Computational Morphology: trends, finite-states and open-source

(***Evolución de la morfología computacional: nuevas posibilidades***)

XII Simposio de Comunicación Social

Santiago de Cuba. 2011

HANDOUT - NOTAS

Bibliography

[1] Beesley K. R. and Karttunen L. 2003. *Finite State Morphology*. CSLI Publications, Palo Alto, CA. http://www.stanford.edu/~laurik/fsmbook/home.html

[2] Oflazer K. *Computational Morphology* (slides). Tutorial. FSMNLP2009. http://fsmnlp2009.fastar.org/Program_files/Oflazer%20-%20slides.pdf

[3] Alegria I., Etxeberria I., Hulden H., Maritxalar M. 2009. Porting Basque Morphological Grammars to foma, an Open-Source Tool. FSMNLP2009. Pretoria. South Africa.

[4] Antworth, E.L. 1990. *PC-KIMMO: a two-level processor for morphological analysis*. Occasional Publications in Academic Computing, vol. 16.

Slides and examples

https://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/1294744449/publikoak/zip
http://foma.sf.net/lrec2010
http://foma.sf.net/dokuwiki

Tools

foma: http://foma.sf.net/
Xerox tools: http://www.stanford.edu/~laurik/.book2software/

Additional links

Graphviz: http://www.graphviz.org (for automata visualization)
Graphviz for OSX: http://www.pixelglow.net (for automata visualization on Macs)
Beesley & Karttunen FSM book: http://www.fsmbook.com

Foma: frequently used commands

apropos <keyword>	short help on keyword
help <keyword>	long help on keyword
regex regular-expression	compile regular expression
define name regular-expression	compile and name a regular expression
define name	name the top FSM on stack
clear/clear stack	clear stack
pop/pop stack	pop the top FSM off stack
down/apply down	enter apply down mode
down <word>	apply a single word
up/apply up	enter apply up mode
up <word>	apply a single word
med/apply med	enter minimum edit distance mode (automata only)
load stack <filename>	load binary FSM
load defined <filename>	load definitions from binary file
save defined <filename>	save definitions into binary file
save stack <filename>	save all the Fhttps://ixa.si.ehu.es/Ixa/Argitalpenak/Artikuluak/1294744449/publikoak/zipSMs on the stack into binary file
source <filename>	compile a file of foma-commands
read lexc <filename>	compile a lexc file
print words/words	print all words in a FSM
print upper-words/upper-words	print all words on upper side
print lower-words/lower-words	print all words on lower side
print random-lower/random-lower	print a random selection of words (lower)
print random-upper/random-upper	print a random selection of words (upper)
print net/net	print information about top FSM on stack
view net/view	display top FSM visually

Foma: regular expressions

Standard:

`A B`	Concatenation
`A | B`	Union
`A & B`	Intersection
`A*`	Kleene star
`A+`	Kleene plus
`$A`	“Contains” a string from A
`A-B`	Subtraction
`~A`	Complement of A
`A.r`	Reverse of A
`(A)`	Optionally A (same as `A | 0`)

Transducer-related:

`A:B`	Cross-product of A and B
`A .o. B`	Composition of A and B
`A.i`	Invert A
`A.u`	Extract upper side (domain) of A
`A.l`	Extract lower side (range) of A
`A .P. B`	Priority union of A and B

Rewrite operations:

`A -> B`	Rewrite strings in A as B
`A (->) B`	Optionally rewrite A as B
`A -> B || C _ D`	Conditional rewrite of A as B (between C and D)
`[..] -> B || C _ D`	Insert a single B between C and D
`A -> B , C -> D ,...`	Multiple simultaneous rewrites (w/ or w/o contexts)
`A -> B ... C`	Markup: insert B before and C after A (w/ or w/o contexts)

Special symbols:

`0 or []`	Epsilon (the empty string)
`?`	The “any” symbol
`.#.`	Word boundary in rewrite rules
`[ and ]`	Grouping symbols for forcing precedence
`" "`	Reserved symbols need to be escaped by quotes

Examples for Spanish

spanish.lexc

```
Multichar_Symbols +N +Sg +Pl

LEXICON Root
alumno      NINFL;
amigo       NINFL;
bebé        NINFL;
chica       NINFL;
composición NINFL;
doctor      NINFL;
flor        NINFL;
interés     NINFL;
lápiz       NINFL;
nación      NINFL;
papel       NINFL;
pez         NINFL;
presidente  NINFL;
rey         NINFL;
tapiz       NINFL;
tisú        NINFL;

LEXICON NINFL
+N+Sg:0  #;
+N+Pl:^s #;
```

spanish.script

```
read lexc spanish.lexc
define Lexicon;

define V [a|e|i|o|u|ü|y|á|é|í|ó|ú];
define Vacc [á|é|í|ó|ú];
define C [b|v|c|c h|d|f|g|h|j|k|l|l l|m|n|ñ|p|q|r|s|t|v|x|z];

define ZRule z -> c || _ %^ s;
define AddE [..] -> e || [C|y|[Vacc - é]] %^ _ s ;
define RemoveAccent á -> a , é -> e , í -> i, ó -> o, ú -> u ||
                            _ V* C+ %^ e s;

define Cleanup %^ -> 0;

define Rules ZRule .o. AddE .o. RemoveAccent .o. Cleanup;

regex Lexicon .o. Rules;
```