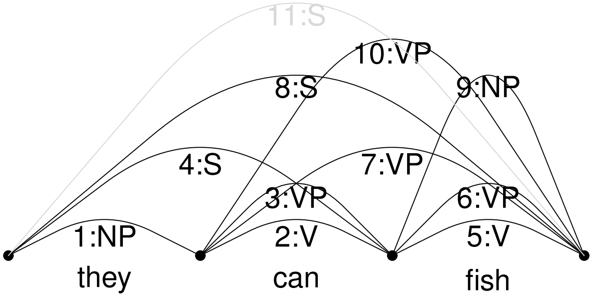
Bottom up parsing: example



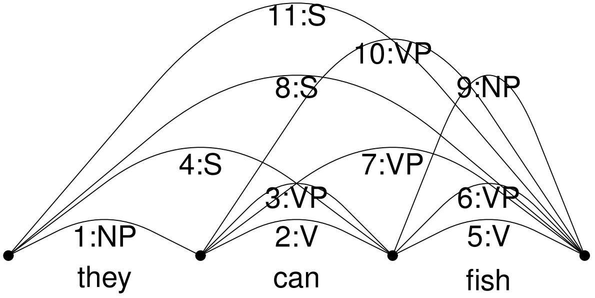
Bottom up parsing: example



Bottom up parsing: example



Bottom up parsing: example



Resulting chart

```
. they . can . fish .
0      1     2      3
```

id	left	right	mother	daughters
1	0	1	NP	(they)
2	1	2	V	(can)
3	1	2	VP	(2)
4	0	2	S	(1 3)
5	2	3	V	(fish)
6	2	3	VP	(5)
7	1	3	VP	(2 6)
8	0	3	S	(1 7)
9	2	3	NP	(fish)
10	1	3	VP	(2 9)
11	0	3	S	(1 10)

Output results for spanning edges

Spanning edges are 8 and 11:

Output results for 8

```
(S (NP they) (VP (V can) (VP (V fish))))
```

Output results for 11

```
(S (NP they) (VP (V can) (NP fish)))
```

A bottom-up chart parser

Parse:
Initialize the chart
For each word *word*, let *from* be left vtx,
to right vtx and *dtrs* be (*word*)
For each category *category*
lexically associated with *word*
Add new edge *from*, *to*, *category*, *dtrs*
Output results for all spanning edges

Inner function

Add new edge *from*, *to*, *category*, *dtrs*:
  Put edge in chart: [*id*,*from*,*to*, *category*,*dtrs*]
  For each *rule* $lhs \rightarrow cat_1 \ldots cat_{n-1}, category$
    Find sets of contiguous edges
    [id_1,$from_1$,to_1, cat_1,$dtrs_1$] ...
      [id_{n-1},$from_{n-1}$,*from*, cat_{n-1},$dtrs_{n-1}$]
    (such that $to_1 = from_2$ etc)
    For each set of edges,
      Add new edge $from_1$, *to*, *lhs*, ($id_1 \ldots id$)

Packing

if we dont use packing, then对于已经找到的edge，我们在加入新的edge之后，要把已经找到的edge再搜索一遍。如果用packing，意味着把已经找到的edge打包装好，新的edge只需要加在原有的edge上即可。目的：为了接受computation cost

To make parsing more efficient:

- don’t add equivalent edges as whole new edges
- *dtrs* is a set of lists of edges (to allow for alternatives)

about to add: [*id*,*l_vtx*, *right_vtx*,*ma_cat*, *dtrs*]
and there is an existing edge:

[*id-old*,*l_vtx*, *right_vtx*,*ma_cat*, *dtrs-old*]

we simply modify the old edge to record the new dtrs:

[*id-old*,*l_vtx*, *right_vtx*,*ma_cat*, *dtrs-old* ∪ *dtrs*]

and do not recurse on it: never need to continue computation with a packable edge.

