DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Integración de datos - Cadenas

Fernando Berzal, berzal@acm.org

Integración de datos

- Descripción de fuentes de datos
- Emparejamiento de cadenas [string matching]
  - Medidas de similitud
  - Escalabilidad
- Integración de esquemas
  - Emparejamiento de esquemas [schema matching]
  - Correspondencias entre esquemas [schema mapping]
  - Gestión de modelos
- Emparejamiento de datos [data matching]
- Wrappers
- Apéndice: Procesamiento de consultas

Emparejamiento de cadenas

Problema

Encontrar cadenas que se refieren a la misma entidad.

p.ej. MSFT & Microsoft & Microsoft Corporation
Gran Vía, 35 & G.Vía 35
F. Berzal & Fernando Berzal

Crítico para muchas tareas:

- Integración de esquemas [schema matching & mapping]
- Integración de datos [data matching]
- Extracción de información
- …

Emparejamiento de cadenas

Formalización del problema

Dados dos conjuntos de cadenas X e Y,
encontrar todos los pares (x,y), con $x \in X$ e $y \in Y$,
que hacen referencia a la misma entidad en el mundo real.

Cada par (x,y) identificado será un emparejamiento [match].

Set *X*	Set *Y*	Matches
x_1 = Dave Smith	y_1 = David D. Smith	(x_1, y_1)
x_2 = Joe Wilson	y_2 = Daniel W. Smith	(x_3, y_2)
x_3 = Dan Smith		

Emparejamiento de cadenas

Desafíos prácticos

- **Precisión [accuracy]:**
  Las cadenas que debemos emparejar no siempre son iguales (typos, errores de OCR, formatos diferentes, abreviaturas y omisiones, apodos, cambios de orden…).

- **Escalabilidad [scalability]:**
  Emparejar cada cadena con todas las demás no es práctico, $O(n^2)$, por lo que deberemos reducir el número de comprobaciones necesario.

Medidas de similitud

Las cadenas que desearíamos emparejar no siempre aparecen de la misma forma:

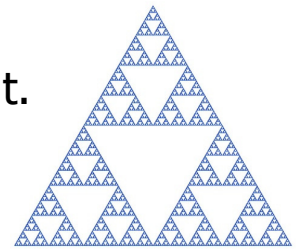
- Errores mecanográficos
  David vs. Davod
- Errores de OCR
  datos vs. dalos
- Abreviaturas (en ocasiones, no estándar) y omisiones
  Calle Real vs. C/ Real vs. C./ Real vs. Cl. Real vs. Call. Real
- Diferentes nombres y apodos
  José vs. Jose vs. Pepe
- Cambios de orden en subcadenas
  ETSIIT, Universidad de Granada
  vs. Universidad de Granada, ETSIIT

Medidas de similitud

Solución

Definir una medida de similitud $s(x,y) \in [0,1]$

- Cuanto mayor sea la similitud s(x,y), mayor es la probabilidad de que x e y casen.
- Normalmente, x e y emparejan si $s(x,y) \geq t$.



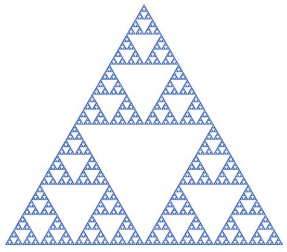
NOTA

También se pueden utilizar funciones de coste o métricas de distancia: cuanto menor sea su valor, mayor es la similitud.

Medidas de similitud

Distintas formas de medir la similitud entre cadenas:

- **Medidas basadas en secuencias:** Distancia de edición, Needleman-Wunch, affine gap, Smith-Waterman, Jaro, Jaro-Winkler…
- **Medidas basadas en conjuntos:** solapamiento, Jaccard, TF/IDF…
- **Medidas híbridas** (e.g. Monge-Elkan)
- **Medidas fonéticas** (e.g. Soundex)



Medidas de similitud

Distancia de edición (a.k.a. Distancia Levenshtein)

d(s,t) mide la diferencia entre dos cadenas s y t como el número mínimo de operaciones de edición que hay que realizar para convertir una cadena en otra:

d("data mining", "data minino") = 1
d("efecto", "defecto") = 1
d("poda", "boda") = 1
d("night","natch") = d("natch","noche") = 3

Aplicaciones: Correctores ortográficos, reconocimiento de voz, detección de plagios, análisis de ADN, traducción...

NOTA: Para datos binarios, es la distancia de Hamming.

Medidas de similitud

Distancia de edición (a.k.a. Distancia Levenshtein)

Operadores de edición (de coste 1)

- Borrar un carácter.
- Insertar un carácter.
- Sustituir un carácter por otro.

Puede adaptarse para incorporar diferentes errores mecanográficos típicos (p.ej. intercambiar dos caracteres)

Medidas de similitud

Distancia de edición (a.k.a. Distancia Levenshtein)

Cálculo mediante **programación dinámica**.

Definición recursiva de la solución:

$$d(i,j) = \begin{cases} d(i-1,j-1) & si \quad s[i] = t[j] \\ 1 + \min\{d(i-1,j), d(i,j-1), d(i-1,j-1)\} & si \quad s[i] \neq t[j] \end{cases}$$

Casos

- Mismo carácter: d(i-1,j-1)
- Inserción: 1 + d(i-1,j)
- Borrado: 1 + d(i,j-1)
- Sustitución: 1 + d(i-1,j-1)

Medidas de similitud

Distancia de edición (a.k.a. Distancia Levenshtein)

```
int LevenshteinDistance (string s[1..m], string t[1..n])
{
  for (i=0; i<=m; i++) d[i,0]=i;
  for (j=0; j<=n; j++) d[0,j]=j;

  for (j=1; j<=n; j++)
      for (i=1; i<=m; i++)
          if (s[i]==t[j])
             d[i,j] = d[i-1, j-1]
          else
             d[i,j] = 1+ min(d[i-1,j],d[i,j-1],d[i-1,j-1]);

  return d[m,n];
}
```

