Morphological analysis

ESSLLI 2013: Computational Morphology

Jirka Hana & Anna Feldman

Processing morphology

1. Lemmatization: word → lemma
   saw → { *see*, *saw* }
2. Morphological analysis (MA): word → setOf(lemma + tag), ignores context
   saw → { ⟨*see*, verb.past⟩, ⟨*saw*, noun.sg⟩, }
3. Tagging: word → tag (often also lemma), considers context
   saw @ *Peter saw her.* → { ⟨*see*, verb.past⟩ }
4. Morpheme segmentation: *de-nation-al-iz-ation*
5. Generation: *see* + verb.past → *saw*

Applications

- Parsing/chunking (used in machine translation, grammar correction, etc.)
- Text Generation
- Search and information retrieval. One usually searches for a lexeme not for a particular form.
- Text-to-speech synthesis.
  $read_{present}$ [rid] vs. $read_{past}$ [rɛd]
  Russian: $sn\grave{e}ga_{noun.masc.sg.gen}$ 'snow' vs. $sneg\grave{a}_{noun.masc.pl.nom/acc}$
- Spell checking
- (Computer assisted) language learning.

Creation/Acquisition

1. manually provided rules
2. use machine learning
    1. supervised – deduced from an annotated corpus
    2. unsupervised – deduced from plain text
3. hybrid

Morphological analysis

MA: form $\rightarrow$ set(lemma $\times$ set(tag))

Morphological analysis

MA: form $\rightarrow$ set(lemma $\times$ set(tag))

English:	*her*	$\rightarrow$	{ (*she*,	{PP }),	
			(*her* ,	{PP$}) }	
Czech:	*ženou*	$\rightarrow$	{ (*žena* 'woman',	{noun fem sing inst }),	
			(*hnát* 'hurry',	{verb pres pl 3rd }) }	
	ženy	$\rightarrow$	{ (*žena* 'woman',	{noun fem sing gen,	
				noun fem pl nom,	
				noun fem pl acc,	
				noun fem pl voc }) }	

Complications

- Stem internal (non-concatenative) alternations:
  German: *Stuhl* → *Stühl-e*, *Vater* → *Väter*
- Irregularities.
  English: *goose* → *geese*, *sheep* → *sheep*
  Russian plural: *knig-a* → *knig-i*, *stol* → *stol-y*, but *kofe* → *kofe*
- Phonological/graphemic alternations:
  English: *knife* → *knive-s*, *city* → *citi-es*
- Homonymy:
  English *-s* – 3rd person singular of verbs vs. plural of nouns;
  Czech *-a* / *-e* (see the tables in the first lecture).

Different Approaches

Two different ways to address phonological/graphemic variations and complex paradigm systems when designing a morphological analyzer:

1. A linguistic approach.
   A phonological component accompanying the simple concatenative process of attaching an ending
2. An engineering approach.
   - No (or very rudimentary) phonological component
   - Phonological changes and irregularities are factored into endings and a higher number of paradigms

Approaches: Comparison

	woman	owl	draft	iceberg	vapor	fly
S1	žen-a	sov-a	skic-a	kr-a	pár-a	mouch-a
S2	žen-y	sov-y	skic-i	kr-y	pár-y	mouch-y
S3	žen-ě	sov-ě	skic-e	kř-e	pář-e	mouš-e
:						
P2	žen-0	sov-0	skic-0	ker-0	par-0	much-0

A linguistic approach

$$\textit{žen} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases} \quad \textit{sov} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases} \quad \textit{skic} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases} \quad \textit{kr} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases} \quad \textit{pár} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases} \quad \textit{mouch} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases}$$

