DAT630

Retrieval Models

Search Engines, Chapter 7

28/08/2017

Krisztian Balog | University of Stavanger

So far…

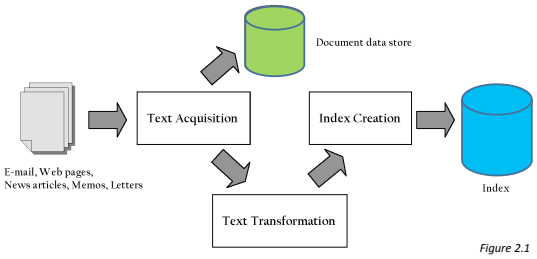


Figure 2.1

Today

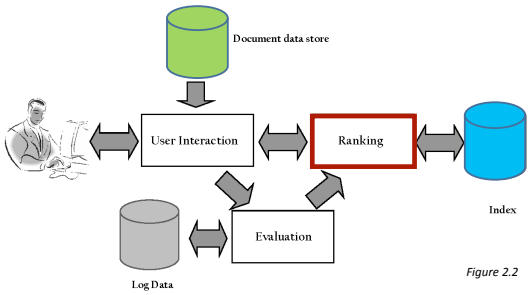


Figure 2.2

Boolean Retrieval

Boolean Retrieval

- Two possible outcomes for query processing
  - TRUE and FALSE (relevance is binary)
  - "Exact-match" retrieval
- Query usually specified using Boolean operators
  - AND, OR, NOT
  - Can be extended with wildcard and proximity operators
- Assumes that all documents in the retrieved set are equally relevant

Boolean Retrieval

- Many search systems you still use are Boolean:
  - Email, library catalog, …
- Very effective in some specific domains
  - E.g., legal search
  - E.g., patent search
  - Expert users

Boolean View of a Collection

- Each row represents the view of a particular term: What documents contain this term?
  - Like an inverted list
- To execute a query
  - Pick out rows corresponding to query terms
  - Apply the logic table of the corresponding Boolean operator

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
aid	0	0	0	1	0	0	0	1
all	0	1	0	1	0	1	0	0
back	1	0	1	0	0	0	1	0
brown	1	0	1	0	1	0	1	0
come	0	1	0	1	0	1	0	1
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0
good	0	1	0	1	0	1	0	1
jump	0	0	1	0	0	0	0	0
lazy	1	0	1	0	1	0	1	0
men	0	1	0	1	0	0	0	1
now	0	1	0	0	0	1	0	1
over	1	0	1	0	1	0	1	1
party	0	0	0	0	0	1	0	1
quick	1	0	1	0	0	0	0	0
their	1	0	0	0	1	0	1	0
time	0	1	0	1	0	1	0	0

Example Queries

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0

dog ∧ fox	?	dog AND fox → ?
dog ∨ fox	?	dog OR fox → ?
dog ¬ fox	?	dog AND NOT fox → ?
fox ¬ dog	?	fox AND NOT dog → ?

Example Queries

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0

dog ∧ fox	?	dog AND fox → Doc 3, Doc 5
dog ∨ fox	?	dog OR fox → Doc 3, Doc 5, Doc 7
dog ¬ fox	?	dog AND NOT fox → empty
fox ¬ dog	?	fox AND NOT dog → Doc 7

Example Query

good AND party AND NOT over

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
good	0	1	0	1	0	1	0	1
party	0	0	0	0	0	1	0	1
over	1	0	1	0	1	0	1	1

g ∧ p	0	0	0	0	0	1	0	1	good AND party → Doc 6, Doc 8
over	1	0	1	0	1	0	1	1	
g ∧ p ¬ o	0	0	0	0	0	1	0	0	good AND party AND NOT over → Doc 6

Example of Query (Re)formulation

lincoln

- Retrieves a large number of documents
- User may attempt to narrow the scope

president AND lincoln

- Also retrieves documents about the management of the Ford Motor Company and Lincoln cars

Ford Motor Company today announced that Darrly Hazel will succeed Brian Kelly as **president** of **Lincoln** Mercury.

