




# Information Retrieval Project

Elena Buscaroli  
January 2022



# Aim of the project

Implement an Information Retrieval system able to

- Perform ranked retrieval using the Vector Space Model:  
answer a query retrieving the K documents with highest scores
- Answer queries given in free-form text – as a sequence of words  
> “ the crystalline lens in vertebrates including humans”
- Allow relevance feedback:  
user interaction to provide a set of relevant documents, given a query
- Allow pseudo-relevance feedback:  
relevance feedback without the user interaction

Evaluate the system on a set of queries.

# The vector space model

In the Vector Space Model (VSM):

- **Terms:** elements of the canonical base of  $\mathbb{R}^n$
- **Documents:** points in the  $n$  –dimensional space

$$\vec{V}(d) = (\text{tfidf}_{t_1,d}, \text{tfidf}_{t_2,d}, \dots, \text{tfidf}_{t_n,d})$$

- **Queries:** points in the  $n$  –dimensional space

$$\vec{V}(q) = (0, 1, 0, \dots, 0)$$

$\text{tfidf}_{t,d} = \text{tf}_{t,d} * \text{idf}_t$ , with:

- $\text{tf}_{t,d} \rightarrow$  higher importance to terms with high frequency
- $\text{idf}_t = \log\left(\frac{N}{\text{df}_t}\right) \rightarrow$  lower importance to terms observed in many documents

# The vector space model

Given a query  $q$ , each document is ranked according to a similarity measure.

The measure used in this project is the cosine similarity:

$$\text{Score}(d_j, q) = \text{sim}(\vec{V}(d_j), \vec{V}(q))$$

Where:

$$\text{sim}(\vec{V}(d_j), \vec{V}(q)) = \frac{\vec{V}(d_j) \cdot \vec{V}(q)}{|\vec{V}(d_j)| |\vec{V}(q)|}$$

# Relevance feedback

The user can provide information about relevant documents for a specific query.

In this project, relevance feedback is implemented according to the Rocchio algorithm.

Given the initial query  $\vec{q}_0$ , the optimized query vector is computed as follows:

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d} \in D_r} \vec{d} - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d} \in D_{nr}} \vec{d}$$

Where:

- $D_r$  is the set of known relevant documents,
- $D_{nr}$  is the set of known non-relevant documents,
- $\alpha = 1, \beta = 0.75, \gamma = 0.15$ .

# Pseudo-relevance feedback

The pseudo-relevance feedback allows an optimization of the original query, this time without the information given from the user.

In this project the pseudo-relevance feedback is implemented according to a pseudo-Rocchio algorithm:

1. The set of relevant documents is computed as the top K documents, given the query  $q_0$ ,
2. The Rocchio algorithm is applied, to compute the optimized query.

# Evaluation

The system is evaluated on a set of queries:

- Precision:  $\frac{\text{relevant} \cap \text{retrieved}}{\text{retrieved}}$       Recall:  $\frac{\text{relevant} \cap \text{retrieved}}{\text{relevant}}$   
computed for each query varying the number of retrieved documents  $K$
- Mean Average Precision:  $\text{MAP}(Q) = \frac{1}{N} \sum_{j=1}^N \left( \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk}) \right)$ 
  - $N$  = number of queries,
  - $m_j$  = number of relevant documents for query  $j$ ,
  - $k$  = number of retrieved documents.

# The dataset

The dataset used for evaluating the program is the Medline collection – a collection of articles from a medical journal

- 1032 documents;
- 13391 terms.

There is an associated set of 30 queries with known relevant documents for each query.



# The dataset

Example of a document:

.I 1

.W

correlation between maternal and fetal plasma levels of glucose and free fatty acids .

correlation coefficients have been determined between the levels of glucose and ffa in maternal and fetal plasma collected at delivery . significant correlations were obtained between the maternal and fetal glucose levels and the maternal and fetal ffa levels . from the size of the correlation coefficients and the slopes of regression lines it appears that the fetal plasma glucose level at delivery is very strongly dependent upon the maternal level whereas the fetal ffa level at delivery is only slightly dependent upon the maternal level .

Example of a query:

the crystalline lens in vertebrates, including humans.

# Corpus pre-processing

Pre-processing on each document and query:

- **Normalization:** punctuation removal and to lowercase
- **Tokenization:** the document has been converted to a list of terms, split by space
- **Stop words removal:** using the default English stop list from the NLTK package
- **Stemming:** through PorterStemmer() function from the NLTK package

After pre-processing of each document → from 13391 to 9685 terms in the dictionary.

# Documents and query vectors

- The corpus is an array of document vectors

```
corpus.shape
```

```
> (1032, 9685) # (D, N)
```

```
corpus
```

```
> [[doc0], [doc1], [doc2], ...]
```

- Each documents is an N-dimensional array

```
corpus[0] # array for document 0
```

```
> [10.420554129725344, 5.272411187248268, 23.968889801250405,  
   0.33239202330297984, 23.661130034925105, ..., 3.9948149668750674,  
   1.075622770443411, 2.931920760809037, 0.6551197849707061, ...,  
   0.0, 0.0, 0.0, 0.0, 0.0, ...]
```

# Documents and query vectors

- Each query is a N-dimensional array

`example_query`

`> [1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, ...]`

- Scores for each document represented as a dictionary `{docID: score, ...}`

`scores`

`> {0: 0, 1: 0, 2: 0, 3: 0.08250173329228949, 4: 0, 5: 0.10556477516258402, ...}`

# Top-K retrieval

Given a query and a value of K, the system returns the list of K documents with the highest cosine similarity  $\text{sim}(q, d_k)$ .

