Spelling and grammar correction with FSTs

Mans Hulden
Ikerbasque/University of the Basque Country
Iñaki Alegria
University of The Basque Country

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

ixa

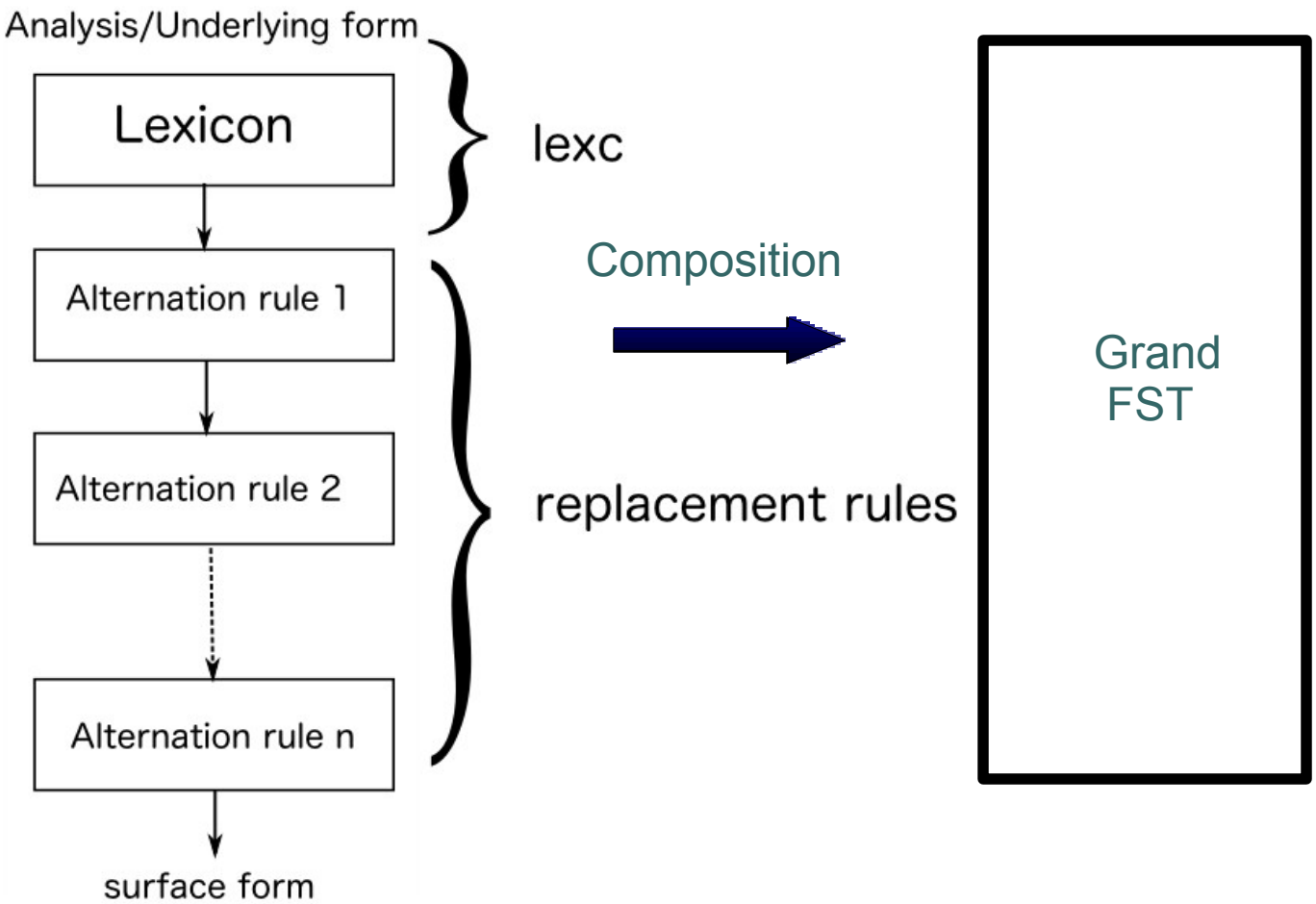
ikerbasque
Basque Foundation for Science