Medidas de similitud

Distancia de edición (a.k.a. Distancia Levenshtein)

EJEMPLO

x = dva
y = dave

		y0	y1	y2	y3	y4
			d	**a**	**v**	**e**
x0		0	1	2	3	4
x1	**d**	1	0	1		
x2	**v**	2				
x3	**a**	3				

		y0	y1	y2	y3	y4
			d	**a**	**v**	**e**
x0		0	1	2	3	4
x1	**d**	1	0	1	2	3
x2	**v**	2	1	1	1	2
x3	**a**	3	2	1	2	2

d(x,y)=2

x = d – v a
| | | |
y = d a v e

Algoritmo O(|x||y|)

Medidas de similitud

Distancia de edición (a.k.a. Distancia Levenshtein)

¿Cómo convertimos una medida de distancia en una medida de similitud?

```
s(x,y) = 1 - d(x,y) / max { length(x), length(y) }
```

EJEMPLO

d ('David Smiths', 'Davidd Simth') = 4
s ('David Smiths', 'Davidd Simth') = 1 – 4 / max(12, 12) = 0.67

Medidas de similitud

Medida de Needleman-Wunch

Generalización de la distancia de edición de Levenshtein.

IDEA BÁSICA: Alineación de secuencias

Correspondencia entre los caracteres de x e y, permitiendo la existencia de huecos

-	C	T	G	A	C	C	T	A	C	C	T
C	C	T	G	A	C	-	T	A	C	A	T

Se asigna un coste a cada alineación y se devuelve la asignación de menor coste…

Medidas de similitud

Medida de Needleman-Wunch

Dadas dos secuencias,
$X = (x_1\ x_2\ \ldots\ x_m)$ e $Y = (y_1\ y_2\ \ldots\ y_n)$,
encontrar la forma de alinearlas con un coste mínimo.

¿Cómo medimos ese coste?

- δ [gap penalty], cuando en una cadena no aparece un símbolo que sí está en la otra.
- α_{pq} [mismatch penalty], cuando en la cadena X aparece el símbolo p pero en la cadena Y aparece q.

Medidas de similitud

Medida de Needleman-Wunch

EJEMPLOS

C	T	G	A	C	C	T	A	C	C	T
C	C	T	G	A	C	T	A	C	A	T

$$\alpha_{TC} + \alpha_{GT} + \alpha_{AG} + 2\alpha_{CA}$$

-	C	T	G	A	C	C	T	A	C	C	T
C	C	T	G	A	C	-	T	A	C	A	T

$$2\delta + \alpha_{CA}$$

- δ [gap penalty]
- α_{pq} [mismatch penalty]

Medidas de similitud

Medida de Needleman-Wunch

- Una alineación M es un conjunto de pares (x_i,y_j) tal que cada uno de los elementos x_i e y_j aparece como mucho en un par y no se produce ningún cruce.
- Los pares (x_i,y_j) y $(x_{i'},y_{j'})$ se cruzan si $i<i'$ pero $j>j'$.
- El coste de la alineación M, por tanto, viene dado por:

$$\text{coste}(M) = \underbrace{\sum_{(x_i,y_j)\in M} \alpha_{x_i y_j}}_{\text{errores}} + \underbrace{\sum_{x_i \notin M} \delta + \sum_{y_j \notin M} \delta}_{\text{huecos}}$$

Medidas de similitud

Medida de Needleman-Wunch

EJEMPLO

CTACCG vs. TACATG

	x_1	x_2	x_3	x_4	x_5		x_6
	C	T	A	C	C	-	G
	-	T	A	C	A	T	G
		y_1	y_2	y_3	y_4	y_5	y_6

$$\text{coste}(M) = \underbrace{\sum_{(x_i, y_j) \in M} \alpha_{x_i y_j}}_{\text{errores}} + \underbrace{\sum_{x_i \notin M} \delta + \sum_{y_j \notin M} \delta}_{\text{huecos}}$$

Alineación M = { (x_2,y_1), (x_3,y_2), (x_4,y_3), (x_5,y_4), (x_6,y_6) }

Coste de la alineación: coste(M) = $2\delta + \alpha_{CA}$

Medidas de similitud

Medida de Needleman-Wunch

OPT(i, j) Coste mínimo de alineación de las secuencias $X_i=(x_1\ x_2\ \ldots\ x_i)$ e $Y_j=(y_1\ y_2\ \ldots\ y_j)$.

- Caso 1: (x_i,y_j) está en la mejor alineación M_{OPT}
  Hay que pagar el coste de emparejar x_i con y_j, a lo que habrá que añadir el coste de alinear las secuencias $X_{i-1}=(x_1\ x_2\ \ldots\ x_{i-1})$ e $Y_{j-1}=(y_1\ y_2\ \ldots\ y_{j-1})$.
- Caso 2a: M_{OPT} deja x_i sin emparejar
  Hay que pagar una penalización δ más el coste de alinear las cadenas $X_{i-1}=(x_1\ x_2\ \ldots\ x_{i-1})$ e $Y_j=(y_1\ y_2\ \ldots\ y_j)$.
- Caso 2b: M_{OPT} deja y_j sin emparejar
  Penalización δ más el coste de alinear las cadenas $X_i=(x_1\ x_2\ \ldots\ x_i)$ e $Y_{j-1}=(y_1\ y_2\ \ldots\ y_{j-1})$.

Medidas de similitud

Medida de Needleman-Wunch

OPT(i, j) Coste mínimo de alineación de las secuencias $X_i=(x_1\ x_2\ \ldots\ x_i)$ e $Y_j=(y_1\ y_2\ \ldots\ y_j)$.