An engineering approach

$$\textit{žen} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases} \quad \textit{sov} + \begin{cases}\text{a}\\ \text{y}\\ \text{ě}\\ 0\end{cases} \quad \textit{skic} + \begin{cases}\text{a}\\ \textbf{i}\\ \textbf{e}\\ 0\end{cases} \quad \textbf{k} + \begin{cases}\text{ra}\\ \text{ry}\\ \text{ře}\\ \text{er}\end{cases} \quad \textbf{p} + \begin{cases}\text{ára}\\ \text{áry}\\ \text{áře}\\ \text{ar}\end{cases} \quad \textbf{m} + \begin{cases}\text{oucha}\\ \text{ouchy}\\ \text{ouše}\\ \text{uch}\end{cases}$$

Linguistic Approach

- Phonological component accompanying the simple concatenative process of attaching an ending;

Linguistic Approach

- Phonological component accompanying the simple concatenative process of attaching an ending;
- Advantages:
  - Small set of paradigms and morphemes
  - Captures linguistics generalizations

Linguistic Approach

- Phonological component accompanying the simple concatenative process of attaching an ending;
- Advantages:
  - Small set of paradigms and morphemes
  - Captures linguistics generalizations
- Problems:
  - Requires a lot of linguistic work and expertise
  - For many languages, the linguistic knowledge is not precise enough
  - It is usually not straightforward to translate even a precisely formulated linguistic description of a morphology into the representation recognized by such a system

Linguistic Approach: Finite-State Morphology

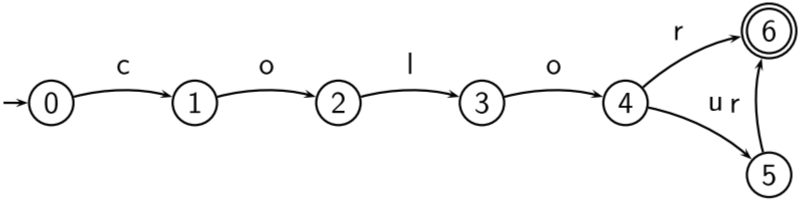
- Morphology analyzed by finite-state automata/transducers.
- It is by far the most popular approach in the field.
- (**johnson:1972**; **kaplan-kay:81**; **beesley-karttunen:03**).
- Two-level morphology (**koskenniemi:1983**; **koskenniemi:1984**)

What is finite state automaton (FSA)?

- Introduced by (**kleene:56**).
- A kind of directed graph:
  - Nodes are called states
  - Each edge is labeled with an accepted string (possibly empty)
  - One node is called the start state
  - One or more nodes are called stopping (or accepting) states
- Recognize/generate regular languages, i.e., languages specified by regular expressions.

An example

- Regular expression: `colou?r`
- Finite state machine:



Some properties of finite state machines

- Recognition problem can be solved in linear time (independent of the size of the automaton).
- There is an algorithm to transform each automaton into a unique equivalent automaton with the least number of states.

Deterministic Finite State Automata

A finite state automaton is deterministic iff it has

- no ϵ (empty) transitions and
- for each state and each symbol there is at most one applicable transition.

Every non-deterministic automaton can be transformed into a deterministic one:

- Define new states representing a disjunction of old states for each non-determinacy which arises.
- Define arcs for these states corresponding to each transition which is defined in the non-deterministic automaton for one of the disjuncts in the new state names.

Finite State Transducers

- Translate strings from one language to strings from another language
- Like a FSA, but each edge is associated with two strings.

Two-level morphology

- Uses 2 levels
  - lexical/underlying/deep forms
  - surface forms
  - one-one correspondence between symbols
c	o	u	n	t	r	y	0	+	s
c	o	u	n	t	r	i	e	0	s

Two-level morphology

- Uses 2 levels
  - lexical/underlying/deep forms
  - surface forms
  - one-one correspondence between symbols
  -

c	o	u	n	t	r	y	0	+	s
c	o	u	n	t	r	i	e	0	s

- Two components
  - Linked lexicons – sets of (underlying forms of) morphemes
  - Phonological rules – relate lexical and surface forms

Two-level morphology – Complextity

- All this can be compiled into one big FST.

Two-level morphology – Complextity

- All this can be compiled into one big FST.
- Looks fast and efficient, but can encode any NP problem.