Example of Query (Re)formulation

- User may try to eliminate documents about cars

president AND lincoln
AND NOT (automobile OR car)

- This would remove any document that contains even of the single mention of "automobile" or "car"
- For example, sentence in biography

Lincoln's body departs Washington in a nine-**car** funeral train.

Example of Query (Re)formulation

- If the retrieved set is too large, the user may try to further narrow the query by adding additional words that occur in biographies

president AND lincoln
AND (biography OR life OR birthplace)
AND NOT (automobile OR car)

- This query may do a reasonable job at retrieving a set containing some relevant documents
- But it does not provide a **ranking** of documents

Example

- WestLaw.com: Largest commercial (paying subscribers) legal search service
- Example query:
  - *What is the statute of limitations in cases involving the federal tort claims act?*
  - **LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM**
    - ! = wildcard, /3 = within 3 words, /S = in same sentence

Boolean Retrieval

- Advantages
  - Results are relatively easy to explain
  - Many different features can be incorporated
  - Efficient processing since many documents can be eliminated from search
  - We do not miss any relevant document

Boolean Retrieval

- Disadvantages
  - Effectiveness depends entirely on user
    - Simple queries usually don't work well
    - Complex queries are difficult to create accurately
  - No ranking
  - No control over result set size: either too many docs or none
  - What about partial matches? Documents that "don't quite match" the query may be useful also

Ranked Retrieval

General Scoring Formula

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

Relevance score
It is computed for each document *d* in the collection for a given input query *q*

Documents are returned in decreasing order of this score

It is enough to consider terms in the query

Term's weight in the document

Term's weight in the query

Example 1: Term presence/absence

- The score is the number of matching query terms in the document

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

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

$f_{t,q}$

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

Term Weighting

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

Example 2: Log-frequency Weighting

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

Raw term frequency →

$f_{t,d}$	$w_{t,d}$
0	0
1	1
2	1.3
10	2
1000	4

Example 2: Log-frequency Weighting

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

$$score(d,q) = \sum_{t \in q} (1 + \log f_{t,d}) \cdot f_{t,q}$$

The Vector Space Model

The 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

Representation

- Documents and query represented by a **vector of term weights**

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{it}) \qquad Q = (q_1, q_2, \ldots, q_t)$$

- Collection represented by a matrix of term weights

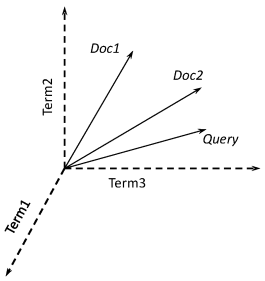
	$Term_1$	$Term_2$	$\ldots$	$Term_t$
Doc_1	d_{11}	d_{12}	$\ldots$	d_{1t}
Doc_2	d_{21}	d_{22}	$\ldots$	d_{2t}
$\vdots$	$\vdots$			
Doc_n	d_{n1}	d_{n2}	$\ldots$	d_{nt}

Bag-of-Words Model

- Vector representation doesn't consider the ordering of words in a document
- "*John is quicker than Mary*" and "*Mary is quicker than John*" have the same vectors

Scoring Documents

- Documents "near" the query's vector (i.e., more similar to the query) are more likely to be relevant to the query



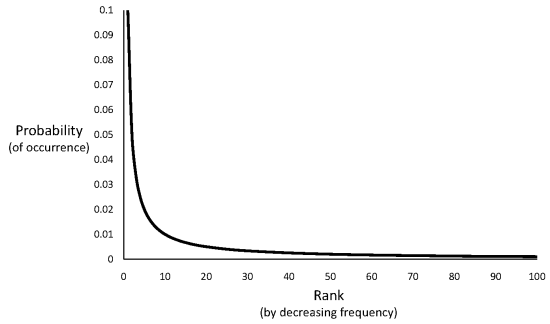
Scoring Documents

- The score for a document is computed using the cosine similarity of the document and query vectors

$$cosine(d, q) = \frac{\sum_t w_{t,d} \cdot w_{t,q}}{\sqrt{\sum_t w_{t,d}^2}\sqrt{\sum_t w_{t,q}^2}}$$

