Natural Language Processing

Ann Copestake

Computer Laboratory
University of Cambridge

October 2007
Outline of today’s lecture

Lecture 1: Introduction
Overview of the course
Why NLP is hard.
Scope of NLP.
A sample application: sentiment classification
More NLP applications
NLP components.
NLP and linguistics

NLP: the automatic processing of human language.

1. Morphology — the structure of words: lecture 2.
2. Syntax — the way words are used to form phrases: lectures 3, 4 and 5.
3. Semantics
   - Compositional semantics — the construction of meaning based on syntax: lecture 6.
   - Lexical semantics — the meaning of individual words: lecture 6.
Also note:

- Exercises: pre-lecture and post-lecture
- Glossary
- **Recommended Book:** Jurafsky and Martin (2000) (and second edition draft)
Querying a knowledge base

User query:
- Has my order number 4291 been shipped yet?

Database:

ORDER
Order number Date ordered Date shipped
4290 2/2/02 2/2/02
4291 2/2/02 2/2/02
4292 2/2/02

USER: Has my order number 4291 been shipped yet?
DB QUERY: order(number=4291,date_shipped=?)
RESPONSE: Order number 4291 was shipped on 2/2/02
Why is this difficult?

Similar strings mean different things, different strings mean the same thing:

1. How fast is the 505G?
2. How fast will my 505G arrive?
3. Please tell me when I can expect the 505G I ordered.

Ambiguity:

- Do you sell Sony laptops and disk drives?
- Do you sell (Sony (laptops and disk drives))? 
- Do you sell (Sony laptops) and disk drives?
Why is this difficult?

Similar strings mean different things, different strings mean the same thing:

1. How fast is the 505G?
2. How fast will my 505G arrive?
3. Please tell me when I can expect the 505G I ordered.

Ambiguity:

- Do you sell Sony laptops and disk drives?
- Do you sell (Sony (laptops and disk drives))? 
- Do you sell (Sony laptops) and disk drives?
Why is this difficult?

Similar strings mean different things, different strings mean the same thing:

1. How fast is the 505G?
2. How fast will my 505G arrive?
3. Please tell me when I can expect the 505G I ordered.

Ambiguity:

- Do you sell Sony laptops and disk drives?
- Do you sell (Sony (laptops and disk drives))? 
- Do you sell (Sony laptops) and disk drives?
Why is this difficult?

Similar strings mean different things, different strings mean the same thing:

1. How fast is the 505G?
2. How fast will my 505G arrive?
3. Please tell me when I can expect the 505G I ordered.

Ambiguity:

- Do you sell Sony laptops and disk drives?
- Do you sell (Sony (laptops and disk drives))? 
- Do you sell (Sony laptops) and disk drives?
Why is this difficult?

Similar strings mean different things, different strings mean the same thing:

1. How fast is the 505G?
2. How fast will my 505G arrive?
3. Please tell me when I can expect the 505G I ordered.

Ambiguity:

- Do you sell Sony laptops and disk drives?
- Do you sell (Sony (laptops and disk drives))?
- Do you sell (Sony laptops) and disk drives?
Why is this difficult?

Similar strings mean different things, different strings mean the same thing:

1. How fast is the 505G?
2. How fast will my 505G arrive?
3. Please tell me when I can expect the 505G I ordered.

Ambiguity:

- Do you sell Sony laptops and disk drives?
- Do you sell (Sony (laptops and disk drives))?
- Do you sell (Sony laptops) and disk drives?
Some NLP applications

- spelling and grammar checking
- optical character recognition (OCR)
- screen readers
- augmentative and alternative communication
- machine aided translation
- lexicographers’ tools
- information retrieval
- document classification
- document clustering
- information extraction
- question answering
- summarization
- text segmentation
- exam marking
- report generation
- machine translation
- natural language interfaces to databases
- email understanding
- dialogue systems
Sentiment classification: finding out what people think about you.

- Task: scan documents for positive and negative opinions on people, products etc.
- Find all references to entity in some document collection: list as positive, negative (possibly with strength) or neutral.
- Summaries plus text snippets.
- Fine-grained classification: e.g., for phone, opinions about: overall design, keypad, camera.
- Still often done by humans . . .
Motorola KRZR (from the Guardian)

Motorola has struggled to come up with a worthy successor to the RAZR, arguably the most influential phone of the past few years. Its latest attempt is the KRZR, which has the same clamshell design but has some additional features. It has a striking blue finish on the front and the back of the handset is very tactile brushed rubber. Like its predecessors, the KRZR has a laser-etched keypad, but in this instance Motorola has included ridges to make it easier to use. . . . Overall there’s not much to dislike about the phone, but its slightly quirky design means that it probably won’t be as huge or as hot as the RAZR.
Sentiment classification: the research task.

- Full task: information retrieval, cleaning up text structure, named entity recognition, identification of relevant parts of text. Evaluation by humans.
- Research task: preclassified documents, topic known, opinion in text along with some straightforwardly extractable score.
- Movie review corpus, with ratings.

Rating: 9/10

Ooooo. Scary.
The old adage of the simplest ideas being the best is once again demonstrated in this, one of the most entertaining films of the early 80’s, and almost certainly Jon Landis’ best work to date. The script is light and witty, the visuals are great and the atmosphere is top class. Plus there are some great freeze-frame moments to enjoy again and again. Not forgetting, of course, the great transformation scene which still impresses to this day.

In Summary: Top banana
Bag of words technique.

- Treat the reviews as collections of individual words.
- Classify reviews according to positive or negative words.
- Could use word lists prepared by humans, but machine learning based on a portion of the corpus (training set) is preferable.
- Use star rankings for training and evaluation.
- Pang et al, 2002: Chance success is 50% (movie database was artificially balanced), bag-of-words gives 80%.
Some sources of errors.

- **Negation:**

  *Ridley Scott has never directed a bad film.*

- **Overfitting the training data:**
  e.g., if training set includes a lot of films from before 2005, *Ridley* may be a strong positive indicator, but then we test on reviews for ‘Kingdom of Heaven’?

- **Comparisons and contrasts.**
Contrasts in the discourse

This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can’t hold up.
More contrasts

AN AMERICAN WEREWOLF IN PARIS is a failed attempt . . . Julie Delpy is far too good for this movie. She imbues Serafine with spirit, spunk, and humanity. This isn’t necessarily a good thing, since it prevents us from relaxing and enjoying AN AMERICAN WEREWOLF IN PARIS as a completely mindless, campy entertainment experience. Delpy’s injection of class into an otherwise classless production raises the specter of what this film could have been with a better script and a better cast . . . She was radiant, charismatic, and effective . . .
Sample data

http://www.cl.cam.ac.uk/ aac10/sentiment/
(linked from http://www.cl.cam.ac.uk/ aac10/stuff.html)
See test data texts in:
http://www.cl.cam.ac.uk/ aac10/sentiment/test/
classified into positive/negative.
Doing sentiment classification properly?

- Morphology, syntax and compositional semantics: who is talking about what, what terms are associated with what, tense . . .
- Lexical semantics: are words positive or negative in this context? Word senses (e.g., spirit)?
- Pragmatics and discourse structure: what is the topic of this section of text? Pronouns and definite references.
- But getting all this to work on arbitrary text is very hard.
- Ultimately the problem is AI-complete, but can we do well enough for NLP to be useful?
IR, IE and QA

- Information retrieval: return documents in response to a user query (Internet Search is a special case)
- Information extraction: discover specific information from a set of documents (e.g. company joint ventures)
- Question answering: answer a specific user question by returning a section of a document:

> What is the capital of France?
> Paris has been the French capital for many centuries.

Much more about these in Simone Teufel’s course.
MT

- Earliest attempted NLP application
- Quality depends on restricting the domain
- Utility greatly increased with increase in availability of electronic text
- Good applications for bad MT . . .
- Spoken language translation is viable for limited domains
Natural language interfaces and dialogue systems

All rely on a limited domain:

- SHRDLU: (text-based) dialogue system: 1973
- Current spoken dialogue systems: e.g., BA flight information

Limited domain allows disambiguation: e.g., in LUNAR, rock had one sense.
Generic NLP modules

- input preprocessing: speech recogniser, text preprocessor or gesture recogniser.
- morphological analysis
- part of speech tagging
- parsing: this includes syntax and compositional semantics
- disambiguation
- context module
- text planning
- tactical generation
- morphological generation
- output processing: text-to-speech, text formatter, etc.
Natural language interface to a knowledge base

- KB INTERFACE
  - PARSING
    - MORPHOLOGY
      - INPUT PROCESSING
        - user input
    - TACTICAL GENERATION
      - MORPHOLOGY GENERATION
      - OUTPUT PROCESSING
        - output
General comments

- Even ‘simple’ applications might need complex knowledge sources
- Applications cannot be 100% perfect
- Applications that are < 100% perfect can be useful
- Aids to humans are easier than replacements for humans
- NLP interfaces compete with non-language approaches
- Shallow processing on arbitrary input or deep processing on narrow domains
- Limited domain systems require extensive and expensive expertise to port
- External influences on NLP are very important
Outline of the next lecture

Lecture 2: Morphology and finite state techniques.
   A brief introduction to morphology.
   Using morphology.
   Spelling rules.
   Finite state techniques.
   More applications for finite state techniques.
Outline of today’s lecture

Lecture 2: Morphology and finite state techniques.
- A brief introduction to morphology.
- Using morphology.
- Spelling rules.
- Finite state techniques.
- More applications for finite state techniques.
Some terminology.

- **morpheme**: the minimal information carrying unit
- **affix**: morpheme which only occurs in conjunction with other morphemes
- words are made up of a **stem** (more than one in the case of compounds) and zero or more affixes. e.g., *dog* plus plural suffix *+s*
- **affixes**: prefixes, suffixes, infixes and circumfixes
- in English: prefixes and suffixes (prefixes only for derivational morphology)
- **productivity**: whether affix applies generally, whether it applies to new words
Inflectional morphology

- e.g., plural suffix +s, past participle +ed
- sets slots in some paradigm
- e.g., tense, aspect, number, person, gender, case
- inflectional affixes are not combined in English
- generally fully productive (modulo irregular forms)
Derivational morphology

- e.g., *un-*, *re-*, *anti-*, *-ism*, *-ist* etc
- broad range of semantic possibilities, may change part of speech
- indefinite combinations
  e.g., *anti-anti-dis-establishmentarianism*
  *anti-anti-dis-establish-ment-arian-ism*
- generally semi-productive
- zero-derivation (e.g. *tango*, *waltz*)
Internal structure and ambiguity

Morpheme ambiguity: stems and affixes may be individually ambiguous: e.g. *dog* (noun or verb), *+s* (plural or 3persg-verb)

Structural ambiguity: e.g., *shorts/short -s*

*unionised* could be *union -ise -ed* or *un- ion -ise -ed*

Bracketing:

- *un- ion* is not a possible form
- *un-* is ambiguous:
  - with verbs: means ‘reversal’ (e.g., *untie*)
  - with adjectives: means ‘not’ (e.g., *unwise*)
- internal structure of *un- ion -ise -ed* has to be (*un- ((ion -ise) -ed*))

Temporarily skip 2.3
Internal structure and ambiguity

Morpheme ambiguity: stems and affixes may be individually ambiguous: e.g. dog (noun or verb), +s (plural or 3persg-verb)

Structural ambiguity: e.g., shorts/short -s

unionised could be union -ise -ed or un- ion -ise -ed

Bracketing:
- un- ion is not a possible form
- un- is ambiguous:
  - with verbs: means ‘reversal’ (e.g., untie)
  - with adjectives: means ‘not’ (e.g., unwise)
- internal structure of un- ion -ise -ed
  has to be (un- ((ion -ise) -ed))

Temporarily skip 2.3
Applications of morphological processing

- compiling a **full-form** lexicon
- **stemming** for IR (not linguistic stem)
- **lemmatization** (often inflections only): finding stems and affixes as a precursor to parsing
  NB: may use parsing to filter results (see lecture 5)
  e.g., *feed* analysed as *fee-ed* (as well as *feed*)
  but parser blocks (assuming lexicon does not have *fee* as a verb)
- **generation**
  Morphological processing may be **bidirectional**: i.e., parsing and generation.
  