Definición recursiva de la solución:

$$OPT(i,j)=\begin{cases} j\delta & \text{si } i=0 \\ \min\begin{cases}\alpha_{x_i y_j}+OPT(i-1,j-1)\\ \delta+OPT(i-1,j)\\ \delta+OPT(i,j-1)\end{cases} & \text{en otro caso} \\ i\delta & \text{si } j=0 \end{cases}$$

Medidas de similitud

Medida de Needleman-Wunch

```
NeedlemanWunch (X, Y, δ, α)
{
   for i = 0 to X.length
      M[0, i] = i*δ;
   for j = 0 to Y.length
      M[j, 0] = j*δ;
   for i = 1 to X.length
      for j = 1 to Y.length
         M[i, j] = min ( α[X[i]][Y[j]] + M[i-1][j-1],
                         δ + M[i-1][j],
                         δ + M[i][j-1]);
   return M[X.length][Y.length];
}
```

Medidas de similitud

Medida de Needleman-Wunch

Algoritmo basado en programación dinámica

Eficiencia

- Tiempo: $\Theta(|X||Y|)$
- Espacio: $\Theta(|X||Y|)$

Otras aplicaciones…

- Reconocimiento de voz con DTW, |X|<100, |Y|<100, OK.
- Biología computacional: |X|=|Y|=100000
  - 10^{10} operaciones, OK
  - Array con 10^{10} entradas > 10 GB !!!

Medidas de similitud

Medida de Needleman-Wunch

EJEMPLO

```
d - - v a
|     | |
d e e v e
```

s_{pq} [score matrix] =

δ [gap penalty] = 1

	d	v	a	e
d	2	-1	-1	-1
v	-1	2	-1	-1
a	-1	-1	2	-1
e	-1	-1	-1	2

		d	e	e	v	e
	0	-1	-2	-3	-4	-5
d	-1	2	1	0	-1	-2
v	-2	1	1	0	2	1
a	-3	0	0	0	1	1

$$s(i,j) = \max \begin{cases} s(i-1,j-1) + c(xi,yj) \\ s(i-1,j) - c_g \\ s(i,j-1) - c_g \end{cases}$$

$$s(0,j) = -jc_g$$
$$s(i,0) = -ic_g$$

Medidas de similitud

Affine gap

Extensión de la medida de Needleman-Wunch para manejar huecos de longitud variable…

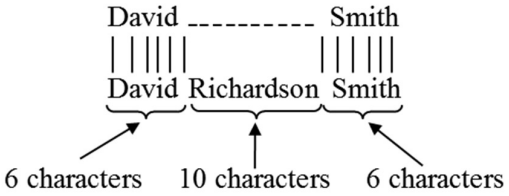
EJEMPLO

"David Smith" vs. "David R. Smith"

- Needleman-Wunch OK (hueco de longitud 2).

"David Smith" vs.

"David Richardson Smith"



- Needleman-Wunch no funciona bien:
  Penalización excesiva por la longitud del hueco :-(

Medidas de similitud

Affine gap

- En la práctica, los huecos suelen ser de longitud > 1.
- Asignarle la misma penalización a cada carácter del hueco penaliza en exceso los huecos largos.

Solución

Definir costes separados
para abrir un hueco y para continuarlo

cost (gap of length k) = $c_0 + (k-1)c_r$

c_0 = cost of opening gap

c_r = cost of continuing gap, $c_0 > c_r$

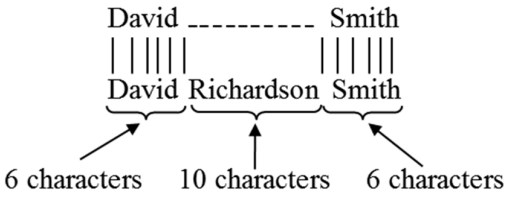
Medidas de similitud

Affine gap

EJEMPLO

"David Smith" vs.

"David Richardson Smith"



- Needleman-Wunch: $score_{NG} = 6*2-10 = 2$
- Affine gap: $score_{AG} = 6*2-1-9*0.5 = 6.5$

  $c_0 = 1$

  $c_r = 0.5$

Medidas de similitud

Affine gap

Cálculo utilizando programación dinámica:

$$s(i,j) = \max \{M(i,j), I_x(i,j), I_y(i,j)\}$$

$$M(i,j) = \max \begin{cases} M(i-1,j-1) + c(x_i,y_j) \\ I_x(i-1,j-1) + c(x_i,y_j) \\ I_y(i-1,j-1) + c(x_i,y_j) \end{cases}$$

$$I_x(i,j) = \max \begin{cases} M(i-1,j) - c_o \\ I_x(i-1,j) - c_r \end{cases}$$

$$I_y(i,j) = \max \begin{cases} M(i,j-1) - c_o \\ I_y(i,j-1) - c_r \end{cases}$$

Medidas de similitud

Medida de Smith-Waterman

MOTIVACIÓN

Las medidas anteriores consideran alineaciones globales (emparejan todos los caracteres de x con todos los de y)

Sin embargo, en algunos casos no resulta adecuado:

ETS Ingeniería Informática
Escuela Técnica Superior de Ingeniería Informática

Prof. Fernando Berzal, Universidad de Granada
Fernando Berzal, Ph.D.

Medidas de similitud

Medida de Smith-Waterman

UNA IDEA MEJOR

Encontrar las subcadenas más similares de x e y.

En los casos anteriores:

Ingeniería Informática
ETS Ingeniería Informática
Escuela Técnica Superior de Ingeniería Informática

Fernando Berzal
Prof. Fernando Berzal, Universidad de Granada
Fernando Berzal, Ph.D.

