800 ANNI 1222 • 2022

UNIVERSITÀ DEGLI STUDI DI PADOVA

Traditional Retrieval Models

Search Engines

Master Degree in Computer Engineering
Master Degree in Data Science
Academic Year 2023/2024

Nicola Ferro

Intelligent Interactive Information Access (IIIA) Hub
Department of Information Engineering
University of Padua

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Outline

- Retrieval Models
- Boolean Model
- Vector Space Model
- Relevance Feedback
- Probabilistic Models
  - Binary Independence Model
  - BM25
  - Language Models

Retrieval Models

- *Problem*: predict which documents the user will find **relevant**
- *Goal*: produce a function that assigns scores to documents with regard to a given query
  - Definition of a logical framework to **represents** documents and queries
  - Definition of a **ranking function** which produces a rank for each document given a query

modellare topic di doc/query
⇓
determinano rilevanza

indice (offline)

qualità di doc (e.g. link collegati a pagina web) ⇐

201.17 ⇒ questo va in run file
Document Score

Retrieval Models

- Provide a mathematical framework for defining the search process
  - includes explanation of assumptions
  - basis of many ranking algorithms
  - can be implicit
- Progress in retrieval models has corresponded with improvements in effectiveness
- Theories about relevance
  - **topical** vs. **user** relevance

A Taxonomy of IR Models

- Set theory ⟶ Boolean model
- Vector space ⟶ Vector models (and Neural Models)
- Probability space ⟶ BM25 and Language models
- But there are other models:
  - Page rank
  - Learning to rank
  - Models for semi-structured text

Boolean Model

The Boolean Model

molto simile a ricerca in DB

- The Boolean model is the simplest retrieval model
- Based on set theory and Boolean algebra
- Very popular in the past, but still important for some domains as legal search, patent search or digital libraries

  ↳ facile capire perché ho preso documenti

  utile inverted index, semplice, efficiente
- Main assumption: index terms are present or absent in a document
- Query q is composed of index terms linked by NOT, AND, OR

  ⟶ Boolean expression

  ⇒ indicare termini adiacenti
  - Proximity operators also used
- The result is not a ranking but a set of documents

© Nicola Ferro

Exercise: The Boolean Model

- Consider the following documents:
  - `D1`: The quokka is a **marsupial** from Western **Australia**, it is **herbivorous** and mainly nocturnal
  - `D2`: The wombat is a **marsupial**, it is mainly crepuscular and nocturnal
  - `D3`: The Tree-kangaroo is a **marsupial** distributed not just in **Australia**, but also in New Guinea and other islands
  - `D4`: A wallaby is a **herbivorous marsupial** native to **Australia** and New Guinea
- What is the output of the query `("marsupial" OR "herbivorous") AND "Australia"`?
- What is the output of the query `("marsupial" OR "herbivorous") AND NOT "Australia"`?

Boolean Model: Pros and Cons

$$R(q,d) = \begin{cases} 1 & \text{if } d \text{ matches the Boolean expression of } q, \\ & \text{i.e. it evaluates to } \texttt{true} \\ 0 & \text{otherwise} \end{cases}$$

"ranking"

- Advantages
  - Results are predictable, relatively easy to explain
  - Many different features can be incorporated
  - Efficient processing since many documents can be eliminated from search
- Disadvantages
  - Effectiveness depends entirely on user
    - The ranking function is basically a binary match, no partial match: documents are relevant or not relevant
    - Does not give a ranking as output (retrieval of too few or too many documents)
  - Does not account for term frequency
  - Simple queries usually do not work well
  - Complex queries are difficult

Vector Space Model

Zipf's Law

- Distribution of word frequencies is very skewed
  - a few words occur very often, many words hardly ever occur
  - e.g., two most common words ("the", "of") make up about 10% of all word occurrences in text documents
- Zipf's "law" is an example of **power law**
  - observation that rank (r) of a word times its frequency (f) is approximately a constant (k)
    assuming words are ranked in order of decreasing frequency
  - P_r is the probability of word occurrence
  - $c \sim 0.1$ for English
  - $\alpha \sim 1$

$$P_r \sim \frac{c}{r^\alpha}$$

- **Principle of the least effort**
  - it is easier for a speaker or a writer of a language to repeat certain words instead of coining new and different words
  - most frequent words tend to be short function words which are easy to coin and whose cost of usage is small

George Kingsley Zipf

Zipf, G. K. (1936). *The Psychobiology of Language*. Routledge, London, UK.
Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge (MA), USA.

Zipf's Law Hands-on

Project

- zipf-wordcloud ~/Documents/progetti/software/search-engines/se-unipd/zip
  - .idea
  - experiment
  - src
    - main
      - java
        - it.unipd.dei.se
          - index
            - BodyField
            - DirectoryIndexer
            - package-info.java
          - parse
            - DocumentParser
            - package-info.java
            - ParsedDocument
            - TipsterParser
          - package-info.java
      - matlab
        - zipf_wordcloud.mlx
  - target
  - pom.xml
  - zipf-wordcloud.iml
- External Libraries
- Scratches and Consoles

zipf_wordcloud.mlx

```
%  Copyright 2021 University of Padua, Italy
%
%  Licensed under the Apache License, Version 2.0 (the "License");
%  you may not use this file except in compliance with the License.
%  You may obtain a copy of the License at
%
%      http://www.apache.org/licenses/LICENSE-2.0
%
%  Unless required by applicable law or agreed to in writing, software
%  distributed under the License is distributed on an "AS IS" BASIS,
%  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
%  See the License for the specific language governing permissions and
%  limitations under the License.
%
%
% @author Nicola Ferro
```

Zipf's Law and Word Clouds

This example shows the Zip's law at work on the TIPSTER collection.

$$P_r \sim \frac{c}{r^\alpha}$$

where:

