

Progetti sperimentali e tesine

Progetti sperimentali e tesine

- ◆ Progetti sperimentali
 - *Progetto 1*: indicizzazione e interrogazione con Lucene
 - *Progetto 2*: Costruzione di un DB che simuli un sistema di IR basato sul modello vettoriale
- ◆ Tesine compilative, su temi di approfondimento, es.:
 - Analisi comparativa di web search engine
 - Analisi comparativa di web search engine semantici
 - Modelli di IR strutturali e sistemi di IR strutturali
 - Tecniche e sistemi per il relevance feedback

Progetto 1: Indicizzazione ed interrogazione con Lucene

- 1) Formare gruppi di 3 studenti
- 2) Selezionare 15-20 documenti su un argomento a piacere
- 3) Indicizzare i documenti con Lucene
 - è possibile costruire più indici introducendo delle varianti nel processo di indicizzazione (operazioni di elaborazione del testo, algoritmo di stemming,...)
- 4) Definire 4 task di retrieval (individuando alcuni specifici bisogni informativi)
- 5) Per ogni task T , valutare la rilevanza di ogni documento rispetto a T :
 - identificare i documenti **rilevanti** per T ed i documenti **non rilevanti** per T
 - ogni membro del gruppo esprime il suo giudizio sulla rilevanza dei documenti
- 6) Per ogni task T di retrieval, formulare almeno 2 query:
 - un'espressione booleana
 - un insieme di termini pesati (i pesi possono essere impostato con l'operatore \wedge)
 - eventuali altre query che usino funzionalità avanzate del linguaggio di interrogazione di Lucene
- 7) Eseguire le query con Lucene sugli indici costruiti al passo 3 (ne basta anche uno solo)
- 8) Calcolare le misure di *recall* e *precision* per tutte le query eseguite
 - Nel caso in cui il risultato si possa interpretare come una lista ordinata, disegnare le curve recall-precision
- 9) Scrivere una relazione sul lavoro svolto

Progetto 1: come costruire le curve recall-precision

Risultato della query = lista ordinata

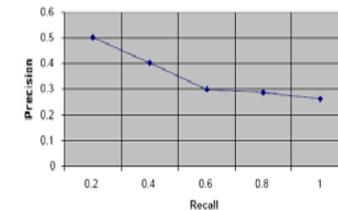
abcdef	NO
xvaxcds	OK
Adilmi	NO
Ljcvh	NO
Jff	OK
Fgv	NO
Fgrer	NO
Gdev	NO
Dgh	NO
Ghkt:i	OK
Loikih	NO
Jlki	NO
Sasd	NO
Dff	OK
Vdvcv	NO
Sfifed33	NO
Lkjh	NO
Lkpi	NO
hliw	OK

Fissando i livelli di recall a

1/5, 2/5, 3/5, 4/5, 5/5

i relativi valori di precision sono

1/2, 2/5, 3/10, 4/14, 5/19



Supponiamo che i (5) documenti rilevanti per la query appaiono nelle posizioni **2,5,10,14** e **19**

Progetto 2: Costruzione di un DB che simuli un sistema di IR basato sul modello vettoriale

INPUT: tre file di testo (formato cvs: valori separati da ";")

- I file possono essere costruiti applicando (su ogni documento di una data collezione) le funzioni di elaborazione del testo presenti in Lucene (e in Luke)

"docs.csv"

```
doc_id;term_id;tf;stemmed_word
1;297;1;span
1;1054;2;part
1;1373;1;bas
...
```

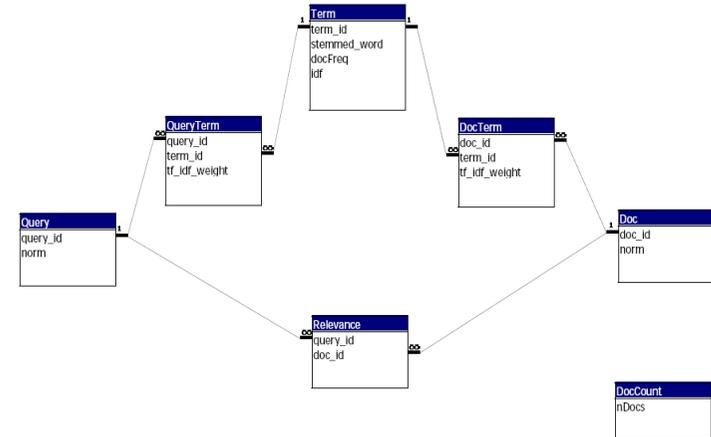
"relevance.csv"

```
query_id;doc_id;
1;184;
1;29;
1;31;
...
```