Medidas de similitud

Medida de Smith-Waterman

ALINEACIÓN LOCAL

Encontrar la mejor alineación local de x e y.

Cambios clave con respecto a Needleman-Wunch:

- El emparejamiento puede empezar en cualquier posición de las cadenas (no limitado al comienzo).
- El emparejamiento puede terminar en cualquier posición de las cadenas (no necesariamente al final): La reconstrucción de la alineación comienza desde el mayor valor de la matriz, no desde su esquina.

Medidas de similitud

Medida de Smith-Waterman

Cálculo utilizando programación dinámica:

		d	a	v	e
	0	0	0	0	0
a	0	0	2	1	0
v	0	0	1	4	3
d	0	2	1	3	3

$$s(i,j) = \max \begin{cases} 0 \\ s(i-1,j-1) + c(xi,yj) \\ s(i-1,j) - c_g \\ s(i,j-1) - c_g \end{cases}$$

$$s(0,j) = 0$$
$$s(i,0) = 0$$

Medidas de similitud

Medida de Jaro

Para comparar cadenas cortas
(p.ej. nombres y apellidos).

- Se encuentran "caracteres comunes" x_i e y_j:
  $x_i = y_j$ tales que $|i-j| \leq \min\{|x|,|y|\}/2$
  (caracteres idénticos posicionalmente cerca)
- Si el i-ésimo carácter común de x no coincide con el i-ésimo carácter común de y, tenemos una trasposición.
- **$jaro(x,y) = 1 / 3[c/|x| + c/|y| + (c - t/2)/c]$,**
  donde c es el número de caracteres comunes
  y t es el número de trasposiciones.

Medidas de similitud

Medida de Jaro

EJEMPLOS

x = jon, y = john

- c = 3 (caracteres comunes {j, o, n})
- t = 0
- jaro(x,y) = 1 / 3(3/3 + 3/4 + 3/3) = **0.917**
- Distancia de edición: Similitud s(x,y) = **0.75**

x = jon, y = ojhn

- common(x) = jon
- common(y) = ojn
- t = 2
- jaro(x,y) = **0.81**

Medidas de similitud

Medida de Jaro-Winkler

Captura casos en los que las cadenas x e y tienen una puntuación baja en la medida de Jaro pero comparten un prefijo, por lo que es probable que emparejen...

jaro-winkler(x,y) = (1 – PL*PW)*jaro(x,y) + PL*PW

PL = Longitud del prefijo común más largo

PW = Peso dado al prefijo

Medidas de similitud

Medidas basadas en conjuntos

Interpretan las cadenas como conjuntos de tokens y utilizan propiedades de esos conjuntos para determinar la similitud entre cadenas.

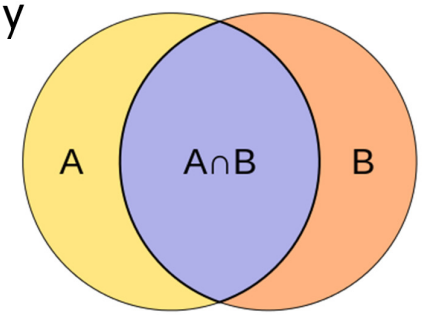
GENERACIÓN DE TOKENS

- Palabras delimitadas por espacios (eliminando "stop words" y usando, opcionalmente, lematización)
  p.ej. "Universidad de Granada" ➔ {Universidad, Granada}
  "integración de datos" ➔ {integr, dat}
- n-gramas (subcadenas de longitud n)
  p.ej. "Granada" ➔ 3-gramas
  {##G, #Gr, Gra, ran, ana, nad, ada, da#, a#}

Medidas de similitud

Medidas basadas en conjuntos

B_x = Conjunto de tokens de la cadena x
B_y = Conjunto de tokens de la cadena y



- **Solapamiento**:
  overlap(x,y) = $|B_x \cap B_y|$
  i.e. número de tokens comunes

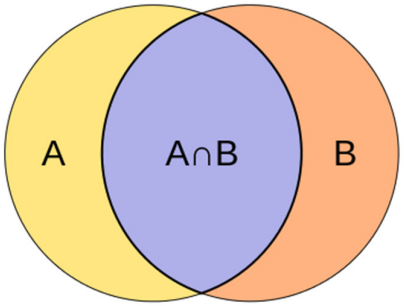
- **Coeficiente de Jaccard**:
  jaccard(x,y) = $|B_x \cap B_y| / |B_x \cup B_y|$

Medidas de similitud

Medidas basadas en conjuntos

EJEMPLO
x = "dave"
y = "dav"

B_x = {#d, da, av, ve, e#}
B_y = {#d, da, av, v#}



- Solapamiento:
  overlap(x,y) = $|B_x \cap B_y|$ = 3
- Coeficiente de Jaccard:
  jaccard(x,y) = $|B_x \cap B_y| / |B_x \cup B_y|$ = 3/6

Medidas de similitud

Coeficiente de Jaccard generalizado

El coeficiente de Jaccard considera los tokens que se solapan en x e y, que deben ser idénticos (demasiado restrictivo en ocasiones).

EJEMPLOS

- Taxonomías
  "Energy & Transportation" vs. "Transportation, Energy, & Gas"
- Errores ortográficos
  energy vs. eneryg vs. energi

Medidas de similitud

Coeficiente de Jaccard generalizado

$B_x = \{x_1, \ldots, x_n\}$

$B_y = \{y_1, \ldots, y_m\}$

Pares de tokens en el conjunto de solapamiento suavizado: Los pares para los que una medida de similitud $s(x_i, y_j) \geq \alpha$ forman un grafo bipartido, para el que se puede encontrar la asignación de peso máximo M [max-weight matching].

