Distance entre deux mots

(introduction à la programmation dynamique)

Philippe Langlais

felipe@iro.umontreal.ca

September 25, 2011

Plan

Distance de Hamming et de Jaro-Winkler

Distance de Levenshtein

Plus longue sous-chaine commune

Applications de la distance d'édition

Algorithme du Soundex

- Pour une très bonne exposition aux algorithmes du texte, lire: (Crochemore, Hancart, and Lecroq 2001).

Distance de Hamming et de Jaro-Winkler

Distance de Levenshtein

Plus longue sous-chaine commune

Applications de la distance d'édition

Algorithme du Soundex

Distance entre deux mots

Définition

- Une fonction $d : \Sigma^\star \times \Sigma^\star \rightarrow \mathscr{R}$ est une distance sur $\Sigma^\star$ si pour tout $u, v \in \Sigma^\star$ on a:
  - $d(u,v) \geq 0$
  - $d(u,v) = 0 \iff u = v$
  - $d(u,v) = d(v,u)$
  - $d(u,v) \leq d(u,w) + d(w,v) \ \forall w \in \Sigma^\star$

- Si l'une de ces propriété n'est pas vérifiée, on parle plutôt de mesure de dissimilarité

Distance de Hamming

entre deux mots de même taille

- Définie pour deux mots u et v de même longueur par le nombre de positions p où $u_p \neq v_p$

- Exemples:
  - d_h(ahuri,aluni) = 2
  - d_h(aluni,ahuri) = 2
  - d_h(salade,salace) = 1
  - d_h(salace,sagace) = 1
  - d_h(salade,sagace) = 2

Distance de Jaro-Winkler

d'après `http://fr.wikipedia.org/wiki/Distance_de_Jaro-Winkler`

- Adaptation par William E. Winkler (1999) d'une distance proposée par Matthew A. Jaro (1989)

- Adaptée pour les chaînes courtes (noms propres, mots de passe, etc.)
- Pas une distance (pas symétrique)

↪ Mesure de similarité entre 0 (absence de similarité) et 1

- Exemples:
  - d_{jw}(intimité,inimitié) = 0.96
  - d_{jw}(opinion,opignon) = 0.95
  - d_{jw}(bernard,bertrand) = 0.92
  - d_{jw}(lagacé,leveillé) = 0.57

Distance de Jaro

$d_j(x, y)$

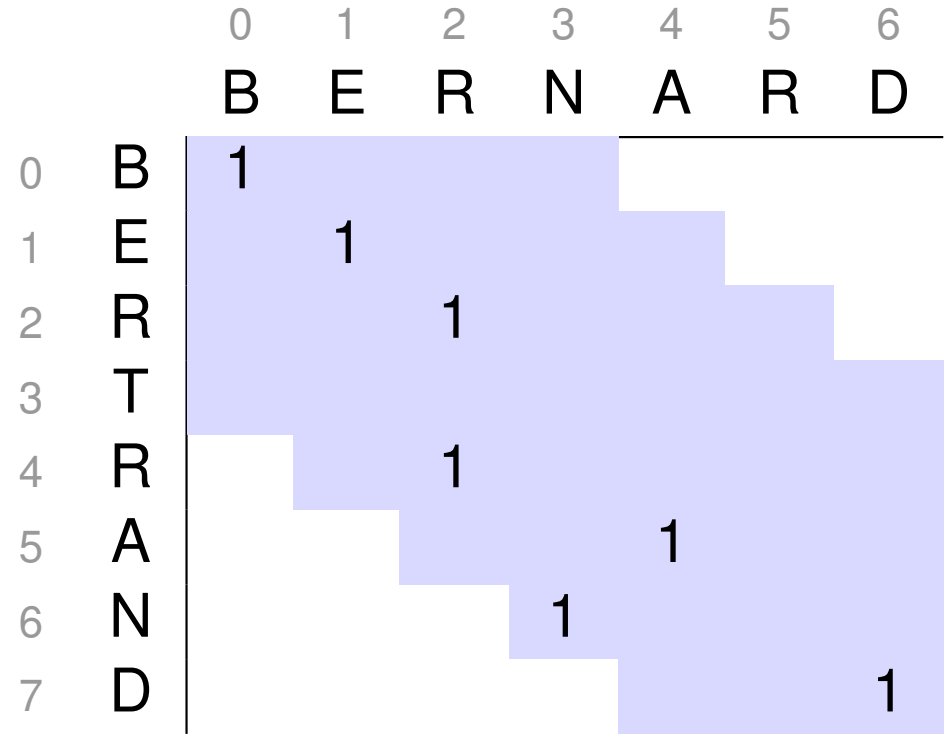
$$d_j(x, y) = \frac{1}{3}\left(\frac{m}{|x|} + \frac{m}{|y|} + \frac{m-t}{m}\right)$$