- $c \sim 0.1$ for English
- $\alpha \sim 1$

Vocabulary setup

Change the current path the home of the project

```
workdir = /Users/ferro/Documents/pr;
cd(workdir)
```

Setup the file containing the vocabulary and import it in a table

```
vocabularyFile = experiment/vocabulary.txt;

vocabulary = readtable(vocabularyFile, "FileType", "text", "Delimiter", "tab", "Format", "%s %f %f");
```

Zipf's law for the TIPSTER collection

Setup the parameters c and α

```
c = 0.1
```

```
c = 0.1000
```

Zipf's Law on TIPSTER

Zipf's Law on TIPSTER

	Term	RawFrequency	DocumentFrequency	Probability
1	'the'	17046212	523493	0.0636
2	'of'	8194282	509907	0.0306
3	'to'	6636615	497631	0.0248
4	'and'	6155918	500998	0.0230
5	'in'	5400372	501511	0.0202
6	'a'	5017581	490969	0.0187
7	'for'	2584861	444871	0.0096
8	'that'	2524882	399779	0.0094
9	'is'	2399716	393357	0.0090
10	'on'	1897055	417959	0.0071
11	'by'	1771885	431611	0.0066
12	'with'	1558296	384853	0.0058
13	'be'	1492303	349977	0.0056
14	'as'	1396109	347482	0.0052
15	'it'	1376709	328226	0.0051
16	'at'	1255877	365510	0.0047
17	'was'	1215162	323112	0.0045
18	'are'	1143758	296997	0.0043
19	'from'	1133760	370436	0.0042
20	'this'	1075308	308818	0.0040

$$T = 267862009$$

Term Weighting

- Idea: not all terms are equally useful for describing the document contents
  - For example: a word which appears in each document of the collection is not useful, on the other hand a word which appears in just a few document is useful because it narrows down the set of potential relevant documents
- To characterise term importance, a **weight** is associated with each index term
- The weights are usually defined upon the **raw frequency** of terms in the documents and in the collections
- Most popular weighting scheme in IR: **Term Frequency (TF)** and **Inverse Document Frequency (IDF)**

Raw Frequency of Terms

- Luhn Assumption: the weight of a term is proportional to its **raw (absolute) frequency**
  - The more often a term occurs in a document, the higher its significance is
  - The justification of measuring word significance by use frequency is based on the fact that a writer normally repeats certain words as he advances or varies his arguments and as he elaborates on an aspect of a subject
- Significant words have a **resolving power**
  - the ability of terms to identify relevant items and distinguish them from not relevant ones
- Caveats
  - elimination of all high-frequency terms may cause losses in recall
  - elimination of low-frequency terms may cause losses in precision
  - how to determine the thresholds C and D?

Hans Peter Luhn

Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Word Cloud on TIPSTER Using Raw Frequency of Terms

Torres Parejo, U., Campaña, J. R., Vila, M. A., and Delgado, M. (2021). A survey of tag clouds as tools for information retrieval and content representation. *Information Visualization*, 20(1):83–97.

Term Frequency

- **Term frequency** must be related a specific document (recall) but it must also distinguish among documents (precision)
  - **Relative frequency** measures terms occurring with substantial frequencies within a document but with a relatively low overall collection frequency

$$tf_{i,k} = \frac{\text{count of term } i \text{ in document } k}{\text{number of terms in document } k} = \frac{f_{i,k}}{\sum_{j=1}^{t} f_{j,k}}$$

- Two occurrences should not necessarily be given twice the weight

TERM WEIGHTING SCHEME

$$tf_{i,k} = \begin{cases} 1 + \log_2(f_{i,k}) & \text{if } f_{i,k} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Gerald Salton

Salton, G. and McGill, M. J. (1983). Introduction to Modern Information Retrieval. *McGraw-Hill*, New York, USA.
Buckley, C., Allan, J., Salton, G., and Singhal, A. (1995). Automatic Query Expansion Using SMART: TREC 3. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*, pages 69–80. National Institute of Standards and Technology (NIST), Special Publication 500-225, Washington, USA.

Inverse Document Frequency

- Inverse Document Frequency (IDF)

$$idf_i = \log_2 \frac{N}{n_i}$$

$N \Rightarrow$ # doc in collezione
$n_i \Rightarrow$ # doc dove c'è parola

  - **Exhaustivity of a document**: coverage it provides for the main topic of the document ~ number of index terms the document contains
  - **Specificity of a term**: how well the term describes a document topic ~ inverse of the number of documents where the term occurs

- **TFIDF** term weighting scheme

$$w_{i,k} = \begin{cases} (1 + \log_2(f_{i,k})) \times \log_2 \frac{N}{n_i} & \text{if } f_{i,k} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Karen Spärck Jones

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

TF and IDF on the Whole TIPSTER Collection

© Nicola Ferro

Word Cloud on TIPSTER Using TFIDF

Vector Space Model

- Goal: allow **partial matching**

indice: doc rappresentati come vettori di feature (elementi sono pesi) ⇒ molto sparsi

- Queries and documents are represented as vectors in a V-dimensional vector space
- The **cosine** is used to compute the **similarity** between queries and documents representations

The closer a document vector to the query vector, the higher its rank position.

$$R(q,d) = \cos\left(\theta(\vec{q}, \vec{d})\right) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|} =$$

$$= \frac{\sum_{j=1}^{V} q_j d_{j,k}}{\left(\sqrt{\sum_{j=1}^{V} q_j^2}\right)\left(\sqrt{\sum_{j=1}^{V} d_{j,k}^2}\right)}$$

interpretazione boolean model: OR pesato

Vector (document) length normalization

Salton, G. (1968). *Automatic Information Organization and Retrieval*. McGraw-Hill, New York, USA.
Salton, G., Wong, A., and Yang, C. S. (1975). A Vector Space Model for Automatic Indexing. *Communications of the ACM (CACM)*, 18(11):613– 620.