Term Weighting

Zipf's Law



Weighting Terms

- Intuition
  - Terms that appear often in a document should get high weights
    - The more often a document contains the term "dog", the more likely that the document is "about" dogs
  - Terms that appear in many documents should get low weights
    - E.g., stopword-like words
- How do we capture this mathematically?
  - Term frequency
  - Inverse document frequency

Term Frequency (TF)

- Reflects the importance of a term in a document (or query)
- Variants
  - binary $tf_{t,d} = \{0, 1\}$
  - raw frequency $tf_{t,d} = f_{t,d}$
  - normalized $tf_{t,d} = f_{t,d}/|d|$
  - log-normalized $tf_{t,d} = 1 + \log f_{t,d}$
  - …
- $f_{t,d}$ is the number of occurrences of term k in the document and |d| is the length of d

Inverse Document Frequency (IDF)

- Reflects the importance of the term in the collection of documents
  - The more documents that a term occurs in, the less *discriminating* the term is between documents, consequently, the less useful for retrieval

$$idf_t = \log \frac{N}{n_t}$$

  - where *N* is the total number of document and n_t is the number of documents that contain term *t*
  - log is used to "dampen" the effect of IDF

Term Weights

- Combine TF and IDF weights by multiplying them:

$$tfidf_{t,d} = tf_{t,d} \cdot idf_t$$

  - Term frequency weight measures importance in document
  - Inverse document frequency measures importance in collection

Scoring Documents

- The score for a document is computed using the cosine similarity of the document and query vectors

$$cosine(d,q) = \frac{\sum_t w_{t,d} \cdot w_{t,q}}{\sqrt{\sum_t w_{t,d}^2}\sqrt{\sum_t w_{t,q}^2}}$$

$$cosine(d,q) = \frac{\sum_t tfidf_{t,d} \cdot tfidf_{t,q}}{\sqrt{\sum_t tfidf_{t,d}^2 \sum_t tfidf_{t,q}^2}}$$

Scoring Documents

- It also fits within our general scoring scheme:
  - Note that we only consider terms that are present in the query

$$Score(q,d) = \sum_{t \in q} w_{t,q} \cdot w_{t,d}$$

$$w_{t,q} = \frac{tfidf_{t,q}}{\sqrt{\sum_t tfidf_{t,q}^2}} \qquad w_{t,d} = \frac{tfidf_{t,d}}{\sqrt{\sum_t tfidf_{t,d}^2}}$$

Scoring Documents

- It also fits within our general scoring scheme:
  - Note that we only consider terms that are present in the query

$$Score(q,d) = \sum_{t \in q} w_{t,q} \cdot w_{t,d}$$

$$w_{t,q} = \frac{tfidf_{t,q}}{\sqrt{\sum_t tfidf_{t,q}^2}} \qquad w_{t,d} = \frac{tfidf_{t,d}}{\sqrt{\sum_t tfidf_{t,d}^2}}$$

may be left out (the same for all docs)

can be pre-computed (and stored in the index)

Basic Algorithm

```
CosineScore(q)
1  float Scores[N] = 0
2  float Length[N]
3  for each query term t
4  do calculate w_{t,q} and fetch postings list for t
5      for each pair(d, tf_{t,d}) in postings list
6      do Scores[d] += w_{t,d} × w_{t,q}
7  Read the array Length
8  for each d
9  do Scores[d] = Scores[d]/Length[d]
10 return Top K components of Scores[]
```