DocID:	71	Score:	0.32326427430871757
DocID:	499	Score:	0.2662094450787526
DocID:	180	Score:	0.2179920012480751
DocID:	170	Score:	0.19139097770770058
DocID:	14	Score:	0.19027283535863754
DocID:	964	Score:	0.18674186882310087
DocID:	165	Score:	0.18475982417434741
DocID:	512	Score:	0.17601476449236989
DocID:	137	Score:	0.16698722946249522
DocID:	359	Score:	0.1503014631326245

# Relevance feedback

- The user can provide a list of known relevant docIDs
  - > [12, 13, 14, 71, 78, 137, 141, 163, 164, 165, ...]
- The system calls the function for the Rocchio algorithm to return the optimized query
  - > [0.026722966488916783, 0.0, 0.0, 0.0, 0.16380065961828127, ...]
- Starting from the  $q_{opt}$  vector we can then retrieve the K documents with highest cosine similarity  $sim(q_{opt}, d_k)$ .

DocID:	498	Score:	0.4975825553600744
DocID:	180	Score:	0.48248989126391234
DocID:	510	Score:	0.472073021470223
DocID:	12	Score:	0.4560654940412232
DocID:	179	Score:	0.45186607965425535
DocID:	508	Score:	0.44791318658586426

...

# Pseudo-relevance feedback

- We can allow pseudo-relevance feedback to obtain a query vector optimised with respect to the top K documents retrieved
  - > [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.25820597884312707, ...]
- On this optimized vector we can then retrieve the K documents with highest cosine similarity  $\text{sim}(q_{opt}, d_k)$ .

DocID:	14	Score:	0.5571136148407742
DocID:	165	Score:	0.5392188837587554
DocID:	512	Score:	0.5216182443784564
DocID:	499	Score:	0.48581512098078977
DocID:	180	Score:	0.46169998787542205
DocID:	170	Score:	0.4382538945134972
...			

# Evaluation of the system

- Set of 30 queries

`example_query`

`> the crystalline lens in vertebrates, including humans.`

- Set of known relevant documents for each query:
  - Minimum of 9 relevant documents per query
  - Maximum of 39 relevant documents per query
  - Average of 23 known relevant documents per query



# Evaluation of the system

- «Standard» scenario:

K=5	Average Precision: 0.68	Average Recall: 0.16
K=6	Average Precision: 0.66	Average Recall: 0.19
K=7	Average Precision: 0.64	Average Recall: 0.22
K=8	Average Precision: 0.64	Average Recall: 0.25
K=9	Average Precision: 0.65	Average Recall: 0.28
K=10	Average Precision: 0.64	Average Recall: 0.31
K=11	Average Precision: 0.62	Average Recall: 0.32
K=12	Average Precision: 0.61	Average Recall: 0.34
K=13	Average Precision: 0.59	Average Recall: 0.36
K=14	Average Precision: 0.58	Average Recall: 0.38

> MAP = 0.6296379359580575

# Evaluation of the system

- Allowing relevance feedback:

K=5	Average Precision: 0.96	Average Recall: 0.24
K=6	Average Precision: 0.95	Average Recall: 0.28
K=7	Average Precision: 0.95	Average Recall: 0.33
K=8	Average Precision: 0.95	Average Recall: 0.37
K=9	Average Precision: 0.94	Average Recall: 0.41
K=10	Average Precision: 0.93	Average Recall: 0.45
K=11	Average Precision: 0.92	Average Recall: 0.49
K=12	Average Precision: 0.91	Average Recall: 0.53
K=13	Average Precision: 0.90	Average Recall: 0.56
K=14	Average Precision: 0.89	Average Recall: 0.59

> MAP = 0.9321421215043031

# Evaluation of the system

- Allowing pseudo-relevance feedback:

K=5	Average Precision: 0.78	Average Recall: 0.19
K=6	Average Precision: 0.76	Average Recall: 0.22
K=7	Average Precision: 0.74	Average Recall: 0.25
K=8	Average Precision: 0.71	Average Recall: 0.28
K=9	Average Precision: 0.70	Average Recall: 0.30
K=10	Average Precision: 0.69	Average Recall: 0.33
K=11	Average Precision: 0.68	Average Recall: 0.36
K=12	Average Precision: 0.68	Average Recall: 0.39
K=13	Average Precision: 0.66	Average Recall: 0.41
K=14	Average Precision: 0.66	Average Recall: 0.43

> MAP = 0.7072055147428038

# Comparison

- «Standard» scenario
  - > MAP = 0.6296379359580575
- Relevance feedback
  - > MAP = 0.9321421215043031
- Pseudo-relevance feedback
  - > MAP = 0.7072055147428038