Processamento e Recuperação de Informação

Information Retrieval Models

Departamento de Engenharia Informática
Instituto Superior Técnico

1º Semestre
2018/2019

Outline

Bibliography

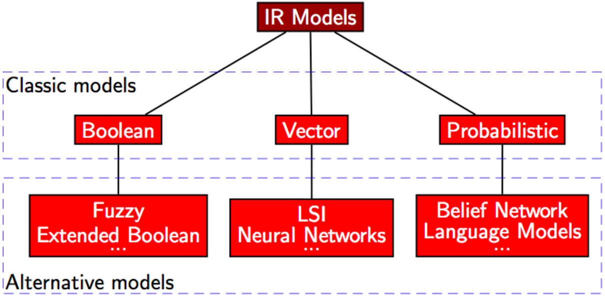
Ricardo Baeza-Yates, Berthier Ribeiro-Neto, Modern Information Retrieval, 2nd edtion. Chapter 3

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval. Chapters 1, 6, 11 and 12

Bing Liu, Web Data Mining - Exploring Hyperlinks, Contents, and Usage Data. Chapter 6.

Outline

Retrieval Models



Index Terms

In the classic IR models, documents are represented by index terms

- full text/selected keywords
- structure/no structure

Not all terms are equally useful

- index terms can be weighted

We assume that terms are mutually independent

- this is, of course, a simplification

Definition of a Document Model

Definition

Let t be the number of index terms in the collection of documents, and let k_i be a generic index term.

- $K = \{k_1, \ldots, k_t\}$ is the set of all index terms.
- A weight $w_{i,j} \geq 0$ is associated with each index term k_i of a document d_j.
- For an index term which does not appear in the document text, $w_{i,j} = 0$.
- Each document d_j is associated a term vector $\vec{d_j}$, represented by $\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$.
- Function $g_i(\vec{d_j})$ returns the weight of index term k_i in vector $\vec{d_j}$.

Outline

Boolean Model Queries

Processamento e Recuperação de Informação

Generic Document Model

The Boolean Model

The Vector Space Model

Probabilistic Models

Comparison of the Different Models

- Follows Boolean algebra syntax and semantics
- Term weights are binary
  - $w_{i,j} \in \{0, 1\}$
  - $w_{i,j} = 1$ — term present,
  - $w_{i,j} = 0$ — term not present
- Queries are Boolean expressions
  - E.g., $q = k_a \wedge (k_b \vee \neg k_c)$
- Documents are considered relevant if the query evaluates to *1* (true)

An Example

d_1

That government is best which governs least

d_2

That government is best which governs not at all

d_3

When men are prepared for it, that will be the kind of government which they will have

An Example

d_1

That government is best which governs least

d_2

That government is best which governs not at all

d_3

When men are prepared for it, that will be the kind of government which they will have

$q = \text{government} \wedge \text{best}$

answer: d_1, d_2

An Example

d_1

That government is best which governs least

d_2

That government is best which governs not at all

d_3

When men are prepared for it, that will be the kind of government which they will have

$q = \text{government} \wedge \text{best}$

answer: d_1, d_2

$q = \text{government} \wedge \text{best} \wedge \neg \text{all}$

answer: d_1

An Example

d_1

That government is best which governs least

d_2

That government is best which governs not at all

d_3

When men are prepared for it, that will be the kind of government which they will have

$q =$ government $\wedge$ best

answer: d_1, d_2

$q =$ government $\wedge$ best $\wedge \neg$ all

answer: d_1

$q =$ government $\vee$ best $\wedge \neg$ all

answer: d_1, d_2, d_3

Document-Query Similarity

- Queries can be translated to a disjunction of conjunctive vectors

$$\vec{q} = k_a \wedge (k_b \vee \neg k_c) \Leftrightarrow (1,1,1) \vee (1,1,0) \vee (1,0,0)$$

each tuple corresponds to a vector (k_a, k_b, k_c)

- Similarity of a document to a query is defined as:

$$sim(d_j, q) = \begin{cases} 1 & \text{if } \exists \vec{q_c} \in \vec{q} | \forall_i, g_i(\vec{d_j}) = g_i(\vec{q_c}) \\ 0 & \text{otherwise} \end{cases}$$

The Boolean Model

Why is it good?

