Information Retrieval (Part III)

[DAT640] Information Retrieval and Text Mining

Krisztian Balog
University of Stavanger

September 23, 2019

Outline

- ~~Search engine architecture, indexing~~
- ~~Evaluation~~
- **Retrieval models** $\Leftarrow$ today
- Query modeling
- Learning-to-rank, Neural IR
- Semantic search

So far

- Document retrieval task: scoring documents against a search query
- Inverted index: special data structure to facilitate large-scale retrieval
- Evaluation: measuring the goodness of a ranking against the ground truth using binary or graded relevance

Retrieval models

- Bag-of-words representation
  - Simplified representation of text as a bag (multiset) of words
  - Disregards word ordering, but keeps multiplicity
- Common form of a retrieval function

$$score(d, q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$$

  - Note: we only consider terms in the query, $t \in q$
  - $w_{t,d}$ is the term's weight in the document
  - $w_{t,q}$ is the term's weight in the query
- $score(d, q)$ is (in principle) to be computed for every document in the collection

Example retrieval functions

- General scoring function

$$score(d, q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$$

- **Example 1**: Count the number of matching query terms in the document

$$w_{t,d} = \begin{cases} 1, & f_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

  - where $f_{t,d}$ is the number of occurrences of term t in document d

$$w_{t,q} = f_{t,q}$$

  - where $f_{t,q}$ is the number of occurrences of term t in query q

Example retrieval functions

- General scoring function

$$score(d,q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$$

- **Example 2**: Instead of using raw term frequencies, assign a weight that reflects the term's importance

$$w_{t,d} = \begin{cases} 1 + \log f_{t,d}, & f_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

  - where $f_{t,d}$ is the number of occurrences of term t in document d

$$w_{t,q} = f_{t,q}$$

  - where $f_{t,q}$ is the number of occurrences of term t in query q

Vector Space Model

Vector space model

- Basis of most IR research in the 1960s and 70s
- Still used
- Provides a simple and intuitively appealing framework for implementing
    - Term weighting
    - Ranking
    - Relevance feedback

Vector space model

- Main underlying assumption: if document d_1 is more similar to the query than another document d_2, then d_1 is *more relevant* than d_2
- Documents and queries are viewed as vectors in a high dimensional space, where each dimension corresponds to a term

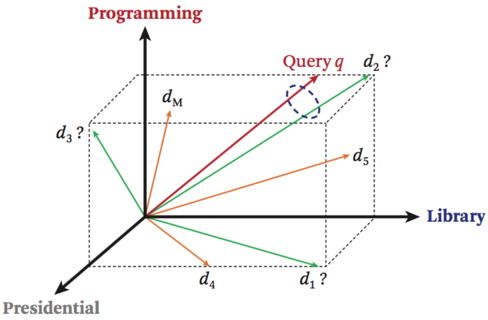


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.2]

Instantiation

- The vector space model provides a *framework* that needs to be instantiated by deciding
  - How to select terms? (i.e., vocabulary construction)
  - How to place documents and queries in the vector space (i.e., term weighting)
  - How to measure the similarity between two vectors (i.e., similarity measure)

Simple instantiation (bit vector representation)

- Each word in the vocabulary V defines a dimension
- Bit vector representation of queries and documents (i.e., only term presence/absence)
- Similarity measure is the dot product

$$sim(q,d) = \vec{q} \cdot \vec{d} = \sum_{t \in V} w_{t,q} \times w_{t,d}$$

  - where $w_{t,q}$ and $w_{t,d}$ are either 0 or 1

Discussion

Question

What are potential shortcomings of this simple instantiation?

Improved instantiation (TF-IDF weighting)

- Idea: incorporate term importance by considering term frequency (TF) and inverse document frequency (IDF)
    - TF rewards terms that occur frequently in the document
    - IDF rewards terms that do not occur in many documents
- A possible ranking function using the TF-IDF weighting scheme:

$$score(d,q) = \sum_{t \in q \cap d} tf_{t,q} \times tf_{t,d} \times idf_t$$

- Note: the above formula uses raw term frequencies and applies IDF only on one of the (document/query) vectors

Many different variants out there!

