Natural Language Processing

Ann Copestake

Computer Laboratory
University of Cambridge

October 2013

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 computational modelling 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 7 and 8.
4. Pragmatics — meaning in context: lecture 9.
5. Language generation — lecture 10.
6. Humans vs machines — lecture 11.

Also note:

- Exercises: pre-lecture and post-lecture
- Glossary
- Recommended Book: Jurafsky and Martin (2008).

Querying a knowledge base

User query:

- Has my order number 4291 been shipped yet?

Database:

ORDER

Order number	Date ordered	Date shipped
4290	2/2/13	2/2/13
4291	2/2/13	2/2/13
4292	2/2/13	

USER: Has my order number 4291 been shipped yet?
DB QUERY: order(number=4291,date_shipped=?)
RESPONSE: Order number 4291 was shipped on 2/2/13

Why is this difficult?

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

1. How fast is the TZ?
2. How fast will my TZ arrive?
3. Please tell me when I can expect the TZ 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 TZ?
2. How fast will my TZ arrive?
3. Please tell me when I can expect the TZ 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 TZ?
2. How fast will my TZ arrive?
3. Please tell me when I can expect the TZ 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 TZ?
2. How fast will my TZ arrive?
3. Please tell me when I can expect the TZ 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 TZ?
2. How fast will my TZ arrive?
3. Please tell me when I can expect the TZ 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 TZ?
2. How fast will my TZ arrive?
3. Please tell me when I can expect the TZ 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)?

Wouldn't it be better if . . . ?

The properties which make natural language difficult to process are essential to human communication:

- Flexible
- Learnable but compact
- Emergent, evolving systems

Synonymy and ambiguity go along with these properties.

Natural language communication can be indefinitely precise:

- Ambiguity is mostly local (for humans)
- Semi-formal additions and conventions for different genres

Wouldn't it be better if . . . ?

The properties which make natural language difficult to process are essential to human communication:

- Flexible
- Learnable but compact
- Emergent, evolving systems

Synonymy and ambiguity go along with these properties.
Natural language communication can be indefinitely precise:

- Ambiguity is mostly local (for humans)
- Semi-formal additions and conventions for different genres

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
- sentiment classification
- question answering

More NLP applications . . .

- summarization
- text segmentation
- exam marking
- language teaching
- 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 ...

Samsung Galaxy Note 3 (from the Guardian)

If you're after a phablet, the Samsung Galaxy Note 3 is the best one available right now.
It's a snappy, lag-free experience, with great battery life and fast charging, but it's just not big enough to be a proper 7in tablet replacement.
It's also likely be too big for most users looking for a smartphone, who will struggle to fit it in their pockets and will find it near-on impossible to use one-handed.
Samsung's TouchWiz customisations to Android are often gimmicky and confusing, but they can be turned off to save frustration and battery life. . . .

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.

IMDb: An American Werewolf in London (1981)

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

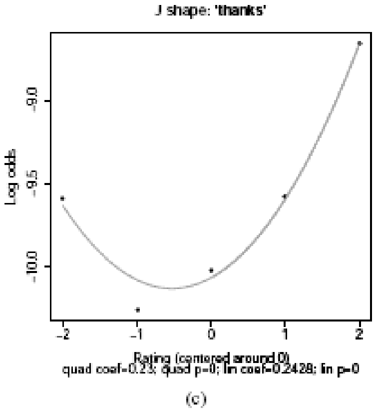
- Treat the reviews as collections of individual words.
- Classify reviews according to positive or negative words.
- Could use word lists prepared by humans, but machine learning based on a portion of the corpus (training set) is preferable.
- Use star rankings for training and evaluation.
- Pang et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

Sentiment words

thanks

Sentiment words

thanks



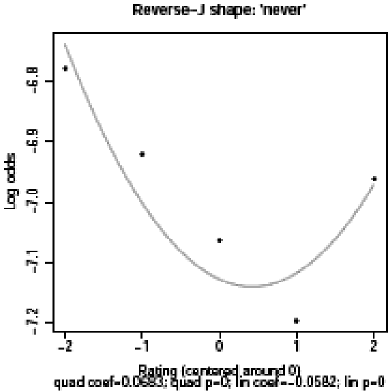
from Potts and Schwarz (2008)

Sentiment words

never

Sentiment words

never



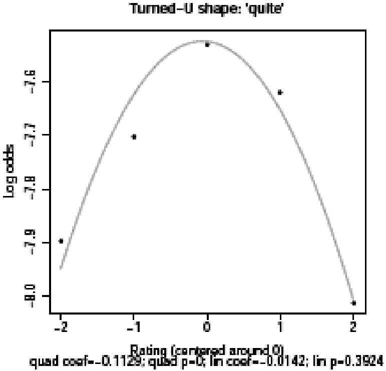
from Potts and Schwarz (2008)

Sentiment words

quite

Sentiment words

quite



(b)

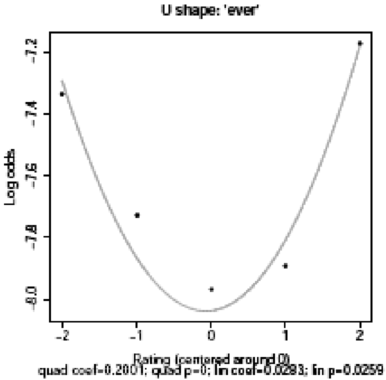
from Potts and Schwarz (2008)

Sentiment words: ever

ever

Sentiment words: ever

ever



(a)

from Potts and Schwarz (2008)

Some sources of errors for bag-of-words

- 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 well on arbitrary text is very hard.
- Ultimately the problem is AI-complete, but can we do well enough for NLP to be useful?

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 well 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.

MT

- Earliest attempted NLP application
- High quality only if the domain is restricted
- Utility greatly increased with increase in availability of electronic text
- Good applications for bad MT . . .
- Spoken language translation is viable for limited domains

Human translation?

Human translation?

I am not in the office at the moment. Please send any work to be translated.

Natural language interfaces and dialogue systems

All rely on a limited domain:

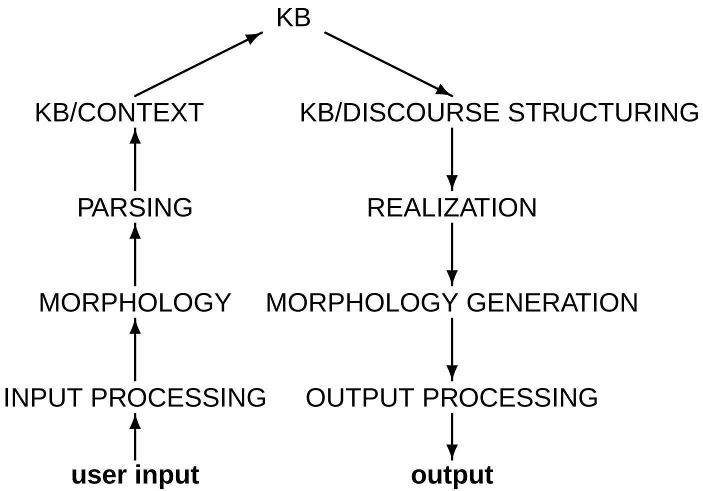
- LUNAR: classic example of a natural language interface to a database (NLID): 1970–1975
- SHRDLU: (text-based) dialogue system: 1973
- Current spoken dialogue systems

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



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

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., *antiantidisestablishmentarianism*
  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`

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)

