Outline of today’s lecture

Lecture 2: Morphology and finite state techniques

- A brief introduction to morphology
- Using morphology in NLP
- Aspects of morphological processing
- Finite state techniques
- More applications for finite state techniques

Stems and affixes

- morpheme: the minimal information carrying unit
- affix: morpheme which only occurs in conjunction with other morphemes
- words made up of stem (more than one for compounds) and zero or more affixes.
  e.g., *dog+s*, *book+shop+s*
- *slither*, *slide*, *slip* etc have somewhat similar meanings, but *sl-* not a morpheme.

Affixation

- suffix: *dog +s*, *truth +ful*
- prefix: *un+ wise* (derivational only)
- infix: *sang* (stem *sing*): not productive
  e.g., (maybe) *absobloodylutely*
- circumfix: not in English
  German *ge+kauf+t* (stem *kauf*, affix *ge-t*)

Productivity

productivity: whether affix applies generally, whether it applies to new words
sing, *sang*, *sung*
ring, *rang*, *rung*
BUT: *ping*, *pinged*, *pinged*
So this infixation pattern is not productive:
sing, *ring* are irregular

Productivity

productivity: whether affix applies generally, whether it applies to new words
sing, *sang*, *sung*
ring, *rang*, *rung*
BUT: *ping*, *pinged*, *pinged*
So this infixation pattern is not productive:
sing, *ring* are irregular

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: e.g., *escapee*, *textee*, *?dropee*, *?snoree*, **cricketee* (* and ?)
- 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* or *short -s*
unionised could be *union -ise -ed* or *un- ion -ise -ed*
Bracketing: *un- ion -ise -ed*

- *un- ion* is not a possible form, so not *((un- ion) -ise) -ed*
- *un-* is ambiguous:
  - with verbs: means 'reversal' (e.g., *untie*)
  - with adjectives: means 'not' (e.g., *unwise, unsurprised*)
- therefore *(un- ((ion -ise) -ed))*

Using morphological processing in NLP

- compiling a full-form lexicon
- stemming for IR (not linguistic stem)
- lemmatization (often inflections only): finding stems and affixes as a precursor to parsing
  morphosyntax: interaction between morphology and syntax
- generation
  Morphological processing may be bidirectional: i.e., parsing and generation.

```
party + PLURAL <-> parties
sleep + PAST_VERB <-> slept
```

Spelling rules

- 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)

  morphophonology
- in English, description is independent of particular stems/affixes

e-insertion

e.g. *box*ˆ*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
ˆ	affix boundary — stem ˆ 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*

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

- 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

Lexical requirements for morphological processing

- affixes, plus the associated information conveyed by the affix

```
ed PAST_VERB
ed PSP_VERB
s PLURAL_NOUN
```

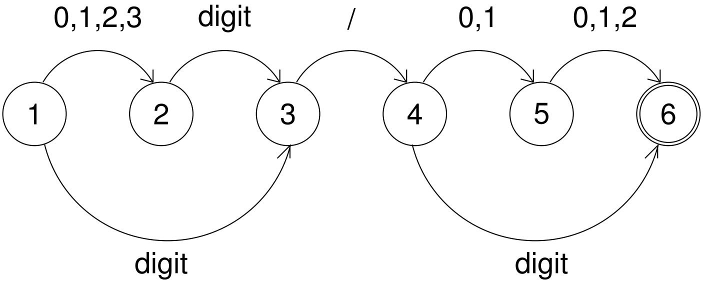
- irregular forms, with associated information similar to that for affixes

```
began PAST_VERB begin
begun PSP_VERB begin
```

- stems with syntactic categories (plus more)

Finite state automata for recognition

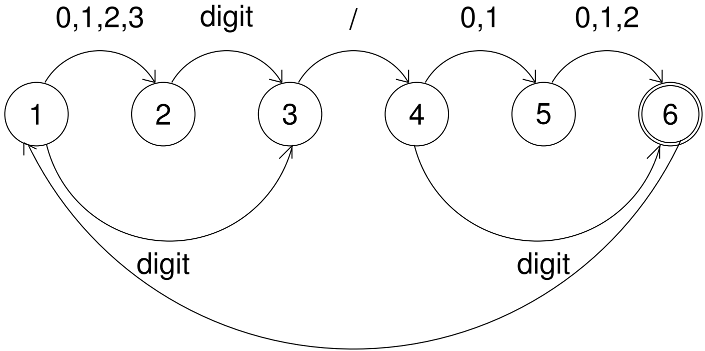
day/month pairs:



