# Probabilistic IR: BIM model Project #3

Valeria Insogna University of Trieste

July 3, 2023

# Outline

- BIM model
- 2 Dataset
- 3 Implementation
- 4 Preprocessing
- Conclusion

What is a probabilistic IR system?

#### Goal

Estimate the probability P of a document d being relevant (R = 1) with respect to a query q based on its content.

$$P(R=1|d,q) \tag{1}$$

## Retrieval

Rank documents by decreasing estimated probabilities of being relevant

#### Assumptions

Binary: documents, queries as binary vectors

$$\vec{x} = [x_1, x_2, \cdots, x_m]$$

where  $x_i = 1$  or 0 if present or not.

- Independence:
  - terms occur in document independently → no associations
  - relevance of each document is independent from others
- terms not in the query appear equally in relevant and non relevant documents

Rank the document  $\vec{x}$  by their odds O of relevance R wrt  $\vec{q}$ :

$$O(R|\vec{x}, \vec{q}) = \frac{P(R=1|\vec{x}, \vec{q})}{P(R=0|\vec{x}, \vec{q})}$$
(2)

By Bayes' rule and assumptions:

$$\propto \prod_{i:x_i=1;q_i=1} \frac{p_i}{u_i} \frac{1-u_i}{1-p_i}$$
 (3)

where

$$p_i = P(x_i = 1|, R = 1, \vec{q})$$
  

$$u_i = P(x_i = 1|, R = 0, \vec{q})$$
(4)

# BIM model RSV

For each document *d* use Retrieved Status Value for ranking:

$$RSV_d = \sum_{i:x_i=1;q_i=1} c_i \tag{5}$$

where  $c_i$  is the weight of the i-th term of the dictionary:

$$ci = \log\left(\frac{p_i}{u_i} \frac{1 - u_i}{1 - p_i}\right) \tag{6}$$

#### Estimation

Pre-compute  $c_i$  by setting  $p_i$  and  $u_i$  initial values.

#### Assumptions:

non relevant documents are majority in collection

$$\log \frac{1 - u_i}{u_i} = \log \frac{N - df_i}{df_i} \approx \log \frac{N}{df_i} = idf_i$$
 (7)

• each term has even probabilities of appearing or not in relevant documents  $\implies pi = \frac{1}{2}$  for each term

Therefore,

$$c_i = \log \frac{1 - u_i}{u_i} \tag{8}$$

$$RSV_d = \sum_{i:x_i=1;q_i=1} \log \frac{1-u_i}{u_i} \approx \sum_{i:x_i=1;q_i=1} idf_i$$
 (9)

#### Relevance feedback

Use relevance feedback to estimate the probabilities in  $RSV_d$ .

- start with  $u_i = df_i$  and  $p_i = \frac{1}{2}$
- retrieve a set of V docs
- user gives feedback on the set VR of relevant docs
- update probabilities:

$$p_i = \frac{|VR_i| + \frac{1}{2}}{|VR| + 1} \qquad u_i = \frac{df - |VR_i| + \frac{1}{2}}{N - |VR_i| + 1}$$
(10)

where  $|VR_i|$  is the number of relevant docs according to the user that contain  $x_i$ .

#### Pseduo-Relevance feedback

- start with  $u_i = df_i$  and  $p_i = \frac{1}{2}$
- retrieve first k highest ranked docs as the set V
- consider all of them as relevant docs
- ullet update probabilities as in eq.10, with V instead of VR
- repeat until ranking converges

# Dataset

#### Intro

Cranfield collection consisting of 1400 documents (1.6 MB) from the aerodynamics field with 225 queries with relevance feedback.

# Original format:

- .1 : doc id
- .T : doc title
- .A : doc author(s)
- .B: bibliography
- .W: doc content (♠ contains also the title!)

 $\triangle$  queries file has only .W and .I tags.

Only .W tags have been considered for text content and .I for identification.

#### Structure

- cran.all: documents (1400)
- cran.qry: queries (225)
- cranqrel: relevance assesments. In 3 columns: query number, relevant document number, relevancy code.
- cranqrel\_bin: binary relevance assessments in TREC format. In 4 columns: query number, iteration (always zero), relevant document number, relevancy code (0 for non relevant, 1 for relevant)

Relevance scores (3<sup>rd</sup> col) have been imported for each query (1<sup>st</sup> col) from cranqrel\_bin file only with R = 1.

#### Folder structure

## Available github repository

- data/cran/: dataset in original format
- data/preprocessed/: import, normalize, tokenize
   import.py output articles.pkl, queries.pkl, relevance.pkl
- utils/: functions.py
- bim.py: implementation bim model  $\xrightarrow{output}$  index.pkl
- index.pkl: index built by bim model
- run\_all.sh: bash script to run import.py and bim.py
- test\_model.ipynb: jupyter notebook to test the model

import.py

For articles and queries, use functions:

- import\_data() splits the articles/queries into entries at the marker ./ returning a list of strings.
- get\_text\_only(): takes only text enclosed by marker .W and returns a list of lists of articles/queries where each sub-list contains all the terms presents in each article. It performs the tokenization and normalization too.

For relevant documents use function import\_relevance(): returns a dictionary (keys are queries' IDs and values are docIDs of the relevant documents to that query.

The results have been saved in .pkl files.

functions.py

Useful functions for files bim.py and import.py.