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."*

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."*

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."*

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*

$$\varepsilon \rightarrow \mathrm{e} / \left\{ \begin{array}{c} \mathrm{s} \\ \mathrm{x} \\ \mathrm{z} \end{array} \right\} \hat{} \ \_ \ s$$

- map 'underlying' form to surface form
- mapping is left of the slash, context to the right
- notation:

_	position of mapping
ε	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*

$$\varepsilon \rightarrow \mathrm{e} / \left\{ \begin{array}{c} \mathrm{s} \\ \mathrm{x} \\ \mathrm{z} \end{array} \right\} \hat{}\ \_\ s$$

- map ‘underlying’ form to surface form
- mapping is left of the slash, context to the right
- notation:

_	position of mapping
ε	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*

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

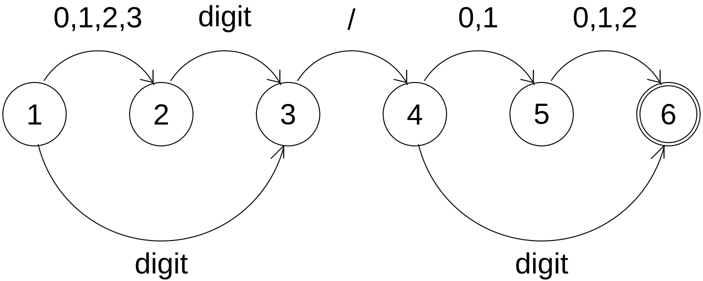
- map 'underlying' form to surface form
- mapping is left of the slash, context to the right
- notation:

_	position of mapping
ε	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

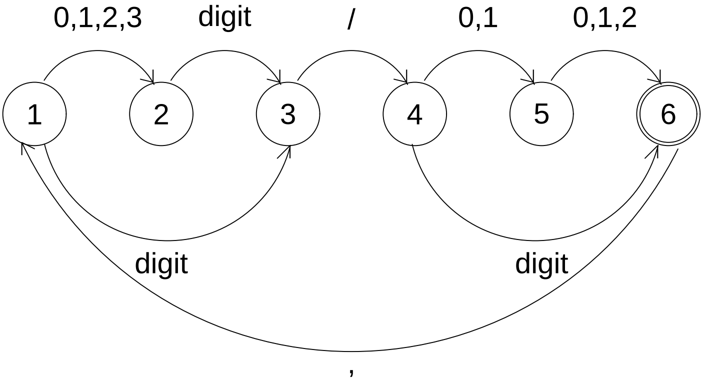
day/month pairs:



- 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

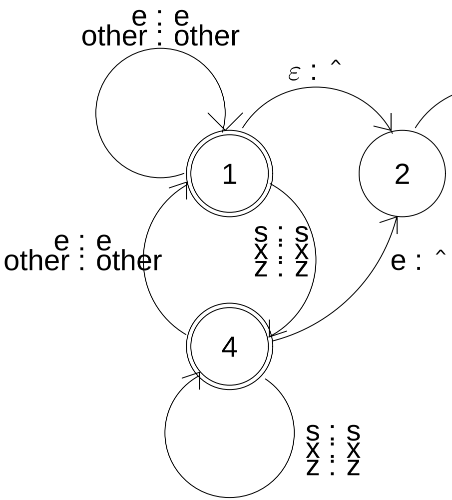
Recursive FSA

comma-separated list of day/month pairs:



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

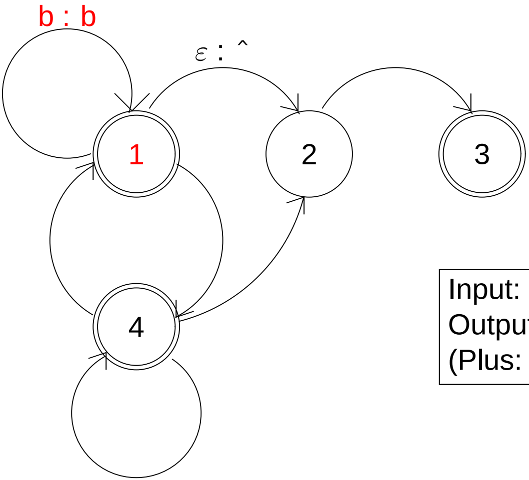
Finite state transducer



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

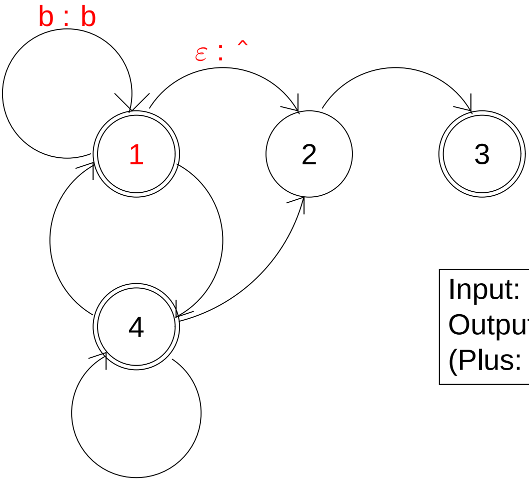
surface : underlying
c a k e s ↔ c a k e ^ s
b o x e s ↔ b o x ^ s

Analysing *b o x e s*



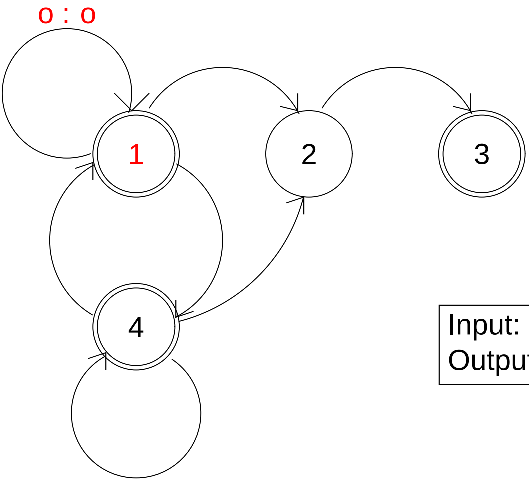
Input: b
Output: b
(Plus: ϵ . ^)

Analysing *b o x e s*



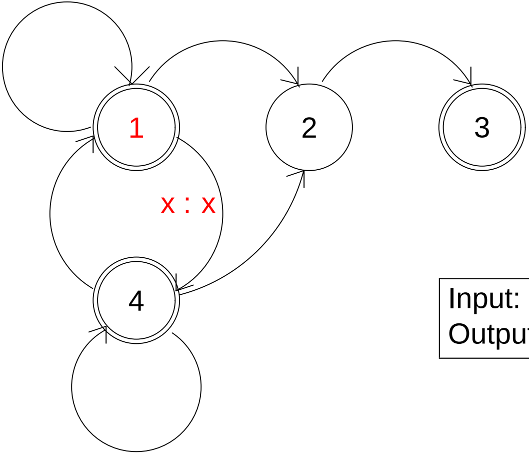
Input: b
Output: b
(Plus: ϵ . ^)