Où:

- m est le nombre de caractères qui se correspondent dans x et y:
  - **Def:** x_i et y_j se correspondent **ssi**:
    - $x_i = y_j$
    - $|i-j| \leq \rho \equiv \left\lfloor \frac{max(|x|,|y|)}{2} \right\rfloor$ -1
- t est le nombre de transpositions, cad: le nombre de caractères qui diffèrent lorsqu'on dresse la séquence des caractères de chaque mot qui ont un correspondant dans l'autre chaîne, divisé par 2

Distance de Jaro

$d_j(BERTRAND, BERNARD)$



BERTRAND	$\rightarrow$	B	E	R	R	A	N	D
BERNARD	$\rightarrow$	B	E	R	N	A	D	

- $\rho = 3$, matchs = 7, transpositions = 3 ($m = 7$, $t = 3/2 = 1.5$)
- $d_j(BERTRAND, BERNARD) = \frac{1}{3}\left(\frac{7}{7} + \frac{7}{8} + \frac{5.5}{7}\right) = 0.89$

Distance de Jaro-Winkler

$d_{jw}(BERTRAND, BERNARD)$

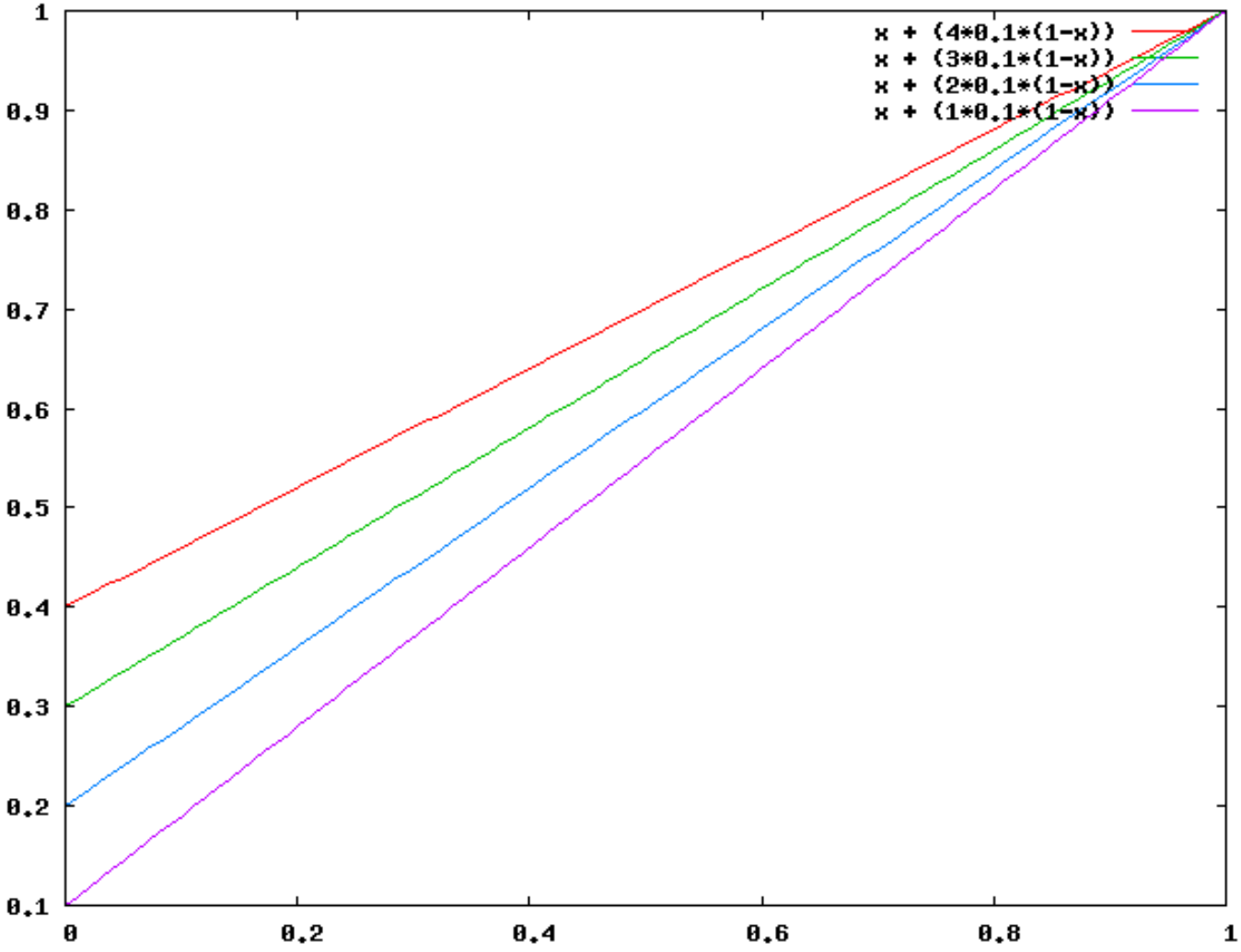
- Soit l la taille du plus grand préfixe commun entre les deux chaînes ($l \leq 4$)
- Soit p un coefficient ($p = 0.1$):

