Overview of Natural Language Processing
Part II & ACS L90
Lecture 2: Morphology

Weiwei Sun and Michael Schlichtkrull

Department of Computer Science and Technology
University of Cambridge

Michaelmas 2023/24
Lecture 2: Morphology

1. Morphology
2. Relevant NLP tasks
3. Finite state techniques
4. Byte-pair encoding

Some yinkish dripners blorked quastofically into the nindin with the pidibs... dipn+ER+S blork+ED quastofical+LY into the nindin with the pidib+S
Morphology
Morphemes are the *smallest meaningful units* of language. Words are composed of morpheme(s).
Morpheme

*Morphemes* are the *smallest meaningful units* of language. Words are composed of morpheme(s).

**Affix**: morpheme which only occurs in conjunction with other morphemes.
- suffix *(units)*, prefix *(incomplete)*, infix, circumfix
Morphemes are the *smallest meaningful units* of language. Words are composed of morpheme(s).

**Affix**: morpheme which only occurs in conjunction with other morphemes.
- suffix (*units*), prefix (*in*complete), infix, circumfix

**Root**: nucleus of the word that affixes attach too.
Infix

Tagalog (Philippines)

infix: -um-

basa  b-um-asa  sulat  s-um-ulat
read  read.PAST  write  wrote

Infix

Tagalog (Philippines)

basa  read
b-um-asar  read.PAST
sulat  write
s-um-ulat  wrote

infix: -um-

Circumfix: occur on both sides

Dutch collectives

circumfix: ge-X-te

berg  mountain
ge-berg-te  mountains

*ge-berg
*beg-te

Source: J Hana & A Feldman. ESSLLI 2013: Computational Morphology.
Inflection and derivation

**Inflection** creates new forms of the same word
- e.g. *bring*, *brought*, *brings*, *bringing*
- generally fully productive (modulo irregular forms)
- tends to affect only its *syntactic function*

**Derivation** creates new words
- e.g. *logic*, *logical*, *illogical*, *illogicality*, *logician*, etc.
- generally semi-productive: e.g., *escapee*, *textee*, *dropee*, *snoree*, *cricketee* (* and ?)
- tends to be more irregular; the meaning is more idiosyncratic and less compositional.
- tends to affect the *meaning* of the word, and may change part-of-speech
Internal structure: ambiguity

Structural ambiguity

unlockable

-able

un-

lock

-able

Not capable of being locked.

Can cross word boundaries

beautiful dancer

-er

beautiful
dance

dance
-er


More about unlockable: en.wiktionary.org/wiki/unlockable
Stem: word without its inflectional affixes = roots + all derivational affixes.
**Stem**: word without its inflectional affixes = roots + all derivational affixes.

*bookshopped*
**Stem**: word without its inflectional affixes = roots + all derivational affixes.

**bookshopped**

![Diagram](image)
**Stem:** word without its inflectional affixes = roots + all derivational affixes.

*bookshopped*

**Lexeme:** the set of all forms related by inflection (but not derivation).

\{bookshops, bookshopped, bookshopping, \ldots\}

**Lemma:** the *canonical/base/dictionary/citation* form of a lexeme chosen by convention.

*bookshop* (cf. the stem—*bookshopp*)
Phonaestheme

*slither, slide, slip* etc have somewhat similar meanings; but *sl-* is not a morpheme.

**Etymology**: *slith, slid* and *slip* are historically related. See www.etymonline.com/word/slide

Phonaestheme

a pattern of sounds systematically paired with a certain meaning in a language

- *cl-*: related to a closing motion of a single object, such as *clam, clamp, clap, clasp, clench, cling, clip, clop, clutch*.

- *gl-*: related to light, as in *glance, glare, glass, gleam, glimmer, glint, glisten, glitter, gloaming, gloom, gloss, glow.*
Compound and multiword expression (1)

2015

![Image of Word of the Year 2015](image-url)
Compound and multiword expression (1)

2016
Compound and multiword expression (1)

2017

Oxford Word of the Year: Youthquake
Compound and multiword expression (1)

2019

[Image of an award with the text: OXFORD WORD OF THE YEAR CLIMATE EMERGENCY]
Root: nucleus of the word that affixes attach too.

