Introduction to Human Language Technologies
3 - Morphology

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

2 de setembre de 2019

Outline

Outline

Motivation

There are lots of NLP tools and applications in which dealing with the morphology of the words is relevant, for instance:

- IR is based on the canonical forms of the words.

  'Normally, houses in the Pyrenees are made of stone.'
  'A typical pyrenean house has litle windows.'

- Spell checkers are based on checking whether words in a document are well-formed or not.

  'This could be an alterantive remedy'

- Syntactic parsing requires lexical information derived from morphological analysis

  'Children are very intelligent'
  'Children is very intelligent'

Outline

Definition of morphology

- Study of the structure of words
  - Phonology: word as a combination of phonemes
  - Orthography: word as a combination of graphemes
  - Morphology: word as a combination of morphemes
- Types of morphemes:
  - Stems: (e.g., 'work', 'of','mak'[e])
  - Affixes: always occur combined with other morphemes (e.g., -s",'in-','-able')
    - Prefixes: in + frequent
    - Suffixes: work + s
    - Infixes: [Arabic] ktb + CuCuC → kutub (books)
    - Circumfixes: en+light+en
- The resulting words can be classified into categories known as Part of Speech (POS): Noun, Verb, Adjective, Adverb, Preposition, ...

Outline

Types of morphology

- Concatenative morphology: builds words up by concatenating morphemes (prefixes, suffixes). The most productive in the Indo-European languages.
  - Inflectional morphology: *word* $\rightarrow$ *new forms of the word*

    Ex: work $\rightarrow$ worked
  - Derivational morphology: *word* $\rightarrow$ *new word*

    Ex: frequent $\rightarrow$ infrequent
  - Compositional morphology: *N word* $\rightarrow$ *new word*

    Ex: fire + man $\rightarrow$ fireman
- Non-concatenative morphology: builds words by other mechanism (infixes, circumfixes).
  - Ex: Root-Pattern morphology

    Ex: [Arabic] ktb + CaCaCa $\rightarrow$ kataba [en: he wrote]

Inflectional morphology

Inflectional morphemes provide morphological information depending on the POS and language of the input word

- Nouns (N):
  - Genre: [Spanish] niñ-o (M), niñ-a (F)
  - Number: [Italian] italian-o (SG), italian-i (PL)
  - Case: [German] die Rolle des Mann-es (Genitive)
- Verbs (V):
  - Tens: want-ed (PAST), will want (no morpho. mark for future)
  - Mode: [Spanish] com-er (indicative), com-ed (imperative)
  - Aspect: want-ed (perfective), I am waiting (no morpho mark for imperfective)
  - Voice: [Sweden] servera-s (PAS) [en: is served]
- Adjectives (A):
  - Genre: [Spanish] blanc-o (M), blanc-a (F) [en: white]
  - Number: [Spanish] blanco (SG), blanco-s (PL) [en: white]
  - Comparison: cheap-er, more similar (not for all adjectives)

Derivational morphology

Derivational morphemes can change the POS and the meaning of the word

- Adjectivization: $V \rightarrow A$ or $N \rightarrow A$

  Ex: react $\rightarrow$ react-ive, employ $\rightarrow$ employ-able
  medicine $\rightarrow$ medicin-al, use $\rightarrow$ use-ful

- Nominalization: $V \rightarrow N$ or $A \rightarrow N$

  Ex: watch $\rightarrow$ watch-er, react $\rightarrow$ react-ion
  useful $\rightarrow$ useful-ness

- Negativization:

  Ex: frequent $\rightarrow$ in-frequent, do $\rightarrow$ un-do

Outline

Goal of morphological analysis

- Morphological recognition

  Does word w belong to language L?

- Morphological parsing

  What is the morphological information related to word $w \in L$?

  Ex: *word POS+Gen+Num+Case+Tense+... LEMMA (stem)*
  men Noun+M+PL man

Resources required for morphological analysis

- Lists of regular (Reg) stems (ambiguities)

  EX: Reg_V: walk
  Reg_N: cat, fox, walk

- Lists of irregular (Irreg) stems (ambiguities)

  Ex: Irreg_pres_V: sing . . . Irreg_past_V: sang sing
  Irreg_sg_N: mouse . . . Irreg_pl_N: mice mouse

- List of suffixes and prefixes (dealing with concatenative morphology)

  Ex: Inflec: s suffix, ing suffix
  Deriv: able suffix, un prefix

- Morphotactics: general rules for combining morphemes

  Ex: Reg_N + s $\rightarrow$ PL
  Reg_V + ing $\rightarrow$ Present_Participle

- Spelling rules: orthographic rules for combining letters

  Ex: E-insertion: -(z,x,s,sh,ch)^s $\rightarrow$ -(z,x,s,sh,ch)es
  Consonant-doubling: -l^ing $\rightarrow$ -lling

Types of morphological processors

- Based on dictionaries: list of word forms [with their corresponding morphological information]

  Ex: (write VPrI write, writes VPrI3S write, wrote VPsI write, . . .)