Overview

- Two part-tutorial:
  - First part: unweighted mostly rule-based spell checkers and correctors (design of actual ones)
  - Second part: weighted checkers correctors
- Primary tool in this part: foma finite-state compiler

  http://foma.googlecode.com
- Spell checking
  - language model from word list
  - language model from morphology by projection (lexicon + rules)
- Spelling correction
  - Typos
  - Competence errors
  - OCR errors

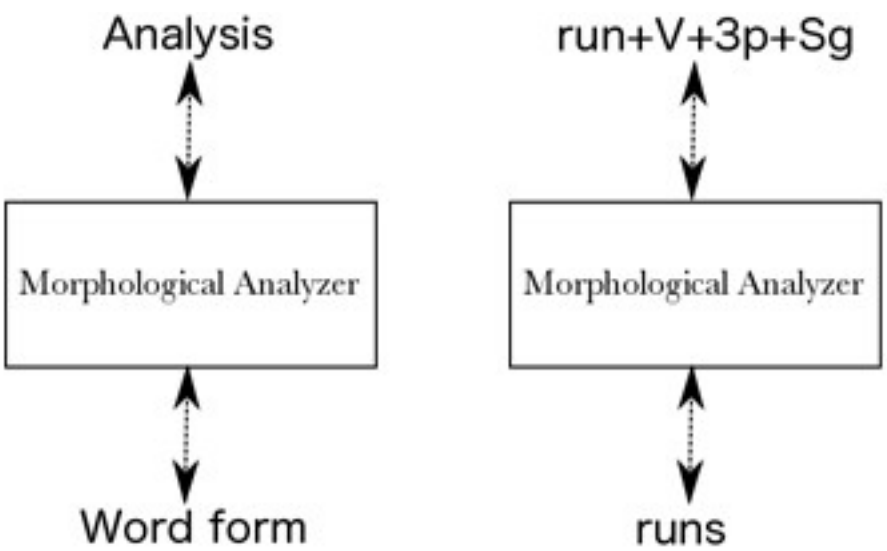
Language model from morphology



For more in-depth explanation, see
https://code.google.com/p/foma/wiki/MorphologicalAnalysisTutorial

Language model from morphology

`Morphology.u` extracts this part



`Morphology.l` extracts this part

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):

```
$ foma -l english.foma
defined Grammar: 2.2 kB. 47 states, 72 arcs, 42 paths.
foma[0]: regex Grammar.l;
1.5 kB. 37 states, 52 arcs, 28 paths.
foma[1]: random-words

[1] begs
[1] talk
[1] panicking
```

Toy grammar. For details, see google code tutorial

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

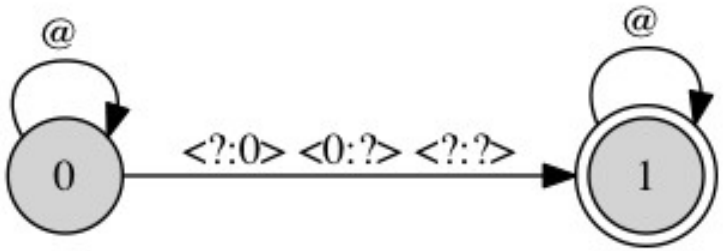
- We can also simply use a word list
  - Example list and compilation into automaton:
  - define W @txt”engwords.txt”;

More detail

- Regular expression trick: define a transducer C1 that makes one change to input words (a deletion, an insertion, or change)
- `define C1 [?* [?:0|0:?|?:?-?] ?*];`