Vector Space Model: Which Weights?

Table 1. Term-weighting components

Term Frequency Component		
b	1.0	binary weight equal to 1 for terms present in a vector (term frequency is ignored)
t	tf	raw term frequency (number of times a term occurs in a document or query text)
n	$0.5 + 0.5 \frac{\text{tf}}{\max \text{tf}}$	augmented normalized term frequency (tf factor normalized by maximum tf in the vector, and further normalized to lie between 0.5 and 1.0)
Collection Frequency Component		
x	1.0	no change in weight; use original term frequency component (b, t, or n)
f	$\log \frac{N}{n}$	multiply original tf factor by an inverse collection frequency factor (N is total number of documents in collection, and n is number of documents to which a term is assigned)
p	$\log \frac{N-n}{n}$	multiply tf factor by a probabilistic inverse collection frequency factor
Normalization Component		
x	1.0	no change; use factors derived from term frequency and collection frequency only (no normalization)
c	$1 \Big/ \sqrt{\sum_{vector} w_i^2}$	use cosine normalization where each term weight w is divided by a factor representing Euclidian vector length

tenere stopword può essere inutile
indice generato da collezione ⇒
query possono usare parole fuori da
collezione (OOV terms)

Table 2. Typical term-weighting formulas

Weighting System	Document term weight	Query Term weight
Best fully weighted system $tfc \cdot nfx$	$\frac{\text{tf} \cdot \log \frac{N}{n}}{\sqrt{\sum_{vector} \left(\text{tf}_i \cdot \log \frac{N}{n_i}\right)^2}}$	$\left(0.5 + \frac{0.5\ \text{tf}}{\max \text{tf}}\right) \cdot \log \frac{N}{n}$
Best weighted probabilistic weight $nxx \cdot bpx$	$0.5 + \frac{0.5\ \text{tf}}{\max \text{tf}}$	$\log \frac{N-n}{n}$
Classical idf weight $bfx \cdot bfx$	$\log \frac{N}{n}$	$\log \frac{N}{n}$
Binary term independence $bxx \cdot bpx$	1	$\log \frac{N-n}{n}$
Standard tf weight: $txc \cdot txx$	$\frac{\text{tf}}{\sqrt{\sum_{vector} (\text{tf}_i)^2}}$	tf
Coordination level $bxx \cdot bxx$	1	1

Salton, G. and Buckley, C. (1988). Term-weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523.

Vector Space Model: Example of Computation

Vector Space Model

Compute the document weights as TFIDF

$$w_{i,k} = tf_{i,k} \cdot idf_i = tf_{i,k} \cdot \log_2 \frac{N}{n_i}$$

```
docWeights = array2table( relativeFrequency{:, :} .* repmat(idf{:, :}, 1, index.NumDocuments), ... % repmat repeats the idf vector for each document
    "RowNames", rawFrequency.Properties.RowNames, ...                                              % the rows are the same as in rawFrequency
    "VariableNames", rawFrequency.Properties.VariableNames)                                        % the columns are the same as in rawFrequency
```

Compute the query weights as IDF

```
% Set the query weights equal to idf, as if the query contained all the terms in the vocabulary
qWeights = idf;
qWeights.Properties.VariableNames = "Weight"; % rename the colum to Weight

% find which terms of the vocabular actually appear in the query
qIdx = ismember(index.Vocabulary, queryTokens(qID).string)
qIdx = qIdx(:); % ensure it is a column vector

% set to zero the weight of the terms in the vocabulary which do not appear in the query
qWeights{~qIdx, :} = 0
```

Compute the score

$$R(q,d) = \frac{\sum_{j=1}^{V} q_j d_{j,k}}{\left(\sqrt{\sum_{j=1}^{V} q_j^2}\right)\left(\sqrt{\sum_{j=1}^{V} d_{j,k}^2}\right)}$$

```
% compute the numerator, i.e. the dot product between the document vectors and the query vector
score = array2table( sum( repmat(qWeights{:, :}, 1, index.NumDocuments) .* docWeights{:, :} ), ... % repmat repeats the query weights vector for each document
    "RowNames", "score", ...                                                                       % there is only one row
    "VariableNames", rawFrequency.Properties.VariableNames)                                        % the columns are the same as in rawFrequency

% divide by the denominator, i.e. normalize the score by the length of the document and query vectors
score{:, :} = score{:, :} ./ ( sqrt(sum(qWeights{:, :}.^2)) .* sqrt(sum(docWeights{:, :}.^2) ) )
```

Explicit Relevance Feedback: The Rocchio Algorithm

chiede input user ⇒ mette query, diamo 1° ranking, user esegue valutazione

- Optimal query: maximizes the difference between the average vector representing the relevant documents and the average vector representing the non-relevant documents ⇒ tra quelli ritornati da sistema

$$\vec{q}' = \alpha \vec{q} + \beta \frac{1}{|Rel|} \sum_{d_k \in Rel} \vec{d_k} - \gamma \frac{1}{|Nonrel|} \sum_{d_k \in Nonrel} \vec{d_k}$$

$$q'_j = \alpha q_j + \beta \frac{1}{|Rel|} \sum_{d_k \in Rel} d_{j,k} - \gamma \frac{1}{|Nonrel|} \sum_{d_k \in Nonrel} d_{j,k}$$

  - typical values are $\alpha = 8, \beta = 16, \gamma = 4$

in origine, usato per migliorare risultati in real-time ⇒ si potrebbero salvare rel

- Query terms with weights that are negative are dropped

⇒ diventa troppo generica ⇒ ↓ precisione, ↑ recall

- To restrict the amount of expansion, typically only a certain number (say, 50) of the terms with the highest average weights in the relevant documents will be added to the query

forma di query expansion

↳ simile a comportamento reale

non sempre utente ha voglia di farlo ⇒ ⇒ si può usare feedback implicito (e.g. click model)

