# Lista de Exercícios - Recuperação de Informação

## Exercício 1

Considere a seguinte coleção com três documentos:

> 1. The old night keeper keeps the keep in the town.
> 2. In the big old house in the big old gown.
> 3. The house in the town had the big old keep.

**a.** Construa a matriz de incidência termo-documento para a coleção.

**b.** Construa o índice invertido para a coleção.

**c.** Construa o índice posicional para a coleção. (Referência: Seção 2.4.2)

## Exercício 2 (1.4)

For the queries below, can we still run through the intersection in time $O(x + y)$, where $x$ and $y$ are the lengths of the postings lists for Brutus and Caesar? If not, what can we achieve?

**a.** Brutus AND NOT Caesar

**b.** Brutus OR NOT Caesar

## Exercício 3 (1.10)

Write out a postings merge algorithm, in the style of Figure 1.6 (page 11), for an $x$ *OR* $y$ query.

## Exercício 4 (1.11)

How should the Boolean query $x$ *AND NOT* $y$ be handled? Why is naive evaluation of this query normally very expensive? Write out a postings merge algorithm that evaluates this query efficiently.

## Exercício 5 (6.9)

What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

## Exercício 6 (6.10 adaptado)

Considere a tabela com a frequência dos termos nos documentos *Doc1, Doc2* e *Doc3* e o valor df para cada termo $t$.

| Termos | Doc1 | Doc2 | Doc3 | $df_t$ |
|---|---|---|---|---|
| car | 27 | 4 | 24 | 500 |
| auto | 3 | 33 | 0 | 300 |
| insurance | 0 | 33 | 29 | 100 |
| best | 14 | 0 | 17 | 800 |

Para uma coleção de documentos de tamanho $N = 1000$, calcule os pesos $tf\text{-}idf$ de cada termo para cada documento.

### Exercício 7 (6.12)

How does the base of the logarithm in $idf_t = log\frac{N}{df_t}$ affect the score calculation in $Score(q, d) = \sum_{t \in q} tf\text{-}idf_{t,d}$? How does the base of the logarithm affect the relative scores of two documents on a given query?

### Exercício 8 (6.15)

Recall the tf-idf weights computed in Exercise 6.10. Compute the Euclidean normalized document vectors for each of the documents, where each vector has four components, one for each of the four terms.

### Exercício 9 (6.17)

With term weights as computed in Exercise 6.15, rank the three documents by computed score for the query *car insurance*, for each of the following cases of term weighting in the query:

**a.** The weight of a term is 1 if present in the query, 0 otherwise.

**b.** Euclidean normalized idf.

### Exercício 10 (6.19)

Compute the vector space similarity between the query "digital cameras" and the document "digital cameras and video cameras" by filling out the empty columns in table. Assume N = 10,000,000, logarithmic term weighting (wf columns) for query and document, idf weighting for the query only and cosine normalization for the document only. Treat *and* as a stop word. Enter term counts in the tf columns. What is the final similarity score?

| Word | tf | wf | df | idf | $q_i$ = wf-idf | tf | wf | $d_i$ = normalized wf | $q_i \cdot d_i$ |
|---|---|---|---|---|---|---|---|---|---|
| digital | | | 10.000 | | | | | | |
| video | | | 10.000 | | | | | | |
| camera | | | 50.000 | | | | | | |