$$d_{jw}(x, y) = d_j(x, y) + (l \times p \times (1 - d_j(x, y)))$$

intuition: favoriser les mots qui partagent un ”grand” préfixe

- $l = 3$ dans notre exemple (BERNARD, BERTRAND)
- d_{jw}(BERTRAND,BERNARD) = 0.92

Distance de Jaro-Winkler



- importance de la taille du préfixe entre les deux chaînes
- x est ici d_j

Distance de Hamming et de Jaro-Winkler

Distance de Levenshtein

Plus longue sous-chaine commune

Applications de la distance d'édition

Algorithme du Soundex

Distance d'édition entre x et y

Opérations d'édition permettant de transformer un mot x en un mot y.

substitution substitution d'une lettre (l_1) de x par une lettre de (l_2) y; soit sub(l_1,l_2) son coût
Ex: x =narine $\Rightarrow$ y =marine en substituant n à m dans x en position 1 dans x

insertion insertion d'une lettre (l) de y dans x; soit ins(l) son coût
Ex: x =marine $\Rightarrow$ y =martine en insérant t en position 4 dans x

suppression opération inverse à la précédente, del(l) son coût

- Le nombre minimum d'opérations à appliquer pour transformer une chaîne en une autre est appelée la distance de Levenshtein Levenshtein 1966, ou encore la distance d'édition (edit distance).

$d_L(voiture, moteur)$?

- Séquence de 4 opérations qui transforme la chaîne voiture en la chaîne moteur.

x	opération	coût
VOITURE	del(E)	1
VOITUR	sub(T,E)	1
VOIEUR	sub(I,T)	1
VOTEUR	sub(V,M)	1
MOTEUR		

Chaque opération, autre que *sub(a,a)* a un coût unitaire.

- Le coût de cette séquence de transformations est 4.
- Peut-on faire mieux ?

$d_L(voiture, moteur)$?

- Séquence de 4 opérations qui transforme la chaîne voiture en la chaîne moteur.

x	opération	coût
VOITURE	del(E)	1
VOITUR	sub(T,E)	1
VOIEUR	sub(I,T)	1
VOTEUR	sub(V,M)	1
MOTEUR		

Chaque opération, autre que *sub(a,a)* a un coût unitaire.

- Le coût de cette séquence de transformations est 4.
- Peut-on faire mieux ?
- Réponse en calculant une table d'édition

$d_L(voiture, moteur)$?

Table d'édition

- $T[i,j] = d_L(x[1 \ldots i], y[1 \ldots j])$

	(6)	M	O	T	E	U	R
(7)		(1,d)	(2,d)	(3,d)	(4,d)	(5,d)	(6,d)
V	(1,i)	(1,s)	(2,d)	(3,d)	(4,d)	(5,d)	(6,d)
O	(2,i)	(2,i)	(1,=)	(2,d)	(3,d)	(4,d)	(5,d)
I	(3,i)	(3,i)	(2,i)	(2,s)	(3,d)	(4,d)	(5,d)
T	(4,i)	(4,i)	(3,i)	(2,=)	(3,d)	(4,d)	(5,d)
U	(5,i)	(5,i)	(4,i)	(3,i)	(3,s)	(3,=)	(4,d)
R	(6,i)	(6,i)	(5,i)	(4,i)	(4,i)	(4,i)	(3,=)
E	(7,i)	(7,i)	(6,i)	(5,i)	(4,=)	(5,d)	(4,i)

- d(élétion)[1], i(nsertion), s(ubstitution), = (égalité)

[1] Je sais c'est une horreur...

$d_L(voiture, moteur)$?

Récurrence

- On peut démontrer que:

$$T[i,j] = min \begin{cases} T[i-1,j-1] + sub(x[i], y[j]) \\ T[i-1,j] + ins(x[i]) \\ T[i,j-1] + del(y[j]) \end{cases}$$

- Cas particuliers aux bornes:

$$\begin{aligned} T[0,0] &= 0 \\ T[i,0] &= T[i-1,0] + ins(x[i]) \\ T[0,j] &= T[0,j-1] + del(y[j]) \end{aligned}$$

$d_L(voiture, moteur)$?

