Basi Dati multimediali
Fase 1 della sperimentazione di approfondimento

Descrizione: In questa attivita' di sperimentazione, approfondiremo questioni relative a:

- text processing,
- modello vettoriale
- grafi sociali

- Vi viene fornito lo schema DBLP (iche contiene le relazioni `authors`, `papers`, `citation`, `coauthorswpaper`, e `writtenby`) ed i dati relativi ad una rete parziale (sotto forma di un dump SQL).
- Scaricate i dati relativi alla rete di citazioni scientifiche DBLP. Scegliete voi se memorizzare i dati in un database relazionale (ad es. MySql) o se creare una struttura dati da tenere in memoria centrale per memorizzare la rete che vi e' stata fornita.

Task 1: Scrivete un progarmma che consideri tutti gli abstract nel database e ne elimini le stop words - a tal fine potete utilizzare gli elenchi di stop words al link

http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

Per il parsing del testo potete usare qualsiasi strumento pubblicamente disponibile per l'analisi del testo, ad esempio le librerie per l'estrazione delle keywords di Lucene

http://lucene.apache.org/core/4_2_1/core/org/apache/lucene/analysis/package-summary.html#package_description

o altri packages, quale ad esempio http://www-nlp.stanford.edu/software/tokenizer.shtml

Il programma da voi scritto dovra' creare e memorizzare un keyword vector per ciascun abstract analizzato, utilizzando sia il modello TF che il modello TF-IDF per i pesi.

Create un'interfaccia a linea di comando

`print_paper_vector` ***paperid model***

che stampi il keyword vector (sotto forma di sequenza di coppie $\langle keyword, weight \rangle$, ordinate in modo decrescente rispetto ai pesi) per l'articolo considerato, rispetto al modello di pesi considerato.

Task 2: Scrivere un programma che consideri tutti gli articoli scritti da un certo autore per creare un "combined keyword vector" per quell'autore. Nei "combined keyword vector", gli articoli piu' recenti devono pesare di piu' di quelli piu' vecchi.

Create un'interfaccia a linea di comando

`print_author_vector` ***authorid model***

che stampi il keyword vector per un dato autore, secondo il modello vettoriale dato.

Task 3: Scrivere un programma che consideri tutti i cautori di un dato autore a_i, e metta in luce il modo in cui gli articoli dell'autore dato *differiscano* da quelli dei suoi coautori. A tal fine, considererete i due modelli TF-IDF2 and PF:

- Nel modello TF-IDF2, per calcolare la componente TF considererete l'insieme di tutti gli articoli scritti dall'autore dato, mentre per il calcolo dell'IDF considererete l'insieme $coauthor\_and\_self(a_i)$ di tutti gli articoli scritti dall'autore e dai suoi coautori.

- Nel modello PF, identificherete il peso $u_{i,j}$ della chiave k_j per l'autore a_i sulla base del seguente meccanismo di feedback probabilistico (che discuteremo a lezione piu' avanti)

$$u_{i,j} = log \frac{r_{i,j}/(R_i - r_{i,j})}{(n_{i,j} - r_{i,j})/(N_i - n_{i,j} - R_i + r_{i,j})} \times \left| \frac{r_{i,j}}{R_i} - \frac{n_{i,j} - r_{i,j}}{N_i - R_i} \right|,$$

  dove:

  - $r_{i,j}$ e' il numero di articoli dei coautori in $coauthor\_papers(a_i)$ che non contengono la chiave k_j;
  - $n_{i,j}$ e' il numero di articoli in $coauthor\_and\_self(a_i)$ che non contengono la chiave k_j;
  - $R_i = \|coauthor\_papers(a_i)\|$, e
  - $N_i = \|coauthor\_and\_self(a_i)\|$.

  Create un'interfaccia a linea di comando

  `differentiate_author` ***authorid model***

  che stampi il vettore delle chiavi per un dato autore rispetto al modello vettriale considerato.