  `sleep + PAST_VERB <--> slept`
Morphology in a deep processing system (cf lec 1)
Lexical requirements for morphological processing

- affixes, plus the associated information conveyed by the affix
  
  ed  PAST_VERB
  ed  PSP_VERB
  s  PLURAL_NOUN

- irregular forms, with associated information similar to that for affixes
  began  PAST_VERB begin
  begun  PSP_VERB begin

- stems with syntactic categories (plus more)
A zookeeper was ordering extra animals for his zoo. He started the letter:

"Dear Sir, I need two mongeese."

This didn’t sound right, so he tried again:

"Dear Sir, I need two mongooses."

But this sounded terrible too. Finally, he ended up with:

"Dear Sir, I need a mongoose, and while you’re at it, send me another one as well."
Mongoose

A zookeeper was ordering extra animals for his zoo. He started the letter:

“Dear Sir, I need two mongeese.”

This didn’t sound right, so he tried again:

“Dear Sir, I need two mongooses.”

But this sounded terrible too. Finally, he ended up with:

“Dear Sir, I need a mongoose, and while you’re at it, send me another one as well.”
A zookeeper was ordering extra animals for his zoo. He started the letter:

“Dear Sir, I need two mongeese.”

This didn’t sound right, so he tried again:

“Dear Sir, I need two mongooses.”

But this sounded terrible too. Finally, he ended up with:

“Dear Sir, I need a mongoose, and while you’re at it, send me another one as well.”
Spelling rules (sec 2.3)

- English morphology is essentially concatenative
- irregular morphology — inflectional forms have to be listed
- regular phonological and spelling changes associated with affixation, e.g.
  - -s is pronounced differently with stem ending in s, x or z
  - spelling reflects this with the addition of an e (boxes etc)
- in English, description is independent of particular stems/affixes
e-insertion

e.g. box\(^s\) to boxes

\[
\varepsilon \rightarrow e/ \left\{ \begin{array}{c}
  s \\
  x \\
  z
\end{array} \right\} ^\_ s
\]

- map ‘underlying’ form to surface form
- mapping is left of the slash, context to the right
- notation:
  \_ \quad \text{position of mapping}
  \varepsilon \quad \text{empty string}
  ^\_ \quad \text{affix boundary} \quad \text{stem} \quad ^\_ \quad \text{affix}

- same rule for plural and 3sg verb
- formalisable/implementable as a finite state transducer
e-insertion

e.g. box's to boxes

\[ \varepsilon \rightarrow e/ \left\{ \begin{array}{c} s \\ x \\ z \end{array} \right\} \hat{n}_{s} \]

- map ‘underlying’ form to surface form
- mapping is left of the slash, context to the right
- notation:
  - _ position of mapping
  - \varepsilon empty string
  - ^ affix boundary — stem ^ affix

- same rule for plural and 3sg verb
- formalisable/implementable as a finite state transducer
e-insertion

e.g. \( box^s \) to \( boxes \)

\[
\varepsilon \rightarrow e/ \left\{ \begin{array}{c} s \\ x \\ z \end{array} \right\} ^s \\
\]

- map ‘underlying’ form to surface form
- mapping is left of the slash, context to the right
- notation:
  - \( _\_ \) position of mapping
  - \( \varepsilon \) empty string
  - \( ^\_ \) affix boundary — stem ^ affix

- same rule for plural and 3sg verb
- formalisable/implementable as a finite state transducer
Finite state automata for recognition

day/month pairs:

0,1,2,3 digit / 0,1 0,1,2

1 — 2 — 3

4 — 5 — 6

— non-deterministic — after input of ‘2’, in state 2 and state 3.
— double circle indicates accept state
— accepts e.g., 11/3 and 3/12
— also accepts 37/00 — overgeneration
Recursive FSA

comma-separated list of day/month pairs:

0,1,2,3 digit / 0,1 0,1,2

digit
digit,

- list of indefinite length
- e.g., 11/3, 5/6, 12/04
Finite state transducer

1. $e : e$
2. other : other
3. $\varepsilon : \wedge$
4. $s : s$

$s : s$
$x : x$
$z : z$

$\varepsilon \rightarrow e/ \{ s, x \} \wedge s$

surface : underlying

cakes ↔ cake^s
boxes ↔ box^s
Analysing $b o x e s$

Input: b
Output: b
(Plus: $\epsilon \cdot ^\wedge$)
Analysing $b o x e s$

Input: $b$
Output: $b$
(Plus: $\epsilon \cdot \wedge$)
Analysing $b o x e s$

Input: $b o$
Output: $b o$
Analysing $b\ o\ x\ e\ s$

Input: $b\ o\ x$
Output: $b\ o\ x$
Lecture 2: Morphology and finite state techniques.

Finite state techniques.

Analysing **b o x e s**

Input: b o x e
Output: b o x e

Input: b o x e
Output: b o x e

Input: b o x e
Output: b o x e
Analysing $b\ o\ x\ e\ \epsilon\ s$
Analysing *b o x e s*

**Input:** b o x e s
**Output:** b o x ^ s

**Input:** b o x e
**Output:** b o x e ^ s

**Input:** b o x e e s
**Output:** b o x e ^ s
Analyzing `box`es

Input: `box`es
Accept output: `box`es

Input: `box`e
Accept output: `box`e

Input: `box`s
Accept output: `box`s

Input: `box`es
Accept output: `box`es
Using FSTs

- FSTs assume **tokenization** (word boundaries) and words split into characters. One character pair per transition!
- Analysis: return character list with affix boundaries, so enabling lexical lookup.
- Generation: input comes from stem and affix lexicons.
- One FST per spelling rule: either compile to big FST or run in parallel.
- FSTs do not allow for internal structure:
  - can’t model *un- ion -ize -d* bracketing.
  - can’t condition on prior transitions, so potential redundancy (cf 2006/7 exam q)
Some other uses of finite state techniques in NLP

- Grammars for simple spoken dialogue systems (directly written or compiled)
- Partial grammars for named entity recognition
- Dialogue models for spoken dialogue systems (SDS)
  e.g. obtaining a date:
  1. No information. System prompts for month and day.
  2. Month only is known. System prompts for day.
  3. Day only is known. System prompts for month.
  4. Month and day known.
Example FSA for dialogue
Example of probabilistic FSA for dialogue
Next lecture

Lecture 3: Prediction and part-of-speech tagging.
Corpora in NLP
Word prediction
Part-of-speech (POS) tagging
Evaluation in general, evaluation of POS tagging
Outline of today’s lecture

Lecture 3: Prediction and part-of-speech tagging.

Corpora in NLP
Word prediction
Part-of-speech (POS) tagging
Evaluation in general, evaluation of POS tagging

First of three lectures that concern syntax (i.e., how words fit together). This lecture: ‘shallow’ syntax: word sequences and POS tags. Next lectures: more detailed syntactic structures.
Corpora

Changes in NLP research over the last 15-20 years are largely due to increased availability of electronic corpora.

- **corpus**: text that has been collected for some purpose.
- **balanced corpus**: texts representing different genres
  - **genre**: is a type of text (vs domain)
- **tagged corpus**: a corpus annotated with POS tags
- **treebank**: a corpus annotated with parse trees
- **specialist corpora** — e.g., collected to train or evaluate particular applications
  - Movie reviews for sentiment classification
  - Data collected from simulation of a dialogue system
Prediction

Guess the missing words:

Illustrations produced by any package can be transferred with consummate _____ to another.

Wright tells her story with great _____.

consummate 

Wright tells her story with great _____.
Prediction

Guess the missing words:

Illustrations produced by any package can be transferred with consummate ___ease___ to another.

Wright tells her story with great ____.
Prediction

Guess the missing words:

Illustrations produced by any package can be transferred with consummate ____ease____ to another.

Wright tells her story with great ____professionalism____.
Prediction

Prediction is relevant for:

- language modelling for speech recognition: e.g., using N-grams. This is an alternative to finite state grammars, suitable for large-scale recognition
- word prediction for communication aids (augmentative and alternative communication). e.g., to help enter text that’s input to a synthesiser
- text entry on mobile phones and similar devices
- OCR, spelling correction, text segmentation
- estimation of entropy
bigrams (N-gram with N=2)

A probability is assigned to a word based on the previous word:

$$P(w_n|w_{n-1})$$

where $w_n$ is the $n$th word in a sentence.

Probability of a sequence of words (assuming independence):

$$P(W_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

Probability is estimated from counts in a training corpus:

$$\frac{C(w_{n-1} w_n)}{\sum_w C(w_{n-1} w)} \approx \frac{C(w_{n-1} w_n)}{C(w_{n-1})}$$

i.e. count of a particular bigram in the corpus divided by the count of all bigrams starting with the prior word.
Calculating bigrams

\[
\langle s \rangle \text{ good morning } \langle s \rangle \text{ good afternoon } \langle s \rangle \text{ good afternoon } \langle s \rangle \text{ it is very good } \langle s \rangle \text{ it is good } \langle s \rangle
\]

<table>
<thead>
<tr>
<th>sequence</th>
<th>count</th>
<th>bigram probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>\langle s \rangle</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>\langle s \rangle good</td>
<td>3</td>
<td>.6</td>
</tr>
<tr>
<td>\langle s \rangle it</td>
<td>2</td>
<td>.4</td>
</tr>
<tr>
<td>good</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>good morning</td>
<td>1</td>
<td>.2</td>
</tr>
<tr>
<td>good afternoon</td>
<td>2</td>
<td>.4</td>
</tr>
<tr>
<td>good \langle s \rangle</td>
<td>2</td>
<td>.4</td>
</tr>
<tr>
<td>morning</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>morning \langle s \rangle</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Practical application

- word prediction: guess the word from initial letters (user confirms each word)
- speech recognition: maximize likelihood of a sequence (implemented using the Viterbi algorithm)

Problems because of sparse data:
- smoothing: distribute ‘extra’ probability between rare and unseen events
- backoff: approximate unseen probabilities by a more general probability, e.g. unigrams
Part of speech tagging

They can fish.