Analysing *b o x e s*



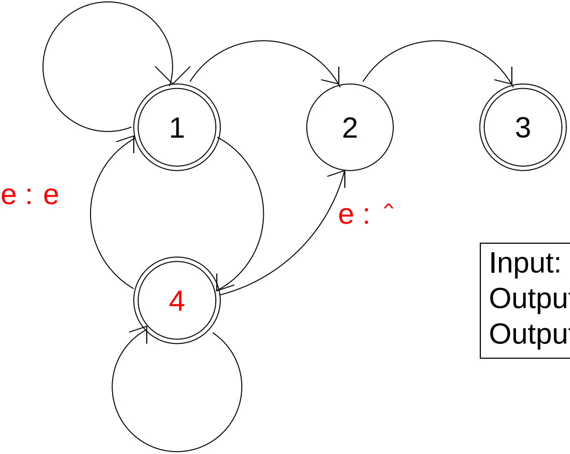
Input: b o
Output: b o

Analysing *b o x e s*



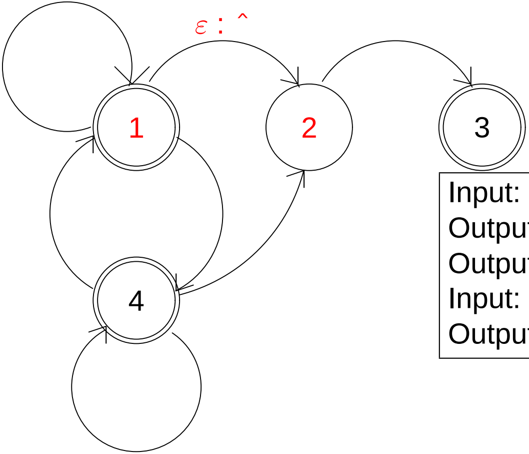
Input: b o x
Output: b o x

Analysing *b o x e s*



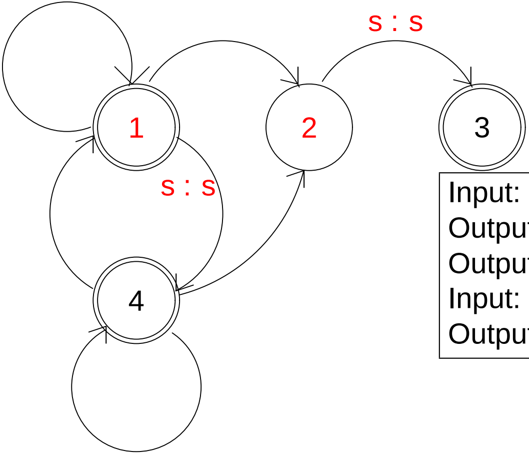
Input: b o x e
Output: b o x ^
Output: b o x e

Analysing *b o x e* ϵ *s*



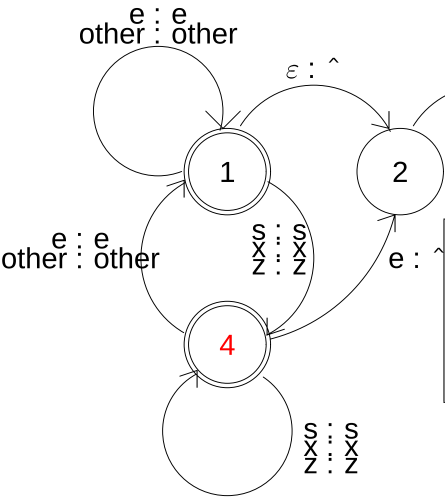
Input: b o x e
Output: b o x ^
Output: b o x e
Input: b o x e ϵ
Output: b o x e ^

Analysing *b o x e s*



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

Analysing *b o x e s*



Input: b o x e s
Accept output: b o x ^ s
Accept output: b o x e s
Input: b o x e ϵ s
Accept output: b o x e ^ s

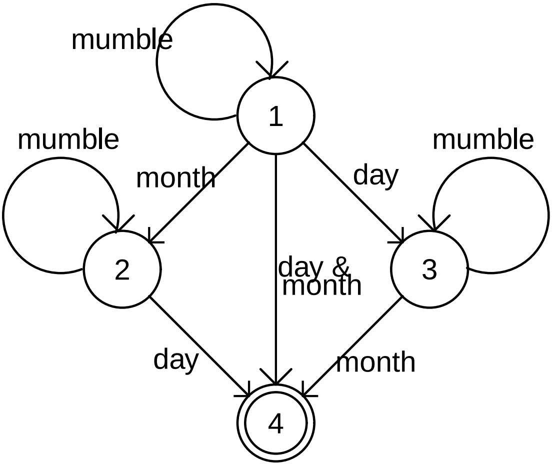
Using FSTs

- 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)

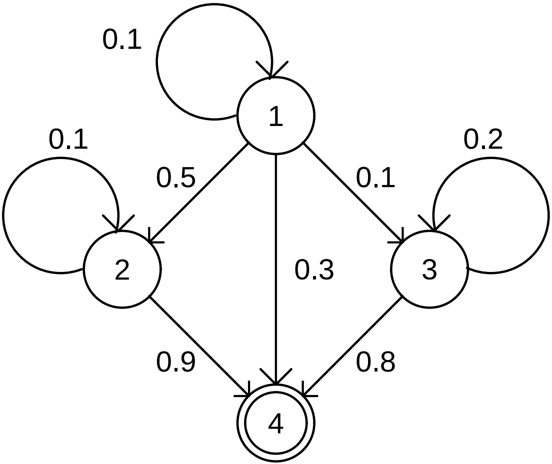
Some other uses of finite state techniques in NLP

- 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

Statistical techniques: NLP and linguistics

But it must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term. (Chomsky 1969)

Whenever I fire a linguist our system performance improves. (Jelinek, 1988?)

Statistical techniques: NLP and linguistics

But it must be recognized that the notion ‘probability of a sentence’ is an entirely useless one, under any known interpretation of this term. (Chomsky 1969)

Whenever I fire a linguist our system performance improves. (Jelinek, 1988?)

Prediction

Guess the missing words:

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

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 to disambiguate results from signal processing: e.g., using n-grams. (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 nth 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

⟨s⟩ good morning ⟨/s⟩ ⟨s⟩ good afternoon ⟨/s⟩ ⟨s⟩ good afternoon ⟨/s⟩ ⟨s⟩ it is very good ⟨/s⟩ ⟨s⟩ it is good ⟨/s⟩

sequence	count	bigram probability
⟨s⟩	5	
⟨s⟩ good	3	.6
⟨s⟩ it	2	.4
good	5	
good morning	1	.2
good afternoon	2	.4
good ⟨/s⟩	2	.4
⟨/s⟩	5	
⟨/s⟩ ⟨s⟩	4	1

Sentence probabilities

Probability of $\langle$s$\rangle$ it is good afternoon $\langle$/s$\rangle$ is estimated as:
$P(\text{it}|\langle\text{s}\rangle)P(\text{is}|\text{it})P(\text{good}|\text{is})P(\text{afternoon}|\text{good})P(\langle\text{/s}\rangle|\text{afternoon})$
$= .4 \times 1 \times .5 \times .4 \times 1 = .08$
Problems because of sparse data (cf Chomsky comment):

- smoothing: distribute ‘extra’ probability between rare and unseen events
- backoff: approximate unseen probabilities by a more general probability, e.g. unigrams

Practical application

- Word prediction: guess the word from initial letters. User confirms each word, so we predict on the basis of individual bigrams consistent with letters.
- Speech recognition: given an input which is a lattice of possible words, we find the sequence with maximum likelihood.
  Implemented efficiently using dynamic programming (Viterbi algorithm).

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:

NN1	singular noun	NN2	plural noun
PNP	personal pronoun	VM0	modal auxiliary verb
VVB	base form of verb	VVI	infinitive form of verb

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

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: fast, so applicable to very large corpora.
- Some linguistic research and lexicography: e.g., how often is *tango* used as a verb? *dog*?
- 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.

