Basics of Computational Morphology

NPFL096 Computational Morphology 2011

Jirka Hana

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: $snèga_{noun.masc.sg.gen}$ 'snow' vs.
  $snegà_{noun.masc.pl.nom/acc}$
- 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

Tags and tagsets

- **(morphological) tag** – a symbol encoding morphological properties of a word
- **tagset** – set of tags, depends on language and application

Tags and tagsets

- **(morphological) tag** – a symbol encoding morphological properties of a word
- **tagset** – set of tags, depends on language and application

- Penn Tagset: about 40 tags; `VBD` – verb in past tense
- Czech Positional Tagset: about 4000 tags; `VpNS---XR-AA---`

Tagsets for English: Penn Treebank

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	and, but, or	SYM	Symbol	+,%, &
CD	Cardinal number	one, two, three	TO	'to'	to
DT	Determiner	a, the	UH	Interjection	ah, oops
EX	Existential	'there' there	VB	Verb, base form	eat
FW	Foreign word	mea culpa	VBD	Verb, past tense	ate
IN	Preposition/sub-conj	of, in, by	VBG	Verb, gerund	eating
JJ	Adjective	yellow	VBN	Verb, past participle	eaten
JJR	Adj., comparative	bigger	VBP	Verb, non-3sg pres	eat
JJS	Adj., superlative	wildest	VBZ	Verb, 3sg pres	eats
LS	List item marker	1, 2, One	WDT	Wh-determiner	which, that
MD	Modal	can, should	WP	Wh-pronoun	what, who
NN	Noun, sing. or mass	llama	WP$	Possessive wh-	whose
NNS	Noun, plural	llamas	WRB	Wh-adverb	how, where
NNP	Proper noun, singular	IBM	$	Dollar sign	$
NNPS	Proper noun, plural	Carolinas	#	Pound sign	#
PDT	Predeterminer	all, both	“	Left quote	(‘ or “)
POS	Possessive ending	's	”	Right quote	(’ or ”)
PP	Personal pronoun	I, you, he	(	Left parenthesis	([, (, {,<)
PP$	Possessive pronoun	your, one's	)	Right parenthesis	( ],), }, >)
RB	Adverb	quickly, never	,	Comma	,
RBR	Adverb, comparative	faster	.	Sentence-final punc	(. !?)
RBS	Adverb, superlative	fastest	:	Mid-sentence punc	(: ; ... '-)
RP	Particle	up, off			

Czech positional tagset

Position	Name	Description	Example *viděl* 'saw'	
1	POS	part of speech	`V`	verb
2	SubPOS	detailed part of speech	`p`	past participle
3	gender	gender	`N`	neuter
4	number	number	`S`	singular
5	case	case	`--`	n/a
6	possgender	possessor's gender	`--`	n/a
7	possnumber	possessor's number	`--`	n/a
8	person	person	`X`	any
9	tense	tense	`R`	past tense
10	grade	degree of comparison	`--`	n/a
11	negation	negation	`A`	affirmative
12	voice	voice	`A`	active voice
13	reserve1	unused	`--`	n/a
14	reserve2	unused	`--`	n/a
15	var	variant, register	`--`	basic variant

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 Table next slide).

Complications (cont.)

Table: Homonymy of the *a* ending in Czech

form	lemma	gloss		category
měst-a	město	town	NS2	noun neut sg gen
			NP1 (5)	noun neut pl nom (voc)
			NP4	noun neut pl acc
tém-a	téma	theme	NS1 (5)	noun neut sg nom (voc)
			NS4	noun neut sg acc
žen-a	žena	woman	FS1	noun fem sg nom
pán-a	pán	man	MS2	noun masc anim sg gen
			MS4	noun masc anim sg acc
ostrov-a	ostrov	island	IS2	noun masc inanim sg gen
předsed-a	předseda	president	MS1	noun masc anim sg nom
vidě-l-a	vidět	see		verb past fem sg
				verb past neut pl
vidě-n-a				verb passive fem sg
				verb passive neut pl
vid-a				verb transgressive masc sg
dv-a	dv-a	two		numeral masc sg nom
				numeral masc sg acc

Complications (cont.)