*Compounds* contain more than one root.

(1) a. youthquake  
    b. post-truth  
    c. railway  
    d. sunset

*Multiword expression*: combinations of two or more words that exhibit syntactic and semantic idiosyncratic behavior.

(2) a. climate emergency  
    b. computer science  
    c. random variable
Different types of multiword expressions

<table>
<thead>
<tr>
<th>Fixed</th>
<th>(Syntactically) flexible</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>by and large</em></td>
<td><em>put on the clothes</em></td>
</tr>
<tr>
<td></td>
<td><em>put the clothes on</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-compositional</th>
<th>Semi-compositional</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>kick the bucket</em></td>
<td><em>spill the beans</em></td>
<td><em>strong tea</em></td>
</tr>
<tr>
<td>(reveal the secret)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Multiword expression and grammatical errors

(3)  a. *At this moment* Carole was living with her husband but they didn’t love each other any more.
  → At the moment

b. It is a *dream becomes true* and was really unexpected for me!
  → dream come true

c. They go together in groups, then they prepare power point presentations and *at least* they present it in front of the other pupils and teachers.
  → finally

d. *By the other side*, I have never climbed a mountain but I always wanted to do it.
  → On the other hand

e. I tried to *take it on my stride* but I couldn’t.
  → take it in my stride

f. However, I told my teacher that I am willing to *give a hand* next time.
  → lend a hand
Code-mixed languages

Code-switching
a speaker alternates between two or more languages in the context of a single conversation or situation.

Cantonese-English (widely used in Hong Kong)
The English word “sure”/“cute” is mixed into an otherwise Cantonese sentence.

• 我唔sure
• cu唔cute啊
Text normalization

- Not using any punctuation at all
  
  *Eh speak english mi malay not tt good* (Eh, speak English! My Ma-lay is not that good.)

- Using spell-ing/punctuation for emphasis
  
  *gooooooood Sunday morning !!!!!!!* (Good Sunday morning!)

- Using phonetic spelling
  
  *dat iz enuf* (That is enough)

- Dropping vowel
  
  *i hv cm to c my luv.* (I have come to see my love.)

- Introducing local flavor
  
  *yar lor where u go juz now* (yes, where did you go just now?)

- Dropping verb
  
  *l hv 2 go. Dinner w parents.* (I have to go. Have dinner with parents.)


More: [noisy-text.github.io/norm-shared-task.html](https://noisy-text.github.io/norm-shared-task.html)
Relevant NLP Tasks
Form transformation

natural language expressions

comprehension

production

\( \mathcal{R} \) \{ morphological structure,
syntactic structure,
semantic structure,
discourse structure,
application-related structure \}
Computational tasks

LEMMATIZATION → word *saw*

TAGGING → contextualized word *saw @ J saw M*

SEGMENTATION → word *meaningful*

GENERATION ← word *saw*
antidisestablishmentarianism ⇒ anti-dis-establish-mentarianism

en.wikipedia.org/wiki/Antidisestablishmentarianism
www.etymonline.com/word/antidisestablishmentarianism

important for some application, e.g. bioinformatics
Word segmentation

Goal

• The written systems for some languages, e.g. Japanese and Chinese contain no word delimiters such as spaces.
• There is a need to develop algorithms that are able to automatically divide a string into its component words.

Example

解放大道路面积水问题

解放 / 大道 / 路面 / 积水 / 问题

解 / 放大 / 道路 / 面积 / 水 / 问题
Finite State Techniques
Language Is An Inherently Temporal Phenomenon

Orders matter!

• talk-ed \neq *ed-talk
• re-write \neq *write-re
• un-kind-ly \neq *kind-un-ly
Language Is An Inherently Temporal Phenomenon

Orders matter!

