Probabilistic IR: BIM model

Project #3

Valeria Insogna
University of Trieste

July 3, 2023

Outline

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

BIM model

Assumptions

1. **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.

2. **Independence**:
   - terms occur in document independently → no associations
   - relevance of each document is independent from others
3. 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 $\vec{q}$:

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

By Bayes' rule and assumptions:

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

where

$$\begin{aligned} p_i &= P(x_i = 1|, R = 1, \vec{q}) \\ u_i &= P(x_i = 1|, R = 0, \vec{q}) \end{aligned} \tag{4}$$

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:

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

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

- non relevant documents are majority in collection

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

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

Therefore,

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

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

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:

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

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

- 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
- update probabilities as in eq.10, with V instead of VR
- repeat until ranking converges

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

Original format:

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

⚠ queries file has only *.W* and *.I* tags.

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

Dataset
Import

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 assesments 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^{rd} **col) have been imported** for each query (1^{st} col) from `cranqrel_bin` file **only with** $R = 1$.

Implementation

Folder structure

Available github repository

- `data/cran/`: dataset in original format
- `data/preprocessed/`: import, normalize, tokenize
  `import.py` $\xrightarrow{\textit{output}}$ `articles.pkl`, `queries.pkl`, `relevance.pkl`
- `utils/`: `functions.py`
- `bim.py`: implementation bim model $\xrightarrow{\textit{output}}$ `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

For articles and queries, use functions:

1. `import_data()` splits the articles/queries into entries at the marker *.I* returning a list of strings.
2. `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.

Implementation
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)
- `doc_frequency` and `inverse_doc_frequency`: used to evaluate for the first time RSV_d by eq.9

In file `bim.py` class `BIM_IRModel` is implemented and runned on

`articles.pkl` $\xrightarrow{output}$ `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_i weights of each term
- `rank`: list of ordered rank of retrieved docs with *RSV*

Methods

- `compute_term_weights`: c_i 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

Directly when importing the dataset or answering free-form text queries via `make_tokens()` in `utils/functions.py`.

Normalization:

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

Tokenization ⟹ Natural Language Toolkit `nltk` via `tokenize()`.

Both implemented by `nltk` library:

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

They are performed when:

- creating index for docs
  via `make_inverted_index()` in `utils/functions.py`
- answering queries
  via `rsv_doc_query()` in `bim.py`

Results

Effectiveness metrics 1

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

$$\textbf{Precision} = \frac{\text{relevant} \cap \text{retrieved}}{\text{retrieved}} \qquad \textbf{Recall} = \frac{\text{relevant} \cap \text{retrieved}}{\text{relevant}} \tag{11}$$

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

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

$$\textbf{R-Precision} = \text{Precision of top R ranked doc returned by query} \tag{13}$$

where R is total num of relevant docs for a query

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

Results

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

Results

Relevance Feedback

```
1  bim.print_first_k_rel_docs(queries[1])
✓ 0.1s

Position: 1, Doc id: 11, RSV score: 17.064
Position: 2, Doc id: 13, RSV score: 14.493
Position: 3, Doc id: 171, RSV score: 13.164
Position: 4, Doc id: 1379, RSV score: 12.711
Position: 5, Doc id: 485, RSV score: 11.744
Position: 6, Doc id: 1088, RSV score: 11.596
Position: 7, Doc id: 745, RSV score: 11.511
Position: 8, Doc id: 77, RSV score: 11.474
Position: 9, Doc id: 50, RSV score: 10.926
Position: 10, Doc id: 363, RSV score: 10.825
```

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

```
1  bim.precision_recall(queries[1], relevance[1])
✓ 0.1s

The precision is: 0.4, the recall is: 0.167
```

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

```
1  bim.relevance_feedback(queries[1], relevance[1],10)
✓ 0.3s

Position: 1, Doc id: 11, RSV score: 11.782
Position: 2, Doc id: 13, RSV score: 9.531
Position: 3, Doc id: 77, RSV score: 9.125
Position: 4, Doc id: 485, RSV score: 8.425
Position: 5, Doc id: 183, RSV score: 8.265
Position: 6, Doc id: 201, RSV score: 8.265
Position: 7, Doc id: 1379, RSV score: 8.135
Position: 8, Doc id: 50, RSV score: 7.368
Position: 9, Doc id: 745, RSV score: 7.164
Position: 10, Doc id: 657, RSV score: 7.144
```

