Modulo V

N. Galesi, S. Guerrini
Lab Prog. A.A. 2004/2005 – II Semestre

1 Modulo V

Dopo aver costruito il dizionario globale, ci occuperemo di effettuare la ricerca di una o più parole e di restituire la lista di pagine web in cui tali parole occorrono ordinate secondo il *rank*, un parametro che si ottiene combinando il PageRank con informazioni sul tipo di occorrenze delle parole nella pagina.

2 Calcolo del Rank

Consideriamo i due casi:

1. *ricerca semplice* di una singola parola;
2. *ricerca multipla*.

Nel Modulo V implementeremo solo la ricerca semplice, ma daremo dei cenni per la soluzione della ricerca multipla.

2.1 Ricerca singola

Vediamo come calcolare il rank con un esempio. Supponiamo di cercare la parola word e di voler calcolare il rank del file pagina, uno degli ipertesti in cui tale parola occorre.

Per prima cosa associamo un peso ad ogni possibile *stato* in cui la parola può trovarsi. Lo stato di una parola si ottiene specifando dove si trova la parola (titolo o corpo), il tipo, il font, la dimensione, ecc. La tabella in Figura 1 descrive come associare dei pesi ad ogni possibile stato di una parola (la X indica che la caratteristica è presente, la $*$ che può esserci o non esserci e lo spazio vuoto che non è presente).

La tabella in Figura 1 si può visualizzare come un vettore di pesi di dimensione pari al numero di stati differenti (25). Denotiamo quindi con `s[i]` il peso corrispondente allo stato i-esimo.

Usando la hitlist di word per il file pagina, calcoliamo il valore `n[i]` che conta il numero di volte che la parola word è nello stato i-esimo nell'ipertesto pagina. Possiamo quindi calcolare il peso complessivo

$$p = \sum_{i=1,\ldots,25} (s[i] * n[i])$$

di word in pagina. I pesi p ottenuti per gli ipertesti in cui occorre word verranno poi normalizzati in una scala tra 1 e 100. Il *rank* di pagina si otterrà sommando il peso complessivo delle occorrenze normalizzato tra 1 e 100 al suo PageRank.

Peso	Titolo	Link	bold	it	large	normal	small
1	X	*	*	*	*	*	*
0.8		X	X	X	X		
0.8		X	X	X		X	
0.8		X	X	X			X
0.7		X	X		X		
0.7		X	X			X	
0.7		X	X				X
0.6		X		X	X		
0.6		X		X		X	
0.6		X		X			X
0.55		X			X		
0.55		X				X	
0.55		X					X
0.5			X	X	X		
0.5			X	X		X	
0.5			X	X			X
0.4			X		X		
0.4			X			X	
0.4			X				X
0.3				X	X		
0.3				X		X	
0.3				X			X
0.2					X		
0.2						X	
0.2							X

Figura 1: Tabella Pesi degli Stati

3 Cosa Fare nel Modulo 5

Data una query composta da una parola, il modulo V deve costruire una lista di ipertesti che contengono la parola, calcolare per ognuno di essi il rank ed infine restituire la lista degli ipertesti ordinata secondo il rank.

La lista degli ipertesti va implementata con la seguente struttura

```
struct LiIp{
       char *it_name; /* nome del'ipertesto contenente la parola */
       struct info_ipertesto *it; /* ptr all'elemento del DI */
       int rank; /* rank dell'ipertesto*/
       struct LiIp *next_el;
}
```

La tabella dei pesi degli stati, `tsw`, è definita da:

```
#define NUM_STATI 25
double tsw[NUM_STATI];
```

Il Modulo V deve usare `nodes.h` e l'header `rank.h` che contiene la definizione dei tipi di dato precedentemente descritti e i prototipi delle funzioni da implementare.

Le funzioni da implementare sono:

```
struct LiIp *build_LiIp(struct dgp *, char *x)
```

che riceve come input il dizionario globale e la parola da ricercare e costruisce la lista degli ipertesti che ne contengono delle occorrenze;

```
struct *LiIP calc_rank_LiIp(char *, struct LiIp *)
```

che riceve in input una parola e una lista di ipertesti che la contengono e calcola il rank degli ipertesti;

```
struct LiIp *ord_LiIp(struct LiIp *)
```

che riceve una lista di ipertesti per i quali è già stato calcolato il rank e la restituisce ordinata rispetto a valori decrescenti del rank.

4 Cenni sul Calcolo del Rank nel caso della Ricerca Multipla

Nel caso della ricerca multipla la situazione è più complessa e dobbiamo considerare anche il fattore della vicinanza delle parole in un ipertesto. In questa sezione daremo dei cenni per la sua soluzione anche se nel Modulo V non ne viene richiesta l'implementazione.

Suponiamo che una query sia formata dalle parole word1,..., wordr. Sia pagina un ipertesto in cui $k > 1$ parole delle r della query occorrono. Per calcolare il

rank di pagina consideriamo una lista, che chiamiamo LCP da Lista Combinazioni Parole, con tanti elementi quante sono le differenti combinazioni possibili delle occorrrenze delle parole word1,..., wordk in pagina. (Ad esempio se $k = 3$ e se word1 occorre x volte, word2 occorre y volte e word3 occorre z volte, la lista contiene $x * y * z$ elementi.)

Il rank dell'ipertesto pagina verrà quindi calcolato sulla base:

1. del peso dello stato in cui ogni combinazione si trova e
2. sulla *vicinanza* delle k parole nel testo di pagina.

Vediamo come calcolare questi due parametri. Il peso di una combinazione di parole è dato dal massimo peso di una parola nella combinazione. Il fattore vicinanza può assumere 6 valori in funzione della distanza massima tra due occorrenze di parole nella combinazione di k parole, secondo la seguente tabella.

Fattore Vicinanza	Distanza massima
1	$\leq k - 1$
0.5	$\leq k + 5$
0.3	$\leq k + 10$
0.2	$\leq k + 20$
0.1	$\leq k + 50$
0.01	$> k + 50$

A questo punto per calcolare il rank possiamo procedere analogamente al caso della ricerca singola. Invece della tabella dei pesi degli stati, consideriamo una tabella formata da tutte le possibili coppie *(stato, fattore vicinanza)* (25*6 = 150 combinazioni possibili), ed invece della hitlist di una parola consideriamo la lista di combinazioni LC. Supponiamo quindi di memorizzare 150 pesi per ognuna delle possibili coppie in un vettore di pesi $t[i, j]$, $i = 1, \ldots, 25$, $j = 1, \ldots, 6$. Dopo aver calcolato calcolato il numero $n[i, j]$ di elementi in LC che si trovano nello stato i e hanno fattore di vicinanza j, il rank di pagina è calcolato come somma del suo PageRank più la quantità $\sum_{i,j} t[i, j]n[i, j]$ scalata da 1 a 100.