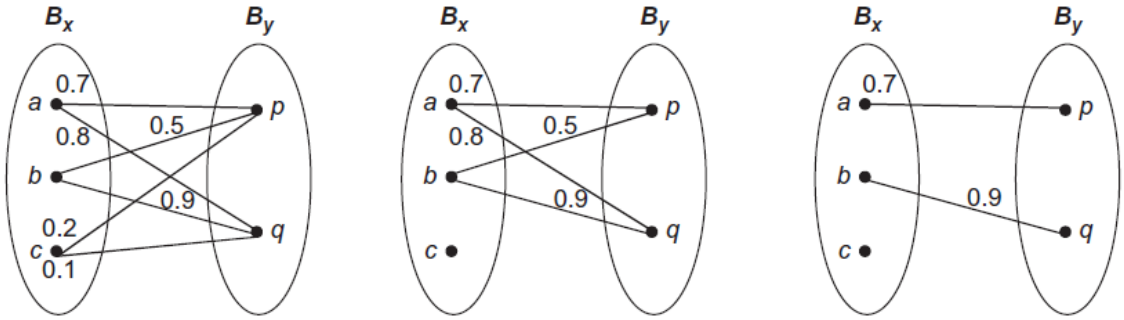
Coeficiente generalizado de Jaccard:
Peso normalizado de M

$$GJ(x,y) = \sum_{(xi,yj)\in M} s(x_i,y_j) / (|B_x| + |B_y| - |M|)$$

Medidas de similitud

Coeficiente de Jaccard generalizado

$$GJ(x,y) = \sum_{(x_i,y_j)\in M} s(x_i,y_j) / (|B_x| + |B_y| - |M|)$$



GJ(x,y) = (0.7 + 0.9)/(3 + 2 – 2) = 0.53

Medidas de similitud

TF/IDF [Term Frequency / Inverse Document Frequency]

Muy usada en recuperación de información [IR]:

Dos cadenas son similares si comparten términos distintivos.

EJEMPLO x = Apple Corporation
y = IBM Corporation
z = Apple Corp.

Emparejamiento incorrecto: s(x,y)>s(x,z) usando la distancia de edición o el coeficiente de Jaccard.

TF/IDF reconoce que Apple es el término distintivo, mientras que "Corporation" es un término más común.

Medidas de similitud

TF/IDF [Term Frequency / Inverse Document Frequency]

Cada cadena se convierte en un conjunto de términos (a los que llamaremos documento):

- Frecuencia de un término: **tf(t,d)**
  Número de veces que aparece en un documento.
- Frecuencia inversa de documento: **idf(t)** = N / N_t
  Número de documentos en la colección partido por el número de documentos en los que aparece el término.
  NOTA: Usualmente se toman logaritmos: **log(idf(t))** = log (N / N_t).
- Vector de características asociado a cada documento:
  v_d(t) = tf(t,d) * idf(t)

Medidas de similitud

TF/IDF [Term Frequency / Inverse Document Frequency]

EJEMPLO

v_d(t) = tf(t,d) * idf(t)

$x = aab \Rightarrow B_x = \{a, a, b\}$

$y = ac \Rightarrow B_y = \{a, c\}$

$z = a \Rightarrow B_z = \{a\}$

	a	b	c
v_x	2	3	0
v_y	3	0	3
v_z	3	0	0

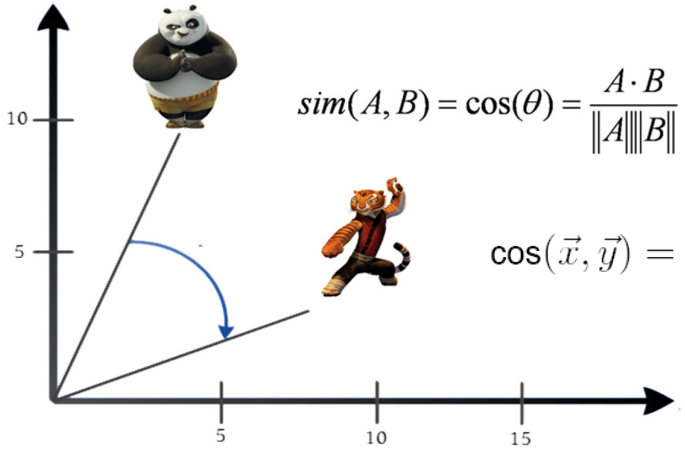
tf(a, x) = 2
tf(b, x) = 1
...
tf(c, z) = 0

idf(a) = 3/3 = 1
idf(b) = 3/1 = 3
idf(c) = 3/1 = 3

Medidas de similitud

TF/IDF [Term Frequency / Inverse Document Frequency]

Dados los vectores $\mathbf{v_d(t) = tf(t,d) * idf(t)}$,
calculamos la similitud entre dos cadenas [documentos]
utilizando la distancia del coseno:



$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

$$\cos(\vec{x},\vec{y}) = \sum_i \frac{x_i \cdot y_i}{\sqrt{\sum_i x_i^2} \cdot \sqrt{\sum_i y_i^2}}$$

Medidas de similitud

TF/IDF [Term Frequency / Inverse Document Frequency]

- TF/IDF es alto si las cadenas comparten muchos términos frecuentes (TF alto) a no ser que los términos sean comunes en otras cadenas (IDF bajo).
- En la práctica, se suele suavizar la expresión:
  Se suele usar $\mathbf{v_d(t) = log\ (\ tf(t,d) + 1\ ) * log(idf(t))}$
  en lugar de $v_d(t) = tf(t,d) * idf(t)$.
- Los vectores de características se suelen normalizar:
  $\mathbf{v_d(t) = v_d(t) / sqrt(\Sigma v_d(t)^2)}$

Medidas de similitud

Soft TF/IDF

Similar al coeficiente generalizado de Jaccard, salvo que usa TF/IDF como medida de similitud:

- close(x,y,k) = Conjunto de términos $t \in \mathbf{B}_x$ que tienen al menos un término cercano $u \in \mathbf{B}_y$, i.e., $s'(t,u) \geq k$
- La similitud s(x,y) se calcula como en TF/IDF, pero ponderando cada componente TF/IDF por s':

  $s(x,y) = \sum_{t \in close(x,y,k)} \mathbf{v}_x(t) * \mathbf{v}_y(u^*) * s'(t,u^*)$

  $u^* \in \mathbf{B}_y$ maximiza $s'(t,u)$ $\forall$ $u \in \mathbf{B}_y$

Medidas de similitud

Soft TF/IDF

EJEMPLO

x = abcd

y = aa'b'cd'

B_x = {a, b, c, d}

1 | 0.7 | 0.8 | 1 | 0.6

B_y = {a, a', b', c, d'}

close (x, y, 0.75) = {a, b, c}

$s(x,y) = v_x(a) \cdot v_y(a) \cdot 1 + v_x(b) \cdot v_y(b') \cdot 0.8 + v_x(c) \cdot v_y(c) \cdot 1$

Medidas de similitud

Similitud de Monge-Elkan

Mayor control sobre la forma de medir la similitud

- Se descomponen las cadenas x e y en múltiples subcadenas $x=A_1..A_n$ e $y=B_1..B_m$.
- Similitud $\mathbf{s(x,y) = 1/n * \Sigma_i \max_j s'(A_i, B_j)}$ donde s' es una medida de similitud secundaria, p.ej. Jaro-Winkler
- Interpretación intuitiva: Se ignora el orden en el que se emparejan las subcadenas y sólo se consideran los mejores emparejamientos de cada subcadena.

Medidas de similitud

Similitud de Monge-Elkan

Mayor control sobre la forma de medir la similitud

EJEMPLO

- x = Comput. Sci. and Eng. Dept., University of California, San Diego
- y = Department of Computer Science, Univ. Calif., San Dieg

NOTA: Se escoge una medida de similitud secundaria s' que funcione bien con abreviaturas y acrónimos.

Muy útil en idiomas como el inglés :-)

Medidas de similitud

Medidas fonéticas

IDEA:
Emparejar cadenas de acuerdo a su pronunciación
en vez de atender a su ortografía
(que, además, no siempre es correcta).

Muy práctico para nombres propios
(especialmente en idiomas como el inglés):

Meyer, Meier & Mire
Smith, Smithe & Smythe
Leticia & Letizia

Medidas de similitud

Medidas fonéticas

Algoritmo más popular: SOUNDEX

- Step 1: Keep the first letter of x,
  subsequent steps are performed on the rest of x
- Step 2:
  Remove all occurences of W and H.
  Replace the remaining letters with digits as follows:
  B, F, P, V with 1; C, G, J, K, Q, S, X, Z with 2;
  D, T with 3; L with 4, M, N with 5; R with 6
- Step 3:
  Replace sequence of identical digits by the digit itself.
- Step 4:
  Drop all non-digit letters,
  return the first four letters as the soundex code.

NOTE: Soundex code is padded with 0 if there are not enough digits

Medidas de similitud

Medidas fonéticas

Algoritmo más popular: SOUNDEX

Se codifica cada apellido usando un código de 4 letras y dos apellidos se consideran similares si comparten el mismo código.

Robert & Rupert ➔ R163

Funciona bien para muchos nombres (p.ej. occidentales), aunque no para otros de distinto origen, como los asiáticos que utilizan las vocales para discriminar…

Medidas de similitud

Medidas fonéticas

Algoritmo más popular: SOUNDEX

x = Ashcraft

- Step 1: A
- Step 2: A226a13
- Step 3: A26a13
- Step 4: A2613 ➔ A261
- Resultado: A261

El mismo resultado para Ashcroft, Ascroft o Ascrofte
http://www.surnamedb.com/Surname/Ashcroft

Escalabilidad

Problema práctico

Emparejar cada cadena con todas las demás no es práctico, $O(n^2)$, por lo que deberemos reducir el número de pruebas necesario.

Solución

Calcular s(x,y) sólo para las parejas más prometedoras

```
for each x ∈ X
    Z = candidatos(x)   // Z ⊆ Y
    for each y ∈ Z
        if s(x,y) ≥ t
           return (x,y) as a matched pair
```

Escalabilidad

Blocking

Técnicas que permiten reducir el número necesario de comparaciones de cadenas.

Umbrella set ("conjunto paraguas")

El conjunto Z de candidatos para una cadena x.

```
for each x ∈ X
    Z = candidatos(x)   // Z ⊆ Y
    for each y ∈ Z
        if s(x,y) ≥ t
           return (x,y) as a matched pair
```

Escalabilidad

Técnicas

- Índice invertido [inverted index]
- Filtrado por longitud [size filtering]
- Filtrado por prefijos [prefix filtering]
- Filtrado por posición [position filtering]
- Filtrado por cotas [bound filtering]

Escalabilidad

Índice invertido [inverted index]

- Se construye un índice invertido sobre Y:
  para cada término de los que aparecen en Y,
  se mantiene la lista de cadenas en las que aparece.
- Dado un término t, se utiliza el índice para acceder rápidamente a las cadenas de Y que lo contienen.

Limitaciones

- La lista de cadenas correspondientes a algunos términos (p.ej. "stop words") puede ser muy larga.
- Requiere enumerar todos los pares de cadenas que comparten al menos una palabra.