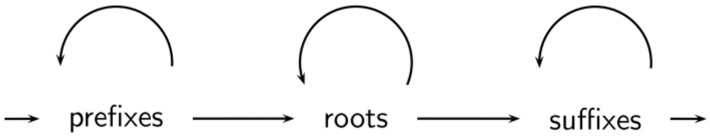
Two-level morphology – Complextity

- All this can be compiled into one big FST.
- Looks fast and efficient, but can encode any NP problem.
- Unrestricted null-characters make it even more complex.

Two-level morphology – Complextity

- All this can be compiled into one big FST.
- Looks fast and efficient, but can encode any NP problem.
- Unrestricted null-characters make it even more complex.
- Reasonable morphology specifications are practically computationally tractable.

Two-level morphology – Linked lexicons



Two-level morphology > Linked lexicons: Example

Two-level morphology – Rules

- relate underlying and surface forms
- applied simultaneously
- Form: lexical symbol : surface symbol operator context

Composite rule	x:y	$\Leftrightarrow$	LeftCtx __ RightCtx
x can is realized as y inn the given cxt			
Context restriction rule	x:y	$\Rightarrow$	LeftCtx __ RightCtx
x can be realized as y only in the given cxt			
Surface coercion rule	x:y	$\Leftarrow$	LeftCtx __ RightCtx
x must be realized as y in the given cxt			
Exclusion rule	x:y	$\not\Leftarrow$	LeftCtx __ RightCtx
x cannot be realized as y in the given cxt			

- `y:i` $\Leftrightarrow$ __ `0:e` ($y - ie$)

- days, spying, played, carryable
- rallies, spies, spied, happily

$$\text{y} \leftrightarrow \text{i} \;/$$

- days, spying, played, carryable
- rallies, spies, spied, happily

$$\text{y} \leftrightarrow \text{i} \;/\; \text{C} \;\_\_ + \text{not}\{\text{i,a}\}$$

hard + *i/ě* – depends on the origin of the vowel

k+ě/i→ č+e/i (1st)	*matka*	*matk+in → matčin*
k+ě/i→ c+e/i (2nd)	*matka*	*matk+ě → matce*
k+j → č	*tlak*	*tlač+jen → tlačen*
h+ě/i→ ž+e/i (1st)	*bůh*	*bůh+ě → bože*
h+ě/i→ z+e/i (2nd)	*bůh*	*bůh+i → bozi*
h+j → ž	*mnoho*	*množ+jení → množení*
g+ě/i→ ž+e/i (1st)	*Jaga*	*Jag+in → Jažin*
g+ě/i→ z+e/i (2nd)	*Jaga*	*Jag+ě → Jaze*
g+j → ž	*pedagog*	*pedagog+jení → pedagožení* ??
d+ě → /ďe/ → dě	*rada*	*rad+ě → radě*
d+j → z	*sladit*	*slad+jení → slazení/sladění*
t+ě → /ťe/ → tě	*teta*	*tet+ě → tetě*
t+je → ce	*platit*	*plat+jení → placení* not productive
ch → š	*moucha*	*mouch+ě → mouše* ; *muší*
n → /ň/ → n	*hon*	*hon+it → honit* ; *honěný*
r → ř	*var*	*var+it → vařit* ; *vaření*

Neutral consonant + ě depends on the origin of ě

b+ě → b+ě	*vrba*	*vrb+ě → vrbě*
b+je → b+e	*zlobit*	*zlob+jení → zlobení*
m+ě → m+ě		
m+je → m+e	*zlomit*	*zlom+jený → zlomený*
p+ě → p+ě		
p+je → p+e	*kropit*	*krop+jení → kropení*
v+ě → v+ě		
v+je → v+e	*lovit*	*lov+jení → lovení*
s+ě → s+e	*vosa*	*vos+ě → vose*
s+je → š+e	*prosit*	*pros+jení → prošení*
→ s+e	*kosit*	*kos+jení → kosení*
z+ě → z+e	*koza*	*koz+ě → koze*
z+je → ž+e	*kazit*	*kazjení → kažení*
→ z+e	*řetězit*	*řetěz+jení → řetězení*
l+ě → l+e	*škola*	*škol+ě → škole*
l+je → l+e	*školit*	*škol+jení → školení*