- talk-ed ≠ *ed-talk
- re-write ≠ *write-re
- un-kind-ly ≠ *kind-un-ly

\[ \text{sign} \]
Turing machine
Finite-state automata

- Circles are states of the automaton.
- Arrows are called transitions.
- The automaton changes states by following transitions.
- The double circle indicates that this state is an accepting state. The automaton accepts the string if it ends in an accepting state.
Finite-state automata

- Circles are states of the automaton.
- Arrows are called transitions.
- The automaton changes states by following transitions.
- The double circle indicates that this state is an accepting state. The automaton accepts the string if it ends in an accepting state.

- Form Transformation: augmenting transitions

  \[\text{input} \rightarrow \text{input:output}\]
Finite state transducer

- \( \text{cakes} \rightarrow \text{cake}\#s \)
- \( \text{boxes} \rightarrow \text{box}\#s \)
Analysing boxes

OUTPUT

INPUT

| b | o | x | e | s |

\[
\begin{align*}
q_1 & \xrightarrow{e:e} q_1 \\
q_1 & \xrightarrow{other:other} q_4 \\
q_2 & \xrightarrow{\epsilon:}\# q_2 \\
q_2 & \xrightarrow{s:s} q_3 \\
q_4 & \xrightarrow{e:e} q_4 \\
q_4 & \xrightarrow{other:other} q_1 \\
q_4 & \xrightarrow{s:s} q_4 \\
q_4 & \xrightarrow{x:x} q_4 \\
q_4 & \xrightarrow{z:z} q_4 \\
q_4 & \xrightarrow{e:}\# q_4
\end{align*}
\]
Analysing boxes

OUTPUT
INPUT

q1

q2

q3

q4

\begin{array}{cccc}
\text{boxes} & \text{boxes} & \text{boxes} & \text{boxes} \\
\end{array}

\begin{array}{cccc}
\text{other} & \text{other} & \text{other} & \text{other} \\
\end{array}

\begin{array}{cccc}
\text{e}:\# & \text{s}:s & \text{e}:\# & \text{e}:\# \\
\end{array}

\begin{array}{cccc}
\text{s}:s & \text{x}:x & \text{z}:z & \text{e}:\# \\
\end{array}

\begin{array}{cccc}
\text{s}:s & \text{x}:x & \text{z}:z & \text{e}:\# \\
\end{array}

\begin{array}{cccc}
\text{s}:s & \text{x}:x & \text{z}:z & \text{e}:\# \\
\end{array}

\begin{array}{cccc}
\text{s}:s & \text{x}:x & \text{z}:z & \text{e}:\# \\
\end{array}
Analysing boxes

OUTPUT
b
INPUT
b o x e s

\[ q_1 \xrightarrow{\epsilon: \#} q_2 \xrightarrow{s:s} q_3 \]

\[ q_4 \xrightarrow{\epsilon: \#} q_1 \]

\[ q_1 \xrightarrow{\epsilon: e} q_4 \]

\[ q_1 \xrightarrow{s:s} q_2 \xrightarrow{s:s} q_1 \}

\[ q_2 \xrightarrow{s:s} q_3 \]

\[ q_2 \xrightarrow{x:x} q_1 \]

\[ q_2 \xrightarrow{z:z} q_4 \]
Analysing boxes

OUTPUT

INPUT

\begin{array}{cccc}
\text{b} & \text{b} & \text{boxes} & \text{b} \\
\end{array}

\begin{tikzpicture}
    \node (q1) [state, initial, accepting] {q1};
    \node (q2) [state, accepting, right of=q1] {q2};
    \node (q3) [state, right of=q2] {q3};
    \node (q4) [state, below of=q1] {q4};

    \draw [->] (q1) edge [loop above] node {e:e} (q1);
    \draw [->] (q1) edge [above] node {\text{other}:\text{other}} (q2);
    \draw [->] (q2) edge [above] node {\text{e}:\#} (q3);
    \draw [->] (q2) edge [below] node {\text{s}:s} (q1);
    \draw [->] (q3) edge [loop right] node {\text{e}:\#} (q3);
    \draw [->] (q4) edge [loop below] node {\text{s}:s} (q4);
    \draw [->] (q4) edge [left] node {\text{e}:\#} (q2);
    \draw [->] (q4) edge [right] node {\text{s}:s} (q3);
    \draw [->] (q4) edge [below] node {\text{x}:x} (q1);
    \draw [->] (q4) edge [below] node {\text{z}:z} (q1);
\end{tikzpicture}
Analysing boxes