Rocchio, J. J. (1971). Relevance Feedback in Information Retrieval. In Salton, G., editor, *The SMART Retrieval System. Experiments in Automatic Document Processing*, pages 313–323. Prentice–Hall, Inc., Englewood Cliff, New Jersey, USA.

equivalente a clustering ⇒ CLUSTERING HYPOTHESIS: doc rilevanti per query distaccati

Pseudo-Relevance Feedback

- **Pseudo/Blind** Relevance Feedback
  - The system generates a document ranking from the initial query
  - The system selects a small number of documents from the top of the ranking
  - The system initiates an iteration of RF by assuming these top-ranked documents are all relevant (the **pseudo-relevant** documents)
- This method can can suffer from one major flaw: **query drift**
  - Query drift occurs when the documents used for relevance feedback contain few or no relevant documents
  - Relevance feedback will add terms to the query that are poor at detecting relevance, and hence in retrieving relevant documents
- Although the pseudo relevance feedback techniques described in this section can improve retrieval **performance** over not using pseudo RF, the problem still remains that it is a **variable** technique: some queries will be improved, others will be harmed

Ruthven, I. and Lalmas, M. (2003). A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(2):95–145.

Harman, D. and Buckley, C. (2009). Overview of the Reliable Information Access Workshop. *Information Retrieval*, 12(6):615–641.

Vector Space Model: Pros and Cons

- Advantages
  - Simple and fast computational framework for ranking
  - Returns a ranking of documents through the cosine function
    - Any similarity measure or term weighting scheme could be used
  - Allows for partial matching of query terms;
  - Document length normalisation built-in;
- Disadvantages
  - "**Bag of Words**" model: term proximity is not considered and terms are mutually independent
  - **Heuristic nature**: simple math and not solidly grounded in a theoretical framework.
    - No predictions about techniques for effective ranking

The Probabilistic Model

Probability Ranking Principle

- If a reference retrieval system's response to **each request** is a **ranking of the documents** in the collection in order of **decreasing probability of relevance** to the user who submitted the request,
  - where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose,
- the **overall effectiveness** of the system to its user will be the **best** that is obtainable on the basis of those data

Stephen E. Robertson

Robertson, S. E. (1977). The probability ranking principle. *Journal of Documentation*, 33(4):294–304.

IR as (Bayes) Classification

Binary relevance

Document

Relevant Documents

Non-Relevant Documents

$$\mathbb{P}(R = 1|D = d, Q = q) > \mathbb{P}(R = 0|D = d, Q = q)$$

Bayes Decision Rule

Sample space: the collection

Sample space: the query log

$$R(q, d) \sim \frac{\mathbb{P}(R = 1|D = d, Q = q)}{\mathbb{P}(R = 0|D = d, Q = q)}$$

C. J. “Keith” van Rijsbergen

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London, England, 2nd edition.

IR as (Bayes) Classification

- Notation

$$\mathbb{P}(r|D,Q) = \mathbb{P}(R=1|D=d,Q=q)$$
$$\mathbb{P}(\bar{r}|D,Q) = \mathbb{P}(R=0|D=d,Q=q)$$

- Bayes rule

$$\mathbb{P}(r|D,Q) = \frac{\mathbb{P}(D|r,Q)\mathbb{P}(r,Q)}{\mathbb{P}(D)} \qquad \mathbb{P}(\bar{r}|D,Q) = \frac{\mathbb{P}(D|\bar{r},Q)\mathbb{P}(\bar{r},Q)}{\mathbb{P}(D)}$$

- Classify a document as relevant if

$$\frac{\mathbb{P}(D|r,Q)}{\mathbb{P}(D|\bar{r},Q)} > \frac{\mathbb{P}(\bar{r},Q)}{\mathbb{P}(r,Q)}$$

$R(q,d)$

Independent from D (negligible for ranking)

Binary Independence Model

- Problem: How to estimate $\mathbb{P}(D|r, Q)$ and $\mathbb{P}(D|\bar{r}, Q)$?
- Assumption **B**: document represented by a vector of **binary** random variables indicating term occurrence (or non-occurrence)
  - We are loosing information about term frequency in documents
- Assumption **T**: given relevance, **terms** are statistically **independent**
  - Is this assumption realistic? No, but it simplifies the problem

$$\mathbb{P}(D|r, Q) = \prod_{i} \mathbb{P}(D_i|r, Q) \qquad \mathbb{P}(D|\bar{r}, Q) = \prod_{i} \mathbb{P}(D_i|\bar{r}, Q)$$

- Assumption **Q**: the presence of a **term** in a document depends on relevance only when that term is **present** in the **query**.
  - Is this assumption realistic? Again no, but it simplifies the problem
  - Terms that are not in the query will have the same probability of occurrence in the relevant and non-relevant documents, so they are negligible for ranking

$$\mathbb{P}(D|r, Q) = \prod_{i \in Q} \mathbb{P}(D_i|r) \qquad \mathbb{P}(D|\bar{r}, Q) = \prod_{i \in Q} \mathbb{P}(D_i|\bar{r})$$

Binary Independence Model

- Notation

$$\begin{cases} \mathbb{P}(D_i = 1|r) = p_i \\ \mathbb{P}(D_i = 0|r) = 1 - p_i \end{cases} \qquad \begin{cases} \mathbb{P}(D_i = 1|\bar{r}) = s_i \\ \mathbb{P}(D_i = 0|\bar{r}) = 1 - s_i \end{cases}$$

$$\prod_{i \in Q} \mathbb{P}(D_i|r) = \prod_{i \in Q \cap D} p_i \prod_{i \in Q \setminus D} 1 - p_i$$

$$\prod_{i \in Q} \mathbb{P}(D_i|\bar{r}) = \prod_{i \in Q \cap D} s_i \prod_{i \in Q \setminus D} 1 - s_i$$

