Introduction to Human Language Technologies

3 - Morphology
Outline

1 Morphology
   - Motivation
   - Definitions
   - Types of morphologies

2 Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3 Spell checkers and spell correctors
Outline

1. Morphology
   - Motivation
     - Definitions
     - Types of morphologies

2. Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3. Spell checkers and spell correctors
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 little 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

1 Morphology
   - Motivation
   - Definitions
   - Types of morphologies

2 Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3 Spell checkers and spell correctors
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
    - Infixedes: [Arabic] ktb + CuCuC → kutub (books)
    - Circumfixedes: en+light+en

- The resulting words can be classified into categories known as Part of Speech (POS): Noun, Verb, Adjective, Adverb, Preposition, ...
Outline

1. Morphology
   - Motivation
   - Definitions
   - Types of morphologies

2. Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3. Spell checkers and spell correctors
Types of morphology

- Concatenative morphology: builds words up by concatenating morphemes (prefixes, suffixes). The most productive in the Indo-European languages.
  - **Inflectional morphology**: \( \text{word} \rightarrow \text{new forms of the word} \)
    Ex: \( \text{work} \rightarrow \text{worked} \)
  - **Derivational morphology**: \( \text{word} \rightarrow \text{new word} \)
    Ex: \( \text{frequent} \rightarrow \text{infrequent} \)
  - **Compositional morphology**: \( N \text{ word} \rightarrow \text{new word} \)
    Ex: \( \text{fire} + \text{man} \rightarrow \text{fireman} \)

- Non-concatenative morphology: builds words by other mechanism (infixes, circumfixes).
  - Ex: Root-Pattern morphology
    Ex: [Arabic] \( \text{ktb} + \text{CaCaCa} \rightarrow \text{kataba} [\text{en: he wrote}] \)
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

1 Morphology
   - Motivation
   - Definitions
   - Types of morphologies

2 Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3 Spell checkers and spell correctors
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:**  
  \[
  \text{word } POS + Gen + Num + Case + Tense + \ldots \ \text{LEMMA (stem)} \\
  \text{men Noun + M + PL man}
  \]
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)
      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 morphomes
  Ex: Reg_N + s → PL
      Reg_V + ing → Gerund

- Spelling rules: orthographic rules for combining letters
  Ex: E-insertion: -(z,x,s,sh,ch) \( ^s \) → -(z,x,s,sh,ch)es
      Consonant-doubling: -l \( ^{\text{ing}} \) → -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 transducers (FSTs)
Outline

1 Morphology
   - Motivation
   - Definitions
   - Types of morphologies

2 Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3 Spell checkers and spell correctors
Finite state automata (FSA)

A FSA defines a function over words \( w \) of a regular language \( L \).

\[ M_L : w \rightarrow \{true, false\} \]

\[ M = \langle Q, \Sigma, q_0, F, \sigma \rangle \]

- \( Q = \{q_0, \ldots, q_n\} \) finite set of states
- \( \Sigma = \{s_0, \ldots, s_k\} \) finite set of symbols
- \( q_0 \in Q \) start state
- \( F \subset Q \) final states

\( \sigma : Q \times \Sigma \rightarrow [Q \lor 2^Q] \) deterministic \( \lor \) non-det. transition function

\begin{align*}
a | (bc) + d & \in \{0, 1\} \\
a & \\
b & \\
bc & \\
bc & \\
bc & \\
b & \\
b & \\
bcd & \\
b & \\
b & \\
b & \\
b & \\
bcd & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
b & \\
\end{align*}
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

<table>
<thead>
<tr>
<th>Reg_N</th>
<th>Irreg_sg_N</th>
<th>Irreg_pl_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>mouse</td>
<td>mice</td>
</tr>
<tr>
<td>fox</td>
<td>foot</td>
<td>feet</td>
</tr>
<tr>
<td>tax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>donkey</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FSAs for morphological recognition

Example: FSA for English number nominal inflection

Morphotactics: List Irreg_N

Morphotactics: noun + s = PL over list Reg_N

SHOULD CORRECT WITH:

Spelling rule:
\( [s,x,z,sh,ch]^s = [s,x,z,sh,ch]es \) over list Reg_N
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

<table>
<thead>
<tr>
<th>Input word (surface form)</th>
<th>Output analysis (lexical form)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>dog+N+SG</td>
</tr>
<tr>
<td>dogs</td>
<td>dog+N+PL</td>
</tr>
<tr>
<td></td>
<td>(word form)</td>
</tr>
<tr>
<td></td>
<td>(lemma+Features)</td>
</tr>
</tbody>
</table>

- A more sophisticated technique is required: FSTs
Outline

1. Morphology
   - Motivation
   - Definitions
   - Types of morphologies

2. Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3. Spell checkers and spell correctors
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 symbols
- $\Delta = \{t_0, \ldots, t_m\}$ finite set of output symbols
- $q_0 \in Q$ start state
- $F \subset Q$ final states
- $\sigma : Q \times \Sigma \rightarrow 2^Q$ transition function
- $\delta : Q \times \Sigma \rightarrow \Delta$ output function

$$
\begin{array}{c|c}
\text{a} | (bc) + d \{0, 1\} & \text{d} | (cb) + a \{0, 1\} \\
\hline
\text{a} & \text{d} \\
\text{bc} & \text{cb} \\
\text{bcd} & \text{cba} \\
\text{bcbcd} & \text{cbcbb} \\
\text{bckcd} & \text{bbcbb} \\
\text{...} & \text{...}
\end{array}
$$
Finite state transducers (FSTs)