  + efficiency
  + can be automatically generated/maintained from the resources
  + language with 'simple' morphology (e.g., English)
  - languages with complex morphology (e.g., German, Finish, ...)
- Based on finite state automata (FSAs)
  - languages with complex morphology
- Based on finite state trandducers (FSTs)

Outline

Finite state automata (FSA)

A FSA defines a function over words w of a regular language L.
$M_L : w \rightarrow \{true, false\}$

$M =< Q, \Sigma, q_0, F, \sigma >$

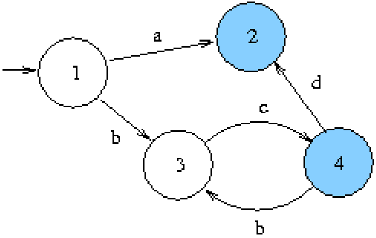
$Q = \{q_0, \ldots, q_n\}$ finite set of states

$\Sigma = \{s_0, \ldots, s_k\}$ finite set of simbols

$q_0 \in Q$ start state

$F \subset Q$ final states

$\sigma : Qx\Sigma \rightarrow [Q \vee 2^Q]$ deterministic $\vee$ non-det. transition function



$a\|(bc)+d\{0,1\}$
a
bc
bcd
bcbcd
...

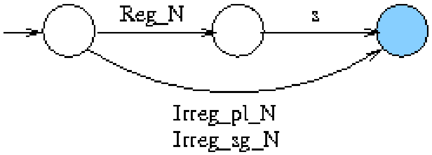
FSAs for morphological recognition

An FSA can be the union of different FSAs:

- FSAs generated from morphological rules
- FSAs generated from spelling rules
- FSAs generated from derivational rules
- FSAs generated from compositional rules

FSAs for morphological recognition

Example: FSA for English number nominal inflection

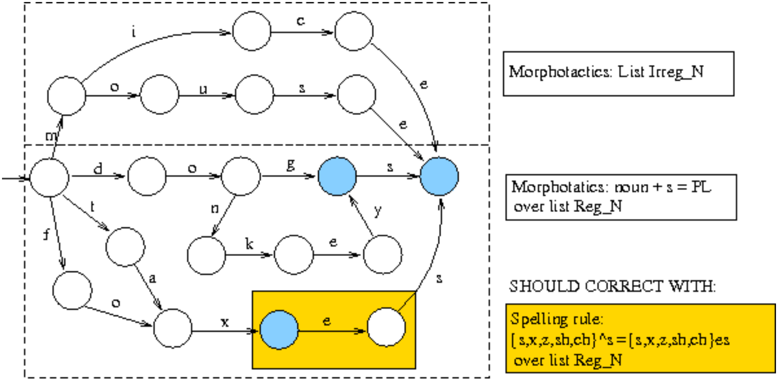


Examples of lists of stems

Reg_N	Irreg_sg_N	Irreg_pl_N
dog	mouse	mice
fox	foot	feet
tax		
donkey		

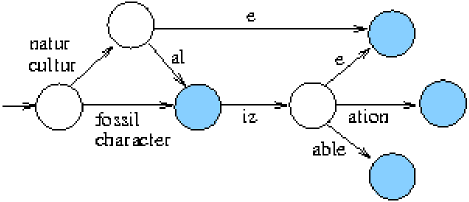
FSAs for morphological recognition

Example: FSA for English number nominal inflection



FSAs for morphological recognition

Example: FSA derived from derivational rules



Not so productive as inflectional rules: 'jail', 'window', ... ?

FSAs for morphological recognition

- FSAs can be useful for recognising words
- FSAs are not able to output a word analysis

Input word (surface form)	Output analysis (lexical form)
dog dogs	dog+N+SG dog+N+PL
(word form)	(lemma+Features)

- A more sophisticated technique is required: FSTs

Outline

Finite state transducers (FSTs)

A FST defines a relation between regular languages L_1 and L_2.