"queries.csv"

```
query_id;term_id;tf;stemmed_word
1;356;1;spee
1;2320;1;aeroelast
1;2726;1;obey
...
```

Lo schema del DB da costruire



Passi per costruire ed usare il DB

1. Importazione dei tre file in tre tabelle distinte

- docs.csv → xDocTerm (**temporanea**)
- queries.csv → xQueryTerm (**temporanea**)
- relevance.csv → Relevance

2. Popolamento delle varie tabelle, mediante query SQL

- DocCount, da xDocTerm (... conto i documenti)
- Term, da DocCount e xDocTerm (... calcolo idf di ogni termine)
- DocTerm, da Term e xDocTerm (... calcolo i pesi tf-idf)
- QueryTerm, da Term e xQueryTerm (... calcolo i pesi tf-idf)
- Doc, da DocTerm (... calcolo la norma di ogni documento)
- Query, da QueryTerm (... calcolo la norma di ogni query)

3. Scrittura di un'istruzione SQL per il ranking dei documenti rispetto ad una data query (... similarità del coseno fra ogni documento e la query)

Popolamento tabella DocCount

```
SELECT COUNT(*) AS nDocs
INTO DocCount
FROM
( SELECT DISTINCT doc_id FROM xDocTerm );
```

Popolamento della tabella *Term*

```
SELECT xDocTerm.term_id, xDocTerm.stemmed_word,  
       COUNT(*) AS docFreq,  
       LOG( MAX(DocCount.nDocs)/COUNT(*) ) / LOG(2) + 1 AS idf  
INTO Term  
FROM xDocTerm, DocCount  
GROUP BY xDocTerm.term_id, xDocTerm.stemmed_word;
```

$$\text{Log}_2 (N/n_i) = \text{Log} (N/n_i) / \text{Log}(2)$$

Popolamento della tabella *DocTerm*

```
SELECT DISTINCT xDocTerm.doc_id, xDocTerm.term_id,  
               (xDocTerm.tf_weight * Term.idf) AS tf_idf_weight  
INTO DocTerm  
FROM xDocTerm, Term  
WHERE Term.term_id = xDocTerm.term_id  
ORDER BY xDocTerm.doc_id, xDocTerm.term_id;
```

Popolamento della tabella *QueryTerm*

```
SELECT DISTINCT xQueryTerm.query_id,  
               xQueryTerm.term_id,  
               (xQueryTerm.tf_weight * Term.idf) AS tf_idf_weight  
INTO QueryTerm  
FROM xQueryTerm, Term  
WHERE Term.term_id= xQueryTerm.term_id  
ORDER BY xQueryTerm.query_id, xQueryTerm.term_id;
```

Popolamento della tabella *Doc*

```
SELECT DISTINCT DocTerm.doc_id,  
               SQR(  
                 SUM( DocTerm.tf_idf_weight * DocTerm.tf_idf_weight )  
               )  
               AS norm  
INTO Doc  
FROM DocTerm  
GROUP BY DocTerm.doc_id;
```

Popolamento della tabella *Query*

```
SELECT DISTINCT query_id,  
       SQR( SUM( tf_idf_weight * tf_idf_weight ) ) AS norm  
INTO Query  
FROM QueryTerm  
GROUP BY query_id;
```

Query per il ranking dei documenti rispetto ad una data query

```
SELECT Doc.doc_id, Cos_Sim FROM (  
  SELECT Doc.doc_id, Doc.norm AS norm,  
         SQR( SUM( DocTerm.tf_idf_weight * QueryTerm.tf_idf_weight ) ) / Doc.norm  
         AS Cos_Sim  
  FROM DocTerm, Doc, QueryTerm  
  WHERE DocTerm.term_id=QueryTerm.term_id  
        AND Doc.doc_id=DocTerm.doc_id  
        AND QueryTerm.query_id=[idQuery]  
  GROUP BY Doc.doc_id, Doc.norm)  
ORDER BY Cos_Sim DESC;
```