Solution 1 (descendante)

Require: $X = x_1, \ldots, x_N$ et $Y = y_1, \ldots, y_M$

function $Solve(i, j)$

if $i = 0$ **then**

if $j = 0$ **then**

return retourne 0

else

return $Solve(0, j-1) + del(Y[j])$

else

if $j = 0$ **then**

return $Solve(i-1, 0) + ins(X[i])$

else

$i_1 \leftarrow Solve(i-1, j-1) + sub(X[i], Y[j])$

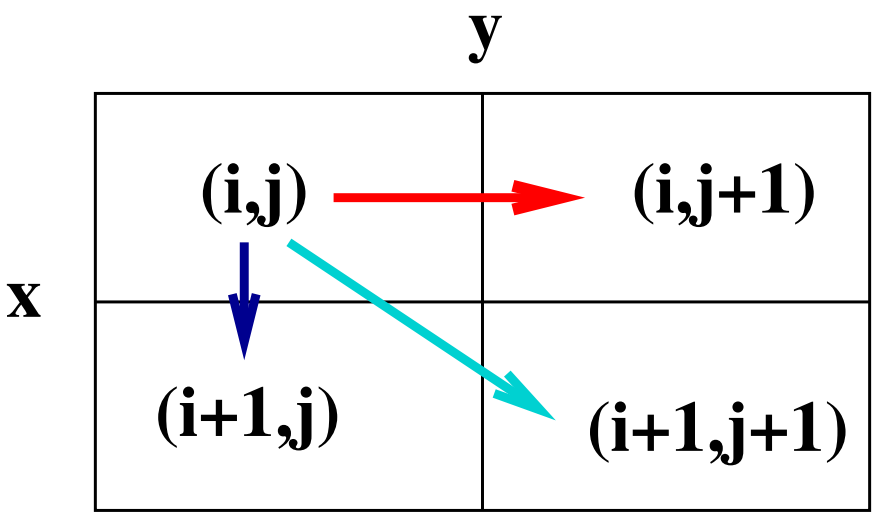
$i_2 \leftarrow Solve(i-1, j) + ins(X[i])$

$i_3 \leftarrow Solve(i, j-1) + del(Y[j])$

return $min(i_1, i_2, i_3)$

$d_L(voiture, moteur)$?

Solution 2 (ascendante): passer de *y* à *x*



suppression de y[j+1]

substitution de x[i+1] par y[j+1]

insertion de x[i+1]

$d_L(voiture, moteur)$?

Remplissage de la table en $\mathcal{O}(I \times J)$

// init au bornes
for i $\leftarrow$ 0 à I **do**
  T[i][0] $\leftarrow i$
for j $\leftarrow$ 1 à J **do**
  T[0][j] $\leftarrow j$

// boucle principale
for i $\leftarrow$ 1 à I **do**
  for j $\leftarrow$ 1 à J **do**

$$\text{T[i][j]} = \min \begin{cases} T[i-1,j-1] & + & sub(x[i], y[j]) \\ T[i-1,j] & + & ins(x[i]) \\ T[i,j-1] & + & del(y[j]) \end{cases}$$

//Le résultat est dans T[I][J]

$d_L(voiture, moteur)$?

