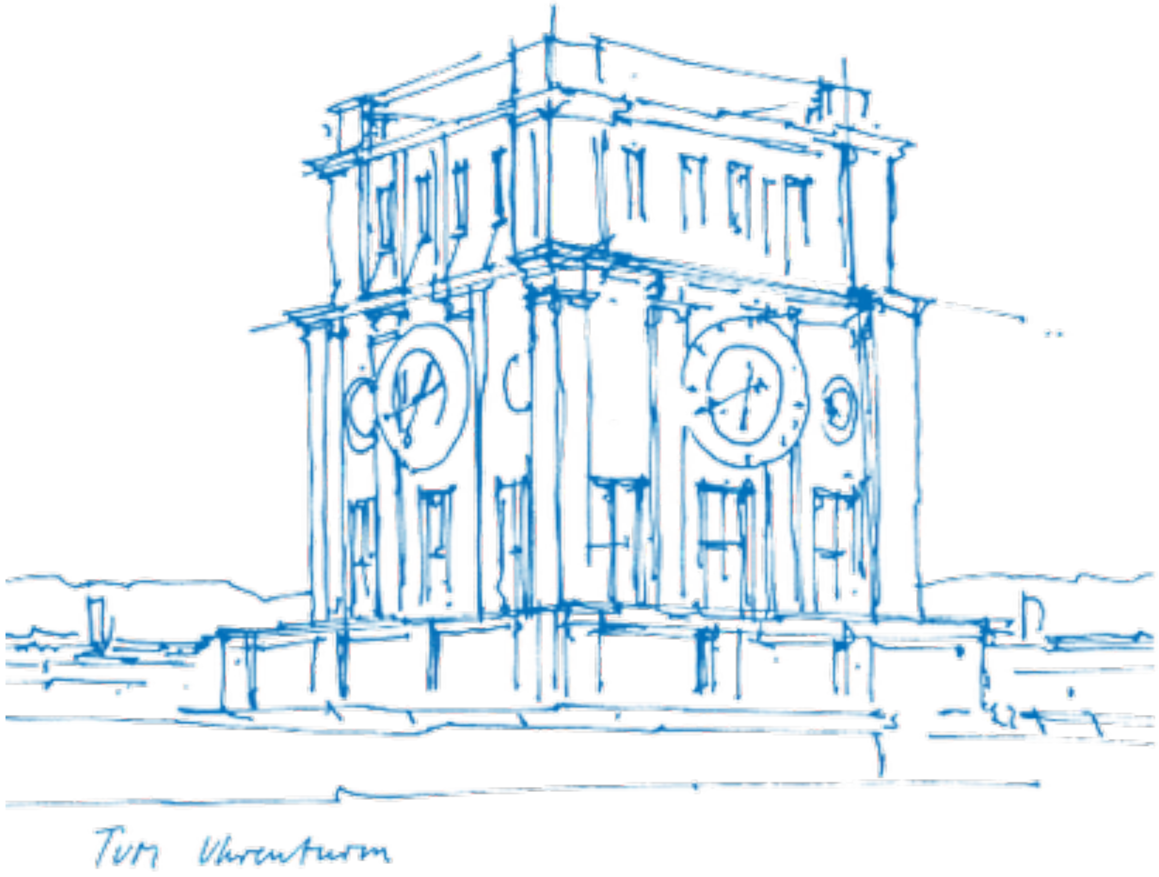
Seminar: Techniques for implementing main memory databases

Text analysis: TF-IDF

Dao Thuy Ngan Tran
Technische Universität München
Fakultät für Informatik
Lehrstuhl für Datenbanksysteme

Garching, 17. Oktober 2017



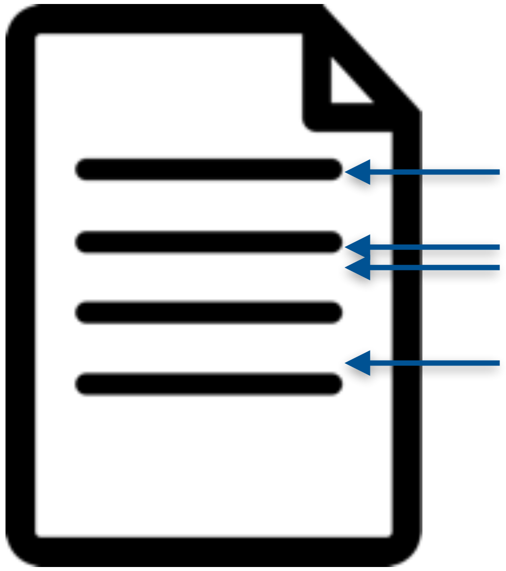
Base Model

TF-IDF as a model

A full text search is performed on a document collection, with each term (token) in a query weighted according to the *Term Frequency* and the *Inverse Document Frequency.*