OUTPUT

INPUT

| b | o | x | e | s |

Diagram:

- States: $q_1$, $q_2$, $q_3$, $q_4$
- Transitions:
  - $q_1$ to $q_2$: $\epsilon$:
  - $q_1$ to $q_3$: $s$:s
  - $q_1$ to $q_4$: $s$:s
  - $q_2$ to $q_3$: $s$:s
  - $q_2$ to $q_4$: $s$:s
  - $q_3$ to $q_4$: $s$:s
  - $q_4$ to $q_1$: $\epsilon$:

- Labels:
  - $b$:o
  - $x$:e
  - $s$:s

- Other:other

- Final States: $q_3$, $q_4$
Analysing boxes

OUTPUT

<table>
<thead>
<tr>
<th>b</th>
<th>o</th>
<th>x</th>
</tr>
</thead>
</table>

INPUT

| b | o | x | e | s |

---

\[ q_1 \xrightarrow{\epsilon:\#} q_2 \xrightarrow{s:s} q_3 \]

\[ q_4 \xrightarrow{s:s} q_4 \]

\[ q_4 \xrightarrow{\text{other:other}} q_1 \xrightarrow{\text{other:other}} q_4 \]

---

\[ e:e \]

\[ s:s \]

\[ x:x \]

\[ z:z \]
Analysing boxes

OUTPUT

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>o</td>
<td>x</td>
</tr>
<tr>
<td>b</td>
<td>o</td>
<td>x</td>
</tr>
</tbody>
</table>

INPUT

1. $q_1$: $\epsilon$:
   - $s$: $s$
   - $x$: $x$
   - $z$: $z$

2. $q_2$: $\epsilon$:
   - $s$: $s$
   - $\epsilon$:
   - $x$: $x$
   - $\epsilon$:
   - $z$: $z$

3. $q_3$: $s$:

4. $q_4$: $\epsilon$:
   - $s$: $s$
   - $\epsilon$:
   - $x$: $x$
   - $\epsilon$:
   - $z$: $z$
Analysing boxes

OUTPUT

INPUT

<table>
<thead>
<tr>
<th>b</th>
<th>o</th>
<th>x</th>
<th>#</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>o</td>
<td>x</td>
<td>e</td>
<td>s</td>
</tr>
</tbody>
</table>

```
q1
  e:e
  other:other

q2
  e:#

q3
  s:s

q4
  e:e
  other:other

s:s
x:x
z:z
```
Finite-state machine

- A symbolic system that can recognize or transform forms.
Finite-state machine

- A symbolic system that can recognize or **transform forms**.
- An automaton remembers only a finite amount of information.
Finite-state machine

- A symbolic system that can recognize or transform forms.
- An automaton remembers only a finite amount of information.
- Information is represented by its states.
Finite-state machine

- A symbolic system that can recognize or transform forms.
- An automaton remembers only a finite amount of information.
- Information is represented by its states.
- State changes in response to inputs and may trigger outputs.
Finite-state machine

- A symbolic system that can recognize or transform forms.
- An automaton remembers only a finite amount of information.
- Information is represented by its states.
- State changes in response to inputs and may trigger outputs.
- Transition rules define how the state changes in response to inputs.
A symbolic system that can recognize or **transform forms**.

- An automaton remembers only a finite amount of information.
- Information is represented by its states.
- State changes in response to inputs and may trigger outputs.
- Transition rules define how the state changes in response to inputs.

- Given a sequence of input symbols, a recognition process starts in the start state and follow the transitions in turn. Input is accepted if this process ends up in an accepting state.
Finite-state machine

- A symbolic system that can recognize or **transform forms**.
- An automaton remembers only a finite amount of information.
- Information is represented by its states.
- State changes in response to inputs and may trigger outputs.
- Transition rules define how the state changes in response to inputs.
- Given a sequence of input symbols, a recognition process starts in the start state and follow the transitions in turn. Input is accepted if this process ends up in an accepting state.
- **Partial grammars for text preprocessing, tokenization, named entity recognition etc.**
Cross-lingual variants

• **English** morphology is essentially concatenative
cf. duplication in **Chinese**, e.g.