Packing example

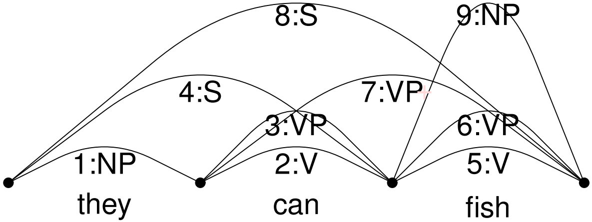
1	0	1	NP	{(they)}
2	1	2	V	{(can)}
3	1	2	VP	{(2)}
4	0	2	S	{(1 3)}
5	2	3	V	{(fish)}
6	2	3	VP	{(5)}
7	1	3	VP	{(2 6)}
8	0	3	S	{(1 7)}
9	2	3	NP	{(fish)}

Instead of edge `10 1 3 VP {(2 9)}`

`7 1 3 VP {(2 6), (2 9)}`

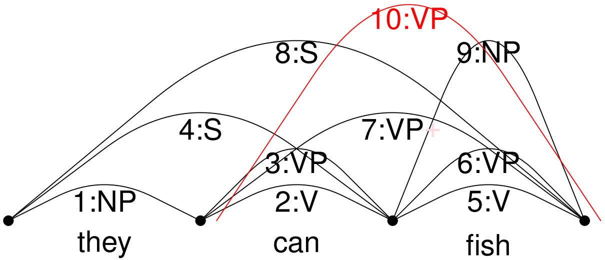
and we're done

Packing example



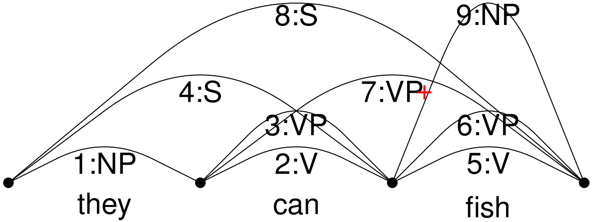
Both spanning results can now be extracted from edge 8.

Packing example



Both spanning results can now be extracted from edge 8.

Packing example



Both spanning results can now be extracted from edge 8.

Probabilistic Parsing

- How can we choose the correct tree for a given sentence?
- Traditional approach: grammar rules hand-written by linguists
  - constraints added to limit unlikely parses for sentences
  - hand-written grammars are not robust: often fail to parse new sentences.
- Current approach: use probabilities
  - Probabilitistic CFG (PCFG)
  - a CFG where each rule is augmented with a probability

An Example PCFG

$S \rightarrow NP\ VP$	.8
$S \rightarrow VP$	.2
$NP \rightarrow D\ N$	.4
$NP \rightarrow NP\ PP$	.4
$NP \rightarrow PN$	.2
$VP \rightarrow V\ NP$	.7
$VP \rightarrow VP\ PP$	.3
$PP \rightarrow P\ NP$	1

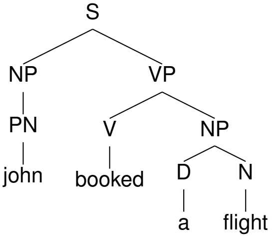
$D \rightarrow the$	.8
$D \rightarrow a$	.2
$N \rightarrow flight$	1
$PN \rightarrow john$	.9
$PN \rightarrow schiphol$	.1
$V \rightarrow booked$	1
$P \rightarrow from$	1

How to compute the probability of a parse tree?

Computing the probability of a parse tree

The probability of a parse tree for a given sentence:

- the product of the probabilities of all the grammar rules used in the sentence derivation.

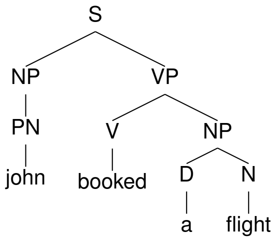


$$\begin{aligned} P(t) = & P(S \rightarrow NP\ VP) \times P(NP \rightarrow PN) \times P(PN \rightarrow john) \times \\ & P(VP \rightarrow V\ NP) \times P(V \rightarrow booked) \times \\ & P(NP \rightarrow D\ N) \times P(D \rightarrow a) \times P(N \rightarrow flight) \\ = & .8 \times .2 \times .9 \times .7 \times .4 \times .2 \times 1 \\ = & .008064 \end{aligned}$$

Computing the probability of a parse tree

The probability of a parse tree for a given sentence:

- the product of the probabilities of all the grammar rules used in the sentence derivation.



$$
\begin{aligned}
P(t) = & P(S \rightarrow NP\ VP) \times P(NP \rightarrow PN) \times P(PN \rightarrow john) \times \\
& P(VP \rightarrow V\ NP) \times P(V \rightarrow booked) \times \\
& P(NP \rightarrow D\ N) \times P(D \rightarrow a) \times P(N \rightarrow flight)
\end{aligned}
$$

$$= .8 \times .2 \times .9 \times .7 \times .4 \times .2 \times 1$$

$$= .008064$$

Computing the probability of a parse tree

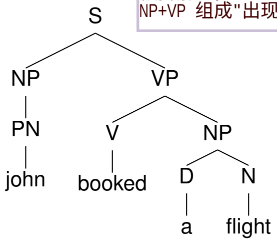
The probability of a parse tree for a given sentence:

- the product of the probabilities of all the grammar rules used in the sentence derivation.