$T =< Q, \Sigma, \Delta, q_0, F, \sigma, \delta >$

$Q = \{q_0, \ldots, q_n\}$ finite set of states

$\Sigma = \{s_0, \ldots, s_k\}$ finite set of input simbols

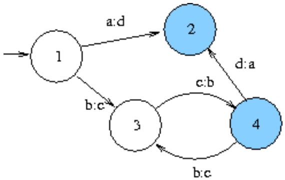
$\Delta = \{t_0, \ldots, t_m\}$ finite set of output simbols

$q_0 \in Q$ start state

$F \subset Q$ final states

$\sigma : Qx\Sigma \rightarrow 2^Q$ transition function

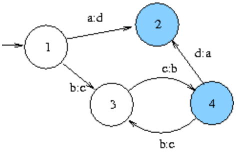
$\delta : Qx\Sigma \rightarrow \Delta$ output function



$a\|(bc)+d\{0,1\}$	$d\|(cb)+a\{0,1\}$
a	d
bc	cb
bcd	cba
bcbc	cbcb
bcbcd	cbcba
...	

Finite state transducers (FSTs)

- Invertion: $T : L_1 \to L_2 \Longrightarrow T^{-1} : L_2 \to L_1$



b:c $\Longrightarrow$ b $\to$ c $\Longrightarrow$ Ex: bcbc $\to$ cbcb

b:c $\Longrightarrow$ b $\leftarrow$ c $\Longrightarrow$ Ex: bcbc $\leftarrow$ cbcb

- Composition: $T_a : L_1 \to L_2 \wedge T_b : L_2 \to L_3 \Longrightarrow T_a \circ T_b : L_1 \to L_3$
- x:x $\equiv$ x
- Non-consumption symbol: $\epsilon \in \Sigma \cup \Delta$

FSTs for morphological analysis

We want a FST being a relation between

- Surface form: $L_1 = \{w | w \text{ is word form}\}$
- Lexical form: $L_2 = \{< l, F > | l \text{ is lemma} \wedge F \text{ are morphological features}\}$

So that we get a morphological parser

- Ex: dogs $\rightarrow$ dog+N+PL
  Ex: dog $\rightarrow$ dog+N+SG

Inverting that FST, we get a word forms generator

- Ex: dog+N+PL $\rightarrow$ dogs
  Ex: dog+N+SG $\rightarrow$ dog

FSTs for morphological analysis

Two-level processing:

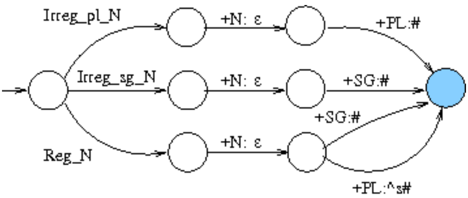
1. A FST that computes morphotactics, T_{lex}
   Ex: Reg_N^s → Reg_N+N+PL.
   Ex: dog^s → dog+N+PL, fox^s → fox+N+PL
2. FSTs each computing a spelling rule, T^i_{inter} (orthographic regularization)
   Ex: -{z,x,s,sh,ch}es → -{z,x,s,sh,ch}^s#

lexical level	fox +N +PL
T_{lex}	↑
intermediate level	fox^s#
$T^1_{inter}, \ldots, T^k_{inter}$	↑
surface level	foxes

FSTs for morphological analysis

1. T_{lex}: FST that computes morphotactics
   Example: FST for English number nominal inflection

T_{num_nouns}



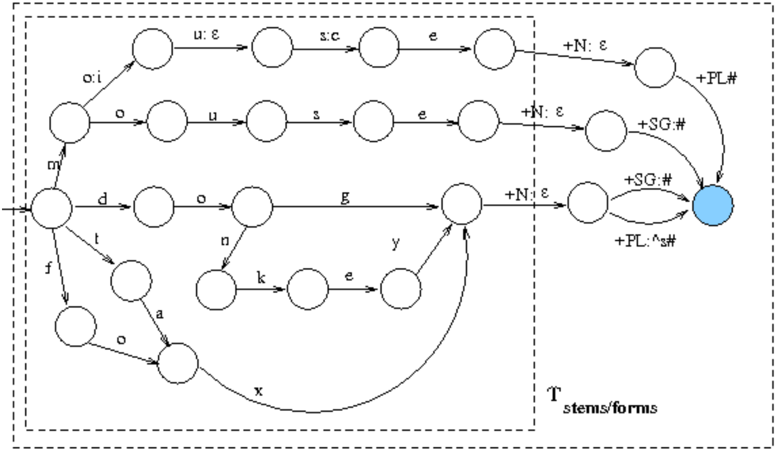
Examples of lists of stems/forms

Reg_N	Irreg_sg_N	Irreg_pl_N
dog	mouse	m o:i u:ϵ s:c e
fox	foot	f o:e o:e t
tax		
donkey		