Variations on Term Weighting

- It is possible to use different term weighting for documents and for queries, for example:

weighting scheme	document term weight	query term weight
1	$f_{i,j} * \log \frac{N}{n_i}$	$(0.5 + 0.5\frac{f_{i,q}}{max_i\ f_{i,q}}) * \log \frac{N}{n_i}$
2	$1 + \log f_{i,j}$	$\log(1 + \frac{N}{n_i})$
3	$(1 + \log f_{i,j}) * \log \frac{N}{n_i}$	$(1 + \log f_{i,q}) * \log \frac{N}{n_i}$

  - See also: https://en.wikipedia.org/wiki/Tf-idf for further variants

Difference from Boolean Retrieval

- Similarity calculation has two factors that distinguish it from Boolean retrieval
  - Number of matching terms affects similarity
  - Weight of matching terms affects similarity
- Documents can be *ranked* by their similarity scores

Exercise

BM25

BM25

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

BM25 Scoring

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

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

BM25: An Intuitive View

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

Terms common between the document and the query => good

BM25: An Intuitive View

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

Repetitions of query terms in the document => good

BM25: An Intuitive View

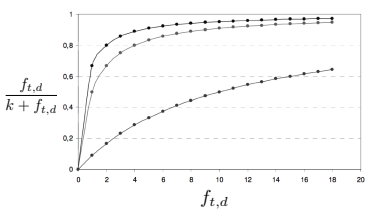
$$score(d,q) = \sum_{t \in q} \frac{f_{t,d} \cdot (1 + k_1)}{f_{t,d} + k_1(1 - b + b\frac{|d|}{avgdl})} \cdot idf_t$$

Term saturation: repetition is less important after a while

BM25: An Intuitive View

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

Term saturation



$\frac{f_{t,d}}{k + f_{t,d}}$ for some k > 0

Asymptotically approaches 1

Middle line is *k=1*
Upper line is lower *k*
Lower line is higher *k*

BM25: An Intuitive View

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

Soft document normalization taking into account document length
Document is more important if relatively long (w.r.t. average)

BM25: An Intuitive View

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

Common terms less important

Parameter Setting

- k_1: calibrating term frequency scaling
  - 0 corresponds to a binary model
  - large values correspond to using raw term frequencies
  - k_1 is set between 1.2 and 2.0, a typical 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

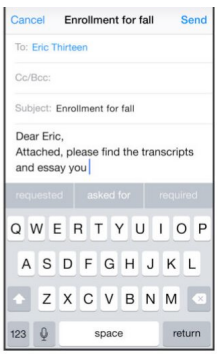
- Based on the notion of probabilities and processes for generating text

Uses

- Speech recognition
  - "I ate a cherry" is a more likely sentence than "Eye eight uh Jerry"
- OCR & Handwriting recognition
  - More probable sentences are more likely correct readings
- Machine translation
  - More likely sentences are probably better translations

Uses

- Completion prediction
  - Please turn off your cell _____
  - Your program does not _____
- *Predictive text input systems* can guess what you are typing and give choices on how to complete it



Ranking Documents using Language Models

- 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?"

Standard Language Modeling approach

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

Document prior
Probability of the document being relevant to *any* query

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

Standard Language Modeling approach (2)

Number of times *t* appears in *q*

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

Document language model
Multinomial probability distribution over the vocabulary of terms