被拆解为, or "a Sentence由NP+VP 组成"出现的概率

名词有许多种可能，这里问名词是人称的可能性

人称是John的可能性



$$P(t) = P(S \rightarrow NP\ VP) \times P(NP \rightarrow PN) \times P(PN \rightarrow john) \times P(VP \rightarrow V\ NP) \times P(V \rightarrow booked) \times P(NP \rightarrow D\ N) \times P(D \rightarrow a) \times P(N \rightarrow flight)$$

$$= .8 \times .2 \times .9 \times .7 \times .4 \times .2 \times 1$$

$$= .008064$$

Disambiguation with PCFGs

These probabilities can provide a criterion for disambiguation:

- i.e. a ranking over possible parses for any sentence
- we can choose the parse tree with the highest probability.

Disambiguation with PCFGs

$S \rightarrow NP\ VP$	.8
$S \rightarrow VP$	.2
$NP \rightarrow D\ N$	.4
$NP \rightarrow NP\ PP$	.4
$NP \rightarrow PN$	.2
$VP \rightarrow V\ NP$	.7
$VP \rightarrow VP\ PP$	.3
$PP \rightarrow P\ NP$	1

$D \rightarrow$ *the*	.8
$D \rightarrow$ *a*	.2
$N \rightarrow$ *flight*	1
$PN \rightarrow$ *john*	.9
$PN \rightarrow$ *schiphol*	.1
$V \rightarrow$ *booked*	1
$P \rightarrow$ *from*	1

John booked a flight from Schiphol

$P(t_1) = 6.4512 \times 10^{-5}$

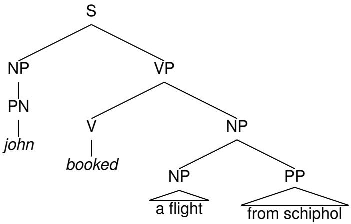
$P(t_2) = 4.8384 \times 10^{-5}$

Disambiguation with PCFGs

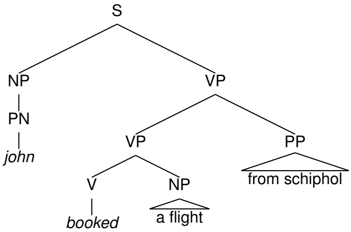
$S \to NP\ VP$	.8
$S \to VP$	.2
$NP \to D\ N$	.4
$NP \to NP\ PP$	.4
$NP \to PN$	.2
$VP \to V\ NP$	.7
$VP \to VP\ PP$	.3
$PP \to P\ NP$	1

$D \to the$	.8
$D \to a$	.2
$N \to flight$	1
$PN \to john$	.9
$PN \to schiphol$	.1
$V \to booked$	1
$P \to from$	1

John booked a flight from Schiphol



$P(t_1) = 6.4512 \times 10^{-5}$



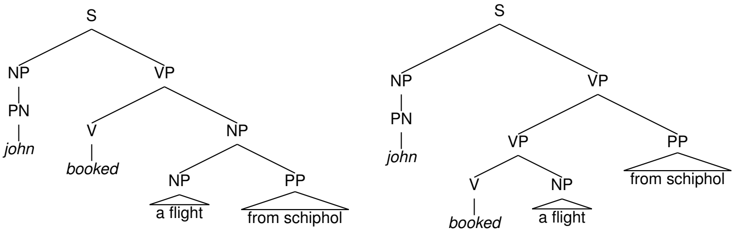
$P(t_2) = 4.8384 \times 10^{-5}$

Disambiguation with PCFGs

$S \rightarrow NP\ VP$	.8
$S \rightarrow VP$	.2
$NP \rightarrow D\ N$	.4
$NP \rightarrow NP\ PP$	.4
$NP \rightarrow PN$	.2
$VP \rightarrow V\ NP$	.7
$VP \rightarrow VP\ PP$	.3
$PP \rightarrow P\ NP$	1