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

```
1  bim.precision_recall(queries[1], relevance[1])
✓ 0.1s

The precision is: 0.7, the recall is: 0.292
```

(d) Precision and Recall improved

Results

Pseudo Relevance Feedback

```
1  bim.precision_recall(queries[2], relevance[2])
✓  0.1s
The precision is: 0.3, the recall is: 0.375
```

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

```
1  bim.pseudo_relevance_feedback(queries[2],k=5)
✓  0.7s
Position: 1, Doc id: 4, RSV score: 20.19
Position: 2, Doc id: 398, RSV score: 20.19
Position: 3, Doc id: 484, RSV score: 20.19
Position: 4, Doc id: 143, RSV score: 18.647
Position: 5, Doc id: 578, RSV score: 18.647
Position: 6, Doc id: 90, RSV score: 17.776
Position: 7, Doc id: 1071, RSV score: 14.791
Position: 8, Doc id: 541, RSV score: 14.352
Position: 9, Doc id: 89, RSV score: 14.089
Position: 10, Doc id: 180, RSV score: 13.679
```

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

```
1  bim.precision_recall(queries[2], relevance[2])
✓  0.1s
The precision is: 0.6, the recall is: 0.75
```

(c) Precision and Recall improved

```
1  bim.print_first_k_rel_docs("dominating factors in structural design of high-speed aircraft")
✓ 0.1s

{'domin', 'aircraft', 'high', 'speed', 'factor', 'structur', 'design'}
Position: 1, Doc id: 1216, RSV score: 3.067
Position: 2, Doc id: 218, RSV score: 2.585
Position: 3, Doc id: 517, RSV score: 2.585
Position: 4, Doc id: 864, RSV score: 2.178
Position: 5, Doc id: 810, RSV score: 2.036
Position: 6, Doc id: 400, RSV score: 1.782
Position: 7, Doc id: 571, RSV score: 1.782
Position: 8, Doc id: 11, RSV score: 1.476
Position: 9, Doc id: 882, RSV score: 0.764
Position: 10, Doc id: 628, RSV score: 0.712
```

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

```
1  print_txt(articles[11])
✓ 0.0s

some structural and aerelastic considerations of high speed flight the dominatin
g factors in structural design of high speed aircraft are thermal and aeroelasti
c in origin the subject matter is concerned largely with a discussion of these f
actors and their interrelation with one another a summary is presented of some o
f the analytical and experimental tools available to aeronautical engineers to m
eet the demands of high speed flight upon aircraft structures the state of the a
rt with respect to heat transfer from the boundary layer into the structure mode
s of failure under combined load as well as thermal inputs and acrothermoelastic
ity is discussed methods of attacking and alleviating structural and aeroelastic
 problems of high speed flight are summarized finally some avenues of fundamenta
l research are suggested
```

(b) Doc 11 before stemming and stopwords removal

```
1  bim.pseudo_relevance_feedback("dominating factors in structural design of high-speed aircraft",k=5)
✓ 1.5s

{'domin', 'aircraft', 'high', 'speed', 'factor', 'structur', 'design'}
Position: 1, Doc id: 11, RSV score: 28.546
Position: 2, Doc id: 1379, RSV score: 24.549
Position: 3, Doc id: 657, RSV score: 21.379
Position: 4, Doc id: 746, RSV score: 21.379
Position: 5, Doc id: 415, RSV score: 20.926
Position: 6, Doc id: 171, RSV score: 17.755
Position: 7, Doc id: 327, RSV score: 17.755
Position: 8, Doc id: 877, RSV score: 17.755
Position: 9, Doc id: 1050, RSV score: 17.755
Position: 10, Doc id: 1245, RSV score: 17.755
```

(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!