They_PNP can_VM0 fish_VVI ._PUN
They_PNP can_VVB fish_NN2 ._PUN
They_PNP can_VM0 fish_NN2 ._PUN no full parse

POS lexicon fragment:

they PNP
can VM0 VVB VVI NN1
fish NN1 NN2 VVB VVI

tagset (CLAWS 5) includes:

<table>
<thead>
<tr>
<th>NN1</th>
<th>singular noun</th>
<th>NN2</th>
<th>plural noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNP</td>
<td>personal pronoun</td>
<td>VM0</td>
<td>modal auxiliary verb</td>
</tr>
<tr>
<td>VVB</td>
<td>base form of verb</td>
<td>VVI</td>
<td>infinitive form of verb</td>
</tr>
</tbody>
</table>
Part of speech tagging

- They_PNP can_VM0 fish_VVI _PUN
- They_PNP can_VVB fish_NN2 _PUN
- They_PNP can_VM0 fish_NN2 _PUN no full parse

POS lexicon fragment:

<table>
<thead>
<tr>
<th>Word</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>they</td>
<td>PNP</td>
</tr>
<tr>
<td>can</td>
<td>VM0 VVB VVI NN1</td>
</tr>
<tr>
<td>fish</td>
<td>NN1 NN2 VVB VVI</td>
</tr>
</tbody>
</table>

**tagset (CLAWS 5) includes:**

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN1</td>
<td>singular noun</td>
</tr>
<tr>
<td>NN2</td>
<td>plural noun</td>
</tr>
<tr>
<td>PNP</td>
<td>personal pronoun</td>
</tr>
<tr>
<td>VM0</td>
<td>modal auxiliary verb</td>
</tr>
<tr>
<td>VVB</td>
<td>base form of verb</td>
</tr>
<tr>
<td>VVI</td>
<td>infinitive form of verb</td>
</tr>
</tbody>
</table>
Part of speech tagging

- They_PNP can_VM0 fish_VVI ._PUN
- They_PNP can_VVB fish_NN2 ._PUN
- They_PNP can_VM0 fish_NN2 ._PUN no full parse

POS lexicon fragment:
- they   PNP
- can    VM0 VVB VVI NN1
- fish   NN1 NN2 VVB VVI

**tagset** (CLAWS 5) includes:
- NN1    singular noun    |    NN2    plural noun
- PNP    personal pronoun |    VM0    modal auxiliary verb
- VVB    base form of verb |    VVI    infinitive form of verb
Why POS tag?

- Coarse-grained syntax / word sense disambiguation: most approaches are quick to run, so applicable to very large corpora.
- Potentially useful for some linguistic research and for lexicography: e.g., how often is *tango* used as a verb? *dog*?
- Basis for named entity recognition and similar tasks (finite state patterns over POS tagged data).
- Features for machine learning e.g., sentiment classification. (e.g., *stink_V vs stink_N*)
- Preliminary processing for full parsing: cut down search space or provide guesses at unknown words.
Stochastic part of speech tagging

1. Start with untagged text.
2. Assign all possible tags to each word in the text on the basis of a lexicon that associates words and tags.
3. Find the most probable sequence (or n-best sequences) of tags, based on probabilities from the training data.
   - lexical probability: e.g., is can most likely to be VM0, VVB, VVI or NN1?
   - and tag sequence probabilities: e.g., is VM0 or NN1 more likely after PNP?

Note: tags are more fine-grained than conventional part of speech. Different possible tagsets: i.e., sets of POS tags.
Training stochastic POS tagging

They_PNP used_VVD to_TO0 can_VVI fish_NN2 in_PRP those_DT0 towns_NN2 ._PUN But_CJC now_AV0 few_DT0 people_NN2 fish_VVB in_PRP these_DT0 areas_NN2 ._PUN

<table>
<thead>
<tr>
<th>sequence</th>
<th>count</th>
<th>bigram probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>NN2 PRP</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>NN2 PUN</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>NN2 VVB</td>
<td>1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Also lexicon: fish NN2 VVB
Training stochastic POS tagging

They_PNP used_VVD to_TO0 can_VVI fish_NN2 in_PRP those_DT0 towns_NN2 ._PUN But_CJC now_AV0 few_DT0 people_NN2 fish_VVB in_PRP these_DT0 areas_NN2 ._PUN

<table>
<thead>
<tr>
<th>sequence</th>
<th>count</th>
<th>bigram probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>NN2 PRP</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>NN2 PUN</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>NN2 VVB</td>
<td>1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Also lexicon: fish NN2 VVB
Training stochastic POS tagging

They\_PNP used\_VVD to\_TO0 can\_VVI fish\_NN2 in\_PRP those\_DT0 towns\_NN2 \_PUN But\_CJC now\_AV0 few\_DT0 people\_NN2 fish\_VVB in\_PRP these\_DT0 areas\_NN2 \_PUN

<table>
<thead>
<tr>
<th>sequence</th>
<th>count</th>
<th>bigram probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN2</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>NN2 PRP</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>NN2 PUN</td>
<td>2</td>
<td>0.5</td>
</tr>
<tr>
<td>NN2 VVB</td>
<td>1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Also lexicon: fish NN2 VVB
Assigning probabilities

More complex than word prediction, because looking at words and tags. $P(T|W)$ is prob of tag sequence T, given word sequence W, but can’t estimate this directly. So, applying Bayes theorem:

$$P(T|W) = \frac{P(T)P(W|T)}{P(W)}$$

$P(W)$ is constant (tagging a known sequence) estimate $P(T)$ as $P(t_i|t_{i-1})$ (bigram assumption) estimate $P(W|T)$ as $P(w_i|t_i)$ (i.e., the probability of each word given its tag)
Example

Tagging: *they fish*
Assume $P(\text{PNP}|\text{they}) = 1$.
Then sequence probability depends on:

$P(\text{NN1}|\text{PNP})P(\text{fish}|\text{NN1})$
$P(\text{NN2}|\text{PNP})P(\text{fish}|\text{NN2})$
$P(\text{VVB}|\text{PNP})P(\text{fish}|\text{VVB})$

etc
Assigning probabilities, more details

- Maximise the overall tag sequence probability — e.g., use Viterbi.
  
  they_PNP can_VVB fish_NN2
  they_PNP can_VM0 fish_VVI
  \[ P(VVI|VM0)P(fish|VVI) \]
  may be lower than
  \[ P(NN2|VVB)P(fish|NN2) \]
  but \[ P(VVB|PNP)P(can|VVB) \]
  versus \[ P(VM0|PNP)P(can|VM0) \]

- Actual systems use trigrams — smoothing and backoff are critical.

- Unseen words: these are not in the lexicon, so use all possible open class tags, possibly restricted by morphology.
Evaluation of POS tagging

- percentage of correct tags
- one tag per word (some systems give multiple tags when uncertain)
- over 95% for English (but note punctuation is unambiguous)
- **baseline** of taking the most common tag gives 90% accuracy
- different tagsets give slightly different results: utility of tag to end users vs predictive power (an open research issue)
Evaluation in general

- **Training data and test data** Test data must be kept unseen, often 90% training and 10% test data.
- **Baseline**
- **Ceiling** Human performance on the task, where the ceiling is the percentage agreement found between two annotators (interannotator agreement)
- **Error analysis** Error rates are nearly always unevenly distributed.
- **Reproducibility**
Representative corpora and data sparsity

- test corpora have to be representative of the actual application
- POS tagging and similar techniques are not always very robust to differences in genre
- balanced corpora may be better, but still don’t cover all text types
- communication aids: extreme difficulty in obtaining data, text corpora don’t give good prediction for real data
Outline of next lecture

Lecture 4: Parsing and generation.
Generative grammar
Simple context free grammars
Random generation with a CFG
Simple chart parsing with CFGs
More advanced chart parsing
Why not finite state?
Parsing (and generation)

Syntactic structure in analysis:
- as a step in assigning semantics
- checking grammaticality
- corpus-based investigations, lexical acquisition etc

Lecture 4: Parsing and generation.
Generative grammar
Simple context free grammars
Random generation with a CFG
Simple chart parsing with CFGs
More advanced chart parsing
Why not finite state?

Next lecture — beyond simple CFGs
Generative grammar

a formally specified grammar that can generate all and only the acceptable sentences of a natural language

Internal structure:

the big dog slept

can be bracketed

((the (big dog)) slept)

constituent a phrase whose components ‘go together’ . . .

weak equivalence grammars generate the same strings

strong equivalence grammars generate the same strings with same brackets
Context free grammars

1. a set of non-terminal symbols (e.g., S, VP);
2. a set of terminal symbols (i.e., the words);
3. a set of rules (productions), where the LHS (mother) is a single non-terminal and the RHS is a sequence of one or more non-terminal or terminal symbols (daughters);

   \[ S \rightarrow NP \ VP \]
   \[ V \rightarrow fish \]

4. a start symbol, conventionally S, which is a non-terminal.

Exclude empty productions, NOT e.g.:

\[ NP \rightarrow \epsilon \]
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
Analyses in the simple CFG

they fish

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

they can fish

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

dey fish in rivers

(S (NP they) (VP (VP (V fish))
(PP (P in) (NP rivers))))
Analyses in the simple CFG

they fish

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

they can fish

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

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

they fish in rivers

(S (NP they) (VP (VP (V fish))
 (PP (P in) (NP rivers))))
Analyses in the simple CFG

they fish

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

they can fish

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

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

they fish in rivers

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

(PP (P in) (NP rivers))))
Structural ambiguity without lexical ambiguity

they fish in rivers in December

(S (NP they)
  (VP (VP (V fish))
    (PP (P in) (NP (NP rivers)
      (PP (P in) (NP December)))))))

(S (NP they)
  (VP (VP (VP (V fish))
    (PP (P in) (NP (NP rivers))))
    (PP (P in) (NP December))))
Structural ambiguity without lexical ambiguity

they fish in rivers in December

(S (NP they)
 (VP (VP (V fish))
  (PP (P in) (NP (NP rivers)
   (PP (P in) (NP December))))))

(S (NP they)
 (VP (VP (VP (V fish))
   (PP (P in) (NP (NP rivers))))
  (PP (P in) (NP December))))
Parse trees

(S (NP they)
 (VP (V can)
  (VP (VP (V fish))
   (PP (P in)
    (NP December)))))
Using a grammar as a random generator

**Expand cat** category sentence-record:
Let possibilities be all lexical items matching category and all rules with LHS category.
If possibilities is empty,
then fail
else
Randomly select a possibility chosen from possibilities.
If chosen is lexical,
then append it to sentence-record
else
expand cat on each rhs category in chosen (left to right) with the updated sentence-record
return sentence-record
Random generator example

Expand cat $S ()$
\[ \text{possibilities} = \{ S \rightarrow \text{NP \ VP} \}, \quad \text{chosen} = S \rightarrow \text{NP \ VP} \]

Expand cat $\text{NP ()}$
\[ \text{possibilities} = \{ \text{it, they, fish} \} \]
\[ \text{chosen} = \text{fish} \]
\[ \text{sentence-record} = (\text{fish}) \]