- make\_tokens: see page 17
- stemming(): see page 18
- make\_inverted\_index(): given tokenized articles, it creates the inverted index after stemming and stopwords removal (see section 18)
- ullet doc\_frequency and inverse\_doc\_frequency: used to evaluate for the first time  $RSV_d$  by eq.9

#### BIM class 2

In file bim.py class BIM\_IRModel is implemented and runned on articles.pkl 

output index.pkl

#### **Members**

- articles: list of list of the tokens contained in each article
- index: dict with inverted index of the terms (make\_inverted\_index)
- df: dict of doc frequencies of each term (inverse\_doc\_frequency)
- N: length of the collection
- query: list with stemmed non-stopwords of query
- u: dict with u probabilities for each term
- p: dict with p probabilities for each term
- term\_weights: dict with c<sub>i</sub> weights of each term
- rank: list of ordered rank of retrieved docs with RSV

# Implementation BIM class 1

#### Methods

- compute\_term\_weights: c<sub>i</sub> for each term eq.8
- update\_p\_u: update p/u during (pseudo) relevant feedback as eq.10
- rsv\_doc\_query: given a query and a doc, compute RSV as eq.5.
- first\_rsv\_doc\_query: compute RSV eq.9
- answer\_query: computes RSV of all docs and rank by decreasing RSV
- query: list with stemmed non-stopwords
- relevance\_feedback: given query and list docIDs relevant uses 10
- pseudo\_relevance\_feedback: with first k documents of answer query as relevant in 10

# Preprocessing

#### Tokenization and Normalization

**Directly when importing the dataset** or answering free-form text queries via make\_tokens() in utils/functions.py.

Normalization:

- replace '-' char with whitespace
  - → avoid double words like 'high-speed'
- ignore capitalization
  - → casefold()
- remove accents and diacritics
  - → unidecode lib
- remove punctuation
  - → string.punctuation
- remove non-alphabetical characters
  - → regex

Tokenization  $\Rightarrow$  Natural Language Toolkit nltk via tokenize().

# Preprocessing

#### Stop words and stemming

## Both implemented by nltk library:

- Stopwords
  - → stopwords.words('english') +
    custom ['isn', 'won', 're', 'll']
- Stemming
  - → PorterStemmer()

# They are performed when:

- creating index for docsvia make\_inverted\_index() in utils/functions.py
- answering queriesvia rsv\_doc\_query() in bim.py

#### Effectiveness metrics 1

Effectiveness of the system is performed on a set of **test queries** by:

$$Precision = \frac{relevant \cap retrieved}{retrieved} \qquad Recall = \frac{relevant \cap retrieved}{relevant} \qquad (11)$$

$$MAP = \frac{1}{Q} \sum_{j=1}^{Q} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk})$$
 (12)

where Q is num of queries in set,  $m_j$  num of relevant docs for query j, k num of retrieved docs,  $R_{jk}$  retrieved docs for j-th query when asking k relevant docs

where R is total num of relevant docs for a query

Mean R-Precision = 
$$\frac{1}{Q} \sum_{i=1}^{Q} \text{R-Precision}(q_i)$$
 (14)

#### Effectiveness metrics 2

In file bim.py class BIM\_IRModel also following methods are implemented.

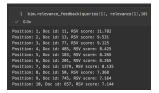
#### Metrics as methods

- precision\_recall: eq.11 given a query, set of relevant documents and number of documents to retrieve
- mean\_average\_precision: eq.12 for a set of queries and the related relevant documents
- r\_precision: eq.13, R is the number of relevant documents for that query
- mean\_r\_precision: eq.14 for a set of queries

#### Relevance Feedback



(a) Retrieval of top 10 docs for 2nd query, no feedback



(c) Retrieval of top 10 docs for 2nd query, after user feedback



(b) Precision and Recall of top 10 retrieved docs for 2nd

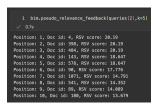


(d) Precision and Recall improved

#### Pseudo Relevance Feedback



(a) Precision and Recall of top 10 retrieved docs for 3rd query, no feedback



(b) Retrieval of top 10 docs for 2nd query, after pseudo feedback using first 5 docs retrieved as relevant



(c) Precision and Recall improved

#### Free form text

```
1 binsprist_first_w_rel_dess("deminating factors in structural design of high-speed aircraft" / Six
("demin, 'aircraft', 'high' 'speed', 'factor', 'structur', 'design')

Bestins 1, Oc. 1d: 28, Six down: 2.58

Resition: 1, Oc. 1d: 28, Six down: 2.58

Position: 4, Doc 1d: 28, KN Score: 2.48

Position: 4, Doc 1d: 28, KN Score: 2.48

Position: 6, Doc 1d: 28, KN Score: 2.48

Position: 7, Oc. 1d: 28, KN Score: 2.48

Position: 8, Doc 1d: 38, KN Score: 1.77

Position: 9, Doc 1d: 38, KN Score: 1.77

Position: 9, Doc 1d: 38, KN Score: 1.77

Position: 9, Doc 1d: 58, KN Score: 8.764

Position: 9, Doc 1d: 58, KN Score: 8.764
```

(a) Retrieval of top 10 docs for free query, no feedback

```
1 print_tet(articles[11])

✓ Obs

some structural and aerelastic considerations of high speed flight the domination of factors in structural design of high speed aircraft are thermal and seroelastic in origin the subject matter is concerned largely with a discussion of these of the subject matter is concerned largely with a discussion of these of the articles of the subject of the subject and the concerned largely with a discussion of these of the articles of the subject of the subject to the sub
```

(b) Doc 11 before stemming and stopwords removal

(c) Retrieval of top 10 docs for free query using pseudo feedback with k=5

## Conclusion

#### Future improvements

- metadata (e.g..A, .B) incorporated into docs
- add eleven-point interpolated average precision metric
- use Edit distance or Soundex algorithm for spelling corrections
- implement OKAPI BM25

# Thank you for your attention!