- 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

e-insertion

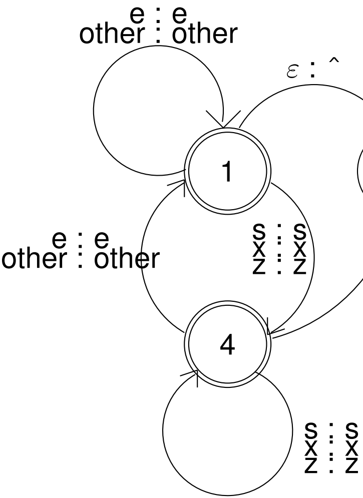
e.g. *box^s* to *boxes*

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

- 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

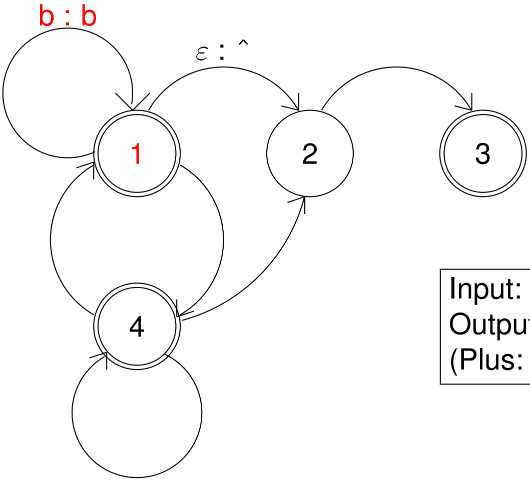
Finite state transducer



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

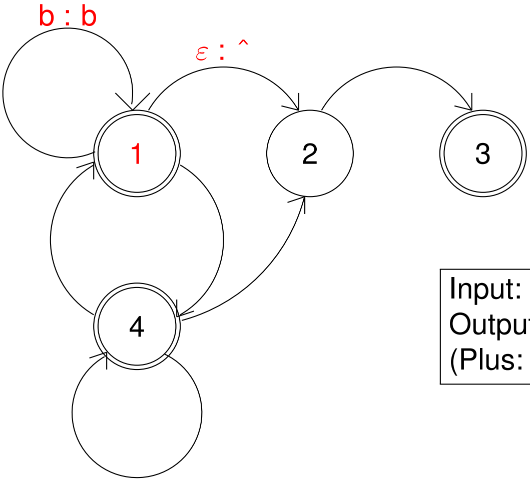
surface : underlying
cakes ↔ cake^s
boxes ↔ box^s

Analysing *b o x e s*



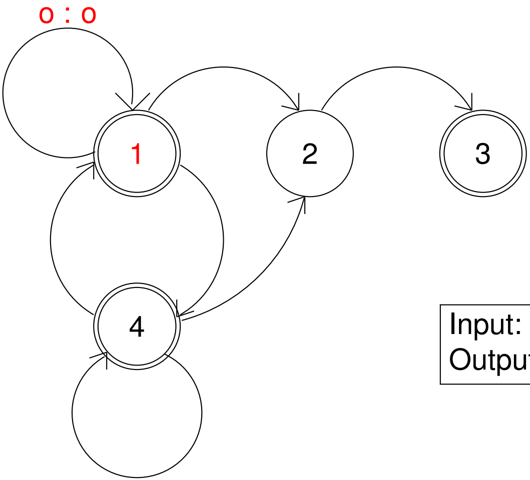
Input: b
Output: b
(Plus: ϵ . ^)

Analysing *b o x e s*



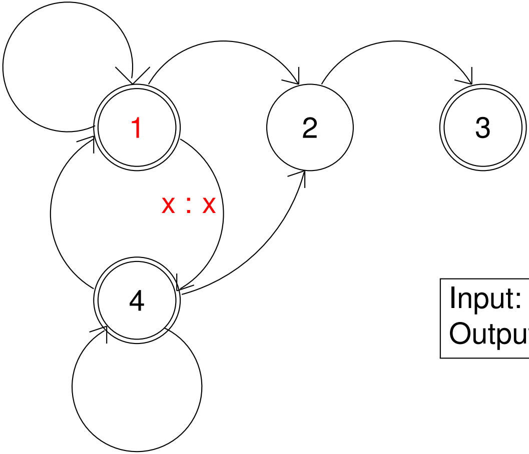
Input: b
Output: b
(Plus: ϵ . ^)

