Introduction to Natural Language Processing

Out of Vocabulary Forms

Spelling Error correction

Jean-Cédric Chappelier

Jean-Cedric.Chappelier@epfl.ch

and

Martin Rajman

Martin.Rajman@epfl.ch

Artificial Intelligence Laboratory

Contents

Out of Vocabulary forms

- Out of Vocabulary (OoV) forms matter: they occur quite frequently (e.g. $\simeq 10\%$ in newspapers)

  What do they consist of?

  – spelling errors: *foget*, *summmary*, *usqge*, ...

  – neologisms: *Internetization*, *Tacherism*, ...

  – borrowings: *gestalt*, *rendez-vous*, ...

  – forms difficult to exhaustively lexicalize: (numbers,) proper names, abbreviations, ...

- identification based on patterns is not well-adapted for all OoV forms

  ☞ We will focus here on spelling errors, neologisms and borrowings

Spelling errors and neologisms

- for **spelling errors** (resp. **neologisms**), distortions (resp. derivations) are modelled by *transformations*, i.e. rewriting rules (sometimes weighted)

  Example:

  – Transposition (distortion): `XY` $\rightarrow$ `YX [1.0]`

    where `X` and `Y` stands for variables

  – tripling (distortion): `XX` $\rightarrow$ `XXX [1.0]`

  – name derivation: `ize:INF` $\rightarrow$ `ization:N [1.0]`

- a given lexicon (regular language) and a set of transformations define the edit space to be explored

  ☞ The aim is to find the position of the OoV forms in the edit space with respect to known (lexicalized) forms (*neighbourhoods*, *similarity*, *distance*)

Spelling errors and neologisms (2)

- if the transformation set is simple enough: automatic (or semi-automatic) learning of the transformation set is possible

  Examples:

  - morphological rules for Spanish
  - transformations for spelling error correction after OCR

Borrowings

For **borrowings** ☞ **identification of the source language**

[when no large coverage lexica are avalaible for the other languages, but only representative texts]

Decomposition into n-grams of characters: *Example*: for trigrams

dribble $\longrightarrow$ (dri,rib,ibb,bbl,ble)

In practice: n varies from 2 to 4

From reference corpora, computation of a frequency matrix (n-gram $\times$ language)

☞ approximation of likelihood of a word to belong to a given language

Example for trigrams:

$$V(\text{dribble}, L) = P(\text{dribble}|L) = P(\text{dri}|L) \cdot \frac{P(\text{rib}|L)}{P(\text{ri}|L)} \cdot \ldots \cdot \frac{P(\text{ble}|L)}{P(\text{bl}|L)}$$

Trigrams for French, English, German and Spanish: 87% discrimination accuracy

n-gram approach

A sequence of xs (letters, words, ...; was letters in the former case)

$(n-1)$-order Markov assumption:

$$P(x_1 \cdots x_N) = P(x_1 \cdots x_{n-1}) \cdot \prod_{i=n}^{N} P(x_i | x_{i-n+1} \cdots x_{i-1})$$

i.e. use this as a score to compare sequences ($n \geq 2$):

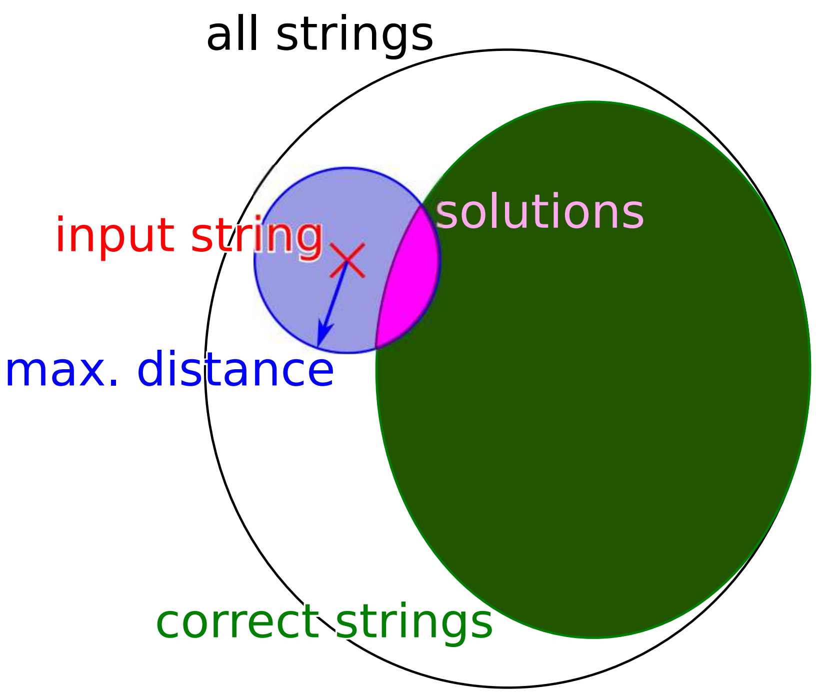
$$\frac{\displaystyle\prod_{i=1}^{N-n+1} P(x_i \cdots x_{i+n-1})}{\displaystyle\prod_{i=2}^{N-n+1} P(x_i \cdots x_{i+n-2})}$$

