

# Information Retrieval Project

**Giovanni Santacatterina**

January 2021

# Outline

---

Main features of the project:

1. Vector space model
2. Relevance feedback and pseudo relevance feedback
3. Free form text queries
4. Python implementation:
  - 4.1. Evaluation of a test dataset
  - 4.2. Tool for retrieval of books

# The dataset

---

The dataset for the evaluation contains 1460 documents collected by the Centre for Inventions and Scientific Information along with 112 related queries.

Every documents contains a unique Id, an author, a title and an abstract. The abstract is the content used for the retrieval.

# The dataset

---

An example of document:

T. Comparisons of Four Types of Lexical Indicators of Content

W. An experiment was conducted to determine which of four types of lexical indicators of content could be utilized best by subjects to determine relevant from irrelevant documents and to answer a set of 100 questions. The results indicate that there were no major differences between the groups using complete text and abstracts to select relevant documents, but the group utilizing the complete text obtained a significantly higher score on the examination.

# The dataset

— — —

The most common words, with stop words included, are:

the, of, and, in , to, is, for, are, **information**, that,  
this, as, on, **library**, by, be, with, an, which, it

The most common are, as expected, stop words. Removing them we have

# Text processing

---

Steps for preprocessing (more later):

1. **Normalization** : removal of punctuation and lowercase
2. **Tokenisation** : splitting based on empty spaces
3. **Stop words removal** : removal of the list of english stopwords (NLTK library)
4. **Stemming**: done using Porter Stemmer (NLTK library)

# Methods implemented

---

I implemented the following methods:

1. Standard retrieval
2. Pseudo relevance feedback with query expansion
3. Pseudo relevance feedback with query drift
4. Relevance feedback with query expansion
5. Relevance feedback with query drift

# Methods

— — —

## 1. **Standard**

It uses the cosine similarity between query and documents. It uses TF-IDF weighting scheme.

## 2. **Query expansion**

After the first proposal, the query is expanded with the top terms found in the proposed relevant documents.

## 3. **Query drift**

After the first proposal, the vectorial representation of the query is moved considering the centroid of relevant and non relevant documents.



# Implementation

---

After the preprocessing of the dataset I obtain a vocabulary of the terms with their inverse document frequency value (IDF), computed as:

$$IDF_t = \log_{10} \left( \frac{|D|}{|d : t \in D|} \right)$$

I also obtain an inverted index for every term, in which I store the DocId containing that term and the corresponding term frequency:

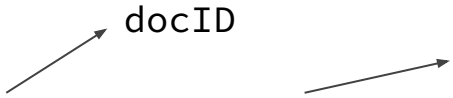
$$tf_{ij} = \frac{n_{ij}}{|d_j|}$$

# Implementation

— — —

Posting list element:

‘women’ : { 4: 1, 187: 1, 944: 1, 1215: 1, 1452: 4 }




docID

term  
frequency

Vocabulary element:

‘women’ : { 2.465... }



inverse  
document  
frequency

# Vectorization vs. Posting List Traversal

---

All methods have been implemented using two paradigms:

1. Vectorization
2. Posting list traversal

# Posting list traversal

---

The idea of posting list traversal is that is not necessary to actually build a vector representation of documents and queries, but relies on the traversal of the posting lists of the terms present in the query.

For some methods, e.g. query drift, I used a weighted query, in which not every entry is either 1 or 0. The entries of the query are used to weight the TF-IDF scores.

# Posting List Traversal

---

Query = “cat on the table” -> “cat” + “table”

I take the IDF of “cat” and “table” and I look at their posting list

“cat: 0.5” : {1 : 2, 7: 1} , “table: 0.9” : { 5: 1, 7:1}

Scores:

$$\text{doc1} = 2 * 0.5 = 1$$

$$\text{doc5} = 1 * 0.9 = 0.9$$

$$\text{doc7} = 1 * 0.5 + 1 * 0.9 = 1.4$$

# Evaluation

---

I evaluated the performance of the IR system in three different setting:

1. Stopwords\_removal = False, Stemming = False
2. Stopwords\_removal = True, Stemming = False
3. Stopwords\_removal = True, Stemming = True

The third setting has proved to be the best one.

# Evaluation

---

I used different metrics:

1. Mean Average Precision (MAP)
2. Mean R-Precision (MRP)
3. Mean Precision@K with  $K = 10$

# Evaluation Standard Methods

— — —

Remove stopwords = True, lemmatize = True

Vocabulary contains 6773 tokens

	MAP	MRP	M@10
Standard	0.26	0.21	0.20
Pseudo Expansion	0.23	0.18	0.18
Pseudo Drift	0.26	0.22	0.21
Feedback Expansion	0.42	0.24	0.24
Feedback Drift	0.48	0.29	0.29



# Evaluation Posting List Traversal Methods

— — —

Remove stopwords = True, lemmatize = True

Vocabulary contains 6773 tokens

	MAP	MRP	M@10
Standard	0.28	0.21	0.20
Pseudo Expansion	0.24	0.18	0.17
Pseudo Drift	0.24	0.22	0.21
Feedback Expansion	0.40	0.24	0.23
Feedback Drift	0.42	0.25	0.25

# VSM Tool

---

I used the CMU Book Summary dataset which contains the summary of 16,559 books extracted from Wikipedia, along with metadata such as Author, Title, DocID, and Genre.

After choosing which method to use, the user can insert a query into the system.

The tool, given a free form text query, returns the  $k$  most relevant documents, where  $k$  is decided by the user.

# VSM Tool

---

A document of the dataset:

The Plague, Albert Camus, 1947

The text of The Plague is divided into five parts. In the town of Oran, thousands of rats, initially unnoticed by the populace, begin to die in the streets. A hysteria develops soon afterward, causing the local newspapers to report the incident. Authorities responding to public pressure order the collection and cremation of the rats, unaware that the collection itself was the catalyst for the spread of the bubonic plague. The main character, Dr. Bernard Rieux, lives comfortably in an apartment building when strangely the building's concierge, M. Michel, a confidante, dies from a fever. Dr. Rieux consults...

The vocabulary contains 87081 terms.

Longest posting list is: `len(POSTING_LIST['one']) = 8252`

# VSM Tool

---

```
jovoni@jovoni: ~/Desktop/UNITs/Information_Retrieval/IR_project
(base) jovoni@jovoni:~/Desktop/UNITs/Information_Retrieval/IR_project$ python VSM_search.py

VSM Search

Available methods:
1: standard
2: pseudo query expansion
3: pseudo query drift
4: feedback query expansion
5: feedback query drift
Select method: 1
How many results you want? 2

-----

Insert query (exit to close): Count

RESULTS:
  The Count of Monte Cristo
  The Castle of Wolfenbach

-----

Insert query (exit to close): exit
(base) jovoni@jovoni:~/Desktop/UNITs/Information_Retrieval/IR_project$
```