Information Retrieval - Exercise

Prof. Dr. Günter Neumann

Substitute: Jörg Steffen

Exercise - Computation of TF-IDF

The words car, auto, best should have the following frequencies in the ten documents Doc1, .., Doc10. Simply assume that all other words in the documents are stop words.

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
car	3	0	0	5	12	0	0	2	8	1
auto	8	6	0	12	0	0	9	1	3	10
best	0	1	7	0	1	5	12	0	2	0

Compute the tf-idf weight for each term using the equations in the slides 17-19.

Solution to Exercise Computation of TF-IDF

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
car	3	0	0	5	12	0	0	2	8	1
auto	8	6	0	12	0	0	9	1	3	10
best	0	1	7	0	1	5	12	0	2	0

Solution to Exercise Computation of TF-IDF

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
car	3	0	0	5	12	0	0	2	8	1
auto	8	6	0	12	0	0	9	1	3	10
best	0	1	7	0	1	5	12	0	2	0

- max(car) = max(auto) =max(best) = 12

Solution to Exercise Computation of TF-IDF

normalized term frequencies:

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
car	0.25	0	0	0.42	1	0	0	0.17	0.67	0.08
auto	0.67	0.5	0	1	0	0	0.75	0.08	0.25	0.84
best	0	0.08	0.58	0	0.08	0.42	1	0	0.17	0

Solution to Exercise Computation of TF-IDF

document frequencies:

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10	
car	3	0	0	5	12	0	0	2	8	1	6
auto	8	6	0	12	0	0	9	1	3	10	7
best	0	1	7	0	1	5	12	0	2	0	6

df(car) = 6

df(auto) = 7

df(best) = 6

Solution to Exercise Computation of TF-IDF

inverse document frequencies:

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10	
car	3	0	0	5	12	0	0	2	8	1	6
auto	8	6	0	12	0	0	9	1	3	10	7
best	0	1	7	0	1	5	12	0	2	0	6

df(car) = 6

df(auto) = 7

df(best) = 6

idf(car) = $\log_2(10/6)=0.74$

idf(auto) = $\log_2(10/7)=0.51$

idf(best) = $\log_2(10/6)=0.74$

Solution to Exercise Computation of TF-IDF

idf(car) = 0.74 idf(auto) = 0.51 idf(best) = 0.74

normalized term frequencies:

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
car	0.25	0	0	0.42	1	0	0	0.17	0.67	0.08
auto	0.67	0.5	0	1	0	0	0.75	0.08	0.25	0.84
best	0	0.08	0.58	0	0.08	0.42	1	0	0.17	0

TF-IDF scores:

term\doc	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9	Doc10
car	0.19	0	0	0.31	0.74	0	0	0.13	0.5	0.06
auto	0.34	0.26	0	0.51	0	0	0.38	0.04	0.13	0.43
best	0	0.06	0.43	0	0.06	0.31	0.74	0	0.13	0

Exercise IR - Inverted Lists

Assume the following documents

doc-id\text	
Doc1	“italy is world champion 2006”
Doc2	“germany and italy played each other in the semifinal”
Doc3	“germany was in the semifinal 2006”
Doc4	“germany won the semifinal in italy 1990”

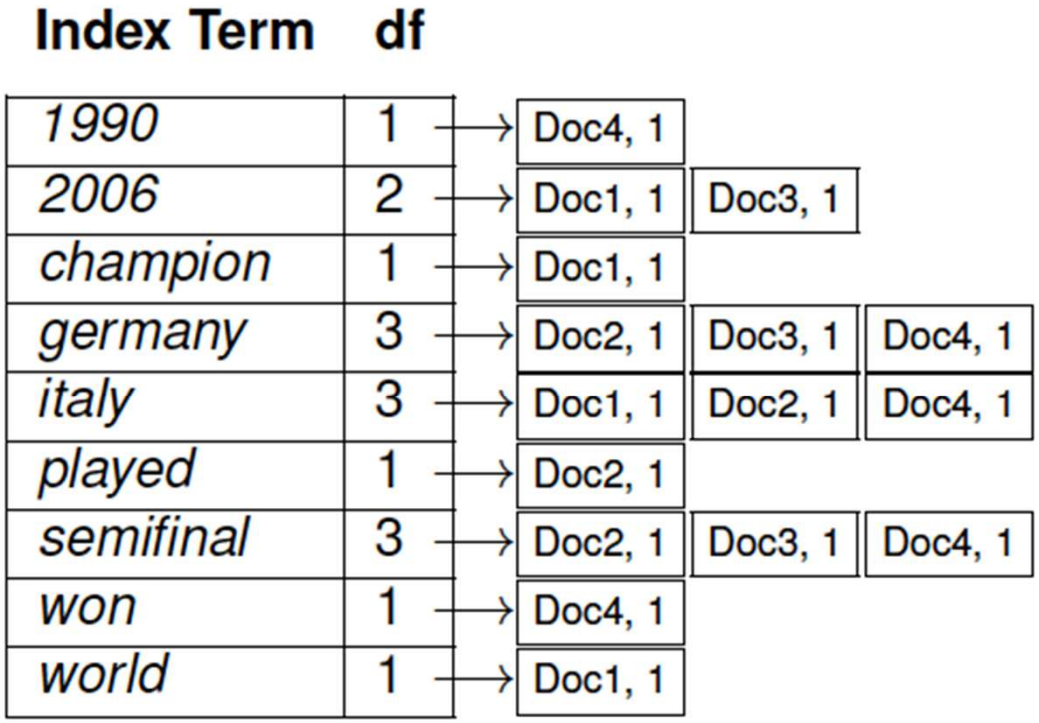
Assume that the following terms are stop words: is, and, in, the, was, each, other. Construct an inverted index.