- Simple model based on Boolean algebra
- Intuitive concept
- Precise semantics
- Clear formal basis
- Widely adopted by early information systems

Boolean Model

Limitations:

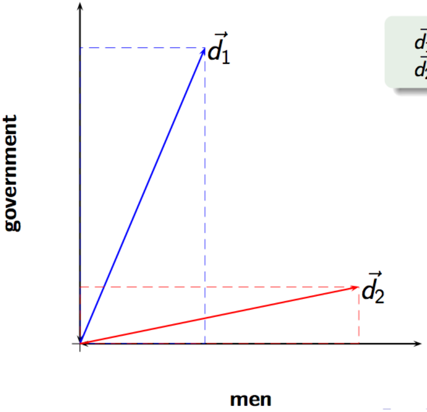
- Retrieval based only on binary decisions
  - More similar to *data retrieval* than *information retrieval*
  - Can retrieve too many, or too little documents
  - Some documents may be more relevant than others
- How do you translate a query to a Boolean expression?
  - Non-expert users may not be able to represent their information needs using Boolean expressions
- Terms are all equally important
  - Index term weighting can bring great improvements in performance

Outline

Documents as Vectors

- Documents are represented as vectors
  - $\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$
  - $w_{i,j}$ is the weight of term i in document j
- Queries are also vectors
  - $\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{t,q})$
- Vector operations can be used to compare queries$\times$documents (or documents$\times$documents)

An Example



$$\vec{d_1} = (2.2, 5.2)$$
$$\vec{d_2} = (4.9, 1.0)$$

Defining Document Vectors

Two questions are still unanswered:

1. How do we define term weights?
2. How do we compare documents to queries?

Defining Term Weights — TF

Term frequency

Term frequency is a measure of term importance within a document

Definition

Let N be the total number of documents in the system and n_i be the number of documents in which term k_i appears. The normalized frequency of a term k_i in document d_j is given by:

$$f_{i,j} = \frac{freq_{i,j}}{\max_l freq_{l,j}}$$

where $freq_{i,j}$ is the number of occurrences of term k_i in document d_j.

Defining Term Weights — IDF

(Inverse) Document frequency

Document frequency is a measure of term importance within a collection

Definition

The inverse document frequency of a term k_i is given by:

$$idf_i = \log\left(\frac{N}{n_i}\right)$$

Defining Term Weights — TF-IDF

Definition

The weight of a term k_i in document d_j for the vector space model is given by the tf-idf formula:

$$w_{i,j} = f_{i,j} \times \log\left(\frac{N}{n_i}\right)$$

Components of TF-IDF

Different TF-IDF formulations consider alternative approaches for the TF and IDF components, and also for normalizing the resulting vectors.

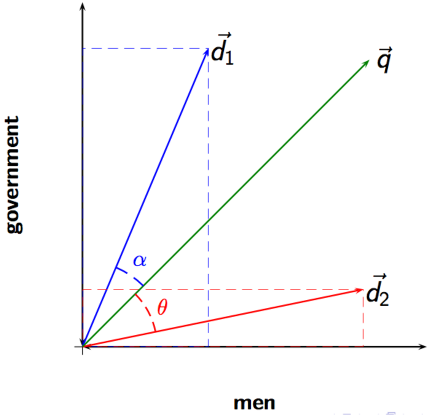
Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Document Similarity

- Similarity between documents and queries is a measure of the correlation between their vectors
- Documents/queries that share the same terms, with similar weights, should be more similar
- Thus, as similarity measure, we use the cosine of the angle between the vectors

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

An Example



$\cos(\alpha) = 0.9$

$\cos(\theta) = 0.8$

The Vector Space Model

Why is it so good?

- Simple model, based on linear algebra
- Term weights are not binary
- Allows computing a continuous degree of similarity between queries and documents
- Thus, allows ranking documents according to their possible relevance

Improving the VSM (1)

The BM25 Model

Consider not only the term frequency and inverse document frequency heuristics, but also the document length as a normalization factor for the term frequency

$$TF_{i,j} = \frac{f_{i,j} \times (k_1 + 1)}{f_{i,j} + k_1 \times \left(1 - b + b\frac{|d_j|}{avgdl}\right)}$$

$$IDF_i = \log \frac{N - n_i + 0.5}{n_i + 0.5}$$

$$sim(d_j, q) = \sum_{i \in q} IDF_i \times TF_{i,j}$$

To be detailed in the next lecture

Improving the VSM (2)