Table: Ending *-e* and noun cases in Czech

case	form	lemma	gender	gloss
nom	kuř-e	kuře	neuter	chicken
gen	muž-e	muž	masc.anim.	man
dat	mouš-e	moucha	feminine	fly
acc	muž-e	muž	masc.anim.	man
voc	pan-e	pán	masc.anim.	mister
loc	mouš-e	moucha	feminine	fly
inst	–	–		

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	k**e**r-0	p**a**r-0	m**u**ch-0

A linguistic approach

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

An engineering approach

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

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

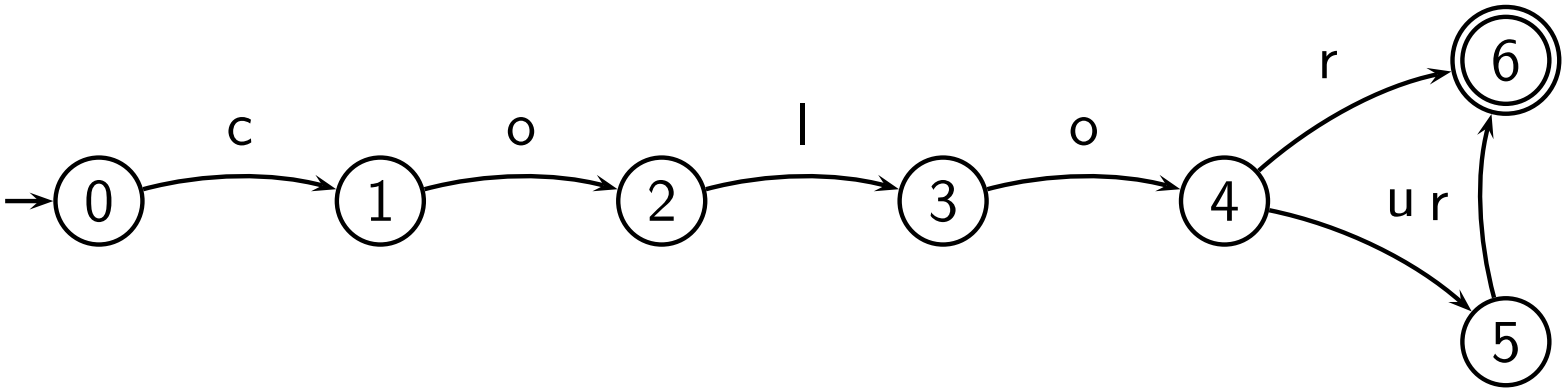
- Morphology analyzed by finite-state automata/transducers.
- It is by far the most popular approach in the field.
- (**johnson:1972**; **kaplan-kay:81**; **beesley-karttunen:03**).
- Two-level morphology (**koskenniemi:1983**; **koskenniemi:1984**)

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.

level morphology

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.

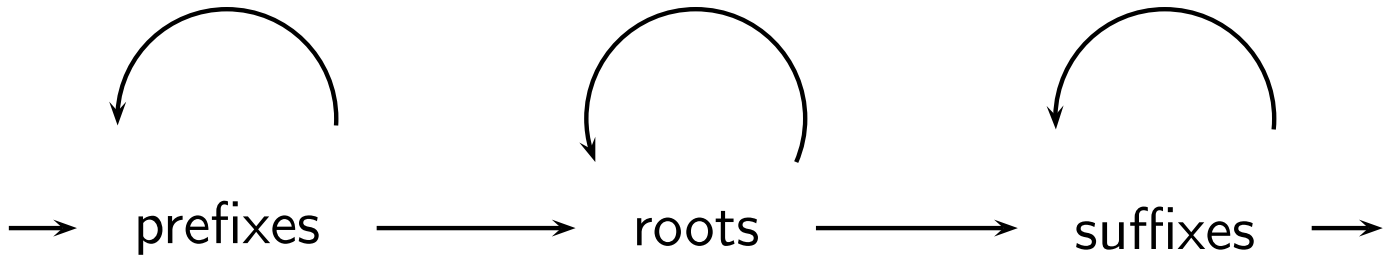
Two-level morphology – Complextity

- 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 – 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` $(y - ie)$

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