Note: tags are more fine-grained than conventional part of speech. Different possible tagsets.

Stochastic part of speech tagging using Hidden Markov Models (HMM)

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?

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
```

sequence	count	bigram probability
NN2	4	
NN2 PRP	1	0.25
NN2 PUN	2	0.5
NN2 VVB	1	0.25

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
```

sequence	count	bigram probability
NN2	4	
NN2 PRP	1	0.25
NN2 PUN	2	0.5
NN2 VVB	1	0.25

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
```

sequence	count	bigram probability
NN2	4	
NN2 PRP	1	0.25
NN2 PUN	2	0.5
NN2 VVB	1	0.25

Also lexicon: fish NN2 VVB

Assigning probabilities

Our estimate of the sequence of n tags is the sequence of n tags with the maximum probability, given the sequence of n words:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

By Bayes theorem:

$$P(t_1^n | w_1^n) = \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

We're tagging a particular sequence of words so $P(w_1^n)$ is constant, giving:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n)$$

Assigning probabilities, continued

Bigram assumption: probability of a tag depends on the previous tag, hence approximate by the product of bigrams:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1})$$

Probability of the word estimated on the basis of its own tag alone:

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^{n} P(w_i | t_i)$$

Hence:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

Example

Tagging: *they fish*
Assume PNP is the only tag for *they*, and that *fish* could be NN2 or VVB.
Then the estimate for PNP NN2 will be:

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

and for PNP VVB:

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

Assigning probabilities, more details

- Maximise the overall tag sequence probability — e.g., use Viterbi.
- 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 on normal corpora (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: Context-free grammars and parsing

- Generative grammar
- Simple context free grammars
- Simple chart parsing with CFGs
- More advanced chart parsing
- Formalism power requirements

Parsing

Syntactic structure in analysis:

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

Lecture 4: Context-free grammars and parsing

Generative grammar
Simple context free grammars
Simple chart parsing with CFGs
More advanced chart parsing
Formalism power requirements

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 -> NP VP
   V -> fish
   ```
4. a start symbol, conventionally S, which is a non-terminal.

Exclude empty productions, NOT e.g.:

NP -> ϵ

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)))

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))))