Expand cat $\text{VP (fish)}$
\[ \text{possibilities} = \{ \text{VP \rightarrow V, VP \rightarrow V \ VP, VP \rightarrow V \ NP} \} \]
\[ \text{chosen} = \text{VP \rightarrow V} \]

Expand cat $V \ (\text{fish})$
\[ \text{possibilities} = \{ \text{fish, can} \} \]
\[ \text{chosen} = \text{fish} \]
\[ \text{sentence-record} = (\text{fish fish}) \]
Chart parsing

A dynamic programming algorithm (memoisation):
- **chart** store partial results of parsing
- **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:

<table>
<thead>
<tr>
<th>id</th>
<th>r</th>
<th>ma</th>
<th>dtrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>3 V</td>
<td>(fish)</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3 VP</td>
<td>(5)</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>3 VP</td>
<td>(3 6)</td>
</tr>
</tbody>
</table>
A bottom-up passive 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: \([id, \text{from}, \text{to}, \text{category}, \text{dtrs}]\)
For each rule \(lhs \rightarrow cat_1 \ldots cat_{n-1}, \text{category}\)
Find sets of contiguous edges
\([id_1, \text{from}_1, \text{to}_1, \text{cat}_1, \text{dtrs}_1]\) \ldots
\([id_{n-1}, \text{from}_{n-1}, \text{from}, \text{cat}_{n-1}, \text{dtrs}_{n-1}]\)
(such that \(to_1 = from_2\) etc)
For each set of edges,
**Add new edge** *from_1, to, lhs, (id_1 \ldots id)*
Bottom up parsing: edges

they can fish
Bottom up parsing: edges

1:NP  they
2:V    can
3:VP   
4:S    
5:V    fish
6:VP   
7:VP   
8:S    
9:NP   
10:VP  
11:S   

Natural Language Processing
Lecture 4: Parsing and generation.
Simple chart parsing with CFGs
Bottom up parsing: edges

- They can fish
- "they" is a noun phrase (NP)
- "can" is a verb (V)
- "fish" is a verb phrase (VP)
- "can" is the head of the verb phrase (VP)
- "they" is the head of the noun phrase (NP)
Bottom up parsing: edges
Bottom up parsing: edges
Bottom up parsing: edges

1:NP they
2:V can
3:VP
4:S
5:V fish
6:VP
7:VP
8:S
9:NP
10:VP
11:S
Bottom up parsing: edges
Bottom up parsing: edges

1: NP  
   they

2: V  
   can

3: VP

4: S

5: V  
   fish

6: VP

7: VP

8: S

9: NP

10: VP

11: S
Bottom up parsing: edges

1:NP  they
2:V  can
3:VP
4:S
5:V  fish
6:VP
7:VP
8:S
9:NP
10:VP
11:S
Bottom up parsing: edges

they can fish

S

NP

V

VP
Bottom up parsing: edges

1:NP  they
2:V    can
3:VP
4:S
5:V    fish
6:VP
7:VP
8:S
9:NP
10:VP
11:S
Parse construction

word = they, categories = {NP}

Add new edge 0, 1, NP, (they)

Matching grammar rules: \{VP \rightarrow V NP, PP \rightarrow P NP\}
No matching edges corresponding to V or P
word = can, categories = \{V\}

**Add new edge** 1, 2, V, (can)

Matching grammar rules: \{VP \rightarrow V\}

recurse on edges \{(2)\}
**Parse construction**

Add new edge 1, 2, VP, (2)
Matching grammar rules: \{S → NP VP, VP → V VP\}
recurse on edges \{(1,3)\}
Parse construction

They can fish

Add new edge 0, 2, S, (1, 3)
No matching grammar rules for S
Matching grammar rules: \{S \rightarrow NP \ \text{VP}, \ \text{VP} \rightarrow V \ \text{VP}\}
No edges for V \text{VP}
word = fish, categories = \{V, NP\}

**Add new edge** 2, 3, V, (fish)

Matching grammar rules: \{VP \rightarrow V\}

recurse on edges \{(5)\}
Parse construction

Add new edge 2, 3, VP, (5)
Matching grammar rules: \{S \rightarrow NP \ VP, \ VP \rightarrow V \ VP\}
No edges match NP
recurse on edges for V VP: \{(2,6)\}
Parse construction

Add new edge 1, 3, VP, (2, 6)
Matching grammar rules: \(S \rightarrow \text{NP VP}, \text{VP} \rightarrow \text{V VP}\)
recurse on edges for NP VP: \{(1,7)\}
Parse construction

Add new edge 0, 3, S, (1, 7)

No matching grammar rules for S

Matching grammar rules: \{S \rightarrow NP \ VP, \ VP \rightarrow V \ VP\}

No edges matching V
Parse construction

Add new edge 2, 3, NP, (fish)
Matching grammar rules: \{VP \rightarrow V \ NP, \ PP \rightarrow P \ NP\}
recurse on edges for V NP \{(2,9)\}

NB: fish as NP
Parse construction

Add new edge 1, 3, VP, (2, 9)
Matching grammar rules: \{S \rightarrow NP VP, VP \rightarrow V VP\}
recurse on edges for NP VP: \{(1, 10)\}
Parse construction

Add new edge 0, 3, S, (1, 10)
No matching grammar rules for S
Matching grammar rules: \{S \rightarrow NP \ VP, \ VP \rightarrow V \ VP\}
No edges corresponding to V \ VP
Matching grammar rules: \{VP \rightarrow V \ NP, \ PP \rightarrow P \ NP\}
No edges corresponding to P NP
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)))

Note: sample chart parsing code in Java is downloadable from the course web page.
Packing

- exponential number of parses means exponential time
- body can be cubic time: 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:

\([id-old, l_vtx, right_vtx, ma_cat, dtrs-old \cup dtrs]\)

and do not recurse on it
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 6))}
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.
Active chart parsing

<table>
<thead>
<tr>
<th>id</th>
<th>r</th>
<th>ma</th>
<th>exp</th>
<th>dtrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>NP</td>
<td>(they)</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>S</td>
<td>VP</td>
</tr>
</tbody>
</table>

- store partial rule applications
- record expected input as well as seen
- one active edge can create more than one passive edge. e.g., *they fish in Scotland* — edge 2 completed by *fish* and *fish in Scotland*. NP is combined with rule once not twice.
- active edges can be packed
Active chart parsing

I store partial rule applications
I record expected input as well as seen
I one active edge can create more than one passive edge.
  e.g., they fish in Scotland — edge 2 completed by fish and fish in Scotland. NP is combined with rule once not twice.
I active edges can be packed
Active chart parsing

- store partial rule applications
- record expected input as well as seen
- one active edge can create more than one passive edge. e.g., *they fish in Scotland* — edge 2 completed by *fish* and *fish in Scotland*. NP is combined with rule once not twice.
- active edges can be packed
Ordering the search space

- agenda: order edges in chart by priority
- top-down parsing: predict possible edges

Producing n-best parses:
- manual weight assignment
- probabilistic CFG — trained on a treebank
  - automatic grammar induction
  - automatic weight assignment to existing grammar
- beam-search
Why not FSA?

centre-embedding:

\[ A \rightarrow \alpha A \beta \]

generate grammars of the form \( a^n b^n \).
For instance:

the students the police arrested complained

However:

? the students the police the journalists criticised arrested complained

Limits on human memory / processing ability
Why not FSA practically?

More importantly for practical application:

1. FSM grammars are very redundant: difficult to build and maintain
2. FSM grammars don’t support composition of semantics but FSMs useful for:
   1. tokenizers (dates, times etc)
   2. named entity recognition in information extraction etc
   3. approximating CFGs in speech recognition
Outline of next lecture

Providing a more adequate treatment of syntax than simple CFGs: replacing the atomic categories by more complex data structures.

Lecture 5: Parsing with constraint-based grammars.

- Problems with simple CFG encoding: agreement, subcategorisation
- Feature structures (informally)
- Encoding agreement
- Parsing with feature structures
- Feature structures more formally
- Encoding subcategorisation
- Interface to morphology
Outline of today’s lecture

Providing a more adequate treatment of syntax than simple CFGs: replacing the atomic categories by more complex data structures.

Lecture 5: Parsing with constraint-based grammars.

Problems with simple CFG encoding: agreement, subcategorisation

Feature structures (informally)
Encoding agreement
Parsing with feature structures
Feature structures more formally
Encoding subcategorisation
Interface to morphology
Overgeneration in atomic category CFGs

- **agreement**: subject verb agreement. e.g., *they fish, it fishes,* *it fish,* *they fishes.* * means ungrammatical
- **case**: pronouns (and maybe who/whom) e.g., *they like them,* *they like they*

\[
\begin{align*}
\text{S} & \rightarrow \text{NP-sg-nom } \text{VP-sg} & \text{NP-sg-nom} & \rightarrow \text{he} \\
\text{S} & \rightarrow \text{NP-pl-nom } \text{VP-pl} & \text{NP-sg-acc} & \rightarrow \text{him} \\
\text{VP-sg} & \rightarrow \text{V-sg } \text{NP-sg-acc} & \text{NP-sg-nom} & \rightarrow \text{fish} \\
\text{VP-sg} & \rightarrow \text{V-sg } \text{NP-pl-acc} & \text{NP-pl-nom} & \rightarrow \text{fish} \\
\text{VP-pl} & \rightarrow \text{V-pl } \text{NP-sg-acc} & \text{NP-sg-acc} & \rightarrow \text{fish} \\
\text{VP-pl} & \rightarrow \text{V-pl } \text{NP-pl-acc} & \text{NP-pl-acc} & \rightarrow \text{fish}
\end{align*}
\]

BUT: very large grammar, misses generalizations, no way of saying when we don’t care about agreement.
Overgeneration in atomic category CFGs

- agreement: subject verb agreement. e.g., *they fish, it fishes, *it fish, *they fishes. * means ungrammatical
- case: pronouns (and maybe who/whom) e.g., *they like them, *they like they

\[ S \rightarrow NP_{-}sg_{-}nom \ VP_{-}sg \quad NP_{-}sg_{-}nom \rightarrow \ he \]
\[ S \rightarrow NP_{-}pl_{-}nom \ VP_{-}pl \quad NP_{-}sg_{-}acc \rightarrow \ him \]
\[ VP_{-}sg \rightarrow V_{-}sg \ NP_{-}sg_{-}acc \quad NP_{-}sg_{-}nom \rightarrow \ fish \]
\[ VP_{-}sg \rightarrow V_{-}sg \ NP_{-}pl_{-}acc \quad NP_{-}pl_{-}nom \rightarrow \ fish \]
\[ VP_{-}pl \rightarrow V_{-}pl \ NP_{-}sg_{-}acc \quad NP_{-}sg_{-}acc \rightarrow \ fish \]
\[ VP_{-}pl \rightarrow V_{-}pl \ NP_{-}pl_{-}acc \quad NP_{-}pl_{-}acc \rightarrow \ fish \]

BUT: very large grammar, misses generalizations, no way of saying when we don’t care about agreement.
Intuitive solution for case and agreement

- Separate slots (features) for CASE and AGR
- Slot values for CASE may be nom (e.g., they), acc (e.g., them) or unspecified (i.e., don’t care)
- Slot values for AGR may be sg, pl or unspecified
- Subjects have the same value for AGR as their verbs
- Subjects have CASE nom, objects have CASE acc
Natural Language Processing

Lecture 5: Parsing with constraint-based grammars.