- Different variants of TF and IDF
- Different TF-IDF weighting for the query and for the document
- Different similarity measure (e.g., cosine)

Exercise #1

- Implement vector space retrieval
- Code skeleton on GitHub: `exercises/lecture_09/exercise_1.ipynb` (make a local copy)

BM25

- BM25 was created as the result of a series of experiments ("Best Match")
- Popular and effective ranking algorithm
- The reasoning behind BM25 is that good term weighting is based on three principles
  - Term frequency
  - Inverse document frequency
  - Document length normalization

BM25 scoring

$$score(d, q) = \sum_{t \in q} \frac{f_{t,d} \times (1 + k_1)}{f_{t,d} + k_1(1 - b + b\frac{|d|}{avgdl})} \times idf_t$$

- Parameters
  - k_1: calibrating term frequency scaling
  - b: document length normalization
- Note: several slight variations of BM25 exist!

Recall: TF transformation

- Many different ways to transform raw term frequency counts

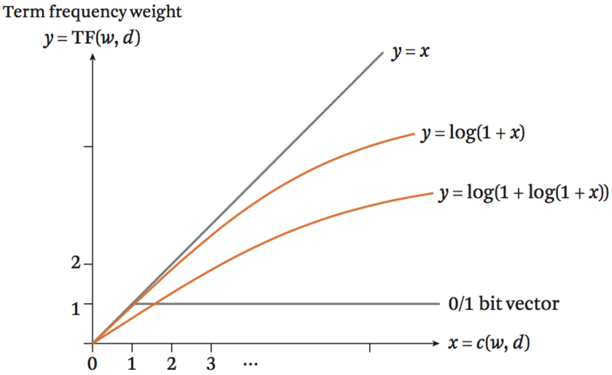


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.14]

BM25 TF transformation

- Idea: term saturation, i.e., repetition is less important after a while

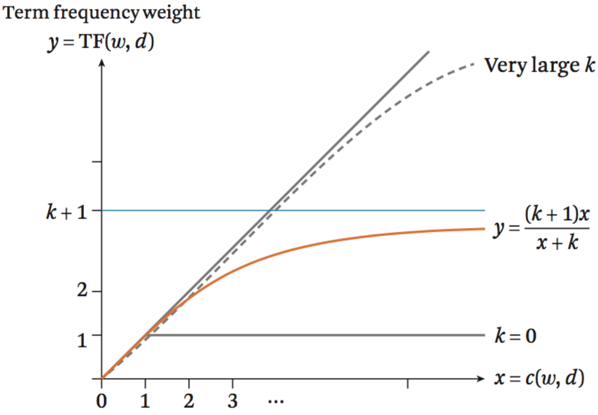


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.15]

BM25 document length normalization

- Idea: penalize long documents w.r.t. average document length (which serves as pivot)

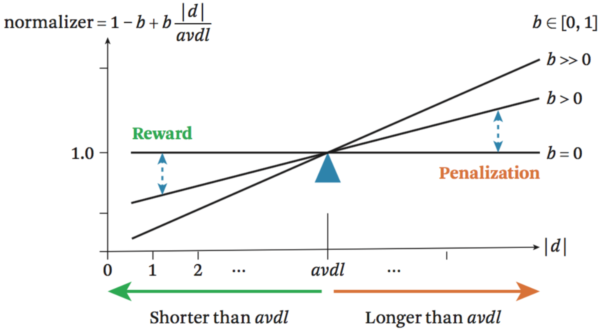


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 6.17]

BM25 parameter setting

- k_1: calibrating term frequency scaling
  - 0 corresponds to a binary model
  - large values correspond to using raw term frequencies
  - typical values are between 1.2 and 2.0; a common default value is 1.2
- b: document length normalization
  - 0: no normalization at all
  - 1: full length normalization
  - typical value: 0.75

Language Models

Language models

- Based on the notion of probabilities and processes for generating text
- Wide range of usage across different applications
  - Speech recognition
    - “I ate a cherry” is a more likely sentence than “Eye eight uh Jerry”
  - OCR and handwriting recognition
    - More probable sentences are more likely correct readings
  - Machine translation
    - More likely sentences are probably better translations

Language models for ranking documents

- Represent each document as a multinomial probability distribution over terms
- Estimate the probability that the query was "generated" by the given document
  - How likely is the search query given the language model of the document?