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 rivers)
                  (PP (P in) (NP December)))))
```

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

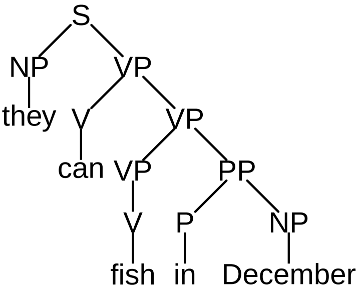
Structural ambiguity without lexical ambiguity

they fish in rivers in December

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

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

Parse trees




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

Chart parsing

A dynamic programming algorithm (memoisation):

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	l	r	ma	dtrs
5	2	3	V	(fish)
6	2	3	VP	(5)
7	1	3	VP	(3 6)

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*,*from*,*to*, *category*,*dtrs*]

For each *rule* $lhs \rightarrow cat_1 \ldots cat_{n-1}, category$

Find sets of contiguous edges

$[id_1, from_1, to_1, cat_1, dtrs_1] \ldots$

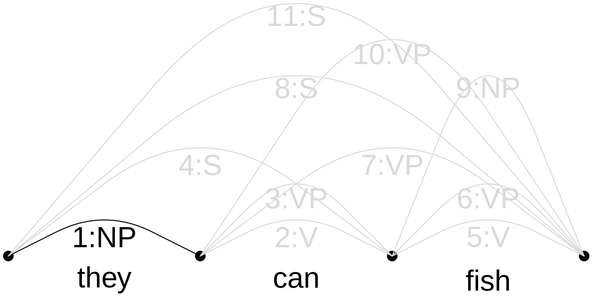
$[id_{n-1}, from_{n-1}, from, cat_{n-1}, dtrs_{n-1}]$

(such that $to_1 = from_2$ etc)

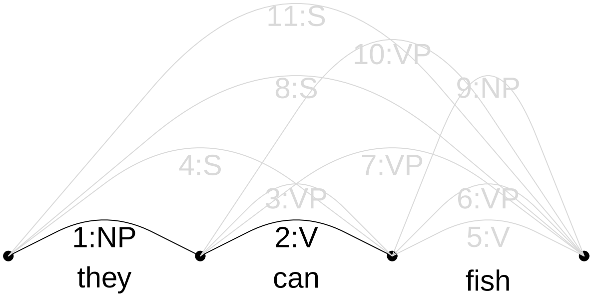
For each set of edges,

Add new edge $from_1$, *to*, *lhs*, $(id_1 \ldots id)$

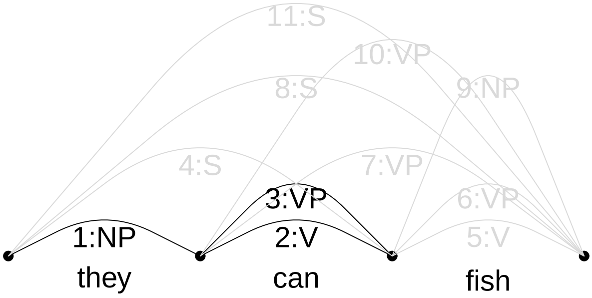
Bottom up parsing: edges



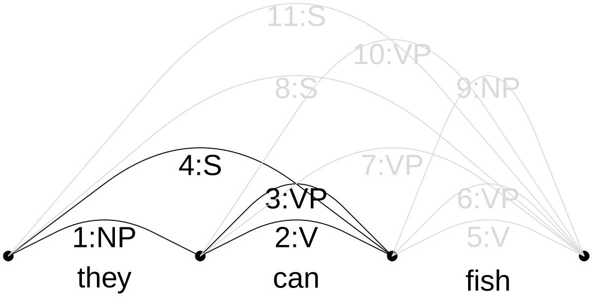
Bottom up parsing: edges



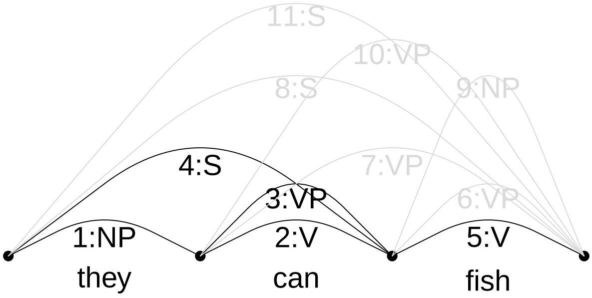
Bottom up parsing: edges



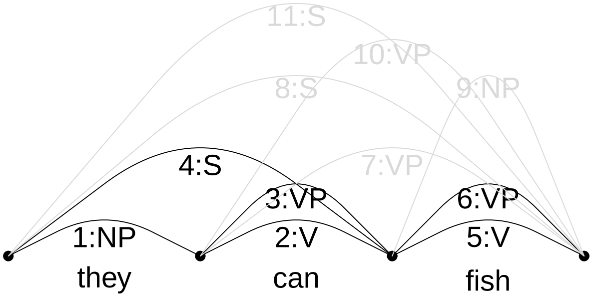
Bottom up parsing: edges



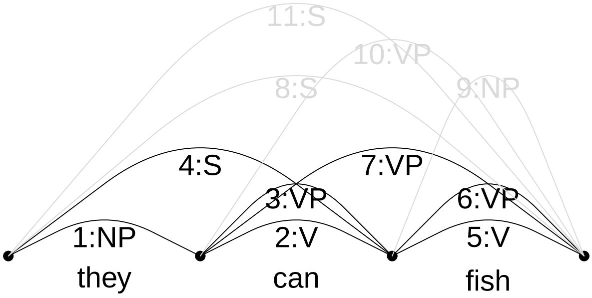
Bottom up parsing: edges



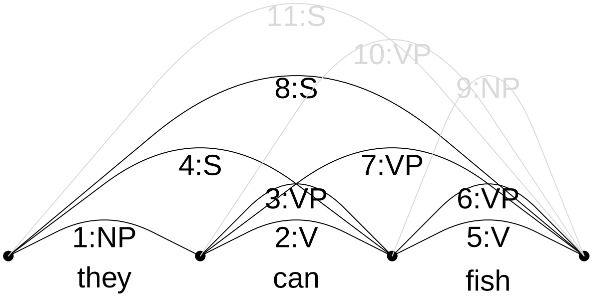
Bottom up parsing: edges



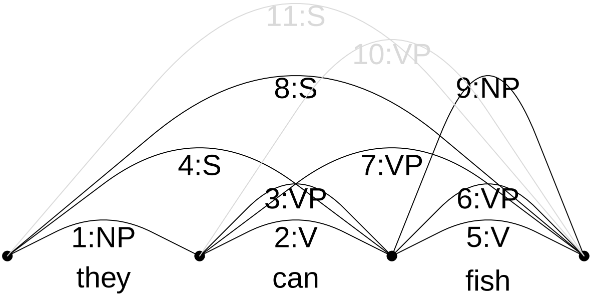
Bottom up parsing: edges



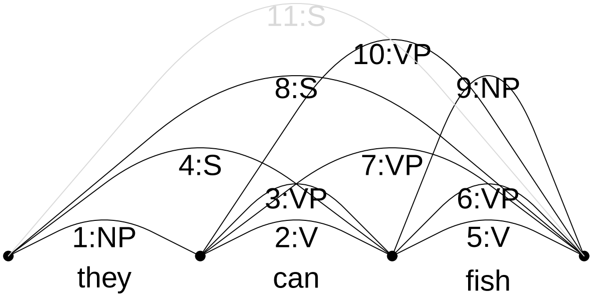
Bottom up parsing: edges



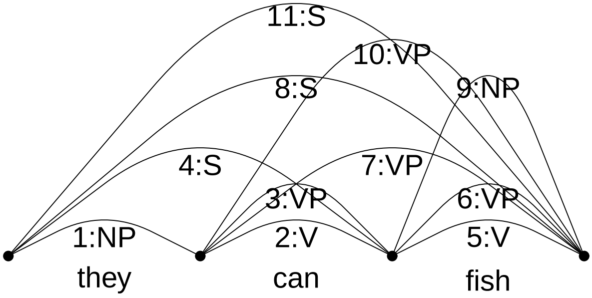
Bottom up parsing: edges



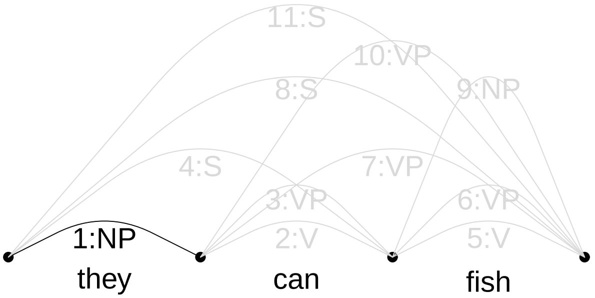
Bottom up parsing: edges



Bottom up parsing: edges

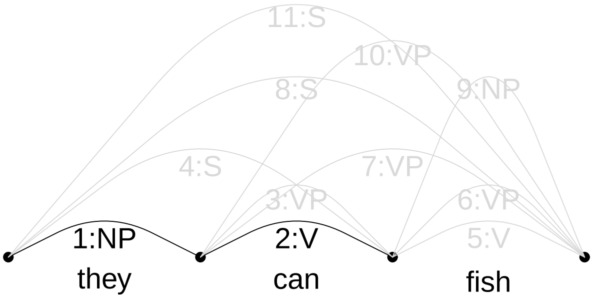


Parse construction



word $=$ they, categories $=$ {NP}
Add new edge 0, 1, NP, (they)
Matching grammar rules: {VP→V NP, PP→P NP}
No matching edges corresponding to V or P

Parse construction



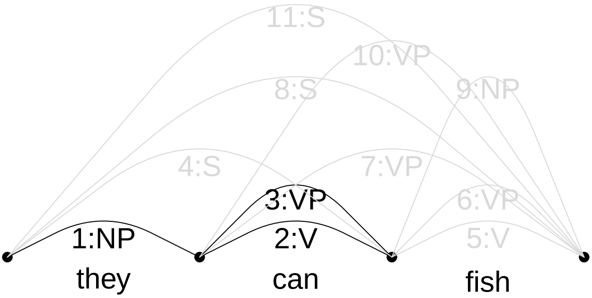
word = can, categories = {V}

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

Matching grammar rules: {VP→V}

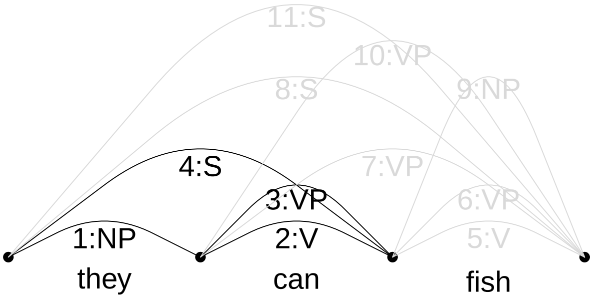
recurse on edges {(2)}

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



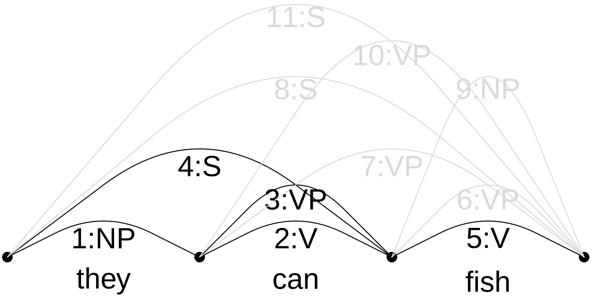
Add new edge 0, 2, S, (1, 3)

No matching grammar rules for S

Matching grammar rules: {S→NP VP, VP→V VP}

No edges for V VP

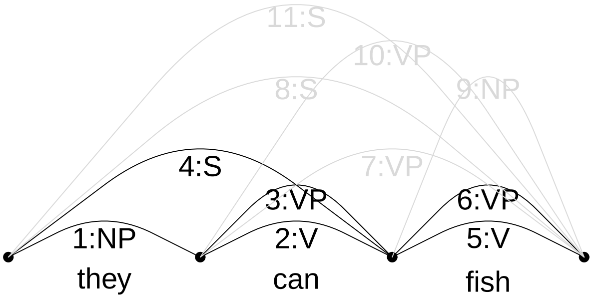
Parse construction



word $=$ fish, categories $=$ {V, NP}
Add new edge 2, 3, V, (fish)
Matching grammar rules: {VP$\rightarrow$V}
recurse on edges {(5)}

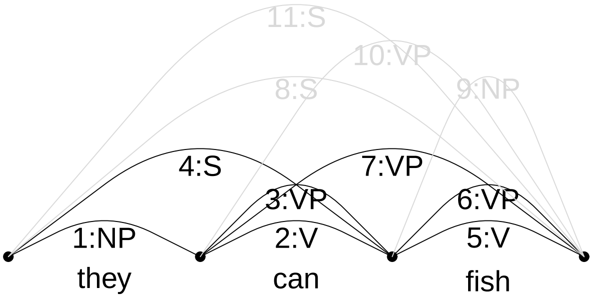
NB: fish as V

Parse construction



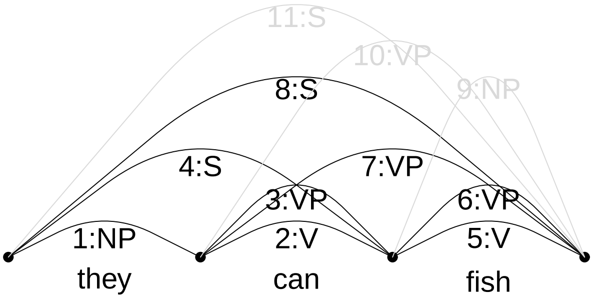
Add new edge 2, 3, VP, (5)
Matching grammar rules: {S $\rightarrow$NP VP, VP $\rightarrow$V VP}
No edges match NP
recurse on edges for V VP: {(2,6)}

Parse construction



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

Parse construction



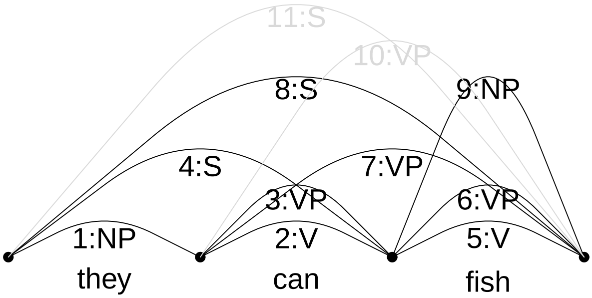
Add new edge 0, 3, S, (1, 7)

No matching grammar rules for S

Matching grammar rules: {S→NP VP, VP →V VP}

No edges matching V

Parse construction



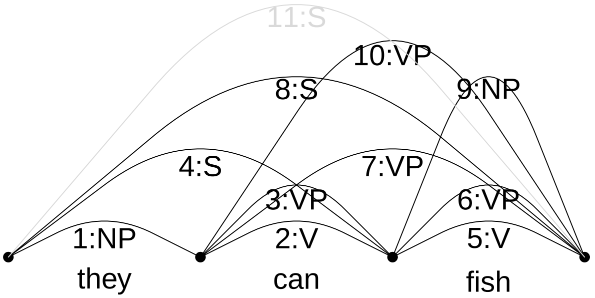
Add new edge 2, 3, NP, (fish)

NB: fish as NP

Matching grammar rules: {VP→V NP, PP→P NP}

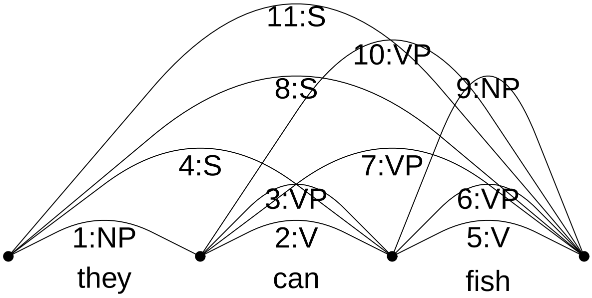
recurse on edges for V NP {(2,9)}

Parse construction



Add new edge 1, 3, VP, (2, 9)
Matching grammar rules: {S→NP VP, VP→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→NP VP, VP→V VP}
No edges corresponding to V VP
Matching grammar rules: {VP→V NP, PP→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* ∪ *dtrs*]

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

Packing example

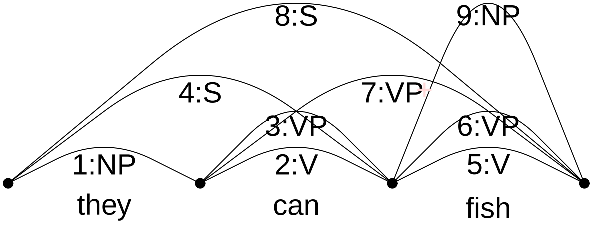
```
1    0    1   NP    {(they)}
2    1    2   V     {(can)}
3    1    2   VP    {(2)}
4    0    2   S     {(1 3)}
5    2    3   V     {(fish)}
6    2    3   VP    {(5)}
7    1    3   VP    {(2 6)}
8    0    3   S     {(1 7)}
9    2    3   NP    {(fish)}
```

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

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

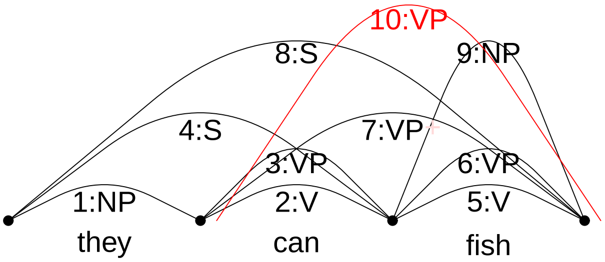
and we're done

Packing example



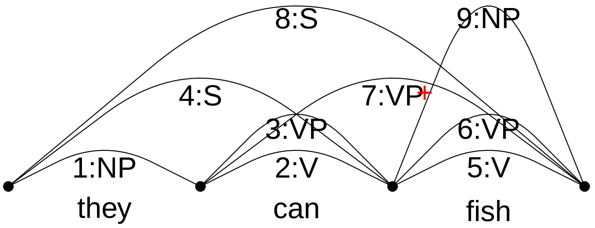
Both spanning results can now be extracted from edge 8.

Packing example



Both spanning results can now be extracted from edge 8.

Packing example



Both spanning results can now be extracted from edge 8.

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, limits on human memory / processing ability:

? the students the police the journalists criticised arrested complained

More importantly:

1. FSM grammars are extremely redundant
2. FSM grammars don't support composition of semantics

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, limits on human memory / processing ability:

? the students the police the journalists criticised arrested complained

More importantly:

1. FSM grammars are extremely redundant
2. FSM grammars don't support composition of semantics

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 -> NP-sg-nom VP-sg          NP-sg-nom -> he
S -> NP-pl-nom VP-pl          NP-sg-acc -> him
VP-sg -> V-sg NP-sg-acc       NP-sg-nom -> fish
VP-sg -> V-sg NP-pl-acc       NP-pl-nom -> fish
VP-pl -> V-pl NP-sg-acc       NP-sg-acc -> fish
VP-pl -> V-pl NP-pl-acc       NP-pl-acc -> fish
```