FSTs for morphological analysis

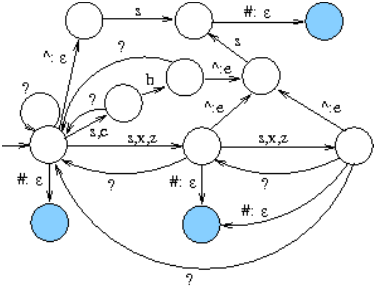
1. T_{lex}: FST that computes morphotactics
   Example: FST for English number nominal inflection

$T_{lex} = T_{stems/forms} \circ T_{num\_nouns}$



FSTs for morphological analysis

2. T^i_{inter}: FSTs that compute spelling rules
   Example: FST for E-insertion rule



'?': other symbol

foxes → fox^s#
bosses → boss^s#
flashes → flash^s#
. . .

FSTs for morphological analysis

2 T^i_{inter}: FST that computes spelling rules

Some other examples of spelling rules:

- **Consonant doubling**: two-syllable word stressed in the last one with ending CVC pattern double last consonant before *-ing*/*-ed*
  EX: control → controlling
- **E-deletion**: Silent *-e* removed before *-ing*/*-ed*
  EX: remove → removed
- **E-insertion**: *-e* added after ending *-s,-z,-x,-ch,-sh*, before *-s*
  EX: flash → flashes
- **Y-replacement**: *-y* changes to *-ie* before *-s* or to *-i* before *-ed*
  EX: cry → cries, cried
- **K-insertion**: verbs ending with *1-vowel+c* add *-k* before *-ed*
  EX: panic → panicked

Exercise

- Generate a FST for the inflection of verbs *sing* and *work*
- Add the inflection of verb *make* to the previous FST

Outline

Spell checkers

- **Goal**: given a piece of text, recognise the word forms that do not belong to the text language L
- **Possible approach**:

  FSA_L OR FST_L
  $S = Tokenizer(text)$ (sequence of forms)

  for each $x \in S$
  if $FSA_L(x)$ then print("x")
  else print("**x**")

Spell correctors

- **Goal**: given a word form, provide a list of possible correct forms.
- **Possible approach**:

$D = \{y_i : y_i \in L\}$ generated by applying FST_L
$S = Tokenizer(text)$ (sequence of forms)

for each $x \in S$
  if $x \in D$ then print(x)
  else
    $D' = \{y \in D : |length(x) - length(y)| \leqslant \gamma\}$
    $C = \emptyset$
    for each $y \in D'$
      $d = distance(x, y)$
      if ($d \leqslant \delta$) then
        $C = C + \{< y, d >\}$
    print_Nbest_candidates(C,N)

$\delta = 2$ and $\gamma = 2$ seem to be enough for standard text

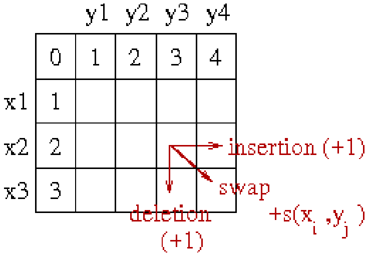
Spell correctors

- Edit distance: minimum number of insertions, deletions, swaps to achieve y from x
- Weighted edit distance: minimum cost of insertions, deletions, swaps to achieve y from x
  - Cost of insertion/deletion $= 1$
  - Cost of swap $= s(a, b)$: (typo - Manhattan distance in a keyboard)

  - Total cost $= d(x, y)$:
    - Compute cost matrix E, with dimension mXn (lengths of x and y) using dynamic programming
    - $d(x, y) = E(m, n)$

Spell correctors

Cost matrix computation



$$E(i,j) = min(Cost_{del}, Cost_{ins}, Cost_{swap})$$

$$\begin{cases} Cost_{del} = E(i-1,j)+1 \\ Cost_{ins} = E(i,j-1)+1 \\ Cost_{swap} = E(i-1,j-1)+s(x_i,y_j) \end{cases}$$

$s(x_i, y_j)$	a	b	c	d	e
a	0				
b	0.5	0			
c	0.3	0.3	0		
d	0.2	0.2	0.1	0	
e	0.3	0.4	0.2	0.1	0

$s(x_i, y_j)$ normalised to 10

Exercise

- Compute de weighted edit distance between ’dom’ and ’come’