$P(x_i \cdots x_{i+n-1})$ being estimated on some corpus

Reminder: $P(x_i \cdots x_{i+n-2}) = \sum_{x} P(x_i \cdots x_{i+n-2}\, x)$

Contents

Spelling error correction



Two approaches:

	Exact lexicon-based	Statistical
correct forms:	lexicon	any string
metric:	edit distance	probability

In this lecture: exact, lexicon-based

For statistical: see http://norvig.com/spell-correct.html

Edit distance

also called Levenshtein distance

☞ distance between 2 forms

= minimal number of transformations to change one into the other

Example of transformations:

- insertion: *exmple* $\longrightarrow$ *example*
- deletion: *example* $\longrightarrow$ *exmple*
- substitution: *exemple* $\longrightarrow$ *example*
- transposition: *exmaple* $\longrightarrow$ *example*

Computation of edit distance (1)

Notations:

X_i: ith char of string X

X_i^j: if $i \leq j$: substring X_i,...,X_j; empty string otherwise

Example: X = castle

X_3 = s $\quad$ X_4^6 = tle $\quad$ X_1^4 = cast $\quad$ X_1^0 = ε

Computation of the distance $D(X, Y)$ by dynamic programming:

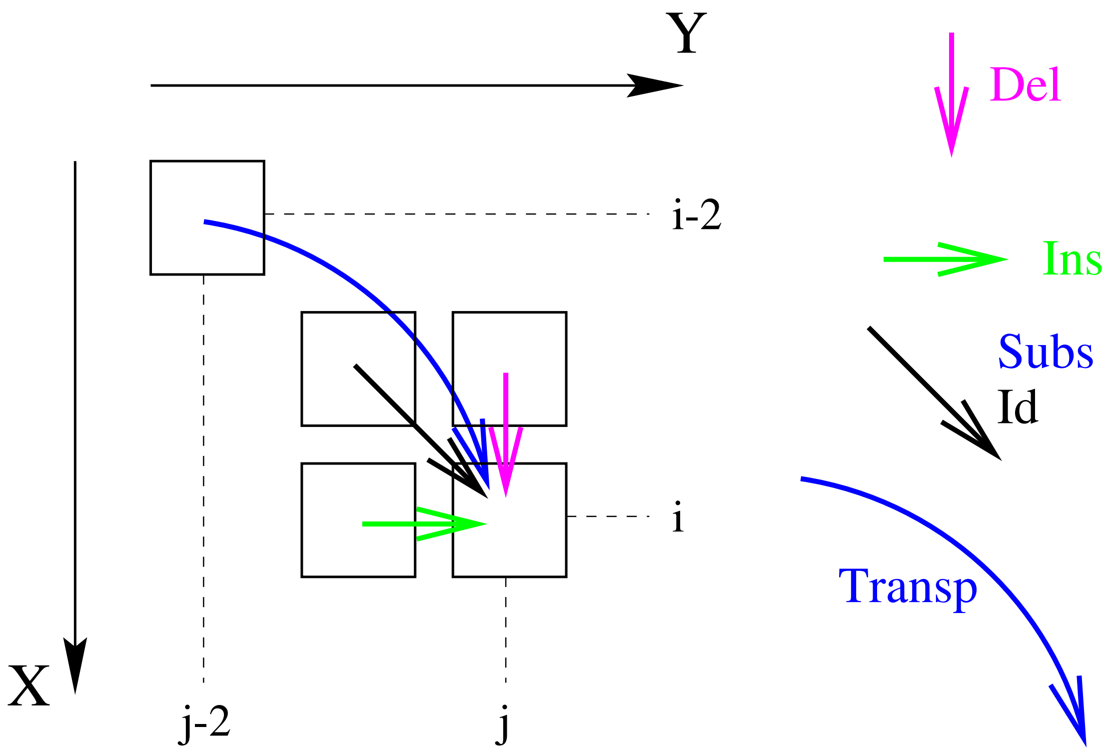
☞ step by step in a chart m where each cell m_{ij} contains the distance between the two substrings X_1^i and Y_1^j :

$$m_{ij} = D(X_1^i, Y_1^j)$$

Computation of edit distance (2)