<table>
<thead>
<tr>
<th>Root Pattern</th>
<th>PoS</th>
<th>Phonological Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktb</td>
<td>C</td>
<td>katav 'wrote'</td>
</tr>
<tr>
<td>ktb hi</td>
<td>CC</td>
<td>hixtiv 'dictated'</td>
</tr>
<tr>
<td>ktb mi</td>
<td>CC</td>
<td>mixtav 'a letter'</td>
</tr>
<tr>
<td>ktb</td>
<td>CC</td>
<td>ktav 'writing, alphabet'</td>
</tr>
</tbody>
</table>

from E. Bender's tutorial (faculty.washington.edu/ebender/papers/100things.pdf)
Cross-lingual variants

- **English** morphology is essentially concatenative cf. duplication in Chinese, e.g.
  
  \[
  \text{高兴} \rightarrow \text{高兴 高兴 高兴 高兴}
  \]
  
  *happy*

- The phones making up a morpheme don’t have to be contiguous, e.g. in Hebrew

<table>
<thead>
<tr>
<th>Root</th>
<th>Pattern</th>
<th>PoS</th>
<th>Phonological Form</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>ktb</td>
<td>CaCaC</td>
<td>v</td>
<td>katav</td>
<td>‘wrote’</td>
</tr>
<tr>
<td>ktb</td>
<td>hiCCiC</td>
<td>v</td>
<td>hixtiv</td>
<td>‘dictated’</td>
</tr>
<tr>
<td>ktb</td>
<td>miCCaC</td>
<td>n</td>
<td>mixtav</td>
<td>‘a letter’</td>
</tr>
<tr>
<td>ktb</td>
<td>CCaC</td>
<td>n</td>
<td>ktav</td>
<td>‘writing, alphabet’</td>
</tr>
</tbody>
</table>

from E. Bender’s tutorial ([faculty.washington.edu/ebender/papers/100things.pdf](faculty.washington.edu/ebender/papers/100things.pdf))
Byte-Pair Encoding
Form transformation

natural language expressions \[ R \] morphological structure
syntactic structure
semantic structure
discourse structure
application-related structure

Are there some magical algorithms that are able to automatically induce useful representations from data?
Subword tokenisation

Isn’t it just one symbol?

\[
\begin{align*}
i &+ l &+ y &+ I &Y &\text{sign} \\
\end{align*}
\]
Subword tokenisation

Isn’t it just one symbol?

Phonaestheme: It is difficult to hard-code the knowledge

- \( cl \)-: related to a closing motion of a single object, such as *clam, clamp, clap, clasp, clench, cling, clip, clop, clutch*. 
Byte-Pair Encoding (BPE)

BPE was initially developed as an algorithm to compress texts, and then used by OpenAI for tokenization when pretraining the GPT model.

- Start from a small base vocabulary, e.g. 256 ASCII code.
- Add new tokens to the vocabulary until the desired vocabulary size is reached by learning merges, which are rules to merge two elements of the existing vocabulary together into a new one.
- At each step, the BPE algorithm search for the most frequent pair, namely two consecutive tokens, of existing tokens.

from https://huggingface.co/learn/nlp-course/chapter6/5?fw=pt
Example


on whiteboard
Readings

Required

• Ann’s lecture notes.
  https://www.cl.cam.ac.uk/teaching/1920/NLP/materials.html
• E. Bender. 100 Things You Always Wanted to Know about Linguistics But Were Afraid to Ask. NAACL-HLT 2012 tutorial.
  faculty.washington.edu/ebender/papers/100things.pdf Please read Numbers #7–#27.

Optional

  ufal.mff.cuni.cz/~hana/teaching/2013-esslli/
  www.aclweb.org/anthology/J97-2003/