- Thus

$$\frac{\mathbb{P}(D|r, Q)}{\mathbb{P}(D|\bar{r}, Q)} = \prod_{i \in Q \cap D} \frac{p_i}{s_i} \prod_{i \in Q \setminus D} \frac{1 - p_i}{1 - s_i}$$

Binary Independence Model

- A bit of manipulation

$$\prod_{i \in Q \cap D} \frac{p_i}{s_i} \prod_{i \in Q \setminus D} \frac{1-p_i}{1-s_i} = \prod_{i \in Q \cap D} \frac{p_i}{s_i} \left[\prod_{i \in Q \cap D} \frac{1-s_i}{1-p_i} \prod_{i \in Q \cap D} \frac{1-p_i}{1-s_i} \right] \prod_{i \in Q \setminus D} \frac{1-p_i}{1-s_i}$$

$$= \prod_{i \in Q \cap D} \frac{p_i(1-s_i)}{s_i(1-p_i)} \prod_{i \in Q} \frac{1-p_i}{1-s_i}$$

Same for all documents (negligible for ranking)

- Use the log to avoid accuracy issues multiplying small numbers

$$R(q,d) \sim \sum_{i \in Q \cap D} \log_2 \frac{p_i(1-s_i)}{s_i(1-p_i)} = \sum_{i \in Q \cap D} \log_2 \frac{p_i}{(1-p_i)} - \sum_{i \in Q \cap D} \log_2 \frac{s_i}{(1-s_i)}$$

Contribution of terms appearing in relevant documents

Contribution of terms appearing in not relevant documents

Estimating Probabilities: Without Information about Relevance

- Assume that p_i is constant
- Assume that s_i could be estimated by using the term occurrences in the whole collection

$$\begin{cases} p_i &= 0.5 \\ s_i &= \frac{n_i}{N} \end{cases}$$

  - the number of relevant documents is much smaller than the total number of documents in the collection
- We obtain an IDF-like scoring function
  - no term frequency component due to the binary assumption

$$R(q,d) \sim \sum_{i \in Q \cap D} \log_2 \frac{0.5(1-\frac{n_i}{N})}{\frac{n_i}{N}(1-0.5)} = \sum_{i \in Q \cap D} \log_2 \frac{N-n_i}{n_i}$$

Becomes negative for $n_i > \frac{N}{2}$

Croft, W. B. and Harper, D. J. (1979). Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4):285–295.

BIM Weights vs IDF

Estimating Probabilities: With Information about Relevance

Number of relevant documents containing term i

	Relevant	Non-relevant	Total
$D_i = 1$	r_i	$n_i - r_i$	n_i
$D_i = 0$	$R - r_i$	$N - n_i - R + r_i$	$N - n_i$
Total	R	$N - R$	N

Total number of relevant documents

$$\begin{cases} p_i &= \frac{0.5+r_i}{R+1} \\ s_i &= \frac{n_i - r_i + 0.5}{N-R+1} \end{cases}$$

Smoothing

- Scoring function, using BIM smoothed weights

$$R(q,d) \sim \sum_{i \in Q \cap D} \log_2 \frac{(r_i + 0.5)(N - n_i - R + r_i + 0.5)}{(n_i - r_i + 0.5)(R - r_i + 0.5)}$$

Robertson, S. E. and Spärck Jones, K. (1976). Relevance Weighting of Search Terms. *Journal of the American Society for Information Science (JASIS)*, 27(3):129–146.

BIM: Example of Computation

Binary Independence Model

Determine which terms match in the documents and the query

```
idx = rawFrequency{:, :} > 0 & repmat(qIdx, 1, index.NumDocuments) % rawFrequency > 0 determines in which document each term appears; repmat repeats the qIdx vector for each document
```

Compute the BIM IDF-like weight of each term in absence of relevance information

$$bim_i = \log_2 \frac{N - n_i}{n_i}$$

```
% we initially use a matrix (and not a table) because logical indexing works simpler with matrices
docWeights = repmat(docFrequency{:, :}, 1, index.NumDocuments);        % repeat the document frequency information for each document
docWeights = log2( (index.NumDocuments - docWeights) ./ docWeights);   % compute the BIM weights
docWeights(~idx) = 0;                                                  % set to zero the weights for the terms not in the query

% wrap in a table
docWeights = array2table( docWeights, ...
    "RowNames", rawFrequency.Properties.RowNames, ...                  % the rows are the same as in rawFrequency
    "VariableNames", rawFrequency.Properties.VariableNames)            % the columns are the same as in rawFrequency
```

Compute the score in absence of relevance information

$$R(q, d) \sim \sum_{i \in Q \cap D} \log_2 \frac{N - n_i}{n_i}$$

```
score = array2table( sum( docWeights{:, :} ), ...
    "RowNames", "score", ...                                   % there is only one row
    "VariableNames", rawFrequency.Properties.VariableNames)    % the columns are the same as in rawFrequency
```

BM25

- Okapi BM25 (Best Match, attempt 25)
  - adds document and query term weights
  - normalizes by document length

The term frequency component, typically $k_1 = 1.2$

$$R(q,d) \sim \sum_{i \in Q \cap D} \log_2 \frac{(r_i + 0.5)(N - n_i - R + r_i + 0.5)}{(n_i - r_i + 0.5)(R - r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{k_1 \left((1-b) + b \frac{dl}{avdl}\right) + f_i} \cdot \frac{(k_2 + 1) qf_i}{k_2 + qf_i}$$

BIM smoothed weight (or any other weight)

The document-length normalization, typically $b = 0.75$

The query term frequency component, typically $k_2 \in [0, 1000]$

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gat- ford, M. (1995). Okapi at TREC–3. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*, pages 109–126. National Institute of Standards and Technology (NIST), Special Publication 500-225, Washington, USA.