Latent Semantic Indexing

- Find a low-rank approximation of the matrix which describes the occurrences of terms in documents
  - Singular Value Decomposition
  - Compare the documents in the low-dimensional space
- The consequence of the rank lowering is that some dimensions are combined (e.g., mitigates the problem of identifying synonymy)
- To be detailed latter in the course

Outline

Probabilistic Models for IR

TF-IDF and VSM produces sufficiently good results in practice, but often criticized for being too ad-hoc or not principled

- Typically outperformed by probabilistic retrieval models and statistical language models in IR benchmarks
- Probabilistic retrieval models
  - use generative models of documents as bags-of-words
  - explicitly model probability of relevance $P(R|d_j, q)$
  - probabilistic justification for TF-IDF-like approaches
- Statistical language models
  - use generative models of documents and queries as sequences-of-words
  - consider likelihood of generating query from document model or divergence of document model and query model

Probabilistic Retrieval Models

Bayes optimal decision rule for set retrieval

$$d_j \text{ is relevant iff } P(R|d_j, q) > P(\overline{R}|d_j, q)$$

- Model the IR problem in a probabilistic framework
- Estimate the probability of document d_j being relevant to the user that that submited the query q
- When considering ranked retrieval, present documents in decreasing order of their estimated probability of relevance

Binary Independence Model (BIM)

Simplifying assumptions for $P(R|d_j, q)$

- A simple probabilistic model can assume that:
  1. probability depends only on query and document
  2. there is a subset R of relevant documents
  3. index terms are independent
  4. non-query terms are equally likely to appear in relevant and non-relevant documents
- Use binary term weights
  - documents and queries as binary term incidence vectors
  - terms not appearing in the query do not affect the ranking

Document Query Similarity

- As a similarity measure, we can use the ratio between the probability of finding the relevant documents and the probability of finding the non-relevant documents

$$sim(d_j, q) = \frac{P(R|\vec{d_j}, \vec{q})}{P(\overline{R}|\vec{d_j}, \vec{q})}$$

- Often refered to as the Retrieval Status Value (RSV)

Similarity Probabilities (1)

Initial Equation

We can simplify the expression leveraging Bayes' theorem and rank equivalence (i.e., remove query-independent constants)

$$sim(d_j, q) = \frac{P(R|\vec{d_j}, \vec{q})}{P(\overline{R}|\vec{d_j}, \vec{q})} = \frac{P(\vec{d_j}, \vec{q}|R) \times P(R)}{P(\vec{d_j}, \vec{q}|\overline{R}) \times P(\overline{R})} \sim \frac{P(\vec{d_j}, \vec{q}|R)}{P(\vec{d_j}, \vec{q}|\overline{R})}$$

Assuming term independence...

$$sim(d_j, q) \sim \frac{(\Pi_{g_i(\vec{d_j})=g_i(\vec{q})=1} P(k_i|R)) \times (\Pi_{g_i(\vec{d_j})=0 \wedge g_i(\vec{q})=1} P(\overline{k_i}|R))}{(\Pi_{g_i(\vec{d_j})=g_i(\vec{q})=1} P(k_i|\overline{R})) \times (\Pi_{g_i(\vec{d_j})=0 \wedge g_i(\vec{q})=1} P(\overline{k_i}|\overline{R}))}$$

Similarity Probabilities (2)

Taking logs and removing constant factors...

$$sim(d_j, q) = \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

Blind assumptions

$$\begin{aligned} P(k_i|R) &= 0.5 \\ P(k_i|\overline{R}) &= \frac{n_i}{N} \end{aligned}$$

- $P(k_i|R)$ reflects that we have no information about relevant documents (i.e., each query term is equally likely to occur in a relevant document)
- $P(k_i|\overline{R})$ reflects a much smaller number of relevant documents than the collection size

Similarity Probabilities (3)

After document retrieval or leveraging training data...

- Let V be the number of returned documents (i.e., number of documents estimated to be relevant)
- Let V_i be the number of returned docs with term k_i

$$P(k_i|R) = \frac{V_i}{V}$$
$$P(k_i|\overline{R}) = \frac{n_i - V_i}{N - V}$$

Similarity Probabilities (4)

Avoiding small values when using blind assumptions...

