BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Indices Invertidos

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 Invertidos

El programa indiza crea un posting-list a partir de un corpus. El posting indiza todas las palabras del vocabulario con su respectiva frecuencia y id del documento.

Las funciones OR, AND y AND-NOT son utilizadas para disminuir el espacio de búsqueda en una colección de documentos en el momento de realizar una consulta, esto es posible gracias a el posting generado por indiza.
Las tablas consecutivas muestran los resultados a obtener para cada uno de las funciones:

Función OR

a	b	S
0	0	0
0	1	1
1	0	1
1	1	1

Función AND

a	b	S
0	0	0
0	1	0
1	0	0
1	1	1

Función AND-NOT

a	S
1	0
0	1

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 [frecuencia] : IdDoc
Palabra [frecuencia] : IdDoc
Palabra [frecuencia] : IdDoc

Código

Indiza

```
awk '{
  ++contador;
  for (i=2; i<=NF; i++) {
    if (!($i in vocabulario)) {
      indice[$i] = indice[$i] "," contador;
      vocabulario[$i]=1;
      df[$i]++;
    }
  }
  delete vocabulario;
}
END {
  for (x in indice) print x " [ " df[x] " ] : " substr(indice[x], 2);
}' $*
```

OR

```
function OR(p1, p2) {
  answer = "";

  na = split(p1, a, ",");
  nb = split(p2, b, ",");

  pp1 = 1;
  pp2 = 1;

  while ((pp1 <= na) || (pp2 <= nb)) {
    if (pp1 > na) {
      answer = answer "," b[pp2];
      pp2++;
    } else {
      if (pp2 > nb) {
        answer = answer "," a[pp1];
        pp1++;
      } else {

        if (a[pp1] == b[pp2]) {
          answer = answer "," a[pp1];
          pp1++;
          pp2++;
        } else {
          if (a[pp1] < b[pp2]) {
            answer = answer "," a[pp1];
            pp1++;
          } else {
            answer = answer "," b[pp2];
            pp2++;
          }
        }
      }
    }
  }

  return substr(answer,2);
}
```

AND

```
function AND(p1, p2) {
 answer = "";

 na = split(p1, a, ",");
 nb = split(p2, b, ",");

 pp1 = 1;
 pp2 = 1;

 while ((pp1 <= na) && (pp2 <= nb)) {
  if (a[pp1] == b[pp2]) {
   answer = answer "," a[pp1];
   pp1++;
   pp2++;
  } else {
   if (a[pp1] < b[pp2]) pp1++;
   else pp2++;
  }
 }
```

AND-NOT

```
function AND_NOT(p1, p2) {
 answer = "";
 na = split(p1, a, ",");
 nb = split(p2, b, ",");
 pp1 = 1;
 pp2 = 1;
 while ((pp1 <= na) && (pp2 <= nb)) {
  if (a[pp1] == b[pp2]) {
   pp1++;
   pp2++;
  } else {
   if (a[pp1] < b[pp2]) {
    answer = answer "," a[pp1];
    pp1++;
   } else pp2++;
  }
```

```
  }
  return substr(answer, 2);
}
```

Ejecución:

1) Genera los el Modelo
bash indiza.awk corpus.txt > Modelo.log

Obtiene la consulta
2) bash NombreFunción.awk Modelo.log corpus.txt test.log