anything (repeat) deletion insertion change anything

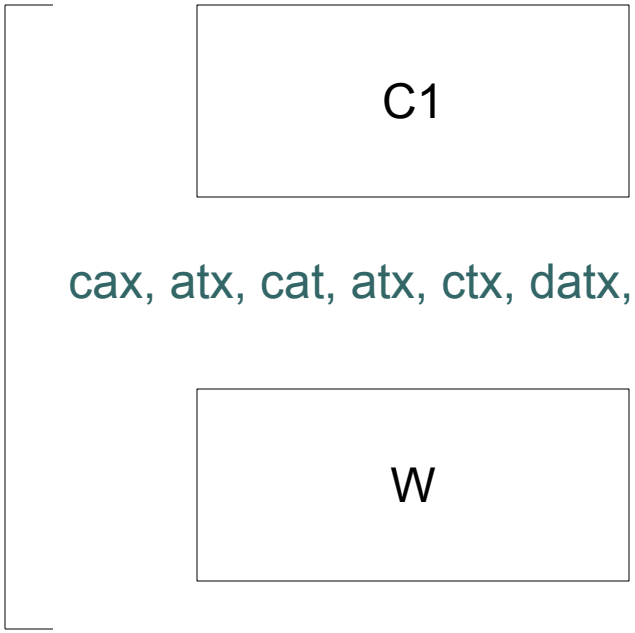
equivalent FST:



Simple spelling correction

- Idea: compose this transducer with a lexicon (W):

catx (input word)



`C1 .o. W`

cax, atx, cat, atx, ctx, datx, catc,... (one change)

cart, cast, cat,... (one change away + exists in lexicon)

Simple spelling correction

Testing:

```
foma[0]: regex C1 .o. W;
21.6 MB. 32302 states, 1415320 arcs, Cyclic.
foma[1]: down
apply down> caxt
cart
cast
cat
apply down> dogx
dogs
dog
```

Simple spelling correction contd.

- What about more edits?
- MED <= 2:

```
define C2 [?* [?:0|0:?|?:?-?] ?*]^<3;
```

```
foma[4]: regex C2 .o. W;
42.7 MB. 48453 states, 2796873 arcs, Cyclic.
```

Original lexicon size: 528.4kB:
the size of the precomposed corrector grows very quickly...

More spelling correction

- Longer edit distances can be lazily evaluated for each word, at some cost of execution speed.
- Idea:

```
foma[1]: regex {catx} .o. C2 .o. W;
2.6 kB. 50 states, 109 arcs, 93 paths.

foma[2]: words
chat
chats
cot
cots
coat
coats
coax
```

Spelling correction

- Or, if we're using foma, we can run minimum-edit distance searches directly against an automaton (with the med/apply command):

```
foma[0]: regex W;
528.4 kB. 16151 states, 33767 arcs, 42404 paths.
foma[1]: med
apply med> grblxal
gradual                 gril--l                 orbital
grblxal                 grblxal                 grblxal
Cost[f]: 3              Cost[f]: 3              Cost[f]: 3
```

Competence errors

- We can also build a more sophisticated error model by specifying weights for different substitutions with *med/apply med*
- 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: s:z z:s x:z z:x s:x x:s
```

script_med_eu

```
regex MORPHO.l ;              # extract lower side of morphology
read cmatrix typo.matrix      # attach matrix
```

Competence errors

```
apply med> leioa
leion
leioa
Cost[f]: 1

leiok
leioa
Cost[f]: 1
```

without confusion matrix

```
apply med> leioa
leihoa
lei-oa
Cost[f]: 1

leion
leioa
Cost[f]: 2
```

with confusion matrix

Manual rules for correction

- We can also specify the “error model” using arbitrary rewrite rules, perhaps in conjunction with edit distance.

```
#transceive, receive, conceive, etc. + teh -> the
define CommonErrors [ c i e v e -> c e i v e ,,
                      t e h -> t h e  || .#. _ .#. ];

define Corrector [CommonErrors .o. W] .P. [C1 .o. W] ;
```

“priority union”: “if CommonPatterns don't produce an output with W, accept `[C1 .o. W]`'s output. The left hand side has “priority” in the union.

Combining manual rules and MED

```
apply down> recieve
relieve
```

```
apply down> recieve
receive
relieve
```

```
apply down> recieve
receive
```

Model: C1 .o. W

(MED 1 search)

[CommonErrors|C1] .o. W

(MED 1 *or* common errors)

[CommonErrors .o. W] .P. [C1 .o. W]

(Common errors have priority over MED 1)

Rules for competence errors using existing morphologies

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:etze

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 (LEX+)
#   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**ait**z**etikan

FST3(comp)

zuaitxetikan

(*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 -l rules_compet_eu