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 -> NP-sg-nom VP-sg          NP-sg-nom -> he
S -> NP-pl-nom VP-pl          NP-sg-acc -> him
VP-sg -> V-sg NP-sg-acc       NP-sg-nom -> fish
VP-sg -> V-sg NP-pl-acc       NP-pl-nom -> fish
VP-pl -> V-pl NP-sg-acc       NP-sg-acc -> fish
VP-pl -> V-pl NP-pl-acc       NP-pl-acc -> fish
```

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

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.

Context-free grammar and language phenomena

- CFGs can encode long-distance dependencies
- Language phenomena that CFGs cannot model (without a bound) are unusual — probably none in English.
- BUT: CFG modelling for English or another NL could be trillions of rules
- Enriched formalisms: CFG equivalent or greater power
- Does CFGness matter?
- Human processing vs linguistic generalisations. Human generalisations?

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: Constraint-based grammars

- From lecture 4
- Beyond simple CFGs
- Feature structures (informally)
- Encoding agreement
- Parsing with feature structures
- Feature stuctures more formally
- Encoding subcategorisation
- Interface to morphology

Outline of today's lecture

Lecture 5: Constraint-based grammars

- From lecture 4
- Beyond simple CFGs
- Feature structures (informally)
- Encoding agreement
- Parsing with feature structures
- Feature stuctures more formally
- Encoding subcategorisation
- Interface to morphology

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.

Context-free grammar and language phenomena

- CFGs can encode long-distance dependencies
- Language phenomena that CFGs cannot model (without a bound) are unusual — probably none in English.
- BUT: CFG modelling for English or another NL could be trillions of rules
- Enriched formalisms: CFG equivalent or greater power
- Does CFGness matter?
- Human processing vs linguistic generalisations. Human generalisations?

Constraint-based grammar (feature structures)

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

- Feature structure formalisms give good linguistic accounts for many languages
- Reasonably computationally tractable
- Bidirectional (parse and generate)
- Used in LFG and HPSG formalisms

Can also think of CFGs as constraints on trees.

Expanded CFG (from last time)