Problems with simple CFG encoding: agreement, subcategorisation, long distance dependencies.

Subcategorization

- intransitive vs transitive etc
- verbs (and other types of words) have different numbers and types of syntactic arguments:
  * Kim adored
  * Kim gave Sandy
  * Kim adored to sleep
  * Kim liked to sleep
  * Kim devoured
  * Kim ate

- Subcategorization is correlated with semantics, but not determined by it.
Overgeneration because of missing subcategorization

Overgeneration:

they fish fish it
(S (NP they) (VP (V fish) (VP (V fish) (NP it)))))

- Informally: need slots on the verbs for their syntactic arguments.
  - intransitive takes no following arguments (complements)
  - simple transitive takes one NP complement
  - *like* may be a simple transitive or take an infinitival complement, etc
Long-distance dependencies

1. which problem did you say you don’t understand?
2. who do you think Kim asked Sandy to hit?
3. which kids did you say were making all that noise?

‘gaps’ (underscores below)

1. which problem did you say you don’t understand _?
2. who do you think Kim asked Sandy to hit _?
3. which kids did you say _ were making all that noise?

In 3, the verb were shows plural agreement.

* what kid did you say _ were making all that noise?

The gap filler has to be plural.

- Informally: need a ‘gap’ slot which is to be filled by something that itself has features.
Feature structures

1. **Features** like AGR with simple values: atomic-valued
2. Unspecified values possible on features: compatible with any value.
3. Values for features for subcat and gap themselves have features: complex-valued
4. **path**: a sequence of features
5. Method of specifying two paths are the same: reentrancy
6. **Unification**: combining two feature structures, retaining all information from each, or fail if information is incompatible.
Feature structures, continued

- Feature structures are singly-rooted directed acyclic graphs, with arcs labelled by features and terminal nodes associated with values.

\[
\begin{bmatrix}
\text{CASE} & [ ] \\
\text{AGR} & \text{sg}
\end{bmatrix}
\]

- In grammars, rules relate FSs — i.e. lexical entries and phrases are represented as FSs
- Rule application by unification
Graphs and AVMs

Example 1:

Here, CAT and AGR are atomic-valued features. NP and sg are values.

Example 2:

HEAD is complex-valued, AGR is unspecified.
Reentrancy indicated by boxed integer in AVM diagram: indicates path goes to the same node.
CFG with agreement

S → NP-sg VP-sg
S → NP-pl VP-pl
VP-sg → V-sg NP-sg
VP-sg → V-sg NP-pl
VP-pl → V-pl NP-sg
VP-pl → V-pl NP-pl
V-pl → like
V-sg → likes
NP-sg → it
NP-pl → they
NP-sg → fish
NP-pl → fish
Encoding agreement

FS grammar fragment encoding agreement

subj-verb rule

$$\text{S} \rightarrow \text{NP} \cdot \text{VP}$$

verb-obj rule

$$\text{VP} \rightarrow \text{V} \cdot \text{NP}$$

Root structure:

$$\text{S}$$

they

$$\text{NP}_{pl}$$

it

$$\text{NP}_{sg}$$

likes

$$\text{V}_{sg}$$

fish

$$\text{NP}_{[]}$$

like

$$\text{V}_{pl}$$
Parsing ‘they like it’

- The lexical structures for *like* and *it* are unified with the corresponding structures on the right hand side of the verb-obj rule (unifications succeed).
- The structure corresponding to the mother of the rule is then:

\[
\begin{array}{c}
\text{CAT} \\
\text{AGR} \\
\end{array}
\begin{array}{c}
\text{VP} \\
\text{pl} \\
\end{array}
\]

- This unifies with the rightmost daughter position of the subj-verb rule.
- The structure for *they* is unified with the leftmost daughter.
- The result unifies with root structure.
Rules as FSs

Rules have features MOTHER, DTR1, DTR2 ... DTRN.

Informally:

\[
\begin{array}{c}
\text{CAT} \quad \text{VP} \\
\text{AGR} \quad [1]
\end{array}
\rightarrow
\begin{array}{c}
\text{CAT} \quad \text{V} \\
\text{AGR} \quad [1]
\end{array},
\begin{array}{c}
\text{CAT} \quad \text{NP} \\
\text{AGR} \quad [ ]
\end{array}
\]

Actually:

\[
\begin{array}{c}
\text{MOTHER} \\
\text{DTR1} \\
\text{DTR2}
\end{array}
\begin{array}{c}
\text{CAT} \quad \text{VP} \\
\text{AGR} \quad [1]
\end{array}
\begin{array}{c}
\text{CAT} \quad \text{V} \\
\text{AGR} \quad [1]
\end{array},
\begin{array}{c}
\text{CAT} \quad \text{NP} \\
\text{AGR} \quad [ ]
\end{array}
\]
Verb-obj rule application

Feature structure for *like* unified with the value of DTR1:

```
MOTHER
DTR1
DTR2
```

```
CAT VP
AGR [1 pl]
```

Feature structure for *it* unified with the value for DTR2:

```
MOTHER
DTR1
DTR2
```

```
CAT VP
AGR [1 pl]
```

```
CAT V
AGR [1]
```

```
CAT NP
AGR [sg]
```
Subject-verb rule application 1

MOTHER value from the verb-object rule acts as the DTR2 of the subject-verb rule:

\[
\begin{align*}
\text{MOTHER} &\quad \text{value from the verb-object rule acts as the DTR2 of the subject-verb rule:} \\
\begin{bmatrix}
\text{CAT} \\
\text{AGR} \\
\text{VP}
\end{bmatrix}_{\text{pl}} &\quad \text{unified with the DTR2 of:} \\
\begin{bmatrix}
\text{DTR1} \\
\text{DTR2}
\end{bmatrix}
\end{align*}
\]

Gives:

\[
\begin{align*}
\begin{bmatrix}
\text{MOTHER} \\
\text{DTR1} \\
\text{DTR2}
\end{bmatrix} \\
\begin{bmatrix}
\text{CAT} \\
\text{AGR} \\
\text{S} \\
\text{1} \\
\text{pl}
\end{bmatrix} \\
\begin{bmatrix}
\text{CAT} \\
\text{AGR} \\
\text{NP} \\
\text{1}
\end{bmatrix} \\
\begin{bmatrix}
\text{CAT} \\
\text{AGR} \\
\text{VP} \\
\text{1}
\end{bmatrix}
\end{align*}
\]
Subject rule application 2

FS for *they*: \[
\begin{bmatrix}
\text{CAT} & \text{NP} \\
\text{AGR} & \text{pl}
\end{bmatrix}
\]

Unification of this with the value of DTR1 succeeds (but adds no new information):

\[
\begin{bmatrix}
\text{MOTHER} \\
\text{DTR1} \\
\text{DTR2}
\end{bmatrix} 
\begin{bmatrix}
\text{CAT} & \text{S} \\
\text{AGR} & 1 \\
\text{CAT} & \text{VP} \\
\text{AGR} & 1
\end{bmatrix}
\]

Final structure unifies with the root structure: \[
\begin{bmatrix}
\text{CAT} & \text{S}
\end{bmatrix}
\]
Properties of FSs

**Connectedness and unique root**  A FS must have a unique root node: apart from the root node, all nodes have one or more parent nodes.

**Unique features**  Any node may have zero or more arcs leading out of it, but the label on each (that is, the feature) must be unique.

**No cycles**  No node may have an arc that points back to the root node or to a node that intervenes between it and the root node.

**Values**  A node which does not have any arcs leading out of it may have an associated atomic value.

**Finiteness**  A FS must have a finite number of nodes.
Subsumption

Feature structures are ordered by information content — FS1 subsumes FS2 if FS2 carries extra information. FS1 subsumes FS2 if and only if the following conditions hold:

**Path values** For every path P in FS1 there is a path P in FS2. If P has a value t in FS1, then P also has value t in FS2.

**Path equivalences** Every pair of paths P and Q which are reentrant in FS1 (i.e., which lead to the same node in the graph) are also reentrant in FS2.

Unification

The unification of two FSs FS1 and FS2 is the most general FS which is subsumed by both FS1 and FS2, if it exists.
Grammar with subcategorisation

Verb-obj rule:

\[
\begin{aligned}
&\text{HEAD } 1 \\
&\text{OBJ filled} \\
&\text{SUBJ 3}
\end{aligned}
\rightarrow
\begin{aligned}
&\text{HEAD } 1 \\
&\text{OBJ 2} \\
&\text{SUBJ 3}
\end{aligned},
\begin{aligned}
&\text{OBJ filled}
\end{aligned}
\]

can (transitive verb):

\[
\begin{aligned}
&\text{HEAD } [\text{CAT verb}] \\
&\text{AGR pl} \\
&\text{OBJ [HEAD [CAT noun]]} \\
&\text{SUBJ [HEAD [CAT noun]]}
\end{aligned}
\]
Grammar with subcategorisation (abbrev for slides)

Verb-obj rule:  
\[
\begin{array}{c}
\text{HEAD 1} \\
\text{OBJ fld} \\
\text{SUBJ 3}
\end{array} \rightarrow \begin{array}{c}
\text{HEAD 1} \\
\text{OBJ 2} \\
\text{SUBJ 3}
\end{array}, \begin{array}{c}
\text{OBJ fld}
\end{array}
\]

can (transitive verb):  
\[
\begin{array}{c}
\text{HEAD} \\
\text{AGR pl} \\
\text{HEAD [CAT n]}
\end{array} \rightarrow \begin{array}{c}
\text{OBJ} \\
\text{HEAD [CAT n]}
\end{array}.
\]
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category

```
S ├── NP └── VP
    │       └── V └── VP
    │           └── V └── P └── NP
```
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category

```
S
 /  
NP  VP+
   /  
  V+  VP
     /  
    VP  PP
      /  
     V   P
       /  
      NP
```
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category

```
S  
|   
NP VP 
  |   
V VP 
    |   
V+ VP+ PP  
   |   
V+ P NP 
```
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category

- **SUBJ**: The subject-verb rule unifies the first daughter of the rule with the SUBJ value of the second. (‘the first dtr fills the SUBJ slot of the second dtr in the rule’)
Concepts for subcategorisation

- **HEAD**: information shared between a lexical entry and the dominating phrases of the same category

- **SUBJ**: The subject-verb rule unifies the first daughter of the rule with the SUBJ value of the second. (‘the first dtr fills the SUBJ slot of the second dtr in the rule’)

- **OBJ**: The verb-object rule unifies the second dtr with the OBJ value of the first. (‘the second dtr fills the OBJ slot of the first dtr in the rule’)

