**BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA**

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

**Indización Posicional**

**EQUIPO:**

*Alvarez Santos Fredy*

*Flores Velázquez David*

*Juárez Escárcega Ilse Lilian*

*Torres Timal Martín Rafael*



*Procesamiento del Lenguaje Natural*

*Dr. David Eduardo Pinto Avendaño*

***Otoño 2012***

índices Posicionales

El programa indiza las posiciones de las palabras de acuerdo aun documento. La idea es poder localizar una frase de acuerdo a la distancia de sus posiciones.

**Algoritmo**

El algoritmo consiste en lo siguiente, dado un corpus de entrenamiento con la siguiente estructura:

*IdDoc, texto …*

*IdDoc, texto …*

*IdDoc, texto …*

Y genera un documento de salida con la siguiente estructura:

*Palabra [frecuenciaTotal] : IdDoc, frecuenciaDoc, posicionPalabra*

*Palabra [frecuencia] : IdDoc, frecuenciaDoc, posicionPalabra*

*Palabra [frecuencia] : IdDoc, frecuenciaDoc, posicionPalabra*

**Código**

Indiza

awk '{

for (i=2; i<=NF; i++) { #para todas las palaras del documento

doc = $1;

$i = tolower($i);

if($i != ""){

indice[$i] = indice[$i] "," doc; #indice[]=indice documento

frecdoc[$i,doc]++; #frecuencia de la palabra por doc

pos[$i,doc]=pos[$i,doc] "," i ; #posicion palabra por documentoo

df[$i]++;

}

}

}

END{

for(x in indice){

aux=0;

d = substr(indice[x], 2);

ln = split(d, a, ",");

cad = "";

for(j = 1; j <= ln; j++){

p = substr(pos[x,a[j]], 2);

if(aux != a[j]){

cad = cad a[j] "," frecdoc[x,a[j]] "," p ";";

}

aux=a[j];

}

print x " , " df[x] " : " cad;

}

}' $\*

awk '

FILENAME==indice{

df[$1]=$2;

posting[$1]=$5;

next;

}

FILENAME==chistes{

chiste[$1]=$0;

next;

}

function abs(x){

return (((x < 0.0) ? -x : x) + 0.0)

}

function PositionalIntersect2(p1, p2, k) {

# entra posting de la sig forma

# idDoc1,pos1,...,posn:idDoc2,pos1,...,posn:.......:idDocn,pos1,...,posn:

answer = "";

longp1 = split(p1, docpos1, ";"); #longitud y subdivision por documento y posicion de p1 # idDoc1, pos1, pos2......posn

# ...

# idDocn, pos1, pos2......posn

longp2 = split(p2, docpos2, ";"); #longitud y subdivision por documento y posicion de p1

idocpos1=1; #indice para moverse entre docpos1

idocpos2=1; #indice para moverse entre docpos1

if(longp1 > 0 && longp2 > 0){

while((idocpos1<longp1)&&(idocpos2<longp2)){

longpos1 = split(docpos1[idocpos1], poss1, ","); #longitud de posiciones y subdivision de las posiciones1 idDoc

# pos1

# ...

# posn

longpos2 = split(docpos2[idocpos2], poss2, ","); #longitud de posiciones y subdivision de las posiciones2

if(poss1[1] == poss2[1]){# verificacmos que sea el mismo id de documento

cadena = "";

iposs1 = 3; #indice para poss1

iposs2 = 3; #indice para poss2

while(iposs1 <= longpos1){ #verificamos que no sea nulo pp1 del libro

while(iposs2 <= longpos2){ #verificamos que no sea nulo pp2 del libro

if(abs(poss1[iposs1]-poss2[iposs2]) <= k){ #checamos que se cumpla el minimo de distancia

cadena = cadena "," poss2[iposs2];

}

else{

if(poss2[iposs2] > poss1[iposs1])

break;

}

iposs2++;

}

if(cadena != "")

longcad = split(cadena, cad, ",");

if(cadena != ""){

longcad = split(cadena, cad, ",");

icad = 2;

while((icad <= longcad) && (abs(cad[icad]-poss1[iposs1]) > k)){

icad++;

}

if(icad<=longcad){

for(i=icad; i<=longcad;i++)

answear = answear ";" poss1[1] "," poss1[iposs1] ","cad[i];

}

}

iposs1++;

}

idocpos1++;

idocpos2++;

}

else{

if(poss1[1] < poss2[1]){

idocpos1++;

idocpos2++;

}

}

}

}

if(answear!="")

answear=substr(answear, 2);

return answear;

}

{

if (NF == 1) {

ne = split(posting[$1], a, ":");

for (i=1; i<=ne; i++) {

split(a[i], idoc, ",");

print chiste[idoc[i]];

print "";

}

}

else {

lista = PositionalIntersect2(posting[$1], posting[$2], $3);

}

}

END{

print "------------------------------------";

print ""

print "------------------------------------";

ne = split(lista, a, ";");

#print lista;

if(ne>0){

aux = -7;

j=1;

for(i=1; i<=ne; i++) {

if(a[i]!=""){

split(a[i], chi, ",");

if(aux!=chi[1]){

print j " : " chiste[chi[1]];

j++;

}

else{

j++;}

aux = chi[1];

}

}

print "------------------------------------";

print ""

print "------------------------------------\n";

}

}' indice=$1 chistes=$2 $\*

**Ejecución:**

1) Genera el modelo a ser probado:

bash indizaPosicional.awk corpusMin.txt > ModeloPosicional.log

2) Consultas:

bash sriPosicional.awk ModeloPosicional.log corpusMin.txt testPosicional.log