```
foma[2]: up zuaitxetikan
zuhaitzetik
foma[2]: up suaitxetikan
zuhaitzetik
foma[2]: up lehioa
leihoa
foma[2]: up lehiotikan
leihotik
```

Normalizing variants

- Is useful for many applications:
  - Digital libraries: dialectal, diachronic, medical texts...
  - New media: SMS, tweets...
- Non-standard lexicon and rules
- Additional rules
  - Phonological/orthographic ones near to the surface
  - Morphological ones near to the lexicon.
- Simplified example for Biscayan Basque:

Lexicon: 2 strategies:

- standard+non-standard
  - similar to competence errors
- standard <->non-standard
  - i.e. Biscayan Basque (Alegria et al., 2010)

```
A -> e || \i _ "+" OpenVowel ;
# alabA+a:alabe+a
```

```
gaude:gagoz     # (standard:non-standard) ~ competence error
gagoz:gaude     # (standard:non-standard) ~ Biscayan standard
```

OCR

Parallel rules

```
define PARAL0 [ m (->) r n , n (->) r i ] ;
define SEQ0 [ m (->) r n .o. n (->) r i ] ;
```

Example of OCR errors

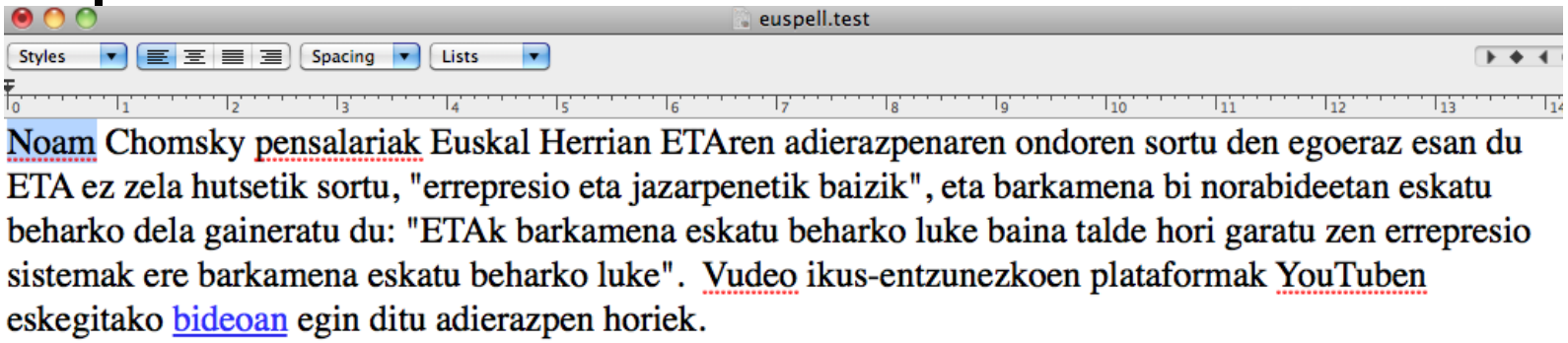
```
define PARAL [c (->) e , e (->) c , c (->) o , o (->)