Example rule application: *they fish 1*

Lexical entry for fish:

```
[HEAD [CAT v ] ],
OBJ [ AGR pl ],
SUBJ [ HEAD [ CAT n ] ]
```

subject-verb rule:

```
[HEAD 1 ],
OBJ [ fld ],
SUBJ [ fld ] \rightarrow 2
```

```
[HEAD [ AGR 3 ] ],
OBJ [ fld ],
SUBJ [ fld ]
```

```
[HEAD 1 [ AGR 3 ] ],
OBJ [ fld ],
SUBJ [ 2 ]
```

unification with second dtr position gives:

```
[HEAD 1 [ CAT v pl ] ],
OBJ [ AGR 3 pl ],
SUBJ [ fld ]
```

```
[HEAD [ CAT n ] ],
OBJ [ AGR 3 ],
SUBJ [ fld ]
```

```
[HEAD 1 ],
OBJ [ fld ],
SUBJ [ 2 ]
```
Lexical entry for *they*:

\[
\begin{bmatrix}
\text{HEAD} & \text{CAT} & n \\
\text{AGR} & pl \\
\text{OBJ} & \text{fld} \\
\text{SUBJ} & \text{fld}
\end{bmatrix}
\]

unify this with first dtr position:

\[
\begin{bmatrix}
\text{HEAD} & 1 \\
\text{CAT} & v \\
\text{AGR} & 3 \\
\text{pl} \\
\text{OBJ} & \text{fld} \\
\text{SUBJ} & \text{fld}
\end{bmatrix} \rightarrow
\begin{bmatrix}
\text{HEAD} & 2 \\
\text{CAT} & n \\
\text{AGR} & 3 \\
\text{OBJ} & \text{fld} \\
\text{SUBJ} & \text{fld}
\end{bmatrix}
\]

Root is:

\[
\begin{bmatrix}
\text{HEAD} & \text{CAT} & v \\
\text{OBJ} & \text{fld} \\
\text{SUBJ} & \text{fld}
\end{bmatrix}
\]

Mother structure unifies with root, so valid.
Parsing with feature structure grammars

- Naive algorithm: standard chart parser with modified rule application
- Rule application:
  1. copy rule
  2. copy daughters (lexical entries or FSs associated with edges)
  3. unify rule and daughters
  4. if successful, add new edge to chart with rule FS as category
- Efficient algorithms reduce copying.
- Packing involves subsumption.
- Probabilistic FS grammars are complex.
Templates

Capture generalizations in the lexicon:

fish INTRANS_VERB
sleep INTRANS_VERB
snore INTRANS_VERB

INTRANS_VERB

[HEAD

OBJ

SUBJ

CAT

AGR

v

pl

fld

n]
Interface to morphology: inflectional affixes as FSs

\[
\begin{bmatrix}
\text{HEAD} \\
\text{AGR}
\end{bmatrix}
\begin{bmatrix}
\text{CAT} \\
\text{n}
\end{bmatrix}
\begin{bmatrix}
\text{pl}
\end{bmatrix}
\]

if stem is:

\[
\begin{bmatrix}
\text{HEAD} \\
\text{OBJ}
\end{bmatrix}
\begin{bmatrix}
\text{CAT} \\
\text{n}
\end{bmatrix}
\begin{bmatrix}
\text{fld}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{SUBJ} \\
\text{fld}
\end{bmatrix}
\]

stem unifies with affix template.

But unification failure would occur with verbs etc, so we get filtering (lecture 2).
Outline of next lecture

Compositional semantics: the construction of meaning (generally expressed as logic) based on syntax.
Lexical semantics: the meaning of individual words.

Lecture 6: Compositional and lexical semantics.
  Compositional semantics in feature structures
  Logical forms
  Meaning postulates
  Lexical semantics: semantic relations
  Polysemy
  Word sense disambiguation
Outline of today’s lecture

Compositional semantics: the construction of meaning (generally expressed as logic) based on syntax. Lexical semantics: the meaning of individual words.

Lecture 6: Compositional and lexical semantics.
  Compositional semantics in feature structures
  Logical forms
  Meaning postulates
  Lexical semantics: semantic relations
  Polysemy
  Word sense disambiguation
Simple compositional semantics in feature structures

- Semantics is built up along with syntax
- Subcategorization ‘slot’ filling instantiates syntax
- Formally equivalent to logical representations (below: predicate calculus with no quantifiers)
- Alternative FS encodings possible

Objective: obtain the following semantics for *they like fish*: 
\[ \text{pron}(x) \land (\text{like}_v(x, y) \land \text{fish}_n(y)) \]
Feature structure encoding of semantics

\[
\begin{align*}
\text{pron}(x) \land & (\text{like}_v(x, y) \land \text{fish}_n(y)) \\
\text{PRED} & & \text{and} \\
\text{ARG1} & & \text{pron} \\
\text{ARG1} & & 1 \\
\text{PRED} & & \text{and} \\
\text{PRED} & & \text{like}_v \\
\text{ARG1} & & 1 \\
\text{ARG2} & & 2 \\
\text{PRED} & & \text{fish}_n \\
\text{ARG1} & & 2 \\
\text{ARG2} & & 2
\end{align*}
\]
Noun entry

Corresponds to fish(x) where the INDEX points to the characteristic variable of the noun (that is x). The INDEX is unambiguous here, but e.g., picture(x, y) ∧ sheep(y)

picture of sheep
Noun entry

```
fish
```

- Corresponds to fish(\(x\)) where the INDEX points to the characteristic variable of the noun (that is \(x\)). The INDEX is unambiguous here, but e.g., picture(\(x, y\)) \(\land\) sheep(\(y\))

\textit{picture of sheep}
Verb entry

Like

- **HEAD**: [CAT v]
- **AGR**: [pl]
- **OBJ**: [f1d]
- **SEM**: [INDEX 2]
- **SUBJ**: [CAT n]
- **SEM**: [INDEX 1]
- **PRED**: [like_v]
- **ARG1**: 1
- **ARG2**: 2
Verb-object rule

- As last time: object of the verb (DTR2) ‘fills’ the OBJ slot
- New: semantics on the mother is the ‘and’ of the semantics of the dtrs
Meaning representation for a sentence is called the logical form.

Standard approach to composition in theoretical linguistics is lambda calculus, building FOPC or higher order representation.

Representation in notes is quantifier-free predicate calculus but possible to build FOPC or higher-order representation in FSs.

Theorem proving.

Generation: starting point is logical form, not string.
Meaning postulates

- e.g.,

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

- usable with compositional semantics and theorem provers
- e.g. from ‘Kim is a bachelor’, we can construct the LF

\[ \text{bachelor}'(\text{Kim}) \]

and then deduce

\[ \text{unmarried}'(\text{Kim}) \]

- OK for narrow domains, but ‘classical’ lexical semantic relations are more generally useful
Lexical semantic relations

**Hyponymy: IS-A:**
- (a sense of) *dog* is a **hyponym** of (a sense of) *animal*
- *animal* is a **hypernym** of *dog*
- hyponymy relationships form a **taxonomy**
- works best for concrete nouns

**Meronomy: 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*
WordNet

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

Overview of adj red:

1. (43) red, reddish, ruddy, blood-red, carmine, cerise, cherry, cherry-red, crimson, ruby, ruby-red, scarlet - (having any of numerous bright or strong colors reminiscent of the color of blood or cherries or tomatoes or rubies)
2. (8) red, reddish - ((used of hair or fur) of a reddish brown color; "red deer"; reddish hair)
Hyponymy in WordNet

Sense 6
big cat, cat

=> leopard, Panthera pardus
  => leopardess
  => panther

=> snow leopard, ounce, Panthera uncia
=> jaguar, panther, Panthera onca, Felis onca

=> lion, king of beasts, Panthera leo
  => lioness
  => lionet

=> tiger, Panthera tigris
  => Bengal tiger
  => tigress
Some uses of lexical semantics

- Semantic classification: e.g., for selectional restrictions (e.g., the object of *eat* has to be something edible) and for named entity recognition
- Shallow inference: ‘X murdered Y’ implies ‘X killed Y’ etc
- Back-off to semantic classes in some statistical approaches
- Word-sense disambiguation
- Machine Translation: if you can’t translate a term, substitute a hypernym
- Query expansion: if a search doesn’t return enough results, one option is to replace an over-specific term with a hypernym
Polysemy

- **homonymy**: unrelated word senses. *bank* (raised land) vs *bank* (financial institution)
- *bank* (financial institution) vs *bank* (in a casino): related but distinct senses.
- *bank* (N) (raised land) vs *bank* (V) (to create some raised land): regular polysemy. Compare *pile*, *heap* etc
- vagueness: *bank* (river vs snow vs cloud)?

No clearcut distinctions.
Dictionaries are not consistent.
Word sense disambiguation

Needed for many applications, problematic for large domains. Assumes that we have a standard set of word senses (e.g., WordNet)

- frequency: e.g., diet: the food sense (or senses) is much more frequent than the parliament sense (Diet of Wurms)
- collocations: e.g. striped bass (the fish) vs bass guitar: syntactically related or in a window of words (latter sometimes called ‘cooccurrence’). Generally ‘one sense per collocation’.
- selectional restrictions/preferences (e.g., Kim eats bass, must refer to fish
WSD techniques

- supervised learning: cf. POS tagging from lecture 3. But sense-tagged corpora are difficult to construct, algorithms need far more data than POS tagging
- unsupervised learning (see below)
- Machine readable dictionaries (MRDs): e.g., look at overlap with words in definitions and example sentences
- selectional preferences: don’t work very well by themselves, useful in combination with other techniques
Disambiguating *plant* (factory vs vegetation senses):

1. Find contexts in training corpus:

<table>
<thead>
<tr>
<th>sense</th>
<th>training example</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>company said that the <em>plant</em> is still operating</td>
</tr>
<tr>
<td>?</td>
<td>although thousands of <em>plant</em> and animal species</td>
</tr>
<tr>
<td>?</td>
<td>zonal distribution of <em>plant</em> life</td>
</tr>
<tr>
<td>?</td>
<td>company manufacturing <em>plant</em> is in Orlando</td>
</tr>
<tr>
<td></td>
<td>etc</td>
</tr>
</tbody>
</table>
Yarowsky (1995): schematically

Initial state
2. Identify some seeds to disambiguate a few uses. e.g., ‘plant life’ for vegetation use (A) ‘manufacturing plant’ for factory use (B):

<table>
<thead>
<tr>
<th>sense</th>
<th>training example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>company said that the <em>plant</em> is still operating</td>
</tr>
<tr>
<td></td>
<td>although thousands of <em>plant</em> and animal species</td>
</tr>
<tr>
<td>A</td>
<td>zonal distribution of <em>plant</em> life</td>
</tr>
<tr>
<td>B</td>
<td>company manufacturing <em>plant</em> is in Orlando etc</td>
</tr>
</tbody>
</table>
Seeds

A A ??
??
A

life

A

A

A

A

manu.

B

B

??

??

??

??

??

??

??

??

??

??

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

<table>
<thead>
<tr>
<th>reliability</th>
<th>criterion</th>
<th>sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.10</td>
<td>plant life</td>
<td>A</td>
</tr>
<tr>
<td>7.58</td>
<td>manufacturing plant</td>
<td>B</td>
</tr>
<tr>
<td>6.27</td>
<td>animal within 10 words of plant etc</td>
<td>A</td>
</tr>
</tbody>
</table>

Decision list classifier: automatically trained if/then statements. Experimenter decides on classes of test by providing definitions of features of interest: system builds specific tests and provides reliability metrics.
4. Apply the classifier to the training set and add reliable examples to A and B sets.