$D(X_1^0;Y_1^j)$	= j	initialization
$D(X_1^i;Y_1^0)$	= i	
$D(X_1^i;Y_1^j)$	= $D(X_1^{i-1};Y_1^{j-1})$	**if** $X_i = Y_j$ (equality)
	= $1 + \min\left\{D(X_1^{i-2};Y_1^{j-2}), D(X_1^{i-1};Y_1^{j}), D(X_1^{i};Y_1^{j-1})\right\}$	**else if** $i \geq 2$ and $j \geq 2$ and $X_{i-1} = Y_j$ and $X_i = Y_{j-1}$ (transposition, deletion, insertion)
	= $1 + \min\left\{D(X_1^{i-1};Y_1^{j-1}), D(X_1^{i-1};Y_1^{j}), D(X_1^{i};Y_1^{j-1})\right\}$	**else** (substitution, deletion, insertion)

Computation order



☞ several possible ways of computing: rowwise, columnwise or diagonal

Computation of edit distance (3)

Example, columnwise:

for all i from 0 to $|X|$ (size of X) **do**

$m_{i0} = i$

for all j from 1 to $|Y|$ **do**

$m_{0j} = j$

for all i from 1 to $|X|$ **do**

if $X_i = Y_j$ **then**

$m_{ij} = m_{i-1,j-1}$

else if $i \geq 2$ and $j \geq 2$ and $X_{i-1} = Y_j$ and $X_i = Y_{j-1}$ **then**

$m_{ij} = 1 + \min\left\{m_{i-2,j-2};\ m_{i,j-1};\ m_{i-1,j}\right\}$

else

$m_{ij} = 1 + \min\left\{m_{i-1,j-1};\ m_{i,j-1};\ m_{i-1,j}\right\}$

Return $m_{|X|,|Y|}$

Edit Distance (example)

D(exmple;exemple)

		e	x	e	m	p	l	e
	0	1	2	3	4	5	6	7
e	1	0	1	2	3	4	5	6
x	2	1	0	1	2	3	4	5
m	3	2	1	1	1	2	3	4
p	4	3	2	2	2	1	2	3
l	5	4	3	3	3	2	1	2
e	6	5	4	3	4	3	2	1

D(exmaple;example)

		e	x	a	m	p	l	e
	0	1	2	3	4	5	6	7
e	1	0	1	2	3	4	5	6
x	2	1	0	1	2	3	4	5
m	3	2	1	1	1	2	3	4
a	4	3	2	1	1	2	3	4
p	5	4	3	2	2	1	2	3
l	6	5	4	3	3	2	1	2
e	7	6	5	4	4	3	2	1

Spelling error correction using a FSA

Problem: approximative search of lexicalized (surface) forms

= within a max. distance range

i.e. Fault-tolerant recognition (within a regular language):

Find **all** ending paths such that the corresponding string is within a distance range less than $\underline{\theta}$ of the given input string.

Remark: a trie is a special case of FSA

Pruning criteria: cut-off edit distance

To make it useful in practice $\Rightarrow$ Fast $\Rightarrow$ good pruning

☞ cut-off edit distance [Oflazer 1996]

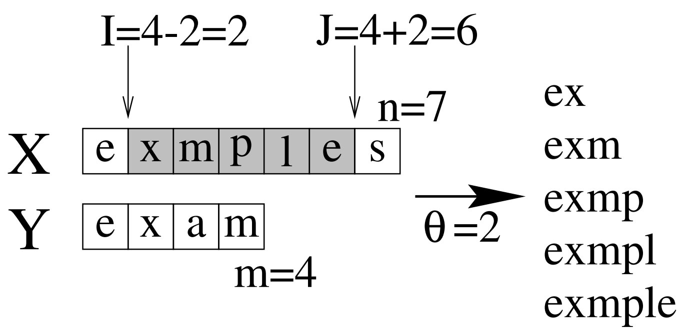
$$D_c(X_1^n, Y_1^m) = \min_{I(m) \leq i \leq J(m)} D(X_1^i; Y_1^m)$$

$$I(m) = \min(n, \max(1, m - \theta)) \qquad J(m) = \min(n, \max(1, m + \theta))$$