Analysing *b o x e s*



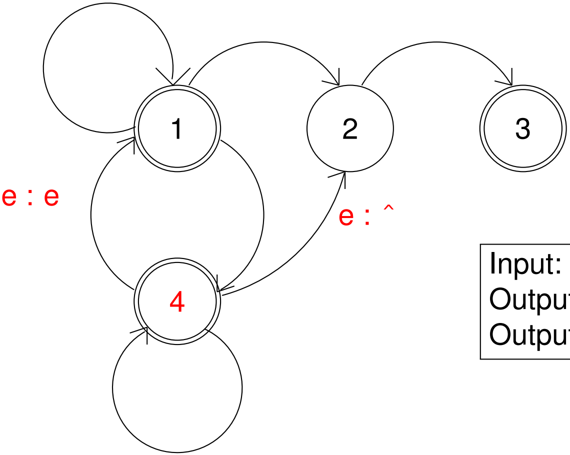
Input: b o
Output: b o

Analysing *b o x e s*



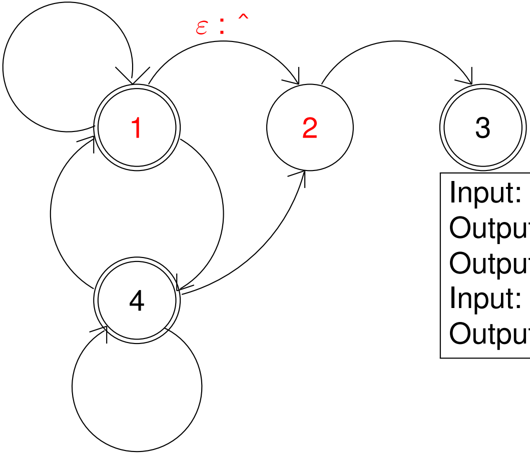
Input: b o x
Output: b o x

Analysing *b o x e s*



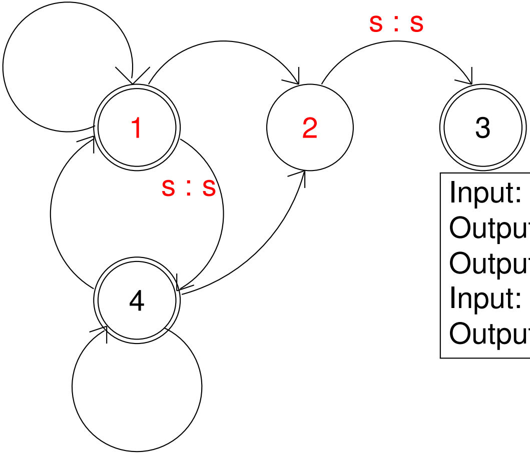
Input: b o x e
Output: b o x ^
Output: b o x e

Analysing *b o x e* ϵ *s*



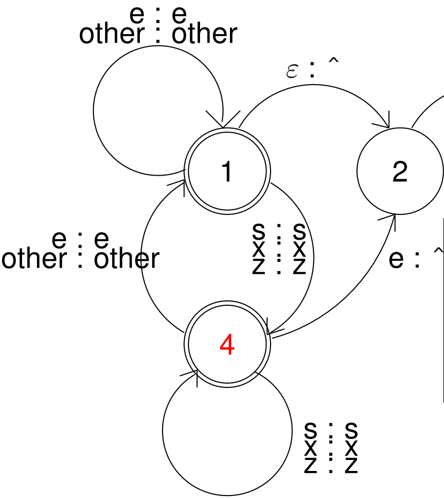
Input: b o x e
Output: b o x ^
Output: b o x e
Input: b o x e ϵ
Output: b o x e ^

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

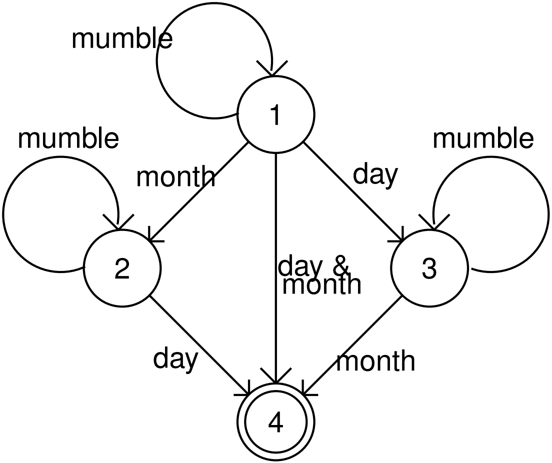
Using FSTs

- 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

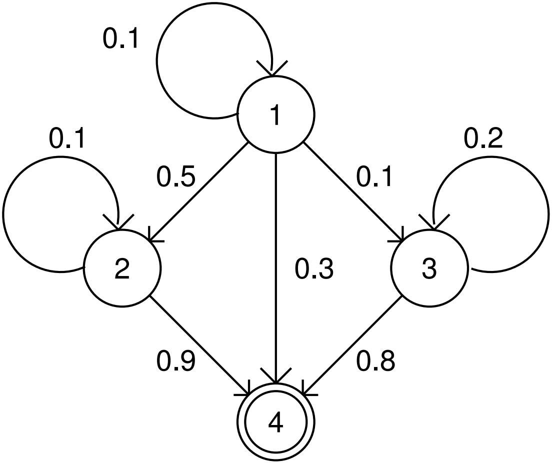
Some other uses of finite state techniques in NLP

- Grammars for simple spoken dialogue systems (directly written or compiled)
- Partial grammars for text preprocessing, tokenization, named entity recognition etc.
- 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



Concluding comments

- English is an outlier among the world's languages: very limited inflectional morphology.
- English inflectional morphology hasn't been a practical problem for NLP systems for decades.
- Limited need for probabilities, small number of possible morphological analyses for a word.
- Lots of other applications of finite-state techniques: fast, supported by toolkits, good initial approach for very limited systems.