Escalabilidad

Índice invertido [inverted index]

Set *X*

1: {lake, mendota}
2: {lake, monona, area}
3: {lake, mendota, monona, dane}

Set *Y*

4: {lake, monona, university}
5: {monona, research, area}
6: {lake, mendota, monona, area}

Terms in *Y*	ID Lists
area	5
lake	4, 6
mendota	6
monona	4, 5, 6
research	5
university	4

Escalabilidad

Filtrado por longitud [size filtering]

Sólo se consideran cadenas de Y determinadas longitudes:

- Dada una cadena $x \in X$, se infiere una restricción sobre la longitud de las cadenas de Y con las que x pueda casar.
- Se utiliza un árbol B como índice para acceder sólo a las cadenas que satisfagan la restricción de longitud.

EJEMPLO: Coeficiente de Jaccard $J(x,y) = |X \cap Y| / |X \cup Y|$

- Dos cadenas emparejan si $J(x,y) \geq t$.
- Dada una cadena x, sólo pueden emparejar con x aquéllas cadenas y tales que $|x|/t \geq |y| \geq |x|*t$

Escalabilidad

Filtrado por longitud [size filtering]

x = {lake, mendota}
t = 0.8

Set *Y*
4: {lake, monona, university}
5: {monona, research, area}
6: {lake, mendota, monona, area}
7: {dane, area, mendota}

Para que $y \in Y$ case con x usando el coeficiente de Jaccard:

$2/0.8 = 2.5 \geq |y| \geq 1.6 = 2*0.8$

Ninguna cadena de Y satisface la restricción.

Escalabilidad

Filtrado por prefijos [prefix filtering]

IDEA
Si dos conjuntos comparten muchos términos, subconjuntos grandes de ellos también los compartirán.

EJEMPLO: Solapamiento $O(x,y) = |X \cap Y|$

- Si $|X \cap Y| \geq k$, cualquier subconjunto $X' \subseteq X$ de tamaño al menos $|X|-(k-1)$ se solapará con Y.
- Para encontrar los pares (x,y) tales que $|X \cap Y| \geq k$, podemos construir un subconjunto X' de tamaño $|X|-(k-1)$ y utilizar un índice invertido para obtener todas las cadenas y que se solapan con x.

Escalabilidad

Filtrado por prefijos [prefix filtering]

$O(x,y) \geq 2$

x: {lake, monona, area}
(x': {lake, monona})

y: {lake, mendota, monona, area}

Set *X*

1: {lake, mendota}
2: {lake, monona, area}
3: {lake, mendota, monona, dane}

Set *Y*

4: {lake, monona, university}
5: {monona, research, area}
6: {lake, mendota, monona, area}
7: {dane, area, mendota}

Terms in *Y*	ID Lists
area	5, 6, 7
lake	4, 6
mendota	6, 7
monona	4, 5, 6
research	5
university	4
dane	7

x_1 = {lake, mendota} ➔ x_1'= {lake}

El índice invertido nos da las cadenas que contienen el término de x_1': $\{y_4, y_6\}$

Escalabilidad

Filtrado por prefijos [prefix filtering]

¿Cómo seleccionar el subconjunto de forma inteligente?

- Se selecciona un subconjunto x' de x para compararlo con el conjunto completo de cadenas de Y.
- Cuanto más pequeño sea el subconjunto de Y que tengamos que considerar, mejor.

IDEA

Imponer un orden sobre los términos (p.ej. frecuencia creciente) y seleccionar los términos menos frecuentes de x para formar el subconjunto x'…

Escalabilidad

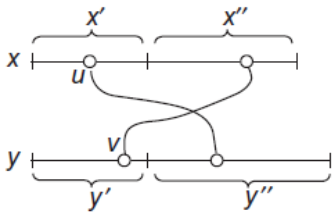
Filtrado por prefijos [prefix filtering]

¿Cómo seleccionar el subconjunto de forma inteligente?

PROPIEDAD

Dados dos conjuntos x e y tales que $|x \cap y| \geq k$,
ordenamos sus elementos de menor a mayor frecuencia.
Sea x' el prefijo de x de tamaño |x|-(k-1)
e y' el prefijo de y de tamaño |y|-(k-1):

x' e y' se solapan.



Escalabilidad

Filtrado por prefijos [prefix filtering]

¿Cómo seleccionar el subconjunto de forma inteligente?

ALGORITMO

- Se ordenan los términos de $x \in X$ e $y \in Y$ en orden creciente de frecuencia.
- Para cada $y \in Y$, se crea y',
  el prefijo de y de tamaño |y|-(k-1).
- Se crea un índice invertido sobre los prefijos y'.
- Para cada $x \in X$, se crea x',
  el prefijo de x de tamaño |x|-(k-1).
- Se utiliza el índice invertido para encontrar las cadenas y para las que el prefijo x' se solapa con el prefijo y'.

Escalabilidad

Filtrado por prefijos [prefix filtering]

¿Cómo seleccionar el subconjunto de forma inteligente?

university < research
< dane < area
< mendota < monona < lake

Reordered Set *X*

1: {mendota, lake}
2: {area, monona, lake}
3: {dane, mendota, monona, lake}

Reordered Set *Y*

4: {university, monona, lake}
5: {research, area, monona}
6: {area, mendota, monona, lake}
7: {dane, area, mendota}

x_1 = {lake, mendota}
➔ x_1'= {mendota}

Índice sobre prefijos

Terms in *Y*	ID Lists
area	5, 6, 7
mendota	6
monona	4, 6
research	5
university	4
dane	7

vs.

Terms in *Y*	ID Lists
area	5, 6, 7
lake	4, 6
mendota	6, 7
monona	4, 5, 6
research	5
university	4
dane	7

NOTA: En la práctica, el índice invertido sobre los prefijos es significativamente más pequeño que el índice invertido sobre las cadenas completas.