Solution to Exercise Inverted Index

doc-id\text	
Doc1	“italy is world champion 2006”
Doc2	“germany and italy played each other in the semifinal”
Doc3	“germany was in the semifinal 2006”
Doc4	“germany won the semifinal in italy 1990”

1990	1
2006	2
champion	1
germany	3
italy	3
played	1
semifinal	3
won	1
world	1

Solution to Exercise Inverted Index



Information Retrieval in Practice: Apache Lucene

What is Apache Lucene ?

- a fast, scalable and feature-rich text search engine
- a well-documented, cross-platform Java library
- free and open-source software (Apache License 2.0)
- widely-used – often indirectly through Apache Solr
- actively developed under the umbrella of the Apache Software Foundation
- Download from: http://lucene.apache.org/

The Lucene Index

- stores a set of documents together with a number of *fields*
- index consists of set of *terms:* field name + *token*
- tokens result from processing field's content:
  tokenization, case normalization, stemming, stoplisting, …
- inverted index: field:token → doc id
- query index: field + search text
- Important: search text transformation into tokens must use the **SAME** preprocessing as during indexing

TF-IDF in Lucene

- search results are ranked by relevance
- the Boolean Model (BM) is combined with the Vector Space Model (VSM)
- documents “approved” by the BM are scored by the VSM
- the weights of the vectors in the VSM are tf-idf values
- Cosine Similarity is used for scoring indexed documents relative to a search query
- More details: http://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Example

- index of books
- fields: author, title, publishing date, abstract, …
- Jeff Hobbs: The Short and Tragic Life of Robert Peace

author: jeff	title: short	date: 23.09.2014	abstract: great
author: hobbs	title: tragic		abstract: young
	title: life		abstract: man
	title: robert		…
	title: peace		

Properties of Index Fields

- stored: the raw content of the field is stored in the index
  → might required a lot of space

- indexed: only indexed fields can be searched

  - analyzed: apply preprocessing to raw field content

  - not analyzed: store raw field content as single token

Lucene Query Language

TYPE	SYNTAX	DESCRIPTION
Token	t	Match token t
Phrase	“cs”	Match tokens in cs in exact order without gaps
Field	f:Q	Match query Q in field f
Wildcard Character	cs1?cs2	Match tokens starting with cs1, ending with cs2, with any char between
Wildcard Sequence	cs1*cs2	Match tokens starting with cs1, ending with cs2, with any char sequence between
Fuzzy	t~	Match token t approximately
Weighted Fuzzy	t~d	Match token t within minimum similarity d
Proximity	P~n	Match tokens in phrase P within distance n
Range, Inclusive	f:[t1 TO t2]	Match tokens lexicographically between tokens t1 and t2 inclusive
Range, Exclusive	f:(t1 TO t2)	Match tokens lexicographically between tokens t1 and t2 exclusive
Boosting	P^d	Match phrase P, boosting score by d
Disjunction	Q1 OR Q2	Match query Q1 or query Q2 (or both)
Conjunction	Q1 AND Q2	Match query Q1 and match query Q2
Difference	Q1 NOT Q2	Match query Q1 but not query Q2
Must	+P	Token or phrase P must appear
Mustn’t	-P	Token or phrase P must not appear

Apache Lucene Demo