Retour d'un meilleur alignement

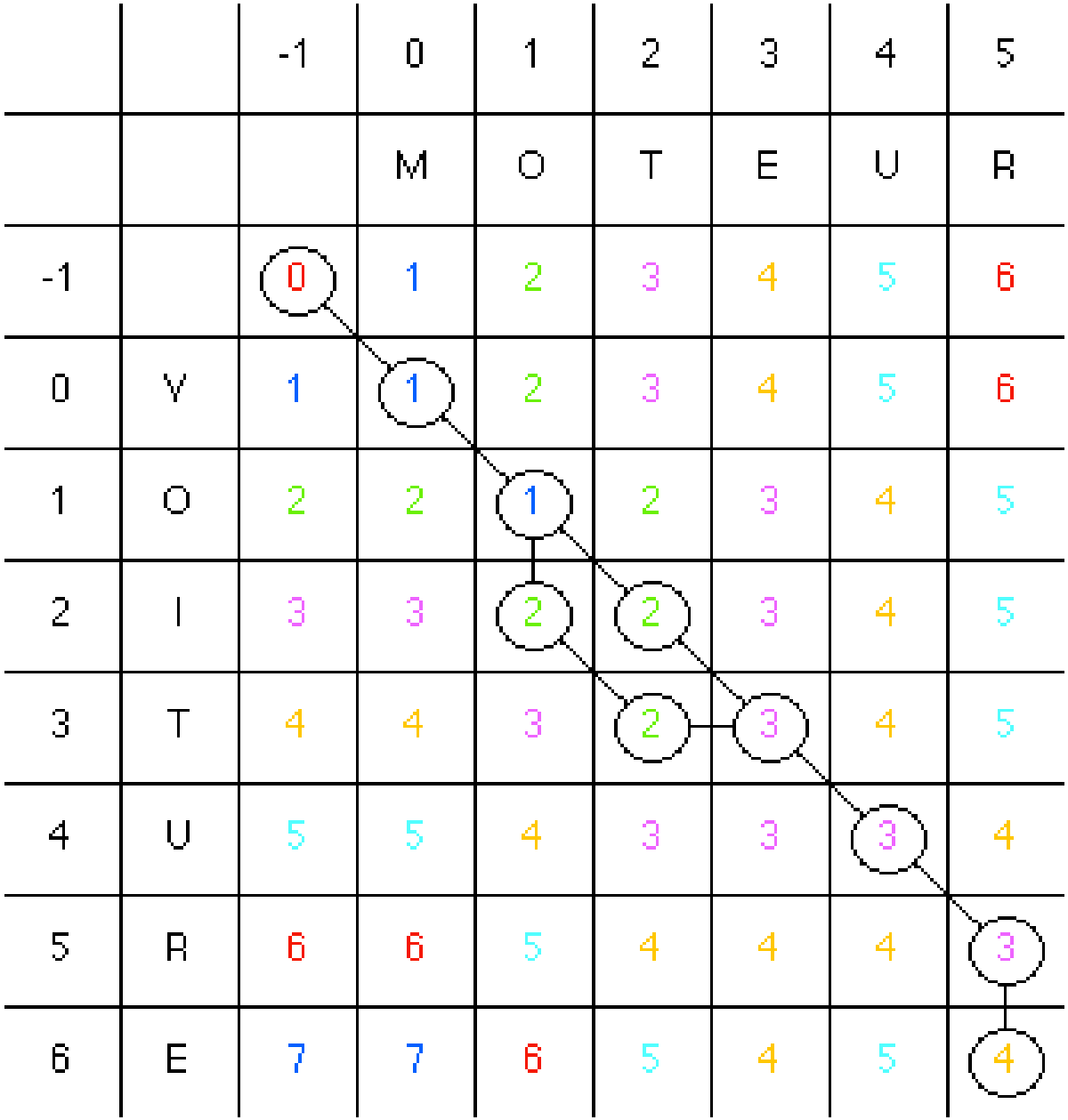
$i \leftarrow I$, $j \leftarrow J$, $a \leftarrow \{\}$

while $(i > 0)$ || $(j > 0)$ **do**
  if T[i][j-1]+del(y[j]) == T[i][j] **then**
    a $\leftarrow$ a $\cup$ del(y[j])
    j $\leftarrow$ j-1
  else if T[i-1][j]+ins(x[i]) == T[i][j] **then**
    a $\leftarrow$ a $\cup$ ins(x[i])
    i $\leftarrow$ i-1
  else
    a $\leftarrow$ a $\cup$ sub(x[i],y[j])
    i $\leftarrow$ i-1 , j $\leftarrow$ j-1

- L'ordre des tests influe sur le meilleur alignement retourné lorsqu'il y en a plusieurs
- On peut éviter cet algorithme en mémorisant dans chaque cellule de la table l'opération (2 bits sont nécessaires).

$d_L(voiture, moteur)$?

http://www-igm.univ-mlv.fr/~lecroq/seqcomp



$d_L(voiture, moteur)$?

Énumérer les meilleurs alignements

- Les deux alignements de la figure précédente (chacun de coût minimal 4):

V	O	I	T	U	R	E
M	O	T	E	U	R	
S	=	S	S	=	=	I

V	O	I	T		U	R	E
M	O		T	E	U	R	
S	=	D	=	I	=	=	I

  - Le second alignement correspond à la table de l'acétate 14
  - On peut représenter l'ensemble des alignements par un automate

Damerau-Levenshtein

Autoriser les transpositions de 2 caractères adjacents

// init au bornes
for i $\leftarrow$ 0 à I **do**
  T[i][0] $\leftarrow i$
for j $\leftarrow$ 1 à J **do**
  T[0][j] $\leftarrow j$

// boucle principale
for i $\leftarrow$ 1 à I **do**
  for j $\leftarrow$ 1 à J **do**

$$\text{T[i][j]} = \min \begin{cases} T[i-1,j-1] & + & sub(x[i],y[j]) \\ T[i-1,j] & + & ins(x[i]) \\ T[i,j-1] & + & del(y[j]) \end{cases}$$

if i $\geq$ 2 **and** j $\geq$ 2 **and** x[i] == y[j-1] **and** x[i-1] == y[j] **then**

$$\text{t[i][j]} = \min \begin{cases} \text{t[i][j]} \\ \text{t[i-2][j-2]} + cost\_transposition \end{cases}$$

//Le résultat est dans T[I][J]

Distance de Needleman-Wunsch

d'après http://fr.wikipedia.org/wiki/Algorithme_de_Needleman-Wunsch

- plus de 6000 citations à (Needleman and Wunsch 1970) !
- matrice de similarité *sim* de taille $\Sigma \times \Sigma$
- recherche les alignements de score maximal de manière analogue à la distance d'édition, en donnant une pénalité de trou identique d à chaque insertion ou suppression.