Smoothing parameter

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

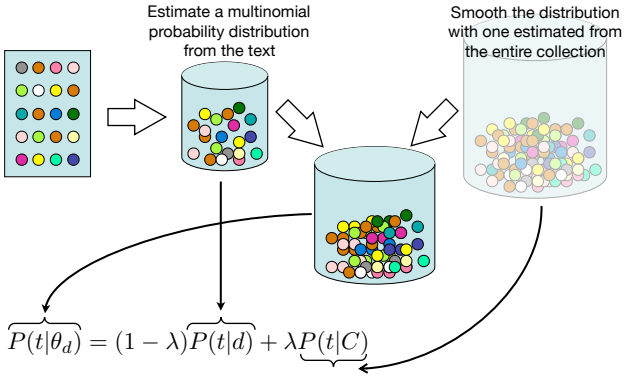
Empirical document model

Maximum likelihood estimates

Collection (a.k.a. background) model

$$\frac{f_{t,d}}{|d|} \qquad \frac{\sum_{d'} f_{t,d'}}{\sum_{d'} |d'|}$$

Language Modeling



Example

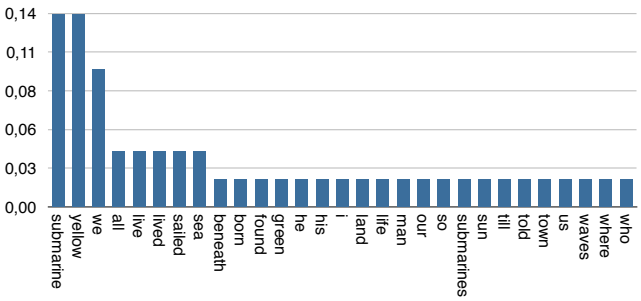
In the town where I was born,
Lived a man who sailed to sea,
And he told us of his life,
In the land of submarines,

So we sailed on to the sun,
Till we found the sea green,
And we lived beneath the waves, In our yellow submarine,

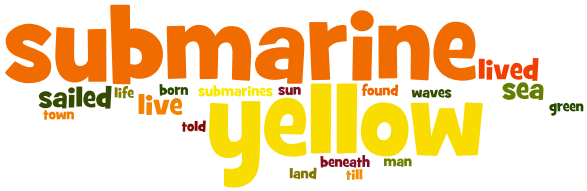
We all live in yellow submarine, yellow submarine, yellow submarine, We all live in yellow submarine, yellow submarine, yellow submarine.

Empirical document LM

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



Alternatively...



Scoring a query

$q = \{\text{sea}, \text{submarine}\}$

$P(q|d) = P(\text{“sea”}|\theta_d) \cdot P(\text{“submarine”}|\theta_d)$

Scoring a query

$q = \{\text{sea}, \text{submarine}\}$

$P(q|d) = \underbrace{P(\text{“sea”}|\theta_d)}_{0.03602} \cdot P(\text{“submarine”}|\theta_d)$

$(1-\lambda)P(\text{“sea”}|d) + \lambda P(\text{“sea”}|C)$

0.9 0.04 0.1 0.0002

t	P(t|d)
submarine	0,14
sea	0,04
...	

t	P(t|C)
submarine	0,0001
sea	0,0002
...	

Scoring a query

$q = \{\text{sea}, \text{submarine}\}$

0.04538 0.03602 0.12601

$$P(q|d) = P(\text{“sea”}|\theta_d) \cdot \underbrace{P(\text{“submarine”}|\theta_d)}$$

0.9 0.14 0.1 0.0001

$$(1-\lambda)P(\text{“submarine”}|d) + \lambda P(\text{“submarine”}|C)$$

t	P(t|d)
submarine	0,14
sea	0,04
...	

t	P(t|C)
submarine	0,0001
sea	0,0002
...	

Smoothing

- **Jelinek-Mercer smoothing**

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

  - Smoothing parameter is λ
  - Same amount of smoothing is applied to all documents
- **Dirichlet smoothing**

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

  - Smoothing parameter is μ
  - Smoothing is inversely proportional to the document length

Relation between Smoothing Methods

- Jelinek Mercer:

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

- by setting:

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

- Dirichlet:

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

Practical Considerations

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

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

$$\underbrace{\log P(q|d)}_{score(d,q)} = \sum_{t \in q} \underbrace{\log P(t|\theta_d)}_{w_{t,d}} \cdot \underbrace{f_{t,q}}_{w_{t,q}}$$

Exercise

Exercise

See 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	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										