$D \rightarrow the$	.8
$D \rightarrow a$	.2
$N \rightarrow flight$	1
$PN \rightarrow john$	.9
$PN \rightarrow schiphol$	.1
$V \rightarrow booked$	1
$P \rightarrow from$	1

John booked a flight from Schiphol



$P(t_1) = 6.4512 \times 10^{-5}$

$P(t_2) = 4.8384 \times 10^{-5}$

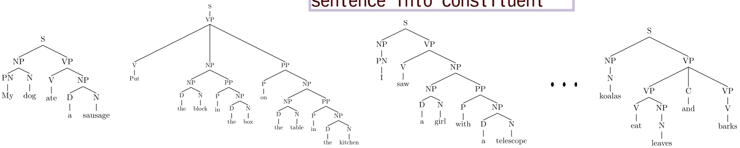
Treebank PCFGs

- Treebanks: instead of paying linguists to write a grammar, pay them to annotate real sentences with parse trees.
- This way, we implicitly get a grammar
(for CFG: read the rules off the trees)
- And we get probabilities for those rules
- We can use these probabilities to improve disambiguation
- and also speed up parsing.

Estimating rule probabilities from a treebank

组分
because we use this
algorithm to break a
sentence into constituent

A treebank: a collection of s ith constituent trees



An estimated probability of a rule (**maximum likelihood** estimates):

$$p(X \to \alpha) = \frac{C(X \to \alpha)}{C(X)}$$

The number of times the rule used in the corpus

how often this rule
occurs in the corpus

The number of times the nonterminal X appears in the treebank

Dependency structure

tells how the each individual words are connected to each other

A dependency structure consists of dependency relations, which are binary and asymmetric.

John hit the ball

A relation consists of

- a head (H) — hit
- a dependent (D) — John
- a label identifying the relation between H and D — Subject

Dependency structure

A dependency structure consists of dependency relations, which are binary and asymmetric.

John hit the ball

A relation consists of

谓语

- a head (H) — hit

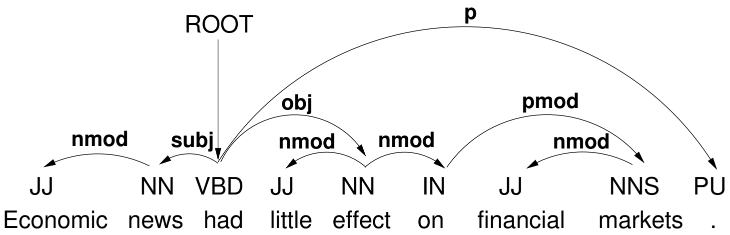
宾语

- a dependent (D) — ball

宾语

- a label identifying the relation between H and D — Object

Example dependency structure



[From Joakim Nivre, Dependency Grammar and Dependency Parsing.]

Dependency parsing

it tells use: who did what to whom

Output a list of dependencies between words in the sentence.

主语 谓语 宾语

John hit the ball.

```
(SUBJ head=hit dep=John)
(OBJ head=hit dep=ball)
(DET head=ball dep=the)
```

Why is it useful?

- dependencies provide an interface to semantics

  "Who did what to whom"

The cost of parsing errors...

Incorrect dependencies

```
(SUBJ head=hit dep=ball)
(OBJ head=hit dep=John)
(DET head=ball dep=the)
```

Outline.

Syntactic parsing

Introduction to semantics & lexical semantics

Semantics

Compositional semantics:

- studies how meanings of phrases are constructed out of the meaning of individual words
- principle of compositionality: meaning of each whole phrase derivable from meaning of its parts
- sentence structure conveys some meaning: obtained by syntactic representation

Lexical semantics:

- studies how the meanings of individual words can be represented and induced

What is lexical meaning?

- recent results in psychology and cognitive neuroscience give us some clues
- but we don't have the whole picture yet
- different representations proposed, e.g.
  - formal semantic representations based on logic,
  - *or* taxonomies relating words to each other,
  - *or* distributional representations in statistical NLP
- but none of the representations gives us a complete account of lexical meaning

How to approach lexical meaning?

- Formal semantics: set-theoretic approach
  e.g., cat′: the set of all cats; bird′: the set of all birds.
- meaning postulates, e.g.

$$\forall x[\text{bachelor}'(x) \rightarrow \text{man}'(x) \land \text{unmarried}'(x)]$$

- Limitations, e.g. *is the current Pope a bachelor?*
- Defining concepts through enumeration of all of their features in practice is highly problematic
- How would you define e.g. *chair, tomato, thought, democracy*? – impossible for most concepts
- Prototype theory offers an alternative to set-theoretic approaches

How to approach lexical meaning?