Query likelihood retrieval model

- Rank documents d according to their likelihood of being relevant given a query q:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \propto P(q|d)P(d)$$

- **Query likelihood**: Probability that query q was "produced" by document d

$$P(q|d) = \prod_{t \in q} P(t|\theta_d)^{f_{t,q}}$$

- **Document prior**, $P(d)$: Probability of the document being relevant to *any* query

Query likelihood

$$P(q|d) = \prod_{t \in q} P(t|\theta_d)^{f_{t,q}}$$

- θ_d is the document language model
  - Multinomial probability distribution over the vocabulary of terms
- $f_{t,q}$ is the raw frequency of term t in the query
- **Smoothing**: ensuring that $P(t|\theta_d)$ is > 0 for all terms

Jelinek-Mercer smoothing

- Linear interpolation between the empirical document model and a collection (background) language model

$$P(t|\theta_d) = (1-\lambda)P(t|d) + \lambda P(t|C)$$

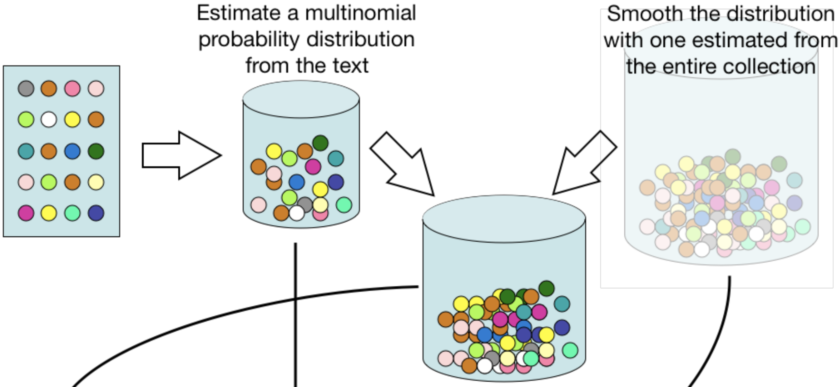
  - $\lambda \in [0,1]$ is the smoothing parameter
  - Empirical document model (maximum likelihood estimate):

$$P(t|d) = \frac{f_{t,d}}{|d|}$$

  - Collection (background) language model (maximum likelihood estimate):

$$P(t|C) = \frac{\sum_{d'} f_{t,d'}}{\sum_{d'} |d'|}$$

Jelinek-Mercer smoothing



$$\overbrace{P(t|\theta_d)} = (1-\lambda)\overbrace{P(t|d)} + \lambda\underbrace{P(t|C)}$$

Dirichlet smoothing

- Smoothing is inversely proportional to the document length

$$P(t|\theta_d) = \frac{f_{t,d} + \mu P(t|C)}{|d| + \mu}$$

  - μ is the smoothing parameter (typically ranges from 10 to 10000)
- Notice that Dirichlet smoothing may also be viewed as a linear interpolation in the style of Jelinek-Mercer smoothing, by setting

$$\lambda = \frac{\mu}{|d| + \mu} \qquad (1 - \lambda) = \frac{|d|}{|d| + \mu}$$

Query likelihood scoring (Example)

- query: “sea submarine”

$$
\begin{aligned}
P(q|d) &= P(\text{sea}|\theta_d) \times P(\text{submarine}|\theta_d) \\
&= \big((1-\lambda)P(\text{sea}|d) + \lambda P(\text{sea}|C)\big) \\
&\quad \times \big((1-\lambda)P(\text{submarine}|d) + \lambda P(\text{submarine}|C)\big)
\end{aligned}
$$

- where
  - $P(\text{sea}|d)$ is the relative frequency of term “sea” in document d
  - $P(\text{sea}|C)$ is the relative frequency of term “sea” in the entire collection
  - ...

Exercise #2

- Document retrieval using language models (paper-based)

Exercise #2 solution (see Excel spreadsheet on GitHub)

	term frequencies					empirical language models					collection language model	smoothed language models				
term	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5		D1	D2	D3	D4	D5
T1		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T2		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T3	3	2	2		1	0,6	0,4	0,5	0	0,25	0,364	0,576	0,396	0,486	0,036	0,261
T4			1	1		0	0	0,25	0,25	0	0,091	0,009	0,009	0,234	0,234	0,009
T5			1	1	1	0	0	0,25	0,25	0,25	0,136	0,014	0,014	0,239	0,239	0,239
T6	2	1		2		0,4	0,2	0	0,5	0	0,227	0,383	0,203	0,023	0,473	0,023
|D|	5	5	4	4	4	1	1	1	1	1	1	1	1	1	1	1
Jelinek-Mercer smoothing																
smoothing parameter						0,1										