Exercise

	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	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

	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	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										

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

Document language model computation

Exercise

	term frequencies					empirical language models					collection language	smoothed language models				
term	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	model	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										

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

Document language model computation

Exercise

	term frequencies					empirical language models					collection language	smoothed language models				
term	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	model	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										

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

Document language model computation

Exercise

	term frequencies					empirical language models					collection language	smoothed language models				
term	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	model	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

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

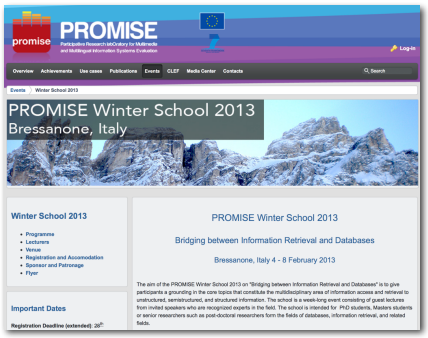
P(q="T2 T1"|D2) = P(T2|D2) * P(T1|D2)

Fielded Variants

Motivation

- Documents are composed of multiple fields
  - E.g., title, body, anchors, etc.
- Modeling internal document structure may be beneficial for retrieval

Example



Unstructured representation

```
PROMISE Winter School 2013
Bridging between Information Retrieval and Databases
Bressanone, Italy 4 - 8 February 2013
The aim of the PROMISE Winter School 2013 on "Bridging between Information
Retrieval and Databases" is to give participants a grounding in the core
topics that constitute the multidisciplinary area of information access and
retrieval to unstructured, semistructured, and structured information. The
school is a week-long event consisting of guest lectures from invited
speakers who are recognized experts in the field. The school is intended for
PhD students, Masters students or senior researchers such as post-doctoral
researchers form the fields of databases, information retrieval, and related
fields.
[...]
```

```
<html>
<head>
   <title>Winter School 2013</title>
   <meta name="keywords" content="PROMISE, school, PhD, IR, DB, [...]" />
   <meta name="description" content="PROMISE Winter School 2013, [...]" />
</head>
<body>
   <h1>PROMISE Winter School 2013</h1>
   <h2>Bridging between Information Retrieval and Databases</h2>
   <h3>Bressanone, Italy 4 - 8 February 2013</h3>
   <p>The aim of the PROMISE Winter School 2013 on "Bridging between
   Information Retrieval and Databases" is to give participants a grounding
   in the core topics that constitute the multidisciplinary area of
   information access and retrieval to unstructured, semistructured, and
   structured information. The school is a week-long event consisting of
   guest lectures from invited speakers who are recognized experts in the
   field. The school is intended for  PhD students, Masters students or
   senior researchers such as post-doctoral researchers form the fields of
   databases, information retrieval, and related fields. </p>
   [...]
</body>
</html>
```

Fielded representation

based on HTML markup

title: Winter School 2013

meta: PROMISE, school, PhD, IR, DB, [...]
PROMISE Winter School 2013, [...]

headings: PROMISE Winter School 2013
Bridging between Information Retrieval and Databases
Bressanone, Italy 4 - 8 February 2013

body: The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as post-doctoral researchers form the fields of databases, information retrieval, and related fields.

Fielded Extension of Retrieval Models

- BM25 => BM25F
- LM => Mixture of Language Models (MLM)

BM25F

- Extension of BM25 incorporating multiple fields
- The soft normalization and term frequencies need to be adjusted
- Original BM25:

$$score(d,q) = \sum_{t \in q} \frac{f_{t,d} \cdot (1+k_1)}{f_{t,d} + k_1 \cdot B} \cdot idf_t$$

where B is the soft normalization:

$$B = (1 - b + b\frac{|d|}{avgdl})$$

BM25F

$$score(d,q) = \sum_{t \in q} \frac{\tilde{f}_{t,d}}{k_1 + \tilde{f}_{t,d}} \cdot idf_t$$