- Formal semantics: set-theoretic approach
  e.g., cat′: the set of all cats; bird′: the set of all birds.
- meaning postulates, e.g.

$$\forall x[\text{bachelor}'(x) \rightarrow \text{man}'(x) \wedge \text{unmarried}'(x)]$$

- Limitations, e.g. *is the current Pope a bachelor?*
- Defining concepts through enumeration of all of their features in practice is highly problematic
- How would you define e.g. *chair, tomato, thought, democracy*? – impossible for most concepts
- Prototype theory offers an alternative to set-theoretic approaches

Prototype theory

- introduced the notion of graded semantic categories
- no clear boundaries
- no requirement that a property or set of properties be shared by all members
- certain members of a category are more central or prototypical (i.e. instantiate the prototype)

furniture: chair is more prototypical than stool

Eleanor Rosch 1975. *Cognitive Representation of Semantic Categories* (J Experimental Psychology)

Prototype theory (continued)

- Categories form around prototypes; new members added on basis of resemblance to prototype
- Features/attributes generally graded
- Category membership a matter of degree
- Categories do not have clear boundaries

Semantic relations

Hyponymy: IS-A

dog is a hyponym of *animal*
animal is a hypernym of *dog*

- hyponymy relationships form a taxonomy
- works best for concrete nouns
- multiple inheritance: e.g., is *coin* a hyponym of both *metal* and *money*?

Other semantic relations

Meronymy: PART-OF e.g., *arm* is a meronym of *body*, *steering wheel* is a meronym of *car* (piece vs part)

Synonymy e.g., *aubergine/eggplant*.

Antonymy e.g., *big/little*

Also:

Near-synonymy/similarity e.g., *exciting/thrilling*
e.g., *slim/slender/thin/skinny*

WordNet

- large scale, open source resource for English
- hand-constructed
- wordnets being built for other languages
- organized into synsets: synonym sets (near-synonyms)
- synsets connected by semantic relations

```
S: (v) interpret, construe, see (make sense of;
 assign a meaning to) - "How do you interpret his
  behavior?"
S: (v) understand, read, interpret, translate (make
 sense of a language) "She understands French";
 "Can you read Greek?"
```

Polysemy and word senses

The children **ran** to the store
If you see this man, **run**!
Service **runs** all the way to Cranbury
She is **running** a relief operation in Sudan
the story or argument **runs** as follows
Does this old car still **run** well?
Interest rates **run** from 5 to 10 percent
Who's **running** for treasurer this year?
They **ran** the tapes over and over again
These dresses **run** small

Polysemy

- homonymy: unrelated word senses. *bank* (raised land) vs *bank* (financial institution)
- *bank* (financial institution) vs *bank* (in a casino): related but distinct senses.
- regular polysemy and sense extension
  - metaphorical senses, e.g. *swallow* [food], *swallow* [information], *swallow* [anger]
  - metonymy, e.g. he played *Bach*; he drank his *glass*.
  - zero-derivation, e.g. *tango* (N) vs *tango* (V)
- vagueness: *nurse, lecturer, driver*
- cultural stereotypes: *nurse, lecturer, driver*

No clearcut distinctions.

Word sense disambiguation

- Needed for many applications
- relies on context, e.g. *striped bass* (the fish) vs *bass guitar*.

Methods:

- supervised learning:
  - Assume a predefined set of word senses, e.g. WordNet
  - Need a large sense-tagged training corpus (difficult to construct)
- semi-supervised learning (Yarowsky, 1995)
  - bootstrap from a few examples
- unsupervised sense induction
  - e.g. cluster contexts in which a word occurs

WSD by semi-supervised learning

Yarowsky, David (1995) *Unsupervised word sense disambiguation rivalling supervised methods*

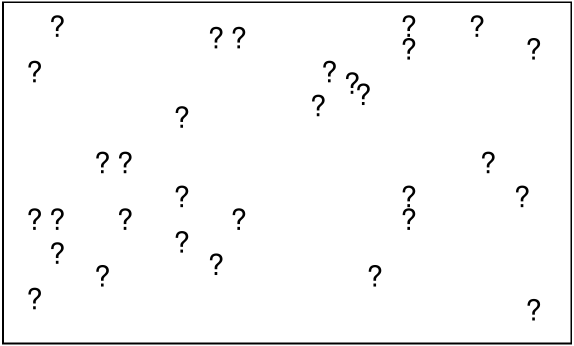
Disambiguating *plant* (factory vs vegetation senses):