Important property:

$$D_c(X, Y) > \theta \implies \forall Z \; D(X, Y + Z) > \theta$$

Cut-off Edit Distance: example



Y

		e	x	a	m
	0	1	2	3	4
e	1	0	1	2	3
x	2	1	0	1	2
m	3	2	1	1	1
p	4	3	2	2	2
l	5	4	3	3	3
e	6	5	4	4	4
s	7	6	5	5	5

X

$$D_c(X,Y) = \min\{2,1,2,3,4\} = 1$$

Walk through a FSA within a θ distance range

Original algorithm

Input: a string to be corrected (X), a lexicon in the form of a FSA and a maximal error threshold (θ)

Push(ε, q_0) — ε: empty string, q_0: inital state

while Stack is not empty **do**

  Pop(Z, p)

  for all $a \in \Sigma$ **do**

    for all q such that $\delta(p, a) = q$ **do**

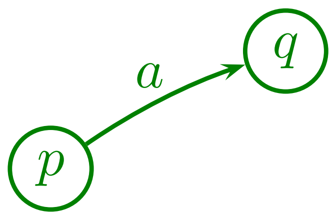
      $Y \leftarrow Z + a$

      if $D_c(X, Y) \leq \theta$ **then**

        Push(Y, q)

        if $q \in F$ **and** $D(X, Y) \leq \theta$ — F: set of final states **then**

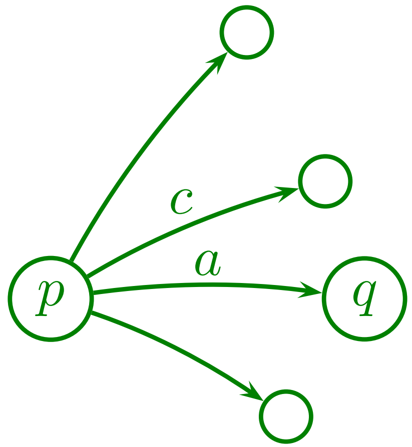
          Add Y to solutions



Walk through a FSA within a θ distance range

Prefix-compatible Depth-first version

Push($\varepsilon, \varepsilon, q_0$)
while Stack is not empty **do**
  Pop(Z, c, p)
  $(q, a) =$ nextAfter(p, c)
  if $(q, a) \neq \emptyset$ **then**
    Push(Z, a, p)
    $Y \leftarrow Z + a$
    if $D_c(X, Y) \leq \theta$ **then**
      Push(Y, ε, q)
      if $q \in F$ **and** $D(X, Y) \leq \theta$ **then**
        Add Y to solutions



where:

nextAfter$(p, c) = \mathrm{Argmin}_\alpha \{(q, \alpha) \in Q \times \Sigma$ such that $\alpha > c$ and $\delta(p, \alpha) = q\}$

Implementation issues

① original version (simpler) .vs. Prefix-compatible Depth first (faster)

② Efficient computation of D_c with the previously described chart :