Spärck Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments – Part 1. *Information Processing & Management*, 36(6):779–808.
Spärck Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments – Part 2. *Information Processing & Management*, 36(6):809–840.

BM25

- Okapi BM25 (Best Match, attempt 25)
  - adds document and query term weights
  - normalizes by document length

TF-like for query terms
Note that it becomes 1
when considering
$qf_i = 1$

$$R(q,d) \sim \sum_{i \in Q \cap D} \log_2 \frac{(r_i + 0.5)(N - n_i - R + r_i + 0.5)}{(n_i - r_i + 0.5)(R - r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{k_1 \left((1-b) + b \frac{dl}{avdl}\right) + f_i} \cdot \frac{(k_2 + 1) qf_i}{k_2 + qf_i}$$

IDF-like
component

TF-like
component

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gat- ford, M. (1995). Okapi at TREC–3. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*, pages 109–126. National Institute of Standards and Technology (NIST), Special Publication 500-225, Washington, USA.

Spärck Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments – Part 1. *Information Processing & Management*, 36(6):779–808.
Spärck Jones, K., Walker, S., and Robertson, S. E. (2000). A probabilistic model of information retrieval: development and comparative experiments – Part 2. *Information Processing & Management*, 36(6):809–840.

BM25: Example of Computation

BM25

Setup the parameters for BM25.

Note that, in our case, we ignore k_2 since the frequency of query terms is $qf_i = 1$ and so $\frac{(k_2+1)qf_i}{k_2+qf_i} = \frac{k_2+1}{k_2+1} = 1$

```
k1 = 1.2;
b = 0.75;

% compute the average document length
avgDocLength = mean(docLength)
```

Compute the smoothed BIM IDF-like weight of each term in absence of relevance information ($R = 0, r_i = 0$)

$$bim_i = \log_2 \frac{(r_i+0.5)(N-n_i-R+r_i+0.5)}{(n_i-r_i+0.5)(R-r_i+0.5)} = \log_2 \frac{N-n_i+0.5}{n_i+0.5}$$

```
% we initially use a matrix (and not a table) because logical indexing works simpler with matrices
bimIDF = repmat(docFrequency{:, :}, 1, index.NumDocuments);              % repeat the document frequency information for each document
bimIDF = log2( (index.NumDocuments - bimIDF + 0.5) ./ (bimIDF + 0.5) );   % compute the smoothed BIM weights
bimIDF(~idx) = 0;                                                        % set to zero the weights for the terms not in the query

% wrap in a table
bimIDF = array2table( bimIDF, ...
    "RowNames", rawFrequency.Properties.RowNames, ...                    % the rows are the same as in rawFrequency
    "VariableNames", rawFrequency.Properties.VariableNames)              % the columns are the same as in rawFrequency
```

Compute the BM25 TF-like factor

$$tf_{i,k} = \frac{(k_1+1)f_i}{k_1\left((1-b)+b\frac{dl_k}{avdl}\right)+f_i}$$

```
% we initially use a matrix (and not a table) because logical indexing works simpler with matrices
bm25TF = ((k1 + 1) .* rawFrequency{:, :}) ./ ...
    ( repmat( k1 .* ( (1-b) + b .* docLength ./ avgDocLength ), index.NumWords, 1 ) + rawFrequency{:, :} );   % repmat repeats the document length normalization for each term
bm25TF(~idx) = 0;                                                                                            % set to zero the weights for the terms not in the query

% wrap in a table
bm25TF = array2table( bm25TF, ...
    "RowNames", rawFrequency.Properties.RowNames, ...                                                        % the rows are the same as in rawFrequency
    "VariableNames", rawFrequency.Properties.VariableNames)                                                  % the columns are the same as in rawFrequency
```

Compute the score in absence of relevance information

$$R(q,d) \sim \sum_{i \in Q \cap D} \log_2 \frac{N-n_i+0.5}{n_i+0.5} \cdot \frac{(k_1+1)f_i}{k_1\left((1-b)+b\frac{dl}{avdl}\right)+f_i}$$

```
score = array2table( sum( bimIDF{:, :} .* bm25TF{:, :} ), ...
    "RowNames", "score", ...                                 % there is only one row
    "VariableNames", rawFrequency.Properties.VariableNames)  % the columns are the same as in rawFrequency
```

BM25: Pros and Cons

- Advantages
  - Simple and fast
  - One of the best-known methods for ranked retrieval, often used as baseline;
  - Competitive with the more modern techniques;
- (Some) Improvements and extensions to BM25
  - ATIRE BM25 (and others): disallowing for negative weights
  - BM25L: the document length normalization of BM25 unfairly prefers shorter documents to longer ones
  - BM25-adpt: a global k_1 applied to all query terms is likely to be less effective than a term-specific k_1
  - **BM25F**: work with fielded documents

Robertson, S. E. and Zaragoza, U. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval (FnTIR)*, 3(4):333–389.

Lv, Y. and Zhai, C. (2011). Adaptive Term Frequency Normalization for BM25. In Ounis, I., Ruthven, I., Berendt, B., de Vries, A. P., and Wenfei, F., editors, *Proc. 20th International Conference on Information and Knowledge Management (CIKM 2011)*, pages 1985–1988. ACM Press, New York, USA.

Lv, Y. and Zhai, C. (2011b). When Documents Are Very Long, BM25 Fails! In Ma, W.-Y., Nie, J.-Y., Baeza-Yaetes, R., Chua, T.-S., and Croft, W. B., editors, *Proc. 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011)*, pages 1103–1104. ACM Press, New York, USA.