<table>
<thead>
<tr>
<th>sense</th>
<th>training example</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>company said that the <em>plant</em> is still operating</td>
</tr>
<tr>
<td>A</td>
<td>although thousands of <em>plant</em> and animal species</td>
</tr>
<tr>
<td>A</td>
<td>zonal distribution of <em>plant</em> life</td>
</tr>
<tr>
<td>B</td>
<td>company manufacturing <em>plant</em> is in Orlando</td>
</tr>
<tr>
<td></td>
<td>etc</td>
</tr>
</tbody>
</table>

5. Iterate the previous steps 3 and 4 until convergence
Iterating:
Final:
6. Apply the classifier to the unseen test data

‘one sense per discourse’: can be used as an additional refinement

e.g., once you’ve disambiguated *plant* one way in a particular text/section of text, you can assign all the instances of *plant* to that sense
Evaluation of WSD

- SENSEVAL competitions
- evaluate against WordNet
- baseline: pick most frequent sense — hard to beat (but don’t always know most frequent sense)
- human ceiling varies with words
- MT task: more objective but sometimes doesn’t correspond to polysemy in source language
Outline of next lecture

Putting sentences together (in text).

Lecture 7: Discourse.
  Relationships between sentences.
  Coherence
  Anaphora (pronouns etc)
  An algorithm for anaphora resolution
Outline of today’s lecture

Putting sentences together (in text).

Lecture 7: Discourse.
Relationships between sentences.
Coherence
Anaphora (pronouns etc)
An algorithm for anaphora resolution
Rhetorical relations

Max fell. John pushed him.

can be interpreted as:

1. Max fell because John pushed him.
   EXPLANATION

or

2. Max fell and then John pushed him.
   NARRATION

Implicit relationship: discourse relation or rhetorical relation
because, and then are examples of cue phrases
Coherence

Discourses have to have connectivity to be coherent:

Kim got into her car. Sandy likes apples.

Can be OK in context:

Kim got into her car. Sandy likes apples, so Kim thought she’d go to the farm shop and see if she could get some.
Coherence in generation

Strategic generation: constructing the logical form. Tactical generation: logical form to string. Strategic generation needs to maintain coherence.

In trading yesterday: Dell was up 4.2%, Safeway was down 3.2%, HP was up 3.1%.

Better:

Computer manufacturers gained in trading yesterday: Dell was up 4.2% and HP was up 3.1%. But retail stocks suffered: Safeway was down 3.2%.

So far this has only been attempted for limited domains: e.g. tutorial dialogues.
Coherence in interpretation

Discourse coherence assumptions can affect interpretation:

Kim’s bike got a puncture. She phoned the AA.

Assumption of coherence (and knowledge about the AA) leads to *bike* interpreted as motorbike rather than pedal cycle.

John likes Bill. He gave him an expensive Christmas present.

If EXPLANATION - ‘he’ is probably Bill.
If JUSTIFICATION (supplying evidence for first sentence), ‘he’ is John.
Factors influencing discourse interpretation

1. Cue phrases.
2. Punctuation (also prosody) and text structure.
   Max fell (John pushed him) and Kim laughed.
   Max fell, John pushed him and Kim laughed.
3. Real world content:
   Max fell. John pushed him as he lay on the ground.
4. Tense and aspect.
   Max fell. John had pushed him.
   Max was falling. John pushed him.

Hard problem, but ‘surfacy techniques’ (punctuation and cue phrases) work to some extent.
Rhetorical relations and summarization

Analysis of text with rhetorical relations generally gives a binary branching structure:

- nucleus and satellite: e.g., EXPLANATION, JUSTIFICATION
- equal weight: e.g., NARRATION

Max fell because John pushed him.
Rhetorical relations and summarization

Analysis of text with rhetorical relations generally gives a binary branching structure:

- **nucleus** and **satellite**: e.g., EXPLANATION, JUSTIFICATION
- **equal weight**: e.g., NARRATION

Max fell because John pushed him.
Summarisation by satellite removal

If we consider a discourse relation as a relationship between two phrases, we get a binary branching tree structure for the discourse. In many relationships, such as Explanation, one phrase depends on the other: e.g., the phrase being explained is the main one and the other is subsidiary. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse.
Summarisation by satellite removal

If we consider a discourse relation as a relationship between two phrases, we get a binary branching tree structure for the discourse. In many relationships, such as Explanation, one phrase depends on the other: e.g., the phrase being explained is the main one and the other is subsidiary. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse.
Summarisation by satellite removal

If we consider a discourse relation as a relationship between two phrases, we get a binary branching tree structure for the discourse. In many relationships, such as Explanation, one phrase depends on the other: e.g., the phrase being explained is the main one and the other is subsidiary. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse.

We get a binary branching tree structure for the discourse. In many relationships one phrase depends on the other. In fact we can get rid of the subsidiary phrases and still have a reasonably coherent discourse.
Referring expressions

Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian’s Oxford study.

- **Referent**: a real world entity that some piece of text (or speech) refers to. The actual Prof. Ferguson
- **Referring expressions**: bits of language used to perform reference by a speaker. ‘Niall Ferguson’, ‘he’, ‘him’
- **Antecedent**: the text evoking a referent. ‘Niall Ferguson’
- **Anaphora**: the phenomenon of referring to an antecedent.
Pronoun resolution

Pronouns: a type of anaphor. Pronoun resolution: generally only consider cases which refer to antecedent noun phrases.

Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian’s Oxford study.
Pronoun resolution

Pronouns: a type of anaphor.
Pronoun resolution: generally only consider cases which refer to antecedent noun phrases.

Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian’s Oxford study.
Pronoun resolution

Pronouns: a type of anaphor. Pronoun resolution: generally only consider cases which refer to antecedent noun phrases.

Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian’s Oxford study.
Pronoun resolution

Pronouns: a type of anaphor. Pronoun resolution: generally only consider cases which refer to antecedent noun phrases.

- hard constraints (e.g., agreement)
- soft preferences / salience (depend on discourse structure)
Hard constraints: Pronoun agreement

- A little girl is at the door — see what she wants, please?
- My dog has hurt his foot — he is in a lot of pain.
- * My dog has hurt his foot — it is in a lot of pain.

Complications:
- The team played really well, but now they are all very tired.
- Kim and Sandy are asleep: they are very tired.
- Kim is snoring and Sandy can’t keep her eyes open: they are both exhausted.
Hard constraints: Reflexives

- $\text{John}_i$ cut himself$_i$ shaving. (herself = John, subscript notation used to indicate this)
- # $\text{John}_i$ cut him$_j$ shaving. ($i \neq j$ — a very odd sentence)

Reflexive pronouns must be coreferential with a preceding argument of the same verb, non-reflexive pronouns cannot be.
Hard constraints: Pleonastic pronouns

Pleonastic pronouns are semantically empty, and don’t refer:

- It is snowing
- It is not easy to think of good examples.
- It is obvious that Kim snores.
- It bothers Sandy that Kim snores.
Soft preferences: Salience

**Recency**  Kim has a fast car. Sandy has an even faster one. Lee likes to drive it.

**Grammatical role** Subjects > objects > everything else: Fred went to the Grafton Centre with Bill. He bought a CD.

**Repeated mention** Entities that have been mentioned more frequently are preferred.

**Parallelism** Entities which share the same role as the pronoun in the same sort of sentence are preferred: Bill went with Fred to the Grafton Centre. Kim went with him to Lion Yard. Him=Fred

**Coherence effects** (mentioned above)
Lappin and Leass

Discourse model: referring NPs in equivalence classes with global salience value (incrementally updated).

For example:

N  Niall Ferguson, him  435
S  Stephen Moss  310
H  the historian  100
O  Oxford study  100

Resolve each pronoun to the entity with the highest weight in the discourse model.
Lappin and Leass’s algorithm

For each sentence:

1. Divide by two the global salience factors
2. Identify referring NPs
3. Calculate global salience factors for each NP (see below)
4. Update the discourse model with the referents and their global salience scores.
5. For each pronoun:
   5.1 Collect potential referents
   5.2 Filter referents
   5.3 Calculate the per pronoun adjustments for each referent (see below).
   5.4 Select the referent with the highest salience value for its equivalence class plus its per-pronoun adjustment.
   5.5 Add the pronoun into the equivalence class for that referent, and increment the salience factor.
Weights

Global salience factors:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Weight</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>recency</td>
<td>100</td>
<td>(current sentence)</td>
</tr>
<tr>
<td>subject</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>existential</td>
<td>70</td>
<td><em>there is a cat</em></td>
</tr>
<tr>
<td>direct object</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>indirect object</td>
<td>40</td>
<td><em>give Sandy a present</em></td>
</tr>
<tr>
<td>oblique complement</td>
<td>40</td>
<td><em>put the cat on a mat</em></td>
</tr>
<tr>
<td>non-embedded noun</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>non-adverbial</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

(i.e., embedded -80 and adverbial -50 but no negative weights)

Per pronoun salience factors:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Weight</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>cataphora</td>
<td>-175</td>
<td>pronoun before NP</td>
</tr>
<tr>
<td>same role</td>
<td>35</td>
<td>e.g., pronoun and NP both subject</td>
</tr>
</tbody>
</table>
Example

Niall Ferguson is prolific, well-paid and a snappy dresser. Stephen Moss hated him — at least until he spent an hour being charmed in the historian’s Oxford study.

Possible antecedents:

N Niall Ferguson, him 435
S Stephen Moss 310

N has score 155 + 280 ((subject + non-embedded + non-adverbial + recency)/2 + (direct object + non-embedded + non-adverbial + recency))
S has score 310 (subject + non-embedded + non-adverbial + recency) + same role per-pronoun 35

So we resolve he to N (wrongly . . . )
Example, continued

Add *he* to the discourse referent equivalence class. Update weights: add 80 because *he* is subject

N Niall Ferguson, him, he 515

Note: no duplicate factors for the same sentence (e.g., no weight added because *he* is non-embedded)
Anaphora for everyone, Kennedy and Boguraev

Modification of Lappin and Leass that doesn’t require a parser.

1. POS tag input text (Lingsoft tagger)
2. Regular expressions to identify NPs (NP chunking), mark expletive *it*
3. Regular expressions for grammatical role
4. Text segmentation: don’t cross document boundaries etc.
5. Heuristics for reflexives
6. Otherwise much as Lappin and Leass
Evaluation

1. LL quoted 86% (computer manuals), KB 75% (mix genres)
2. much less standardized than POS tagging: datasets, metrics
3. results are genre-dependent
4. replication is difficult
Outline of next lecture

Applications and (perhaps) demos.
Outline of today’s lecture

Lecture 8: Applications.
  Spoken dialogue systems
  Question Answering
  Wrapping up
Spoken dialogue systems

1. Single initiative systems (also known as system initiative systems): system controls what happens when.
   System: Do you have your customer reference number? Please say yes or no. User: Yes
   Limited mixed-initiative:
   System: When do you want to leave?
   User: the twenty-third
   OR
   User: the morning of the twenty-third

2. Mixed initiative dialogue. Both participants can control the dialogue to some extent.
   System: Which station do you want to leave from?
   User: I don’t know, tell me which station I need for Cambridge.
Approaches to SDS

- Custom grammars: FSAs or simple CFGs (compiled to FSAs) at each point in a dialogue controlled by an FSA. VoiceXML
  Feature structure grammar compiled to CFG/FSA: e.g., Clarissa (International Space Station).