1. Find contexts in training corpus:

sense	training example
?	company said that the *plant* is still operating
?	although thousands of *plant* and animal species
?	zonal distribution of *plant* life
?	company manufacturing *plant* is in Orlando
	etc

Yarowsky (1995): schematically

Initial state

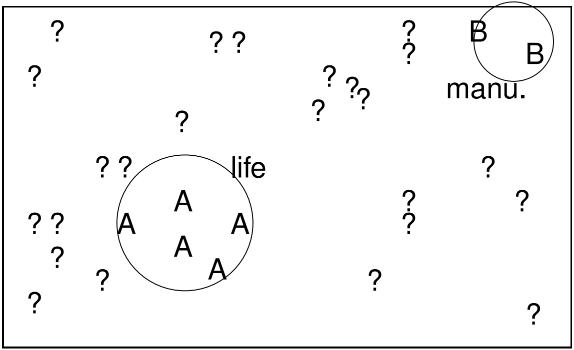


2. Identify some seeds to disambiguate a few uses:

‘*plant* life’ for vegetation use (A)
‘manufacturing *plant*’ for factory use (B)

sense	training example
?	company said that the *plant* is still operating
?	although thousands of *plant* and animal species
A	zonal distribution of *plant* life
B	company manufacturing *plant* is in Orlando
	etc

Seeds



3. Train a decision list classifier on Sense A/Sense B examples.

Rank features by log-likelihood ratio:

$$\log\left(\frac{P(\text{Sense}_A|f_i)}{P(\text{Sense}_B|f_i)}\right)$$

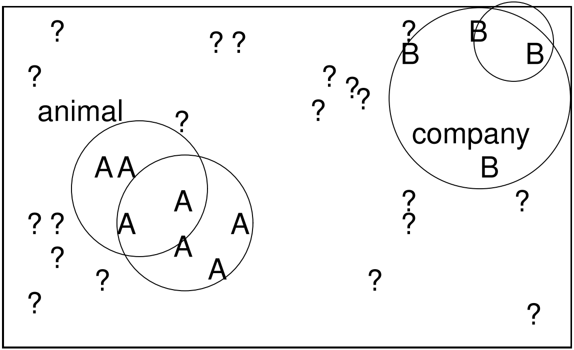
reliability	criterion	sense
8.10	*plant* life	A
7.58	manufacturing *plant*	B
6.27	*animal* within 10 words of *plant*	A
	etc	

4. Apply the classifier to the training set and add reliable examples to A and B sets.

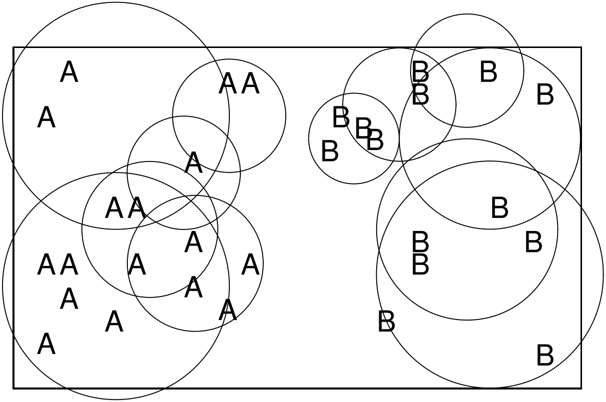
sense	training example
?	company said that the *plant* is still operating
A	although thousands of *plant* and animal species
A	zonal distribution of *plant* life
B	company manufacturing *plant* is in Orlando
	etc

5. Iterate the previous steps 3 and 4 until convergence

Iterating:



Final:



6. Apply the classifier to the unseen test data

- Accuracy of 95%, but...
- Yarowsky's experiments were nearly all on homonyms: these principles may not hold as well for sense extension.

Problems with WSD as supervised classification

- real performance around 75% (supervised)
- need to predefine word senses (not theoretically sound)
- need a very large training corpus (difficult to annotate, humans do not agree)
- learn a model for individual words — no real generalisation

Better way:

- unsupervised sense induction (but a very hard task)

Acknowledgement

Some slides were adapted from Ann Copestake and Tejaswini Deoskar