- With zero counts probability is not well-defined
- Maximum likelihood estimates do not work for rare events
- To avoid zeros add 0.5 to each count (expected likelihood estimation) or use a different type of smoothing

$$
\begin{aligned}
P(k_i|R) &= 0.5 \\
P(k_i|\overline{R}) &= \frac{n_i+0.5}{N+1}
\end{aligned}
$$

Avoiding small values with estimates after retrieval...

$$
\begin{aligned}
P(k_i|R) &= \frac{V_i+\frac{n_i}{N}}{V+1} \\
P(k_i|\overline{R}) &= \frac{n_i-V_i+\frac{n_i}{N}}{N-V+1}
\end{aligned}
$$

Problems of this Simple Probabilistic Model

- There is no accurate estimate for the first run probabilities
- Index terms are not weighted
- Terms are assumed mutually independent

In fact, many different probabilistic retrieval models have been proposed, some addressing the aforementioned limitations!

Another Look at the BIM (1)

Recall the log odds ratio for computing RSV

$$sim(d_j, q) = \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

Denoting $p_i = P(k_i|R)$ and $u_i = P(k_i|\overline{R})$

$$sim(d_j, q) = \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{p_i}{1 - p_i} + \log \frac{1 - u_i}{u_i} \right)$$

Another Look at the BIM (2)

With the blind estimates, does the equation look familiar?

$$P(k_i|R) = p_i = 0.5$$

$$P(k_i|\overline{R}) = u_i = \frac{n_i}{N}$$

Replacing p_i and u_i in the previous equation...

$$\log \frac{p_i}{1 - p_i} = 0$$

$$\log \frac{1 - u_i}{u_i} = \log \frac{N - n_i}{n_i} \approx \log \left(\frac{\mathbf{N}}{\mathbf{n_i}} \right)$$

Another Look at the BIM (3)

- The BIM can be seen as TF-IDF with binary term frequencies and logarithmically dampened inverse document frequencies
- The score for document d_j is just IDF weighting of the query terms present in the document

$$sim(d_j, q) = \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \log\left(\frac{N}{n_i}\right)$$

- Alternative formulation using smoothing

$$sim(d_j, q) = \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \log\left(\frac{N - n_i + 0.5}{n_i + 0.5}\right)$$

The Okapi BM25 Model (1)

- Inspired by the BIM probabilistic formulation
- Considering an alternative for term weighting
- Captures various aspects in a simple formula, tuning each component
    - Inverse Document Frequency (IDF)
    - Term Frequncy (TF)
    - Document length
    - Query term fequency *(in some formulations)*
- BM25 (BestMatch25) is an effective and widely used model for full-text retrieval over large collections

The Okapi BM25 Model (2)

The BM25 Model

$$TF_{i,j} = \frac{f_{i,j} \times (k_1 + 1)}{f_{i,j} + k_1 \times \left(1 - b + b\frac{|d_j|}{avgdl}\right)}$$

$$IDF_i = \log \frac{N - n_i + 0.5}{n_i + 0.5}$$

$$sim(d_j, q) = \sum_{i \in q} IDF_i \times TF_{i,j}$$

- Postulates Poisson (or 2-Poisson-mixture) distributions for terms, instead of Binomial distributions as in BIM
- Parameters k_1 and b need to be tuned
  - k_1 controls impact of term frequency
  - b controls impact of document length
  - Setting $k_1 = 1.5$ and $b = 0.75$ are common defaults

Extending BM25 to Consider Document Fields

- Textual data often found in some sort of structural form
- Retrieval effectiveness can be improved by taking the structure into account
- Simple solution: calculate score for each field and combine the different fields linearly

$$sim(d_j, q) = \sum_{z \in F} \alpha^z \times sim(d_j^z, q)$$

Problems of linear combination

- With similarities per field, IDF can vary highly in different fields (e.g. stopwords scoring highly in the title)
- TF usually non-linear and information gained by observing a term for the first time is greater than observing subsequent occurrences

BM25F and Combining Term Frequencies

$$TF_{i,j} = \sum_{z \in F} \alpha^z \times \frac{f_{i,j}^z \times (k_1 + 1)}{f_{i,j}^z + k_1 \times \left(1 - b^z + b^z \frac{|d_j^z|}{avgdl^z}\right)}$$

$$IDF_i = \log \frac{N - n_i + 0.5}{n_i + 0.5}$$

$$sim(d_j, q) = \sum_{i \in q} IDF_i \times TF_{i,j}$$

Probabilistic Language Models

- Another simple probabilistic retrieval formulation
- Each document d is treated as (the basis for) a probabilistic language model
- Given a query q rank documents based on $P(d|q)$