Trotman, A., Puurula, A., and Burgess, B. (2014). Improvements to BM25 and Language Models Examined. In Culpepper, J. S., Park, L., and Zuccon, G., editors, *Proc. 19th Australasian Document Computing Symposium (ADCS 2014)*, pages 58–65. ACM Press, New York, USA.

Language Models

What is a language model?

A (statistical) language model θ is a probability distribution over word sequences

$$\theta := \begin{cases} \mathbb{P}(\text{“Today is Wednesday”}) = 0.001 \\ \mathbb{P}(\text{“Today Wednesday is”}) = 0.000000001 \\ \mathbb{P}(\text{“The equation has a solution”}) = 0.000000001 \end{cases}$$

- A language model can be **context dependent**
  - The above one may be a reasonable language model for describing general conversations, but it may be inaccurate for describing conversations happening at a mathematics conference
- Given a language model, we can **sample word sequences** according to the distribution to obtain a text sample. In this sense, we may use such a model to “generate” text.
  - Thus, a language model is also often called a **generative model** for text

Unigram Language Models

A **unigram language model** assumes that a word sequence is generated by picking each word **independently**

$$\mathbb{P}(w_1, w_2, \ldots) = \frac{(f_1 + f_2 + \ldots)!}{f_1! \cdot f_2! \cdot \ldots} \prod_i \mathbb{P}(w_i|\theta)^{f_i}$$

- We assume a **multinomial distribution** in order to simplify computations and estimations
  - IR works with “bag of words” and this is a reasonable approximation
- A note on distributions
  - **Bernuolli distrbution**: a discrete binary random variable which takes value 1 with probability p .
    For example, one trial flipping a coin
  - **Binomial distribution**: a discrete probability distribution of the number of successes in a sequence of n independent trials, each one taking value 1 with probability p
    For example, n trials flipping a coin
  - **Multinomial distribution**: a discrete probability distribution of of any particular combination of numbers of successes for k categories in a sequence of n independent trials, where exactly one category can succeed with probability p_i
    For example, n trials rolling a k-sided dice

Unigram Language Models

A **unigram language model** assumes that a word sequence is generated by picking each word **independently**

$$\mathbb{P}(w_1, w_2, \ldots) = \frac{(f_1 + f_2 + \ldots)!}{f_1! \cdot f_2! \cdot \ldots} \prod_i \mathbb{P}(w_i|\theta)^{f_i}$$

All the possible bags of words

Frequency of a word

- We assume a multinomial distribution in order to simplify computations and estimations
  - IR works with "bag of words" and this is a reasonable approximation
- A note on distributions
  - **Bernuolli distrbution**: a discrete binary random variable which takes value 1 with probability p .
    For example, one trial flipping a coin
  - **Binomial distribution**: a discrete probability distribution of the number of successes in a sequence of n independent trials, each one taking value 1 with probability p
    For example, n trials flipping a coin
  - **Multinomial distribution**: a discrete probability distribution of of any particular combination of numbers of successes for k categories in a sequence of n independent trials, where exactly one category can succeed with probability p_i
    For example, n trials rolling a k-sided dice

Why Language Models?

- A language model provides us with a **principled way** to quantify the uncertainties associated with the use of natural language
  - A language model may help answering the question: how likely would a user use a query containing the word “baseball” if the user wants to find information about sports?
- What kind of documents would have higher probabilities?
  - Intuitively it would be those documents that contain many occurrences of the high probability words according to θ
  - The high probability words of θ can indicate the topic captured by θ

Why Language Models? An Example

$$\theta_1 := \begin{cases} \dots \\ \mathbb{P}(\text{“text”}) = 0.2 \\ \mathbb{P}(\text{“mining”}) = 0.1 \\ \mathbb{P}(\text{“association”}) = 0.01 \\ \mathbb{P}(\text{“clustering”}) = 0.02 \\ \dots \\ \mathbb{P}(\text{“food”}) = 0.00001 \\ \dots \end{cases} \qquad \theta_2 := \begin{cases} \dots \\ \mathbb{P}(\text{“food”}) = 0.25 \\ \mathbb{P}(\text{“nutrition”}) = 0.1 \\ \mathbb{P}(\text{“healthy”}) = 0.05 \\ \mathbb{P}(\text{“diet”}) = 0.02 \\ \dots \\ \mathbb{P}(\text{“text”}) = 0.00001 \\ \dots \end{cases}$$

- Suppose we have two language models: θ_1 captures the topic “text mining” while θ_2 the topic “health”
- Suppose we have two documents: D is a text mining paper while D' is a blog article discussing diet control
- We would expect that

$$\mathbb{P}(D|\theta_1) > \mathbb{P}(D|\theta_2) \qquad \mathbb{P}(D|\theta_1) > \mathbb{P}(D'|\theta_1)$$

$$\mathbb{P}(D'|\theta_1) < \mathbb{P}(D'|\theta_2) \qquad \mathbb{P}(D|\theta_2) < \mathbb{P}(D'|\theta_2)$$

Unigram LM: Maximum Likelihood Estimator

- Suppose we have a document D. How do we estimate a model $\hat{\theta}$ which would generate D?
  - In other terms, how do we estimate the probabilities $\mathbb{P}(w_i|\hat{\theta})$ for each word in the document D?
- A common approach is to use a **maximum likelihood estimator** (MLE)

$$\hat{\theta} = \arg\max_{\theta} \mathbb{P}(D|\theta)$$

whose solution is

$$\mathbb{P}(w_i|\hat{\theta}) = \frac{f_i}{\sum_{j=1}^{t} f_j} = tf_i = \frac{\text{count of term } i \text{ in document } D}{\text{number of terms in document } D}$$

Zero probability for terms not occurring in D. It **overfits**

Previously called **relative term frequency**.