```
S -> NP-sg-nom VP-sg          NP-sg-nom -> he
S -> NP-pl-nom VP-pl          NP-sg-acc -> him
VP-sg -> V-sg NP-sg-acc       NP-sg-nom -> fish
VP-sg -> V-sg NP-pl-acc       NP-pl-nom -> fish
VP-pl -> V-pl NP-sg-acc       NP-sg-acc -> fish
VP-pl -> V-pl NP-pl-acc       NP-pl-acc -> fish
```

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**

can (n) $\begin{bmatrix} \text{CASE} & [\,] \\ \text{AGR} & \textbf{sg} \end{bmatrix}$ fish (n) $\begin{bmatrix} \text{CASE} & [\,] \\ \text{AGR} & [\,] \end{bmatrix}$

she $\begin{bmatrix} \text{CASE} & \textbf{nom} \\ \text{AGR} & \textbf{sg} \end{bmatrix}$ them $\begin{bmatrix} \text{CASE} & \textbf{acc} \\ \text{AGR} & \textbf{pl} \end{bmatrix}$

Feature structures

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

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.

Simple unification examples

1. $\begin{bmatrix} \text{CASE} & [\,] \\ \text{AGR} & \textbf{sg} \end{bmatrix} \sqcap \begin{bmatrix} \text{CASE} & \textbf{nom} \\ \text{AGR} & [\,] \end{bmatrix} = \begin{bmatrix} \text{CASE} & \textbf{nom} \\ \text{AGR} & \textbf{sg} \end{bmatrix}$

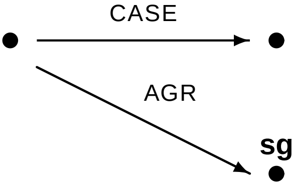
2. $\begin{bmatrix} \text{CASE} & [\,] \\ \text{AGR} & \textbf{sg} \end{bmatrix} \sqcap \begin{bmatrix} \text{AGR} & [\,] \end{bmatrix} = \begin{bmatrix} \text{CASE} & [\,] \\ \text{AGR} & \textbf{sg} \end{bmatrix}$

3. $\begin{bmatrix} \text{CASE} & [\,] \\ \text{AGR} & \textbf{sg} \end{bmatrix} \sqcap \begin{bmatrix} \text{CASE} & \textbf{nom} \\ \text{AGR} & \textbf{pl} \end{bmatrix} = \text{fail}$

Feature structures, continued

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

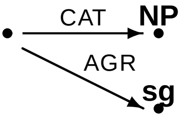
$$\begin{bmatrix} \text{CASE} & [\,] \\ \text{AGR} & \textbf{sg} \end{bmatrix}$$



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

Graphs and AVMs

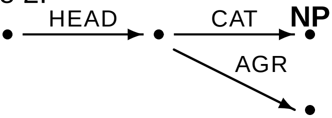
Example 1:



$$\begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \textbf{sg} \end{bmatrix}$$

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

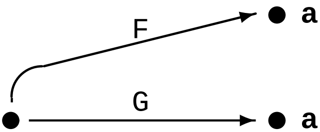
Example 2:



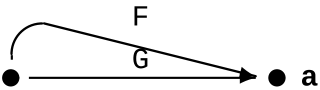
$$\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & [\,] \end{bmatrix} \end{bmatrix}$$

HEAD is complex-valued, AGR is unspecified.

Reentrancy



$$\begin{bmatrix} \mathrm{F} & \mathbf{a} \\ \mathrm{G} & \mathbf{a} \end{bmatrix}$$



$$\begin{bmatrix} \mathrm{F} & \boxed{0}\ \mathbf{a} \\ \mathrm{G} & \boxed{0} \end{bmatrix}$$

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
```

FS grammar fragment encoding agreement

subj-verb rule $\begin{bmatrix} \text{CAT} & \textbf{S} \\ \text{AGR} & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \boxed{1} \end{bmatrix}, \begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1} \end{bmatrix}$

verb-obj rule $\begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT} & \textbf{V} \\ \text{AGR} & \boxed{1} \end{bmatrix}, \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & [\,] \end{bmatrix}$

Root structure: $\begin{bmatrix} \text{CAT} & \textbf{S} \end{bmatrix}$

they $\begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \textbf{pl} \end{bmatrix}$ it $\begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \textbf{sg} \end{bmatrix}$ likes $\begin{bmatrix} \text{CAT} & \textbf{V} \\ \text{AGR} & \textbf{sg} \end{bmatrix}$

fish $\begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & [\,] \end{bmatrix}$ like $\begin{bmatrix} \text{CAT} & \textbf{V} \\ \text{AGR} & \textbf{pl} \end{bmatrix}$

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:

$$\begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \textbf{pl} \end{bmatrix}$$

- 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

But what does the coindexation of parts of the rule mean? Treat rule as a FS: e.g., rule features MOTHER, DTR1, DTR2 . . . DTRN.

informally: $\begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \rightarrow \begin{bmatrix} \text{CAT} & \textbf{V} \\ \text{AGR} & \boxed{1} \end{bmatrix}, \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & [\,] \end{bmatrix}$

actually: $\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR1} & \begin{bmatrix} \text{CAT} & \textbf{V} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR2} & \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & [\,] \end{bmatrix} \end{bmatrix}$

Verb-obj rule application

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

$$\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1}\ \textbf{pl} \end{bmatrix} \\ \text{DTR1} & \begin{bmatrix} \text{CAT} & \textbf{V} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR2} & \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & [\ ] \end{bmatrix} \end{bmatrix}$$

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

$$\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1}\ \textbf{pl} \end{bmatrix} \\ \text{DTR1} & \begin{bmatrix} \text{CAT} & \textbf{V} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR2} & \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \textbf{sg} \end{bmatrix} \end{bmatrix}$$

Subject-verb rule application 1

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

$\begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \textbf{pl} \end{bmatrix}$ unified with the DTR2 of: $\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{CAT} & \textbf{S} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR1} & \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR2} & \begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$

Gives:

$$\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{CAT} & \textbf{S} \\ \text{AGR} & \boxed{1}\ \textbf{pl} \end{bmatrix} \\ \text{DTR1} & \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR2} & \begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Subject rule application 2