Combining term frequencies across fields

$$\tilde{f}_{t,d} = \sum_i w_i \frac{f_{t,d_i}}{B_i}$$

Field weight

Soft normalization for field *i*

$$B_i = (1 - b_i + b_i \frac{|d_i|}{avgdl_i})$$

Parameter b becomes field-specific

Mixture of Language Models

- Build a separate language model for each field
- Take a linear combination of them

$$P(t|\theta_d) = \sum_i w_i P(t|\theta_{d_i})$$

Field weights $\sum_{j=1}^{m} w_j = 1$

Field language model
Smoothed with a collection model built from all document representations of the same type in the collection

Field Language Model

Smoothing parameter

$$P(t|\theta_{d_i}) = (1-\lambda_i)P(t|d_i) + \lambda_i P(t|C_i)$$

Empirical field model $\frac{f_{t,d_i}}{|d_i|}$

Maximum likelihood estimates

Collection field model $\frac{\sum_{d'} f_{t,d'_i}}{\sum_{d'} |d'_i|}$

Example

$q = \{\text{IR}, \text{winter}, \text{school}\}$
$\text{fields} = \{\text{title}, \text{meta}, \text{headings}, \text{body}\}$
$w = \{0.2, 0.1, 0.2, 0.5\}$

$$P(q|\theta_d) = \underbrace{P(\text{“IR”}|\theta_d)}_{P(\text{“IR”}|\theta_d)} \cdot P(\text{“winter”}|\theta_d) \cdot P(\text{“school”}|\theta_d)$$

$$\begin{aligned} P(\text{“IR”}|\theta_d) &= 0.2 \cdot P(\text{“IR”}|\theta_{d_{title}}) \\ &+ 0.1 \cdot P(\text{“IR”}|\theta_{d_{meta}}) \\ &+ 0.2 \cdot P(\text{“IR”}|\theta_{d_{headings}}) \\ &+ 0.5 \cdot P(\text{“IR”}|\theta_{d_{body}}) \end{aligned}$$

Exercise

Parameter Settings

Setting Parameter Values

- Retrieval models often contain parameters that must be tuned to get best performance for specific types of data and queries
- For experiments:
  - Use *training* and *test* data sets
  - If less data available, use *cross-validation* by partitioning the data into *K* subsets

Finding Parameter Values

- Many techniques used to find optimal parameter values given training data
  - Standard problem in machine learning
- In IR, often explore the space of possible parameter values by *grid search ("brute force")*
  - Perform a sweep over the possible parameter values of each parameter, e.g., from 0 to 1 in 0.1 steps

Query Processing

- Strategies for processing the data in the index for producing query results
- Document-at-a-time
  - Calculates complete scores for documents by processing all term lists, one document at a time
- Term-at-a-time
  - Accumulates scores for documents by processing term lists one at a time
- Both approaches have optimization techniques that significantly reduce time required to generate scores

Document-at-a-Time

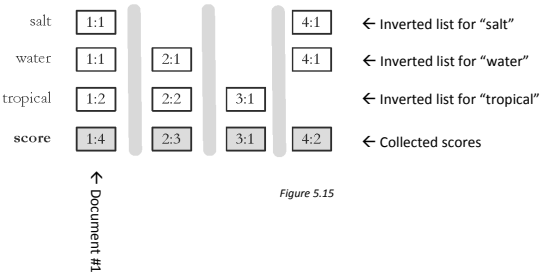


Figure 5.15

Term-at-a-Time

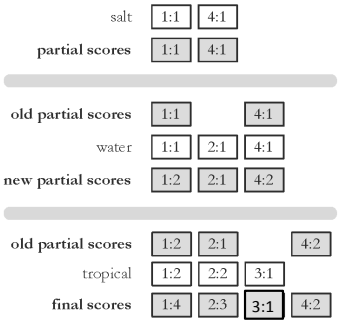


Figure 5.17