   c , l (->) i , i (->) l , l (->) 1 , 1 (->) l , r n
   (->) m , m (->) r n , r i (->) n , n (->) r i , c l
   (->) d , d (->) c l , o (->) "0" , "0" (->) o , r t
   (->) n , r t (->) i t , r m (->) n n , r r i (->) m ,
   m (->) r r i , n i (->) m , l i (->) h , u (->) i r ]
   ;
```

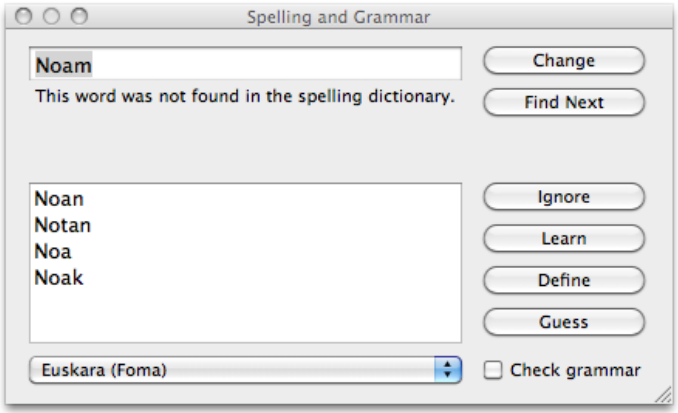
foma -l ocr_simple

Spelling correction APIs (code)

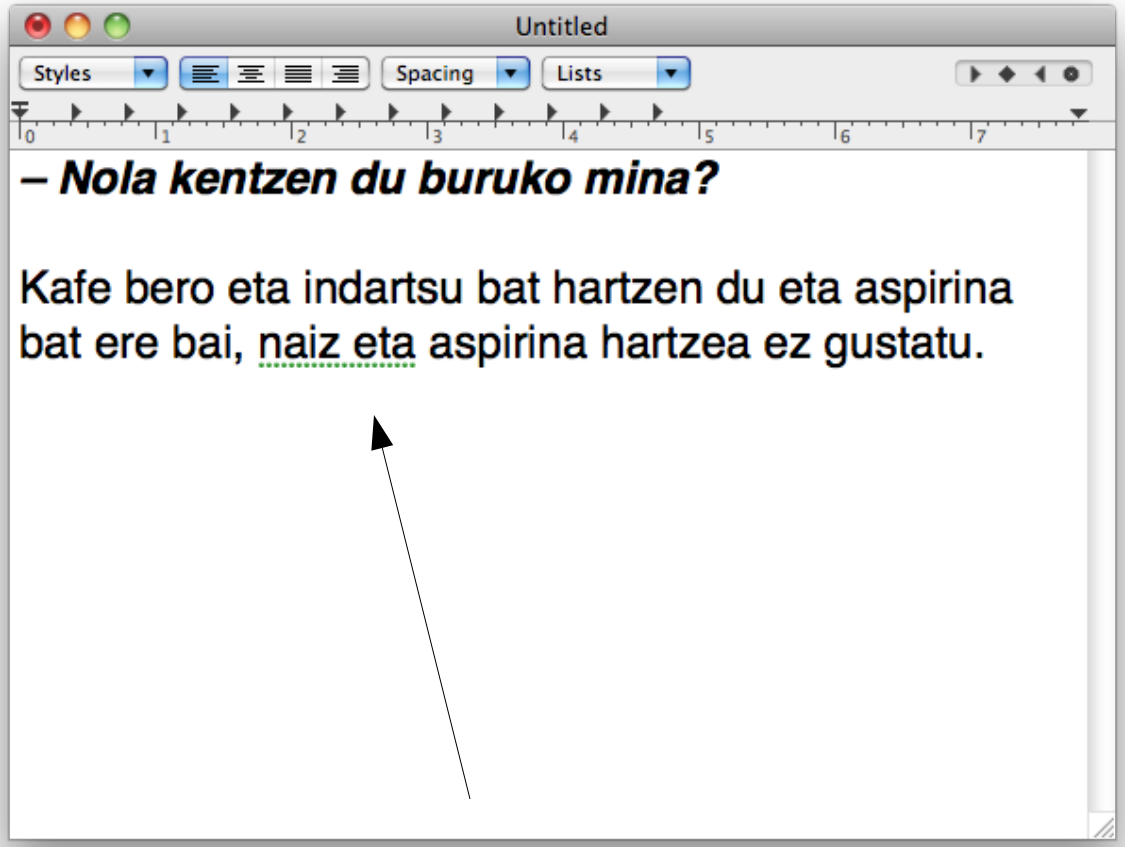
```
#include "fomalib.h"

/* Variables */
struct apply_handle *spell_apply_handle;
struct apply_med_handle *spell_apply_med_handle;
char *word;

/* Initialization */
spell_fsm = fsm_read_binary_file("mywords.foma”));
spell_apply_handle = apply_init(spell_fsm);
…
/* Check spelling */
if (apply_down(spell_apply_handle, word) == NULL)
    return 0;
...
/* Suggest alternatives using built-in med */
suggestedword = apply_med(spell_apply_med_handle, word);
```

Integration with system spell checkers





Integration with system grammar checkers



common competence error: should be “nahiz eta,” but “naiz” is also a real word, and so we give a different warning...