$$P(t|\theta_d) = (1-\lambda)\overbrace{P(t|d)} + \lambda P(t|C)$$

Document language model computation

Exercise #2 solution (see Excel spreadsheet on GitHub)

	term frequencies					empirical language models					collection language model	smoothed language models				
term	**D1**	**D2**	**D3**	**D4**	**D5**	**D1**	**D2**	**D3**	**D4**	**D5**		**D1**	**D2**	**D3**	**D4**	**D5**
T1		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T2		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T3	3	2	2		1	0,6	0,4	0,5	0	0,25	0,364	0,576	0,396	0,486	0,036	0,261
T4			1	1		0	0	0,25	0,25	0	0,091	0,009	0,009	0,234	0,234	0,009
T5			1	1	1	0	0	0,25	0,25	0,25	0,136	0,014	0,014	0,239	0,239	0,239
T6	2	1		2		0,4	0,2	0	0,5	0	0,227	0,383	0,203	0,023	0,473	0,023
|D|	5	5	4	4	4	1	1	1	1	1	1	1	1	1	1	1
Jelinek-Mercer smoothing																
smoothing parameter						0,1										

$$P(t|\theta_d) = (1-\lambda)P(t|d) + \lambda \underbrace{P(t|C)}$$

Document language model computation

Exercise #2 solution (see Excel spreadsheet on GitHub)

	term frequencies					empirical language models					collection language model	smoothed language models				
term	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5		D1	D2	D3	D4	D5
T1		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T2		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T3	3	2	2		1	0,6	0,4	0,5	0	0,25	0,364	0,576	0,396	0,486	0,036	0,261
T4			1	1		0	0	0,25	0,25	0	0,091	0,009	0,009	0,234	0,234	0,009
T5			1	1	1	0	0	0,25	0,25	0,25	0,136	0,014	0,014	0,239	0,239	0,239
T6	2	1		2		0,4	0,2	0	0,5	0	0,227	0,383	0,203	0,023	0,473	0,023
|D|	5	5	4	4	4	1	1	1	1	1	1	1	1	1	1	1
Jelinek-Mercer smoothing																
smoothing parameter						0,1										

$$\overbrace{P(t|\theta_d)} = (1-\lambda)P(t|d) + \lambda P(t|C)$$

Document language model computation

Exercise #2 solution (see Excel spreadsheet on GitHub)

	term frequencies					empirical language models					collection language model	smoothed language models				
term	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5		D1	D2	D3	D4	D5
T1		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T2		1			1	0	0,2	0	0	0,25	0,091	0,009	0,189	0,009	0,009	0,234
T3	3	2	2		1	0,6	0,4	0,5	0	0,25	0,364	0,576	0,396	0,486	0,036	0,261
T4			1	1		0	0	0,25	0,25	0	0,091	0,009	0,009	0,234	0,234	0,009
T5			1	1	1	0	0	0,25	0,25	0,25	0,136	0,014	0,014	0,239	0,239	0,239
T6	2	1		2		0,4	0,2	0	0,5	0	0,227	0,383	0,203	0,023	0,473	0,023
|D|	5	5	4	4	4	1	1	1	1	1	1	1	1	1	1	1
Jelinek-Mercer smoothing																
smoothing parameter						0,1										
q="T3"												0,576	0,396	0,486	0,036	0,261
q="T2 T1"												0,000	0,036	0,000	0,000	0,055
q="T6"												0,383	0,203	0,023	0,473	0,023
q="T3 T1 T3 T2"												0,000	0,006	0,000	0,000	0,004

Scoring a query

$$P(q|d) = \prod_{t \in q} P(t|\theta_d)^{f_{t,q}}$$

P(q="T2 T1"|D2) = P(T2|D2) * P(T1|D2)

Practical considerations

- Since we are multiplying small probabilities, it is better to perform computations in the log space

$$
\begin{aligned}
P(q|d) &= \prod_{t \in q} P(t|\theta_d)^{f_{t,q}} \\
&\Downarrow \\
\log P(q|d) &= \sum_{t \in q} f_{t,q} \times \log P(t|\theta_d)
\end{aligned}
$$

- Notice that it is a particular instantiation of our general scoring function $score(d,q) = \sum_{t \in q} w_{t,d} \times w_{t,q}$ by setting
  - $w_{t,d} = \log P(t|\theta_d)$
  - $w_{t,q} = f_{t,q}$

Reading

- Text Data Management and Analysis (Zhai&Massung), Chapter 6