



Creating LRs with FSTs

Part VI

Simple applications

Mans Hulden

(University of Helsinki)

Iñaki Alegria

(University of The Basque Country)



Overview

- Spell checking
- Spelling correction
 - Competence errors
 - OCR errors
- Morphological guessers



Spell checking

A morphological analyzer transducer contains on its lower side, a grammar for the legitimate word-forms of the language

We can extract this part with the .l-operator (creating an automaton that only accepts English words):

```
...  
defined Grammar: 1.8 kB. 47 states, 70 arcs, 42 paths.  
foma[0]: regex Grammar.l;  
1.5 kB. 37 states, 52 arcs, 28 paths.  
foma[1]: random-words  
cat  
watch  
watching  
making
```



Spelling correction

- We can re-use the word automaton for creating a rudimentary spelling corrector

An example from a larger English grammar:

- (1) Extract the set of words
- (2) Compose this set with a transducers that makes a limited number of changes
- (3) Run the resulting transducer in the upward direction



A simple corrector

foma[0]: **load defined engwords.foma**

Loading definitions from engwords.foma.

foma[0]: **print defined**

Words528.1 kB. 16151 states, 33767 arcs, 42404 paths.

foma[0]: **define C1 [?* [?:0|0:?:?:?-?] ?*];**

defined C1: 294 bytes. 2 states, 5 arcs, Cyclic.

foma[0]: **regex Words .o. C1;**

21.6 MB. 32302 states, 1415320 arcs, Cyclic.



Testing the corrector

foma[1]: **up**

apply up> **caxt**

cast

cart

cat

apply up> **gdog**

dog

apply up> **twinx**

twins

twine

twin

apply up>



MED built in foma

foma[0]: **regex Words;**

528.1 kB. 16151 states, 33767 arcs, 42404 paths.

foma[1]: med

apply med> **caxt**

Calculating heuristic [h]

Using Levenshtein distance.

cart

caxt

Cost[f]: 1

ca*t

caxt

Cost[f]: 1

cast

caxt

Cost[f]: 1

apply med>



Competence errors

- MED for Basque
- Phonologically similar segments are interchanged at lower cost (e.g. h/0 x/s, ...)

typo.matrix

Insert 2

Substitute 2

Delete 2

Cost 1

:h h: x:j j:x t: :t p:b b:p t:d d:t p:f f:p g:j j:g u:o o:u i:l
l:i c:k k:c z:c c:z m:n n:m b:v v:b s:z z:s x:z z:x s:x x:s

script_med_eu

regex MORPHO.1 ;

#extract lower side

read cmatrix typo.matrix

#attach matrix

med ettxea

#test words...

med lehioa



Competence errors

```
Reading confusion matrix from  
  file 'typo.matrix'  
Calculating heuristic [h]  
Using confusion matrix.
```

```
e*txea  
ettxea  
Cost[f]: 1
```

```
le*ihoa  
lehi*oa  
Cost[f]: 2
```

```
et*xea  
ettxea  
Cost[f]: 1
```

```
le*ihoak  
lehi*oa*  
Cost[f]: 4
```

```
e*txean  
ettxea*  
Cost[f]: 3
```

```
le*iho*  
lehi*oa  
Cost[f]: 4
```



Rules for competence errors

Example of rule (for Basque)

usual mistakes and dialectal phonological rules

used in CALL (Computer Aided Lang. Learning)

Sibilants

define Sibilant z | s | x ;

define H1 h (->) 0 ;

 # hoztu:oztu

define H2 [..] (->) h || [Vowel0 | .#.] _ Vowel0 ;

 # leihoa:lehioa

define Sib Sibilant (->) Sibilant ;

 # etxe:etxe

define CompRules H1 .o. H2 .o. Sib ;



Competence errors in the lexicon

Competence errors in the lexicon

For dialectal uses or idiosyncratic changes

New entries in the lexicon (LEXPLUS)

LEXICAL LEVEL: +Etik

INTERM. LEVEL: +Etikan

+Etik:Etikan # ablative case

...

ihardun:jardun # old standard

...

```
define ENHANCED LEXPLUS .o. RULES .o. COMPET;
```

```
define CompCorr MORPHO.i .o. ENHANCED;
```

Enhanced transducer for correction

zuhaitz+**Etik**
↕ FST1(lex+)
zuhaitz+**Etikan**
↕ FST2(rules)
zu**h**aitzetikan
↕ FST3(comp)
zuaitzetikan
(*from the tree*)

1. Analysis using the enhanced transducer

zuhaitz+**Etik**
↕ FST1(lex)
zuhaitz+**Etik**
↕ FST2(rules)
zuhaitzetik
(*from the tree*)

2. Generation using the standard transducer



Testing the enhanced transducer

foma[1]: `regex CompCorr;`

9.4 kB. 262 states, 522 arcs, Cyclic.

foma[2]: `up zuaitzetikan`

zuhaitzetik

foma[2]: `up suaitxetikan`

zuhaitzetik

foma[2]: `up lehioa`

leihoa

foma[2]: `up lehiotikan`

leihotik



Morphological guessers

- For many applications we need to be quite flexible in parsing OOV items (word forms)
- Simple strategy: reuse rule grammar, use an “open” lexicon with *any* phonologically possible word as a stem
- The more accurate the set of phonologically possible stems, the better (cf. run+V→running, eat+V→eating, *sibl+V→sibling)
- We can simply keep the closed-lexicon morphology and guesser separate, or combine the two



An English guesser

```
[englishguesser.foma]
define Stem [C^<3 V C^<3]+;
define GuessLexicon Stem [
    ["+N" "+Pl"] : ["^" s] |
    ["+N" "+Sg"] : 0 |
    ["+V"] : 0 |
    ["+V" "+3P" "+Sg"] : ["^" s] |
    ["+V" "+PastPart"] : ["^" e d ] |
    ["+V" "+PresPart"] : ["^" i n g ]] ;
```

```
foma[1]: up
apply up> sibling
sibling+V
sibling+N+Sg
sible+V+PresPart
sibl+V+PresPart
```



Guessers with priority union

foma[1]: **regex [Grammar.i .P. Guesser.i].i;**

23.8 kB. 82 states, 1475 arcs, Cyclic.

foma[2]: **up**

apply up> **cats**

cat+N+PI

apply up> **catting**

catt+V+PresPart

catte+V+PresPart

catting+V

catting+N+Sg

apply up> **blarks**

blark+V+3P+Sg

blark+N+PI

apply up>



Guessers with priority union

foma[2]: **regex [Grammar.i .P. Guesser.i].i;**

23.8 kB. 82 states, 1475 arcs, Cyclic.

foma[3]: **up**

apply up> **cats**

cat+N+PI

apply up> **catting**

catt+V+PresPart

catte+V+PresPart

catting+V

catting+N+Sg

apply up> **blarks**

blark+V+3P+Sg

blark+N+PI

apply up>