☞ recomputation of the last column (m) **only**

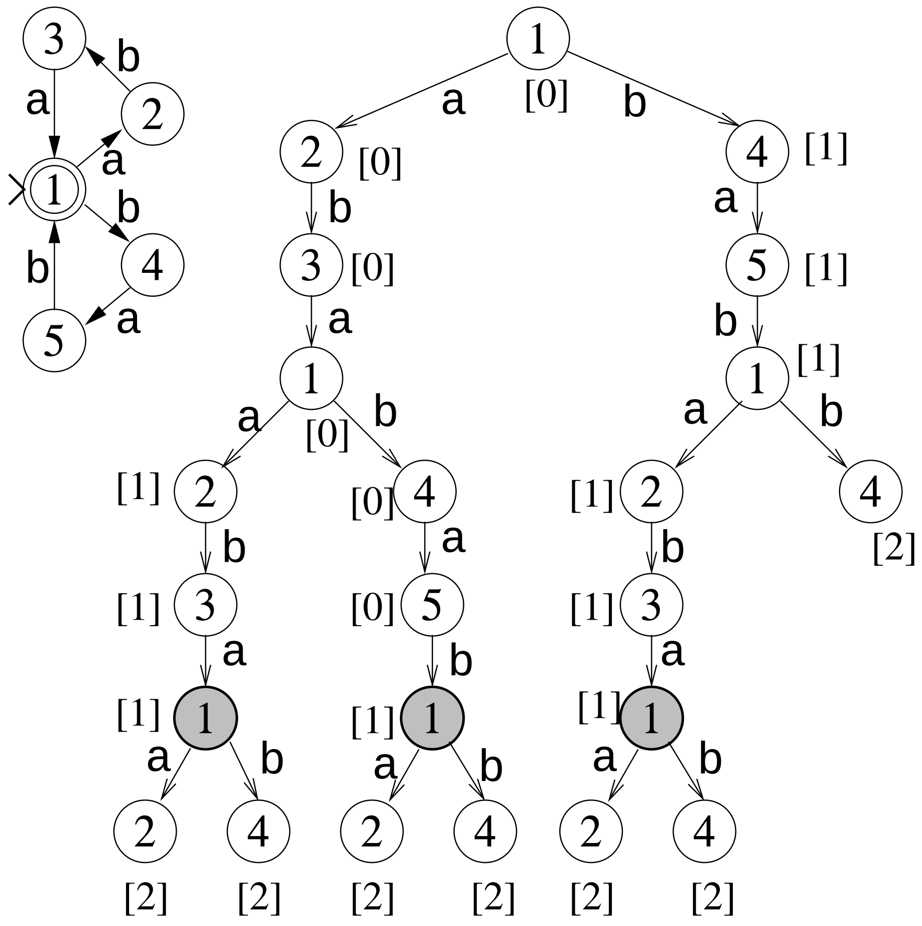
(pay attention to the backtrack case! Original version $\longrightarrow$ the last *two* columns)

☞ Computation of D and D_c in the **same** loop

③ $Y \leftarrow Z + a$: beware (local copies, pointers etc...).

Similarly, do not naively implement "Push(Y, q)".

④ In some languages (especially POO): it could be worth transposing the algorithm: Y (which is changing) for rows and X for columns



X=ababa .vs. (aba|bab)* for $\theta=1$

		a	b	a	a	b	a
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
b	2	1	0	1	2	3	4
a	3	2	1	0	1	2	3
b	4	3	2	1	1	1	2
a	5	4	3	2	1	1	1

Solutions ☞ abaaba, ababab, bababa

Contents

- Out of Vocabulary Forms
- Spelling Error Correction
  - ✈ Edit distance
  - ✈ Spelling error correction with FSA
  - ➤ Weighted edit distance

Limitations?

➥ weighting

Example: diacritics, uppercase

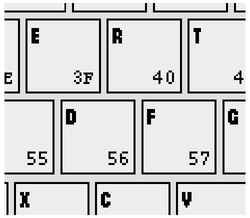
eleves $\longrightarrow$ *élèves* *aloves* $\longrightarrow$ *élèves*

➥ specific transformations

Example: typing errors

tupe $\longrightarrow$ *type* more generally: ***de**uit* $\longrightarrow$ *fruit*

*us**q**ge* $\longrightarrow$ *usage*



➥ whitespaces

theothers $\longrightarrow$ *the others* *othe rs* $\longrightarrow$ *others*

☞ 3 aspects of the **same** problem

Solution: generalization of the edit distance: weighted edit distance

Weighted Edit Distance

weighted transformations such that :

- $C(\mathrm{Id}) = 0$
- $C(\mathrm{f}) > 0 \qquad \mathrm{f} \neq \mathrm{Id}$
- $C(\mathrm{f}^{-1}) = C(\mathrm{f})$
- $C(\mathrm{f} \circ \mathrm{g}) = C(\mathrm{f}) + C(\mathrm{g})$

$$D(X;Y) = \min_{\mathrm{f}:Y=\mathrm{f}(X)} C(f)$$

☞ It is actually a distance on Σ^*

Difference with the preceding distance: $C(\mathrm{f})$ is not necessarily the same (= 1).