- la récurrence:

$$
\begin{aligned}
F_{ij} &= max \begin{cases} F_{i-1,j-1} + sim(x_i, y_j) \\ F_{i,j-1} + d \\ F_{i-1,j} + d \end{cases} \\
F_{0j} &= F_{i0}, \qquad \forall i, j
\end{aligned}
$$

Distance de Needleman-Wunsch

d'après http://fr.wikipedia.org/wiki/Algorithme_de_Needleman-Wunsch

- avec $d = -5$ et *sim*:

	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

- l'alignement suivant de AGACTAGTTAC avec CGAGACGT a un coût de 1:

A	G	A	C	T	A	G	T	T	A	C
C	G	A	–	–	–	G	A	C	G	T
-3	7	10	-5	-5	-5	7	-4	0	-1	0

- Est-ce un alignement optimal (cad de similarité maximale)?

Distance de Hamming et de Jaro-Winkler

Distance de Levenshtein

Plus longue sous-chaine commune

Applications de la distance d'édition

Algorithme du Soundex

Plus longue sous-chaine commune à X et Y

- Soit X = $x_1 \dots x_N$ et $Y = y_1 \dots y_M$
  - $Z = z_1 \dots z_K$ est une sous-chaine de X, notée $Z = SC(X)$, *ssi* il existe $\rho : [1, K] \rightarrow [1, N]$ tel que: $Z = x_{\rho(1)} \dots x_{\rho(K)}$ avec $\rho(i) > \rho(j), \forall i > j$
  - Z est une sous-chaine commune à X et Y, notée $Z = SCC(X, Y)$ *ssi* Z=SC(Y) et Z=SC(X)

- **Exemple:** $X = ACGATCCACGT$, $Y = AGCTACGT$
  - *AAA*, *CT*,*ACGT* sont des sous-chaines de X,
  - *AA*, *AGACT* sont des sous-chaines communes à X et Y

Idée: X et Y sont d'autant plus proches qu'elle possèdent une sous-chaine commune qui est longue.

Plus longue sous-chaine commune à X et Y

Théorème

- Soit $X = x_1 \dots x_N$ $Y = y_1 \dots y_M$ et $Z = z_1 \dots z_K$
- **si** Z est une plus longue sous-chaine de X et Y, ce que l'on note $Z = PLSCC(X, Y)$
- **alors**
  - **si** $x_N = y_M$ **alors** $z_K = x_N$ et $Z_1^{K-1} = PLSCC(X_1^{N-1}, Y_1^{M-1})$
  - **sinon**
    - **si** $z_K \neq x_N$ **alors** $Z_1^K = PLSCC(X_1^{N-1}, Y)$
    - **si** $z_K \neq y_M$ **alors** $Z_1^K = PLSCC(X, Y_1^{M-1})$

Récurrence

- Soit $T[i,j]$ la longueur de la plus longue sous-chaine commune à $X[1...i]$ et $Y[1...j]$:

$$T[i,j] = \begin{cases} 0 & \text{si} \quad i*j=0 \\ T[i-1,j-1]+1 & \text{si} \quad i,j>0 \text{ et } x_i = y_j \\ max \begin{cases} T[i-1,j] \\ T[i,j-1] \end{cases} & \text{si} \quad i,j>0 \text{ et } x_i \neq y_j \end{cases}$$

- **Exemple:**

$$\overbrace{\overbrace{\overbrace{1+\max(1+\overbrace{1+\underbrace{PLSCC(\varepsilon,\varepsilon)}_{0}}^{PLSCC(A,A)=1}}^{PLSCC(AA,AA)=2}, \max(\overbrace{1+\underbrace{PLSCC(A,\varepsilon)}_{0}}^{PLSCC(AA,A)=1}, \underbrace{PLSCC(AAB,\varepsilon)}_{0}))}^{PLSCC(AAB,A)=1}}^{PLSCC(AAB,AA)=2}}^{PLSCC(AABB,AAB)=3}$$

Plus longue sous-chaine commune à X et Y

Programmation dynamique

Require: T une table $N \times M$ dont un élément est T[i,j] = <score,back pointeur>