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

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

$$\begin{bmatrix} \text{MOTHER} & \begin{bmatrix} \text{CAT} & \textbf{S} \\ \text{AGR} & \boxed{1}\ \textbf{pl} \end{bmatrix} \\ \text{DTR1} & \begin{bmatrix} \text{CAT} & \textbf{NP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \\ \text{DTR2} & \begin{bmatrix} \text{CAT} & \textbf{VP} \\ \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Final structure unifies with the root structure: $\begin{bmatrix} \text{CAT} & \textbf{S} \end{bmatrix}$

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: $\begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{OBJ} & \textbf{filled} \\ \text{SUBJ} & \boxed{3} \end{bmatrix} \to \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{OBJ} & \boxed{2} \\ \text{SUBJ} & \boxed{3} \end{bmatrix}, \boxed{2} \begin{bmatrix} \text{OBJ} & \textbf{filled} \end{bmatrix}$

can (transitive verb): $\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{verb} \\ \text{AGR} & \textbf{pl} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{noun} \end{bmatrix} \\ \text{OBJ} & \textbf{filled} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{noun} \end{bmatrix} \end{bmatrix} \end{bmatrix}$

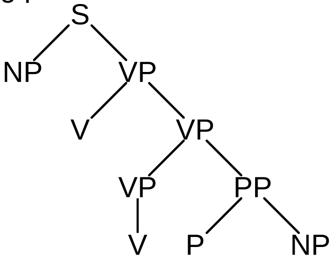
Grammar with subcategorisation (abbrev for slides)

Verb-obj rule: $\begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \boxed{3} \end{bmatrix} \rightarrow \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{OBJ} & \boxed{2} \\ \text{SUBJ} & \boxed{3} \end{bmatrix}, \boxed{2} \begin{bmatrix} \text{OBJ} & \textbf{fld} \end{bmatrix}$

can (transitive verb): $\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{v} \\ \text{AGR} & \textbf{pl} \end{bmatrix} \\ \text{OBJ} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{n} \end{bmatrix} \\ \text{OBJ} & \textbf{fld} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{n} \end{bmatrix} \end{bmatrix} \end{bmatrix}$

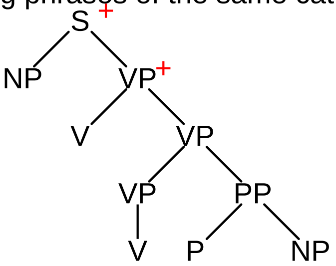
Concepts for subcategorisation

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



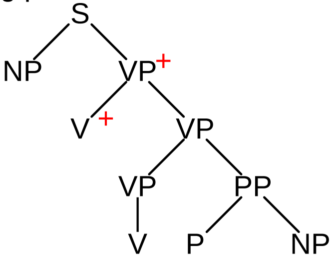
Concepts for subcategorisation

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



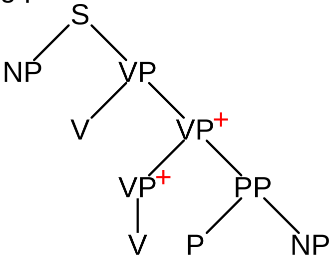
Concepts for subcategorisation

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



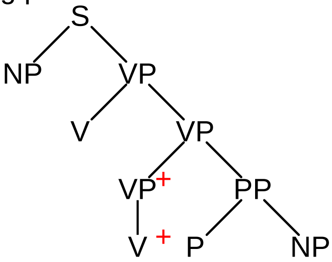
Concepts for subcategorisation

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



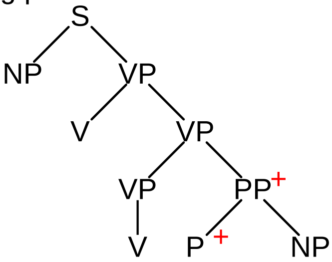
Concepts for subcategorisation

- 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



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:
$$\begin{bmatrix} \text{HEAD} & \begin{bmatrix}\text{CAT} & \textbf{v}\\ \text{AGR} & \textbf{pl}\end{bmatrix}\\ \text{OBJ} & \textbf{fld}\\ \text{SUBJ} & \begin{bmatrix}\text{HEAD} & \begin{bmatrix}\text{CAT} & \textbf{n}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

subject-verb rule:

$$\begin{bmatrix} \text{HEAD} & \boxed{1}\\ \text{OBJ} & \textbf{fld}\\ \text{SUBJ} & \textbf{fld}\end{bmatrix} \rightarrow \boxed{2}\begin{bmatrix} \text{HEAD} & \begin{bmatrix}\text{AGR} & \boxed{3}\end{bmatrix}\\ \text{OBJ} & \textbf{fld}\\ \text{SUBJ} & \textbf{fld}\end{bmatrix}, \begin{bmatrix} \text{HEAD} & \boxed{1}\begin{bmatrix}\text{AGR} & \boxed{3}\end{bmatrix}\\ \text{OBJ} & \textbf{fld}\\ \text{SUBJ} & \boxed{2}\end{bmatrix}$$

unification with second dtr position gives:

$$\begin{bmatrix} \text{HEAD} & \boxed{1}\begin{bmatrix}\text{CAT} & \textbf{v}\\ \text{AGR} & \boxed{3}\ \textbf{pl}\end{bmatrix}\\ \text{OBJ} & \textbf{fld}\\ \text{SUBJ} & \textbf{fld}\end{bmatrix} \rightarrow \boxed{2}\begin{bmatrix} \text{HEAD} & \begin{bmatrix}\text{CAT} & \textbf{n}\\ \text{AGR} & \boxed{3}\end{bmatrix}\\ \text{OBJ} & \textbf{fld}\\ \text{SUBJ} & \textbf{fld}\end{bmatrix}, \begin{bmatrix} \text{HEAD} & \boxed{1}\\ \text{OBJ} & \textbf{fld}\\ \text{SUBJ} & \boxed{2}\end{bmatrix}$$

Lexical entry for *they*: $\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{n} \\ \text{AGR} & \textbf{pl} \end{bmatrix} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \textbf{fld} \end{bmatrix}$

unify this with first dtr position:

$$\begin{bmatrix} \text{HEAD} & \boxed{1} \begin{bmatrix} \text{CAT} & \textbf{v} \\ \text{AGR} & \boxed{3}\ \textbf{pl} \end{bmatrix} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \textbf{fld} \end{bmatrix} \rightarrow \boxed{2} \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{n} \\ \text{AGR} & \boxed{3} \end{bmatrix} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \textbf{fld} \end{bmatrix}, \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \boxed{2} \end{bmatrix}$$

Root is: $\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{v} \end{bmatrix} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \textbf{fld} \end{bmatrix}$

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

$$\begin{bmatrix} \text{HEAD} & \begin{bmatrix}\text{CAT} & \textbf{v}\\ \text{AGR} & \textbf{pl}\end{bmatrix} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \begin{bmatrix}\text{HEAD} & \begin{bmatrix}\text{CAT} & \textbf{n}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Interface to morphology: inflectional affixes as FSs

$$s \quad \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{n} \\ \text{AGR} & \textbf{pl} \end{bmatrix} \end{bmatrix}$$

$$\text{if stem is:} \quad \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{CAT} & \textbf{n} \\ \text{AGR} & [\,] \end{bmatrix} \\ \text{OBJ} & \textbf{fld} \\ \text{SUBJ} & \textbf{fld} \end{bmatrix}$$

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.