Consonant cluster + soft vowel

st+j→ šť	*čistit*	*čišt+jení → čišť+ení → čištění* 'cleaning'
sl+j→ šl	*myslit*	*myšl+jení → myšlení*
sk → šť	*kamarádský*	*kamarádsk+í → kamarádští*
	kamarádský	*kamarádsk+ější → kamarádštější*
ck → čť	*čacký*	*čačk+í → čačtí*
	čacký	*čačk+ější → čačtější*
čk → čc	*žluťoučký*	*žluťoučk+í → žluťoučcí*
čk → čť	*žluťoučký*	*žluťoučk+ější → žluťoučtější*

- Introduce special characters marking stems (simplified):

  ^1P — 1st palatalization
  ^2P — 2nd palatalization
  ^A — Assimilation (*tlak* → *tlačen*).
  ^N — No alternation.

- Introduce special characters marking stems (simplified):

  ^1P — 1st palatalization
  ^2P — 2nd palatalization
  ^A — Assimilation (*tlak* → *tlačen*).
  ^N — No alternation.

- The marker is followed by a number indicating how many chars should be removed before attaching the suffix. (^1P0, ^2P0, ^1P1, ...)

- Introduce special characters marking stems (simplified):

  ˆ1P — 1st palatalization
  ˆ2P — 2nd palatalization
  ˆA — Assimilation (*tlak* → *tlačen*).
  ˆN — No alternation.

- The marker is followed by a number indicating how many chars should be removed before attaching the suffix. (ˆ1P0, ˆ2P0, ˆ1P1, . . .)
- `doktorkaˆ1P1inˆ2P0ých`

- Introduce special characters marking stems (simplified):

  ^1P — 1st palatalization
  ^2P — 2nd palatalization
  ^A — Assimilation (*tlak* → *tlačen*).
  ^N — No alternation.

- The marker is followed by a number indicating how many chars should be removed before attaching the suffix. (^1P0, ^2P0, ^1P1, ...)
- `doktorka^1P1in^2P0ých`
- *doktorka* → *doktorčin*

- Introduce special characters marking stems (simplified):

  ^1P — 1st palatalization
  ^2P — 2nd palatalization
  ^A — Assimilation (*tlak* → *tlačen*).
  ^N — No alternation.

- The marker is followed by a number indicating how many chars should be removed before attaching the suffix. (^1P0, ^2P0, ^1P1, ...)
- `doktorka^1P1in^2P0ých`
- *doktorka* → *doktorčin*
  úředník → *úřednice* → *úředničin*

- Introduce special characters marking stems (simplified):

  ^1P — 1st palatalization
  ^2P — 2nd palatalization
  ^A — Assimilation (*tlak* → *tlačen*).
  ^N — No alternation.

- The marker is followed by a number indicating how many chars should be removed before attaching the suffix. (^1P0, ^2P0, ^1P1, ...)
- `doktorka^1P1in^2P0ých`
- *doktorka* → *doktorčin*
  úředník → *úřednice* → *úředničin*
- 159 paradigms instead of 219

Engineering approach

- No (or very rudimentary) phonological component
- Phonological changes and irregularities are factored into endings and a higher number of paradigms.
  Therefore the terms *stem* and *ending* have slightly different meanings than they traditionally do. A stem is the part of the word that does not change within its paradigm, and the ending is the part of the word that follows such a stem.

Engineering approach (cont.)

- Advantages:
  - high speed;
  - simple implementation;
  - straightforward morphology specification;
- Problems:
  - high number of paradigms (e.g. around 500 for Czech);
  - Impossibility to capture even the simplest and most regular phonological changes and so predict the behavior of new lexemes;
  - in theory, incapable of capturing some languages
- (**hajic:2004**) for Czech; (**mikheev:liubushkina:1995**) for Russian