Escalabilidad

Filtrado por prefijos [prefix filtering]

¿Cómo aplicarlo sobre el coeficiente de Jaccard?

$$J(x,y) \geq t \Leftrightarrow O(x,y) \geq \alpha = \frac{t}{1+t} \cdot (|x| + |y|)$$

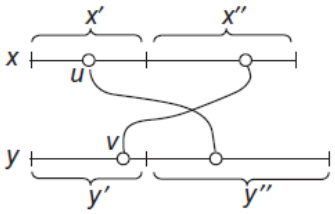
- El umbral α depende de |x| e |y|.
- Se tienen que indexar los prefijos de $y \in Y$ de longitud $|y| - \lceil t\,|y| \rceil + 1$ para garantizar que no se pierden posibles emparejamientos.

Escalabilidad

Filtrado por posición [position filtering]

Limita el conjunto de pares candidatos derivando una cota superior sobre el solapamiento de x e y:

$x = x' \cup x''$
$y = y' \cup y''$



$$O(x,y) \leq |x' \cap y'| + \min\{|x''|, |y''|\}$$

Escalabilidad

Filtrado por posición [position filtering]

x = {dane, area, mendota, monona, lake}
y = {research, dane, mendota, monona, lake}

J(x,y) ≥ 0.8

- Por un lado (filtrado por prefijos),
  O(x,y) ≥ 4.44 = 0.8 / (1+0.8) * (5+5)
- Por otro (filtrado por posición),
  O(x,y) ≤ 4 = 1 + min{3,3}

Sin comparar las cadenas, descartamos el par (x,y).

Escalabilidad

Filtrado por cotas [bound filtering]

Optimización para el coeficiente de Jaccard generalizado:

$$\mathbf{GJ(x,y) = \sum_{(x_i,y_j)\in M} s(x_i,y_j) \,/\, (|B_x| + |B_y| - |M|)}$$

Conjuntos de pares (x_i,y_j):

S1: Para cada $x_i \in B_x$, encontrar el elemento $y_j \in B_y$ de mayor similitud tal que $s(x_i,y_j) \geq \alpha$

S2: Para cada $y_j \in B_y$, encontrar el elemento $x_i \in B_x$ de mayor similitud tal que $s(x_i,y_j) \geq \alpha$

- Cota superior:
  $UB(x,y) = \sum_{(x_i,y_j)\in S1\cup S2} s(x_i,y_j) \,/\, (|B_x| + |B_y| - |S_1 \cup S_2|)$
- Cota inferior:
  $LB(x,y) = \sum_{(x_i,y_j)\in S1\cap S2} s(x_i,y_j) \,/\, (|B_x| + |B_y| - |S_1 \cap S_2|)$

Escalabilidad

Filtrado por cotas [bound filtering]

Optimización para el coeficiente de Jaccard generalizado:

$$\mathbf{GJ(x,y) = \sum_{(x_i,y_j)\in M} s(x_i,y_j) \,/\, (|B_x| + |B_y| - |M|)}$$

Para cada (x,y) se calcula una cota inferior LB(x,y) y una cota superior UB(x,y) sobre GJ(x,y):

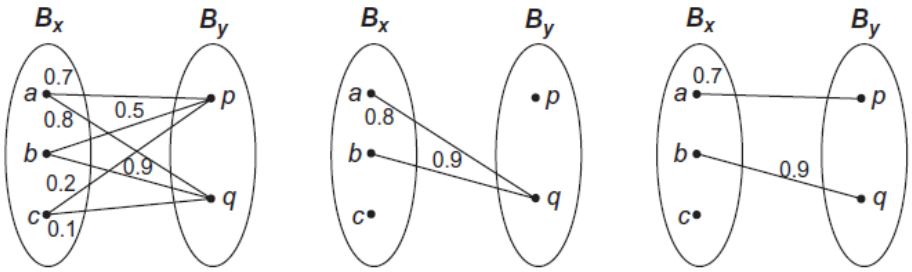
- Si $UB(x,y) \leq t$, el par (x,y) puede ignorarse.
- Si $LB(x,y) \geq t$, el par (x,y) casa [sin compararlo].
- En otro caso, se calcula GJ(x,y).

Escalabilidad

Filtrado por cotas [bound filtering]

Optimización para el coeficiente de Jaccard generalizado:

$GJ(x,y) = \sum_{(x_i,y_j)\in M} s(x_i,y_j) / (|B_x| + |B_y| - |M|)$



S1 = {(a,q), (b,q)}
S2 = {(a,p), (b,q)}

UB(x,y) = (0.8+0.9+0.7+0.9)/(3+2-3) = 1.65

LB(x,y) = 0.9/(3+2-1) = 0.225

Escalabilidad

Extensiones a otras medidas de similitud

Traducción del valor s(x,y) en restricciones sobre otras medidas de similitud para las que funcione la técnica:

- **Distancia de edición**
  Filtrado de prefijos de longitud $q\varepsilon+1$ usando q-gramas.

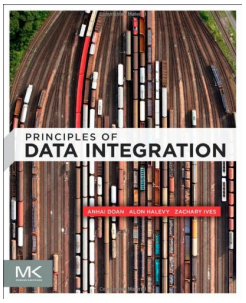
$$d(x,y) \le \epsilon \Rightarrow O(x,y) \ge \alpha = (\max\{|B_x|,|B_y|\} + q - 1) - q\epsilon$$

- **TF/IDF**
  Filtrado de prefijos de longitud $|x| - \lceil t^2|x| \rceil + 1$.

$$C(x,y) \ge t \Leftrightarrow O(x,y) \ge \lceil t \cdot \sqrt{|x||y|} \rceil$$

Bibliografía recomendada

- Hai Doan, Alon Halevy & Zachary Ives:
  Principles of Data Integration
  Morgan Kaufmann, 1st edition, 2012.
  ISBN 0124160441
  http://research.cs.wisc.edu/dibook/



Chapter 4: String Matching