Ensure: T[N,M] contient la longueur d'une PLSSC de X et Y

for $i : 0 \rightarrow N$ **do** T[i,0] = $< 0, \downarrow >$

for $j : 1 \rightarrow M$ **do** T[0,j] = $< 0, \leftarrow >$

for $i : 1 \rightarrow N$ **do**

  for $j : 1 \rightarrow M$ **do**

    if $x_i = y_j$ **then**

      T[i,j] = $< T[i-1, j-1].score + 1, \swarrow >$

    else

      T[i,j] = $< T[i-1, j].score, \downarrow >$

      if T[i,j-1].score $>$ T[i,j].score **then**

        T[i,j] = $< T[i, j-1].score, \leftarrow >$

Plus longue sous-chaine commune à X et Y

Programmation dynamique

T	0	↓	1	↓	2	↓	3	↓	4	↓	5	↓	6	↓	7	↓	8	↙
G	0	↓	1	↓	2	↓	3	↓	4	↓	5	↓	6	↓	7	↙	7	←
C	0	↓	1	↓	2	↙	3	↓	4	↙	5	↙	6	↓	6	←	6	←
A	0	↓	1	↙	2	↓	3	↙	4	↓	5	↓	6	↙	6	←	6	←
G	0	↓	1	↓	2	↓	3	↓	4	↓	5	↓	5	←	6	↙	6	←
C	0	↓	1	↓	2	↙	3	↓	4	↙	5	↙	5	←	5	←	5	←
A	0	↓	1	↙	2	↓	3	↙	4	↓	4	←	5	↙	5	←	5	←
C	0	↓	1	↓	2	↙	3	↓	4	↙	4	↙	4	←	4	←	4	←
G	0	↓	1	↓	2	↓	3	↓	3	←	3	←	3	←	4	↙	4	←
G	0	↓	1	↓	2	↓	3	↓	3	←	3	←	3	←	4	↙	4	←
A	0	↓	1	↙	2	↓	3	↙	3	←	3	←	3	↙	3	←	3	←
G	0	↓	1	↓	2	↓	2	←	2	←	2	←	2	←	3	↙	3	←
C	0	↓	1	↓	2	↙	2	←	2	↙	2	↙	2	←	2	←	2	←
A	0	↓	1	↙	1	←	1	↙	1	←	1	←	1	↙	1	←	1	←
X	0	↓	0	←	0	←	0	←	0	←	0	←	0	←	0	←	0	←
	Y		A		C		A		C		C		A		G		T	

- Y est une PLSCC(X,Y)

Distance de Hamming et de Jaro-Winkler

Distance de Levenshtein

Plus longue sous-chaine commune

Applications de la distance d'édition

Algorithme du Soundex

Mesure de la qualité d'un engin de traduction

WER (Word Error Rate)

SRC	**cependant , il y a ici deux problèmes qui apparaissent**
REF	however , there are two problems here .
SMT	however , there are two problems emerging here . (11%)
SRC	**les limites des circonscriptions électorales**
REF	electoral boundaries
SMT	the electoral boundaries (33%)
SRC	**nous sommes fiers de ces habitants de london et d' autres ca bâtir un monde meilleur .**
REF	we are proud of these londoners and of other canadians who devote
SMT	we are proud of these people of london and other people spend thei
SRC	**quelle plus belle image peut on donner du canada ?**
REF	this is canada at its best .
SMT	what more can be nice to canada ? (100%)

WER

Note

(8)	(7) (0,i)	this (1,d)	is (2,d)	canada (3,d)	at (4,d)	its (5,d)	best (6,d)	. (7,d)
what	(1,i)	(1,s)	(2,d)	(3,d)	(4,d)	(5,d)	(6,d)	(7,d)
more	(2,i)	(2,i)	(2,s)	(3,d)	(4,d)	(5,d)	(6,d)	(7,d)
can	(3,i)	(3,i)	(3,i)	(3,s)	(4,d)	(5,d)	(6,d)	(7,d)
be	(4,i)	(4,i)	(4,i)	(4,i)	(4,s)	(5,d)	(6,d)	(7,d)
nice	(5,i)	(5,i)	(5,i)	(5,i)	(5,i)	(5,s)	(6,d)	(7,d)
to	(6,i)	(6,i)	(6,i)	(6,i)	(6,i)	(6,i)	(6,s)	(7,d)
canada	(7,i)	(7,i)	(7,i)	(6,=)	(7,d)	(7,i)	(7,i)	(7,s)
?	(8,i)	(8,i)	(8,i)	(7,i)	(7,s)	(8,d)	(8,i)	(8,i)

this what	is more	canada can	at be	its nice	best to	. canada	?
sub	sub	sub	sub	sub	sub	sub	I

Application à la correction orthographique

- De nombreux types d'erreurs (qui peuvent être cumulés)
  - fautes d'orthographe: éphémaire, opignon, etc.
  - fautes de frappe: qviron, errueurs
  - impropriété (malaproprism): il est âpre / Il est râpe
  - mots d'emprunts: tu n'as qu'à piper la sortie de ta commande avant de faire ton debugging
  - fautes de grammaire: la femme que j'ai vu.
  - erreurs de césure: il est con fondant/confondant
  - formes extra-lexicales: U2, OQP, etc.