© Nicola Ferro

Unigram Language Models: Example of Computation

Unigram Language Models

This example shows, step-by-step, how to

- estimatate unigram language models using MLE
- compute the probability of a sequence of words given an estimated language model
- generate new text from an estimated language model

Setup

The documents

```
d1 = "The quokka, the only member of the genus Setonix, is a small marsupial abou
d2 = "Wombats are small, short-legged, muscular quadrupedal marsupials that are n
d3 = "Quokkas have little fear of humans and commonly approach people closely, pa

% the collection of documents
corpus = [d1, d2, d3]
```

```
corpus = 1×3 string
   "The quokka, the only member of the genus Setonix, is a small marsu… "Wombats ...
```

Document Pre-processing

Perform the following steps:

- simple tokenization at separators (white spaces)
- removal of punctuation
- lower case transformation

Query Likelihood Ranking

Rank documents by the probability that the query text could be generated by the document language model

$$R(q,d) = \mathbb{P}(Q|\theta_D) \sim \prod_{w_i \in Q} \mathbb{P}(w_i|\theta_D)^{f_i}$$

- For each document in the collection, we estimate its own language model and compute the probability of the query to be generated by that document
  - Even assuming a multinomial distribution, we do not have the factorials (all the possible bags of words) since they are constant for a given query and thus indifferent for ranking
- $\mathbb{P}(Q|\theta_D)$ should be interpreted as $\mathbb{P}(Q = q|D = d, r)$, i.e. the probability that the user would enter the query q in order to retrieve d
  - Conditioning on d provides us with an example of a relevant document, creating a **topic model** for that document

Smoothing

- The maximum likelihood estimator (MLE)

$$\mathbb{P}(w_i|\hat{\theta}_D) = \frac{f_{w_i,D}}{|D|} = \frac{\text{count of term } w_i \text{ in document } D}{\text{number of terms in document } D}$$

  suffers from major problem: if any of the **query words** are **missing** from the **document**, the score given by the whole query likelihood model for $\mathbb{P}(Q|\theta_D)$ will be zero

- Since we are building a topic model for a document, words associated with that topic should have some probability of occurring, even if they were not mentioned in the document
- **Smoothing** is a technique for avoiding this estimation problem and overcoming **data sparsity**.
  - The general approach to smoothing is to **lower** (or discount) the **probability** estimates for **words** that are **seen** in the document text, and **assign** that "**leftover**" probability to the estimates for the **words** that are **not seen** in the text

Jelinek-Mercer Smoothing

$$\mathbb{P}(w_i|\hat{\theta}_D) = (1-\lambda)\underbrace{\frac{f_{w_i,D}}{|D|}}_{\theta_D} + \lambda\underbrace{\frac{f_{w_i,C}}{|C|}}_{\theta_C}$$

- The collection language θ_C model is called the **background model**
- **Small** values of **λ** produce less smoothing, and consequently the query tends to act more like a **Boolean AND** since the absence of any query word will penalize the score substantially. As **λ approaches 1**, the query acts more like a **Boolean OR**
- It has been shown that values of **λ** around **0.1** work well for **short queries**, whereas values around **0.7** are better for much **longer queries**

Jelinek-Mercer Smoothing

$$\mathbb{P}(w_i|\hat{\theta}_D) = (1-\lambda)\underbrace{\frac{f_{w_i,D}}{|D|}}_{\theta_D} + \lambda\underbrace{\frac{f_{w_i,C}}{|C|}}_{\theta_C}$$

Number of times a word appears in the collection

- The collection language θ_C model is called the **background model**
- **Small** values of **λ** produce less smoothing, and consequently the query tends to act more like a **Boolean AND** since the absence of any query word will penalize the score substantially. As **λ approaches 1**, the query acts more like a **Boolean OR**
- It has been shown that values of **λ** around **0.1** work well for **short queries**, whereas values around **0.7** are better for much **longer queries**

Jelinek-Mercer Smoothing

$$\mathbb{P}(w_i|\hat{\theta}_D) = (1-\lambda)\underbrace{\frac{f_{w_i,D}}{|D|}}_{\theta_D} + \lambda \underbrace{\frac{f_{w_i,C}}{|C|}}_{\theta_C}$$

Number of times a word appears in the collection

Number of words in the collection

- The collection language θ_C model is called the **background model**
- **Small** values of **λ** produce less smoothing, and consequently the query tends to act more like a **Boolean AND** since the absence of any query word will penalize the score substantially. As **λ approaches 1**, the query acts more like a **Boolean OR**
- It has been shown that values of **λ** around **0.1** work well for **short queries**, whereas values around **0.7** are better for much **longer queries**

Dirichlet Smoothing

$$\mathbb{P}(w_i|\hat{\theta}_D) = \frac{f_{w_i,D} + \mu \frac{f_{w_i,C}}{|C|}}{|D| + \mu}$$

- Suppose we have **added** an **extra** $\mu > 0$ **words** to each document in the collection and we have distributed them according to the collection language model θ_C . This would change the MLE estimation as above
- It is called Dirichlet smoothing because it can be derived from a Dirichlet distribution, which is the natural way to specify prior knowledge when estimating the probabilities in a multinomial distribution
- Typical values of μ are in the range 1,000 to 2,000 (remember that collection probabilities are very small), and Dirichlet smoothing is generally more effective than Jelinek-Mercer, especially for short queries

Questions?

ASOK LOST HIS SOUL
WE CAN IMPROVE OUR GOOGLE SEARCH RANKING WITH KEY WORDS, INBOUND LINKS AND. . .
Dilbert.com DilbertCartoonist@gmail.com
. . . RITUAL SACRIFICE OF A. . .
2-19-10 ©2010 Scott Adams, Inc./Dist. by UFS, Inc.
I THINK IT'S DOWN TO YOU OR ME.
WHAT ARE YOU IMPLYING?