Remarks

❶ Distance on $\Sigma^* \Rightarrow \forall X\ Y, \quad \exists f : Y = f(X)$

True if Ins and Del are in the transformation set

❷ non overlapping transformations

i.e. cannot apply a transformation to the result of the previous transformation

Counter-Example: $\mathtt{ba} \stackrel{\mathrm{Transp}}{\longrightarrow} \mathtt{ab} \stackrel{\mathrm{Sub}}{\longrightarrow} \mathtt{ac}$

Coherence Constraints

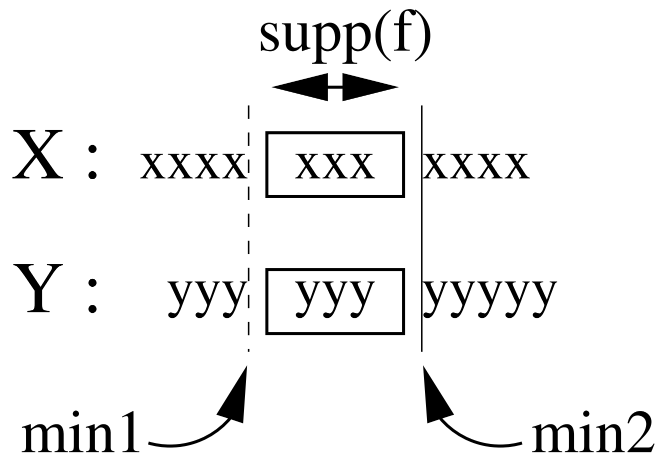
"Semantic Integrity":

- ❒ $C(\mathrm{Del}) + C(\mathrm{Ins}(x)) > C(\mathrm{Sub}(x))$
- ❒ $C(Split) < C(\mathrm{Ins}(x))$ ($\Rightarrow C(Merge) < C(\mathrm{Del})$)
- ❒ $C(\mathrm{Transp}) < C(\mathrm{Ins}(x)) + C(\mathrm{Del})$

☞ Introduction a new f such that $f = \circ_i f_i$, is useful if and only if

$$C(f) < \sum_i C(f_i)$$

Weighted Edit Distance: computation



$$\text{min2} = \min_{f} \left[\, \text{min1}(f) + C(f) \,\right]$$

(min1 and min2 are the values stored in the chart)

Weighted Edit Distance: computation (2)

$D(X_1^0; Y_1^j)$	=	j	initialization
$D(X_1^i; Y_1^0)$	=	i	
$D(X_1^i; Y_1^j)$	=	$D(X_1^{i-1}; Y_1^{j-1})$	if $X_i = Y_j$ (equality)
	=	$C(f) + \min\{\min_1(f)\}$	for all applicable transformations f of the same weight
	=	...	for all possible weights.

increasing $C(f)$

☞ The optimization lies in the grouping of similar cases: same weight and compatible transformations (Example: previously Transp and Sub were incompatible because $C(\mathrm{Transp}) < 2\,C(\mathrm{Sub})$. But each of them is compatible with Del and Ins.)

Note: $\{\min_1(f)\}$ is the set of all the minimal values for all possible f at this point; they shall, of course, already be computed at this point (loop condition)

Example

D(example;exemple)

		e	x	e	m	p	l	e
	0	1	2	3	4	5	6	7
e	1	0	1	2	3	4	5	6
x	2	1	0	1	2	3	4	5
a	3	2	1	1	2	3	4	5
m	4	3	2	2	1	2	3	4
p	5	4	3	3	2	1	2	3
l	6	5	4	4	3	2	1	2
e	7	6	5	4	4	3	2	1

D(exémple;exemple)

		e	x	e	m	p	l	e
	0	1	2	3	4	5	6	7
e	1	0	1	2	3	4	5	6
x	2	1	0	1	2	3	4	5
é	3	2	1	0.1	1.1	2.1	3.1	4.1
m	4	3	2	1.1	0.1	1.1	2.1	3.1
p	5	4	3	2.1	1.1	0.1	1.1	2.1
l	6	5	4	3.1	2.1	1.1	0.1	1.1
e	7	6	5	4	3.1	2.1	1.1	0.1

C(é↔e)=0.1

Keypoints

- One has to handle out of vocabulary forms
- Edit (Levenshtein) distance, weighted edit distance
- Spelling error correction with FSA

References

K. Oflazer, *Error-tolerant Finite State Recognition with Applications to Morphological Analysis and Spelling Correction*, Computational Linguistics, Volume 22, Number 1, 1996.

Section 8.2 in M. Rajman editor, ”Speech and Language Engineering”, EPFL Press, 2006.

Sections 3.10 and 3.11 in D. Jurafsky and J. H. Martin, ”Speech and Language Processing”, Prentice Hall, 2008 (2nd edition).

Section 3.3 in C. D. Manning, P. Raghavan and H. Schütze, ”Introduction to Information Retrieval”, Cambridge University Press. 2008