- Statistical language modelling plus robust customised grammars or keyword spotting

- Statistical language modelling plus grammar induction

- Statistical language modelling plus general purpose grammar
Spoken dialogue system architecture

- **Dialogue Manager**
- **Database (DB)**
- **Speech Recognition**
- **Text-to-Speech**

**Processes**:
- **Parsing**: turn-specific grammars and grammar constraints
- **Slot Filling**: filled slots and speech
- **User Input**
- **Speech Output**
Dialogue management in single-initiative SDS

- Finite-state dialogue manager
- Tightly controls the dialogue: prompts user for specific information
- Separate recognition grammar for every state
- DB may help specify the grammars: e.g.,
  1. prompt for post code
  2. get 100 items on n-best list from recogniser
  3. use first line of addresses from these to build a FS grammar
  4. prompt for ‘first line’ of address
  5. disambiguate post code
- Confirmation strategy is important
QA with parsed corpus

PARSED CORPUS

query match

PARSING

response selection

output

MORPHOLOGY

INPUT PROCESSING

user input
Questions and answers: QA, NLID etc

A valid answer should entail the query (with suitable interpretation of *wh*-terms etc).

*Is a dog barking?*

\[ \exists x [\text{dog}'(x) \land \text{bark}'(x)] \]

*A dog is barking* entails *A dog is barking*

*Rover is barking* and *Rover is a dog* entails *A dog is barking.*

\[ \text{bark}'(\text{Rover}) \land \text{dog}'(\text{Rover}) \text{ entails } \exists x [\text{dog}'(x) \land \text{bark}'(x)] \]

*which dog is barking?*

\[ \text{bark}'(\text{Rover}) \land \text{dog}'(\text{Rover}) \text{ entails } \exists x [\text{dog}'(x) \land \text{bark}'(x)] \]

Bind query term to answer.
Example
What eats jellyfish?
Simplified semantics:
[ a:eat(e), ARG1(a,x), ARG2(a,y), jellyfish(y) ]
So won’t match on *jellyfish eat fish*. 
What eats jellyfish?

Example

Turtles eat jellyfish and they have special hooks in their throats to help them swallow these slimy animals.
What eats jellyfish?

Example
Turtles eat jellyfish and they have special hooks in their throats to help them swallow these slimy animals.

Match on [ a:eat(e), ARG1(a,x), ARG2(a,y), jellyfish(y) ]

A logically valid answer which entails the query since the conjunct can be ignored.
What eats jellyfish?

Example

Sea turtles, ocean sunfish (Mola mola) and blue rockfish all are able to eat large jellyfish, seemingly without being affected by the nematocysts.
What eats jellyfish?

Example
Sea turtles, ocean sunfish (Mola mola) and blue rockfish all are able to eat large jellyfish, seemingly without being affected by the nematocysts.

Pattern matching on semantics:
[ a:eat(e), ARG1(a,x), ARG2(a,y), large(y), jellyfish(y) ]

eat large jellyfish entails eat jellyfish (because large is intersective)
What eats jellyfish?

Example

Also, open ocean-dwelling snails called Janthina and even some seabirds have been known to eat jellyfish.
What eats jellyfish?

Example
Also, open ocean-dwelling snails called Janthina and even some seabirds have been known to eat jellyfish.

\[ \text{[ } a_1 : \text{know}(e), \text{ARG2}(a_1,h_1), \text{qeq}(h_1,lb), \text{lb:a:eat}(e), \text{ARG1}(a,x), \text{ARG2}(a,y), \text{jellyfish}(y) \text{ ]} \]

Logically valid if \textit{know} is taken as truth preserving.

\[ \forall P \forall y[\text{know}(y, P) \implies P] \]

Axioms like this required for logically valid entailment: missing axiom would cause failure to match.
What eats jellyfish?

Example

Also, open ocean-dwelling snails called Janthina and even some seabirds have been known to eat jellyfish.

\[ \text{[ a1:know(e), ARG2(a1,h1), qeq(h1,lb), lb:a:eat(e), ARG1(a,x), ARG2(a,y), jellyfish(y) ]} \]

Logically valid if *know* is taken as truth preserving.

\[ \forall P \forall y[\text{know}(y, P) \implies P] \]

Axioms like this required for logically valid entailment: missing axiom would cause failure to match.
0. Corpus collection. Parse corpus to semantic representation.
1. Process query: (i.e., separate words, detect non-words), run automatic spelling correction (maybe).
2. Morphology and lexical lookup
4. Match question against parsed corpus.
5. Return snippets/documents for best match.
Morphology and lexical lookup

What eats aardvarks?

What eat+3SG aardvark+PL?
Morphology and lexical lookup, 2

spelling rule

plur_noun_orule :=
%suffix (!s !ss) (!ss !ssses) (ss sses)
(!ty !ties) (ch ches) (sh shes) (x xes) (z zes)
lex_rule_infl_affixed &
[ ND-AFF +,
    SYNSEM mass_or_count_synsem &
    [ LOCAL plur_noun ]].
Semantics

What eats aardvarks?

<mrs>
<label vid='1'/><var vid='2'/>
<ep cfrom='0' cto='4'><pred>THING_REL</pred><label vid='3'/>
<fvpair><rargname>ARG0</rargname><var vid='4' sort='x'>
<extrapair><path>PERS</path><value>3</value></extrapair>
<extrapair><path>NUM</path><value>SG</value></extrapair>
<extrapair><path>SF</path><value>PROP</value></extrapair></var></fvpair></ep>
<ep cfrom='0' cto='4'><pred>WHICH_Q_REL</pred><label vid='5'/>
<fvpair><rargname>ARG0</rargname><var vid='4' sort='x'>
<extrapair><path>PERS</path><value>3</value></extrapair>
<extrapair><path>NUM</path><value>SG</value></extrapair>
<extrapair><path>SF</path><value>PROP</value></extrapair></var></fvpair><rargname>RSTR</rargname><var vid='6' sort='h'></var></ep>
<fvpair><rargname>BODY</rargname><var vid='7' sort='h'></var></ep>
<ep cfrom='5' cto='9'><spred>_eat_v_1_rel</spred><label vid='8'/>
<fvpair><rargname>ARG0</rargname><var vid='2' sort='e'>
<extrapair><path>TENSE</path><value>PRES</value></extrapair>
<extrapair><path>MOOD</path><value>INDICATIVE</value></extrapair>
<extrapair><path>PROG</path><value>-</value></extrapair>
<extrapair><path>PERF</path><value>-</value></extrapair>
<extrapair><path>SF</path><value>QUES</value></extrapair></var></fvpair><rargname>ARG1</rargname><var vid='4' sort='x'>
<extrapair><path>PERS</path><value>3</value></extrapair>
<extrapair><path>NUM</path><value>SG</value></extrapair>
<extrapair><path>SF</path><value>PROP</value></extrapair></var></ep>
<ep cfrom='10' cto='20'><spred>_aardvark_n_1_rel</spred><label vid='13'/>
<fvpair><rargname>ARG0</rargname><var vid='9' sort='x'>
<extrapair><path>PERS</path><value>3</value></extrapair>
<extrapair><path>NUM</path><value>PL</value></extrapair>
<extrapair><path>IND</path><value>+</value></extrapair>
<extrapair><path>DIV</path><value>+</value></extrapair>
<extrapair><path>SF</path><value>PROP</value></extrapair></var></fvpair><rargname>RSTR</rargname><var vid='11' sort='h'></var></ep>
<fvpair><rargname>BODY</rargname><var vid='12' sort='h'></var></ep>
<ep cfrom='10' cto='20'><spred>_aardvark_n_1_rel</spred><label vid='13'/>
<fvpair><rargname>ARG0</rargname><var vid='9' sort='x'>
<extrapair><path>PERS</path><value>3</value></extrapair>
<extrapair><path>NUM</path><value>PL</value></extrapair>
<extrapair><path>IND</path><value>+</value></extrapair>
<extrapair><path>DIV</path><value>+</value></extrapair>
<extrapair><path>SF</path><value>PROP</value></extrapair></var></fvpair><rargname>RSTR</rargname><var vid='11' sort='h'></var></ep>
<fvpair><rargname>BODY</rargname><var vid='12' sort='h'></var></ep>
<hcons hreln='qeq'><hi><var vid='6' sort='h'></var></hi><lo><var vid='3' sort='h'></var></lo></hcons>
<hcons hreln='qeq'><hi><var vid='11' sort='h'></var></hi><lo><var vid='13' sort='h'></var></lo></hcons>
</mrs>
DELPH-IN: Deep Linguistic Processing using HPSG

- Informal collaboration on tools and grammars: see http://www.delph-in.net/
- Large grammars for English, German and Japanese; medium/growing for Spanish, Norwegian, Portuguese, Korean, French. Many small grammars.
- Common semantic framework: Minimal Recursion Semantics (MRS) and Robust MRS. RMRS also from shallower parsing, chunking, POS tagging.
- Parsing and generation (realization), integrated shallower processing.
- Grammar Matrix: framework/starter kit for the development of grammars for diverse languages.
Generation with the ERG

- Bidirectional grammar, but want to recognise multiple dialects and generate consistently in an appropriate one
- Full generation so far only used in MT
- Needs further work on speed, selection of realisation (i.e., the generated string) and implementation in a runtime system
Themes: ambiguity

- levels: morphology, syntax, semantic, lexical, discourse
- resolution: local ambiguity, syntax as filter for morphology, selectional restrictions.
- ranking: parse ranking, WSD, anaphora resolution.
- processing efficiency: chart parsing
Themes: evaluation

- training data and test data
- reproducibility
- baseline
- ceiling
- module evaluation vs application evaluation
- nothing is perfect!
Themes: evaluation

- training data and test data
- reproducibility
- baseline
- ceiling
- module evaluation vs application evaluation
- nothing is perfect!
Conclusion

- different processing modules
- different applications blend modules differently
- many different styles of algorithm:
  1. FSAa and FSTs
  2. Markov models and HMMs
  3. CFG (and probabilistic CFGs)
  4. constraint-based frameworks
  5. inheritance hierarchies (WordNet), decision trees (WSD)
  6. mixing hard and soft constraints (Lappin and Leass)
More about speech and language processing

- Simone Teufel’s Part II course
- MPhil course: CSTIT
  - jointly taught with Dept of Engineering
  - see lab web page for details
  - financial support available for some applicants
More about speech and language processing

- Simone Teufel’s Part II course
- MPhil course: CSTIT
  - jointly taught with Dept of Engineering
  - see lab web page for details
  - financial support available for some applicants