- Inversion: \( T : L_1 \rightarrow L_2 \implies T^{-1} : L_2 \rightarrow L_1 \)

![Diagram showing finite state transducers](image)

- b:c \( \implies b \rightarrow c \implies \text{Ex: } bcbc \rightarrow cbcb \)
- b:c \( \implies b \leftarrow c \implies \text{Ex: } bcbc \leftarrow cbcb \)

- Composition: \( T_a : L_1 \rightarrow L_2 \land T_b : L_2 \rightarrow L_3 \implies T_a \circ T_b : L_1 \rightarrow L_3 \)

- 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: \( L_2 = \{ < l, F > | l \text{ is lemma } \land F \text{ are morphological features} \} \)

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: $\text{Reg}_N \hat{s}\# \rightarrow \text{Reg}_N N N + PL$.
   - Ex: $\text{dog} \hat{s}\# \rightarrow \text{dog} N + PL$, $\text{fox} \hat{s}\# \rightarrow \text{fox} N + PL$

2. FSTs each computing a spelling rule, $T_{inter}^i$ (orthographic regularization)
   - Ex: $\{-z,x,s,sh,ch\} \text{es} \rightarrow \{-z,x,s,sh,ch\} \hat{s}\#$

\[
\begin{array}{c}
\text{surface level} \\
T_{inter}^1, \ldots, T_{inter}^k \\
\text{intermediate level} \\
T_{lex} \\
\text{lexical level}
\end{array}
\]

<table>
<thead>
<tr>
<th>dogs</th>
<th>foxes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\text{dog} \hat{s}#$</td>
<td>$\text{fox} \hat{s}#$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>$\downarrow$</td>
</tr>
<tr>
<td>$\text{dog} N + PL$</td>
<td>$\text{fox} N + PL$</td>
</tr>
</tbody>
</table>
FSTs for morphological analysis

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

$T_{\text{num\_nouns}}$

Examples of lists of stems/forms

<table>
<thead>
<tr>
<th>Reg_N</th>
<th>Irreg_sg_N</th>
<th>Irreg_pl_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>mouse</td>
<td>m o:i u:e s:c e f o:e o:e t</td>
</tr>
<tr>
<td>fox</td>
<td>foot</td>
<td></td>
</tr>
<tr>
<td>tax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>donkey</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FSTs for morphological analysis

\( T_{\text{lex}} \): FST that computes morphotactics

Example: FST for English number nominal inflection

\[
T_{\text{lex}} = T_{\text{stems/forms}} \circ T_{\text{num_nouns}}
\]

\[
\text{fox}^s# \rightarrow \text{fox+N+PL} !! \text{ (require spelling rules)}
\]
FSTs for morphological analysis

$T_{\text{inter}}^i$: FSTs that compute spelling rules

Example: FST for E-insertion rule

<table>
<thead>
<tr>
<th>e-insertion cases</th>
<th>regular cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>foxes $\rightarrow$ fox$^s$#</td>
<td>dogs $\rightarrow$ dog$^s$#</td>
</tr>
<tr>
<td>bosses $\rightarrow$ boss$^s$#</td>
<td>...</td>
</tr>
<tr>
<td>flashes $\rightarrow$ flash$^s$#</td>
<td>...</td>
</tr>
</tbody>
</table>

$'?':$ other symbol
FSTs for morphological analysis

\[ T^i_{\text{inter}}: \] 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

1. Morphology
   - Motivation
   - Definitions
   - Types of morphologies

2. Morphological analysis
   - Finite-state automata
   - Finite-state transducers

3. Spell checkers and spell correctors
Spell checkers

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

- **Possible approach**:
  
  \[
  FSA_L \text{ OR } FST_L \\
  S = \text{Tokenizer}(text) \text{ (sequence of forms)} \\
  \text{for each } x \in S \\
  \quad \text{if } FSA_L(x) \text{ then print(”x”) } \\
  \quad \text{else print(”**x**”)}
  \]
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 $FST_L$
  
  $S = \text{Tokenizer}(\text{text})$ (sequence of forms)
  
  for each $x \in S$
  
  if $x \in D$ then print($x$)
  
  else
  
  $D' = \{y : |\text{length}(x) - \text{length}(y)| \leq \gamma\}$
  
  $C = \emptyset$
  
  for each $y \in D'$
  
  $d = \text{distance}(x, y)$
  
  if ($d \leq \delta$) 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 $m \times n$ (lengths of $x$ and $y$) using dynamic programming
  - $d(x, y) = E(m, n)$
Spell correctors

Cost matrix computation

\[
E(i, j) = \min(Cost_{del}, Cost_{ins}, Cost_{swap})
\]

\[
\begin{align*}
Cost_{del} &= E(i - 1, j) + 1 \\
Cost_{ins} &= E(i, j - 1) + 1 \\
Cost_{swap} &= E(i - 1, j - 1) + s(x_i, y_j)
\end{align*}
\]

\[
s(x_i, y_j) \text{ normalised to 1.0}
\]
Exercise

- Compute the weighted edit distance between 'dom' and 'come'