Fondamenti di Informatica ◇ 2019-20

Laboratorio #5 del 05-12-2019

Tag Cloud o Word Cloud

Si vuole realizzare un programma che partendo da un file di testo crei una cosiddetta *Word Cloud* da visualizzarsi mediante una pagina web. Per raggiungere lo scopo, sono dati i seguenti file di testo (disponibili su `piazza.com`):

- `stopWords.it.txt` e `stopWords.txt` che contiene le parole che non vanno considerate in quanto non rilevanti dal punto di vista della semantica (ad esempio congiunzioni, articoli, ...) sia per la lingua italiana che per quella inglese,
- `d3.layout.cloud.js`, `d3.min.js` e `word-cloud.js` file javascript utilizzati dalla pagina web per realizzare la Word Cloud, e
- `wordcloud.htm` pagina html utilizzata per visualizzare la Word Cloud, che include la seguente direttiva:
  `<script type="text/javascript" src="words.js"></script>`
  Il file `words.js` è ciò che il programma deve creare per ottenere il risultato desiderato.

Il programma riceve in ingresso un file ASCII contenente il testo da analizzare e crea il file `words.js` risultato dell'analisi. Il formato di tale file è il seguente:

```
var tags = [
{"key": "programma", "value" : 26},
{"key": "C", "value" : 27},
{"key": "variabile", "value" : 25},
{"key": "ciclo", "value" : 25},
{"key": "costrutto", "value" : 27},
{"key": "tipo", "value" : 25},
{"key": "sottoprogramma", "value" : 24},
{"key": "parametro", "value" : 26},
{"key": "visualizza", "value" : 26},
{"key": "argc", "value" : 25},
...
{"key": "lista", "value" : 6}
];
```

A parte la prima e l'ultima riga, ogni elemento specifica un vocabolo quante volte compare nel testo. Nel realizzare la soluzione, si tengano presente i seguenti aspetti:

- non mettere nel file finale i vocaboli che compaiono meno di 6 volte;
- tutti i vocaboli hanno al più 30 caratteri (simbolo `LEN`);
- i vocaboli contenuti nei file `stopWords.it.txt` e `stopWords.txt` sono tutti minuscoli;

- ◇ i vocaboli nel file da analizzare possono avere maiuscole, si tratta di un testo reale;
- ◇ nel file da analizzare ci possono essere segni di interpunzione (virgola, apostrofo, ...) e devono essere opportunamente gestiti;
- ◇ il file da analizzare potrebbe contenere doppi spazi, trovare i vocaboli

A titolo di esempio di ciò che dovrebbe realizzare il programma si consideri il testo inziale qui riportato

> [...] Ai tempi in cui accaddero i fatti che prendiamo a raccontare, quel borgo, già considerabile, era anche un castello, e aveva perciò l'onore dalloggiare un comandante, e il vantaggio di possedere una stabile guarnigione di soldati spagnoli, che insegnavan la modestia alle fanciulle e alle donne del paese, accarezzavan di tempo in tempo le spalle a qualche marito, a qualche padre; e, sul finir dellestate, non mancavan mai di spandersi nelle vigne, per diradar l'uve, e alleggerire a' contadini le fatiche della vendemmia. Dall'una all'altra di quelle terre [...]

i vocaboli che il programma dovrebbe considerare, prima di aver ignorato quelli presenti nel file `stopWords.it.txt` sono:

> [...] ai tempi in cui accaddero i fatti che prendiamo a raccontare quel borgo già considerabile era anche un castello e aveva perciò onore alloggiare un comandante e il vantaggio di possedere una stabile guarnigione di soldati spagnoli che insegnavan la modestia alle fanciulle e alle donne del paese accarezzavan di tempo in tempo le spalle a qualche marito a qualche padre e sul finir estate non mancavan mai di spandersi nelle vigne per diradar uve e alleggerire contadini le fatiche della vendemmia una altra di quelle terre [...]

utilizzando il file `stopWords.it.txt`, il programma scarterà i vocaboli che compaiono in tale file, arrivando ai seguenti vocaboli (non è importante che non ci siano vocaboli ripetuti)

> tempi fatti prendiamo raccontare borgo considerabile castello onore alloggiare comandante vantaggio possedere stabile guarnigione soldati spagnoli insegnavan modestia fanciulle donne paese accarezzavan tempo tempo spalle marito padre finir estate mancavan spandersi vigne diradar uve alleggerire contadini fatiche vendemmia terre

Si utilizzi il seguente tipo di dato:

```
typedef struct wordlist_s {
  char word[LEN+1];
  int times;
  struct wordlist_s * next;
} wordlist_t;
```

Si considerino già a disposizione i seguenti sottoprogrammi (codice disponibile su `piazza.com`):

- ◇ `wordlist_t * increasing(wordlist_t * h, char * w);`

  inserisce un nuovo elemento in ordine alfabetico crescente rispetto al campo `word` della struttura;
- ◇ `wordlist_t * deleteptr(wordlist_t * h, wordlist_t * e)`

  elimina un elemento dalla lista;
- ◇ `wordlist_t * find(wordlist_t * h , char * w);`

  trova e restituisce della lista con un certo valore nel campo `word` della struttura, se esiste, altrimenti restituisce `NULL`.

Si utilizzino i sottoprogrammi delle librerie `string.h`, `stdlib.h`, `ctype.h` e simili quando utili, senza sviluppare ex-novo cose che già esistono.