TF (Term Frequency)

Is determined per token and per document

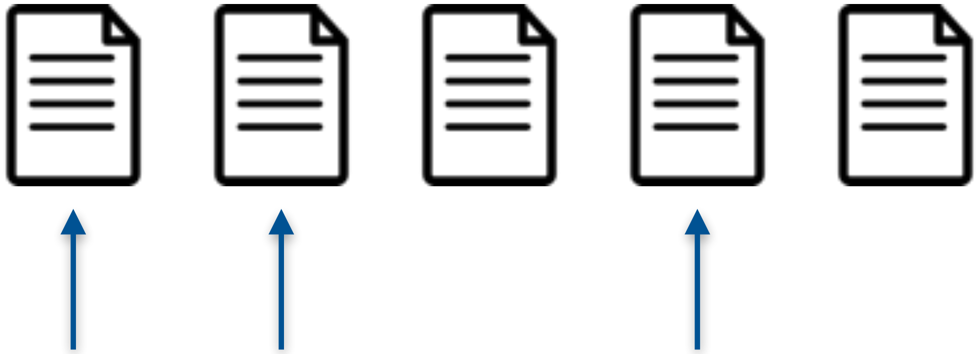


There are 4 occurrences of this token in this document

⇨ **TF is 4**

IDF (Inverse Document Frequency)

Is determined per token over the whole document collection



3 out of 5 documents in this collection contain this token

⇨ **IDF is log(5/3)**

TF and IDF as heuristics

Term Frequency
A term is simply weighted proportional to the number of times it appears in a document. If a document contains the term many times, we can assume that the document is relevant for our search because it might be just about the specific topic of the term.

Inverse Document Frequency
A term that occurs only in few documents, it might be a specific term to this query and is given a higher weight for the search. On the other side, common words like ‚the‘, ‚and‘, etc. …. should not be assigned a high weight for the search. Similar to Shannon’s *expected amount of information.*

Computing the TF-IDF vector (1)

Query: „data and programming languages“

Indexing

All documents in the sample document collection are indexed as followed:

Document: „SQL is a query language, which is used for accessing data bases.“

Resulting index:

```
<[„sql“, 1],
  [„is“, 2],
  [„a“, 1],
  [„language“, 1],
  [„which, 1“],
  [„used“, 1],
  [„for“, 1],
  [„accessing“, 1],
  [„data“, 1]>
```

Term frequencies can be easily obtained from the index

Computing the TF-IDF vector (2)

obtain idf

```
for each token do
   count = 0;
    for each document in collection do
     if(document_index contains token)
     count++;
   idf = log(collection_size/count);
```

compute tf-idf

```
for each token do
      idf = get_idf(token);
          for each document in collection do
          tf_idf += get_tf(token, document) * idf
```

tf-idf vector

The a possible resulting tf-idf vector could look like this:
<[„data“, 27.7536], [„language“, 14.1661], [„programming“, 14.1661], [„and“, 36.7368]>

Extended Ranking Model

Improving TF-IDF

What are drawbacks of the just presented TF-IDF base model?

- long documents tend to achieve a higher term frequency count
- the term frequency increases linearly
- the term frequencies within a document are independent from another

Two TF factors

Relative Intra-Document TF

Compute a TF that is normalized by the average TF of the document: $RITF = TF/avgTF(D)$

In order to bound the factor to 1, $BRITF = RITF/1+RITF$ is used.

This factor rewards documents which seem to be more specific about a query term.

Length Regularized TF

Normalize TF by taking document length into account:

$LRTF = TF * ld(1 + avgDocument\_length/Document\_length)$

In order to bound the factor to 1, $BLRTF = LRTF/1+LRTF$ is used.

This factor punishes long documents.

Combining the two factors

We want both properties, so our modified TF should consist of both factors.

$TFF = w * BRITF + (1-w) * BLRTF \text{ with } w \text{ as } 2/(1+ld(1 + query\_length))$

The main property of w should be its proportional increase to the query length.

Modifying IDF

Average elite set term frequency

Distinguish between term specific documents and only mentioning documents:

$AEF = collection\_tf/df$

We want an function proportionally increasing to inverse term density bounded to 1.

Thus, new_IDF is $IDF * (AEF/1+AEF)$

Using modified TF-IDF for ranking of documents

Each document is assigned a score representing the similarity to the query.

```
Similarity score
for each document in collection do
       simScore = 0;
       for each token in query do
             simScore += tff * new_idf
```

The document(s) with the highest similarity score to the query should be returned.

User-based TF-IDF

TF-ID$_u$F

- Another weighting scheme based on TF-IDF
- The computation remains the same - only the document collection changes
- term weights are determined according to the user’s personal document collection
- the actual search is still performed on the public document collection
- recently downloaded documents of user weighted higher
- hybrid schema for practical use

Thank you for your attention!