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

- The evidence $P(q)$ is the same for all documents, so ignore
- $P(d)$ is the prior
  - often treated as the same for all d
  - we can give a higher prior to "high-quality" documents (e.g., those with high PageRank – to be seen latter)
- $P(q|d)$ is likelihood, i.e. the probability of q given d

How to compute $P(q|d)$

- Conditional independence assumption

$$P(q|d) = P(\{t_1, \ldots, t_{|q|}\}|d) = \prod_{1<=i<=|q|} P(t_i|d)$$

- $|q|$ is length of q
- t_i is the token occurring at position i in q

- The above multinomial model is equivalent to:

$$P(q|d) = \prod_{\text{distinct term } t \in q} P(t|d)^{TF_{t,q}}$$

- Component $TF_{t,q}$ is the term frequency of t in q
- Parameters $P(t|d)$ computed through maximum likelihood estimates

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

Types of Language Models

- The unigram language model (show before)

$$P(d) = P(t_1 t_2 t_3 \ldots) = P(t_1)P(t_2)P(t_3)\cdots$$

- *n*-gram language models (e.g. *bigram* language models)

$$P(d) = P(t_1 t_2 t_3 \ldots) = P(t_1)P(t_2|t_1)P(t_3|t_2)\cdots$$

- More complex langue models, e.g. using *probabilistic context-free grammars*
  - Used for tasks like speech recognition, spelling correction, and machine translation

LM Retrieval and Naïve Bayes

The next class will introduce a simple probabilistic document classifyer, known as the Naïve Bayes approach

- We want to classify document d.
  We want to classify a query q
- Human-defined classes: e.g., politics, economics, sports.
  Each document in the collection is a different class
- Assume that d was produced by the generative model.
  Assume that q was generated by a generative model
- Which of the classes (= class models) is most likely to have generated the document d?
  Which document (=class) is most likely to have generated the query q?
- For which class do we have the most evidence?
  For which document (as source for query) do we have the most evidence?

More on computing $P(q|d)$

Problems with the aforementioned unigram model

$$P(q|d) = P(\{t_1, \ldots, t_{|q|}\}|d) = \prod_{1<=i<=|q|} P(t_i|d)$$

- A document with a single missing query-term will receive a score of zero (similar to Boolean AND)
- Where is the equivalent of the IDF?

Linear interpolation smoothing

$$P(q|d) = \prod_{1<=i<=|q|} (\alpha \times P(t_i|d)) + ((1-\alpha) \times P(t_i|c))$$

Query Likelihood Retrieval with Linear Interpolation Smoothing

- Helps us avoid zero-probabilities
- Another added benefit...
  - Without smoothing, the query-likelihood model ignores how frequently the term occurs in general
  - Linear interpolation smoothing also introduces an IDF-like scoring of documents

$$
\begin{aligned}
P(q|d) &= \prod_{1<=i<=|q|} (\alpha \times P(t_i|d)) + ((1-\alpha) \times P(t_i|c)) \\
&= \prod_{1<=i<=|q|} ((\alpha \times P(t_i|d)) + ((1-\alpha) \times P(t_i|c))) \times \left(\frac{(1-\alpha)\times P(t_i|c)}{(1-\alpha)\times P(t_i|c)}\right) \\
&= \prod_{1<=i<=|q|} \left(\frac{\alpha\times P(t_i|d)}{(1-\alpha)\times P(t_i|c)} + 1\right) \times (1-\alpha) \times P(t_i|c) \\
&= \prod_{1<=i<=|q|} \left(\frac{\alpha\times P(t_i|d)}{(1-\alpha)\times P(t_i|c)} + 1\right) \times \prod_{1<=i<=|q|} (1-\alpha) \times P(t_i|c) \\
&\approx \prod_{1<=i<=|q|} \left(\frac{\alpha\times P(t_i|d)}{(1-\alpha)\times P(t_i|c)} + 1\right)
\end{aligned}
$$

Outline

What makes these Models Work?

Three main term weighting normalization driving features:

- TF - Term Frequency
- IDF - Inverse Document Frequency
- DL - Document Length

Comparison of the Different Models

- Boolean model is considered the weakest
- There is some controversy over which shows better performance: vector space or probabilistic
  - Simple BIM is just IDF weighting of the terms
  - BIM originally designed for short catalog records of fairly consistent length, working reasonably in these contexts
  - BM25 or language models offer a better performance (e.g., paying attention to term frequency and document length)
- Nowadays, BM25 is perhaps the most widely used

Questions?