- Les transparents qui suivent sont des idées simples qui illustrent comment on peut utiliser un modèle de langue pour faire un mini-correcteur orthographique.

Application à la correction orthographique

Approche possible

Soit $\mathscr{L}$ un lexique fini contenant de nombreux mots d'une langue.

- Identifier dans un texte T_1^n les mots inconnus de $\mathscr{L}$ (on écarte ici les formes extra-lexicales chiffrées)
- Pour chaque mot inconnu u, réunir les mots de $\mathscr{L}$ qui lui sont proches au sens d'une métrique à définir.

  soit v_u l'ensemble de ces mots proches de la forme u
- Si $h(u)$ désigne l'historique de u dans le texte, alors corriger u en $\hat{u}$:

$$\hat{u} = \underset{v \in v_u}{\operatorname{argmax}}\, p(v|h(u))$$

j'aime la apriture $\Rightarrow$ j'aime la aperture / friture / aptitude

Distance d'édition

Étude ce cas: Dictionnaire de 100 000 formes

- les mots les plus proches de asfalte:

asphalte	2	spalter	3	state	3
svelte	3	falce	3	fate	3
faute	3	faîte	3	halte	3

- Les mots les plus proches de ammateur:

armateur	1	amateur	1	aviateur	2
amateurs	2	animateur	2	armateurs	2
orateur	3	radiateur	3	sénateur	3

- Les mots les plus proches de courier:

courier	0	courrier	1	courtier	1
courir	1	courber	1	sourir	2
tourner	2	usurier	2	écourter	2

Distance de Hamming et de Jaro-Winkler

Distance de Levenshtein

Plus longue sous-chaine commune

Applications de la distance d'édition

Algorithme du Soundex

L'algorithme du Soundex

- Algorithme simple capable d'associer à chaque mot (suite de caractères) un *"code phonétique"*
- Il existe des soundex pour de nombreuses langues

- **Idée:** les mots proches phonétiquement ont le même code soundex
  soundex(computer) = soundex(camputter) = C513

- brevet déposé par Robert, R. Russel en 1918
- nombreuses variantes (ex: metaphone, NYYSIIS)

Soundex Anglais

(Russel et O'Dell,1918)

1. mettre le mot en majuscule, éliminer les ponctuations
2. garder la première lettre du mot
3. supprimer les occurences de: A E I O U H W Y
4. faire les changements suivants:
   - B F P V $\rightarrow$ 1 — bilabiales
   - C G J K Q S X Z $\rightarrow$ 2 — labiodentales
   - D T $\rightarrow$ 3 — dentales
   - L $\rightarrow$ 4 — alvéolaires
   - M N $\rightarrow$ 5 — vélaires
   - R $\rightarrow$ 6 — laryngales
5. si deux lettres adjacentes dans la chaîne de départ ont le même code, ne garder que la première de ces lettres (élimination des doublons)
6. retourner les 4 premiers caractères (compléter par des zéros le cas échéant)

Soundex Français

- Règles de conversion pour le français:

B P	$\rightarrow$	1	L	$\rightarrow$	4	G J	$\rightarrow$	7
C K Q	$\rightarrow$	2	M N	$\rightarrow$	5	X Z S	$\rightarrow$	8
D T	$\rightarrow$	3	R	$\rightarrow$	6	F V	$\rightarrow$	9

- On peut calculer le soundex du dictionnaire et encoder cela dans une table de hachage multiple de manière à retrouver en temps constant l'ensemble des mots proches d'un mot inconnu.

Références I

Crochemore, Maxime, Christophe Hancart, and Thierry Lecroq (2001). *Algorithmique du texte*. Vuibert Informatique. ISBN-2-7117-8628-5. Vuibert.

Levenshtein, V. I. (1966). “Binary codes capable of correcting deletions, insertions and reversals”. In: *Sov. Phys. Dokl.* 6, pp. 707–710.

Needleman, S. B. and C. D. Wunsch (Mar. 1970). “A general method applicable to the search for similarities in the amino acid sequence of two proteins.” In: *Journal of molecular biology* 48.3, pp. 443–453. ISSN: 0022-2836. URL: `http://view.ncbi.nlm.nih.gov/pubmed/5420325`.