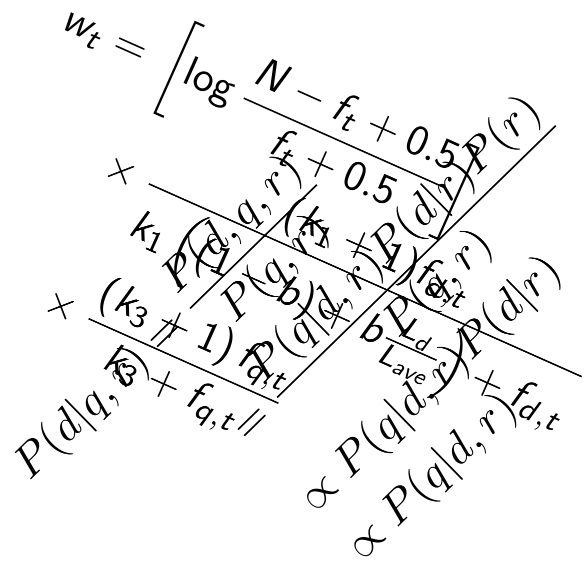


COMP90042 LECTURE 17

RETRIEVAL USING BM25 AND LANGUAGE MODELS

OVERVIEW

- Two leading methods for modern IR
  - BM25
  - Language models
- Inspired by nice theory and highly effective

RECAP: VECTOR SPACE MODEL

- Document are bag-of-words, represented as TF*IDF vectors and normalised
- Queries represented as binary term occurrence vectors
- Cosine measure of similarity between a query and document
- Efficient algorithm for finding ranked list of documents by cosine score

OKAPI BM25

- Why not try other forms of term weighting than TF*IDF?
- Can we capture various aspects in simple formula
  - idf
  - tf
  - document length
  - query tf
- Then seek to tune each component

OKAPI BM25

$$
\begin{aligned}
w_t = & \left[\log \frac{N - f_t + 0.5}{f_t + 0.5}\right] && \text{(idf)} \\
& \times \frac{(k_1 + 1) f_{d,t}}{k_1 \left((1 - b) + b \frac{L_d}{L_{ave}}\right) + f_{d,t}} && \text{(tf and doc. length)} \\
& \times \frac{(k_3 + 1) f_{q,t}}{k_3 + f_{q,t}} && \text{(query tf)}
\end{aligned}
$$

- Parameters k_1, b, k_3 need to be tuned
  - defaults $k_1 = 1.5$, $b = 0.5$, $k_3 = 0$
- BM25 most widely used method in IR

IDF COMPONENT

- Slight difference to standard IDF
  what happens when doc freq, f_t, nears N?

$$\left[\log \frac{N - f_t + 0.5}{f_t + 0.5}\right]$$

- Inspired by the *Binary Independence Model*
  - frames retrieval as *ranking* by probability of relevance, *P(R = 1 | d, q)* where *R = 0* or *1* is (ir-)relevance, *d* = document, *q* = query
  - various simplifying assumptions to make practical (and avoid the need for manual *relevance feedback*)

DOCUMENT TF COMPONENT

- Next component is based on TF

$$\frac{(k_1 + 1) f_{d,t}}{k_1 \left((1 - b) + b \frac{L_d}{L_{ave}} \right) + f_{d,t}}$$

- Consider what happens at extrema
  - $k_1 = 0$ or $k_1 \rightarrow \infty$
  - b = 0 … 1
- b controls length based term to *reward high* frequency terms in *shorter* documents

QUERY TF COMPONENT

$$\frac{(k_3 + 1)\, f_{q,t}}{k_3 + f_{q,t}}$$

- Sometimes we have long form queries
  - E.g., sentences, paragraphs or documents (‘documents like this one’)
- Repeated terms in query might be important
  - tuneable parameter k_3 modulates between *binary* occurrence and query frequency count

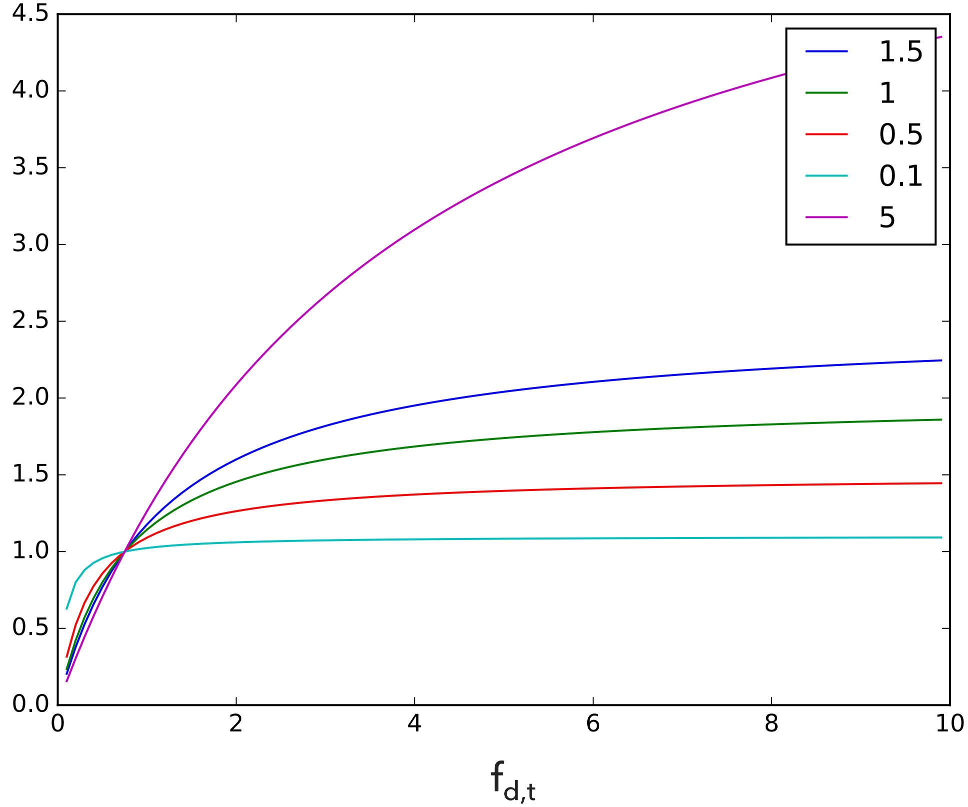
ROLE OF K_1

with b=0.5

BM25

score

(tf part)



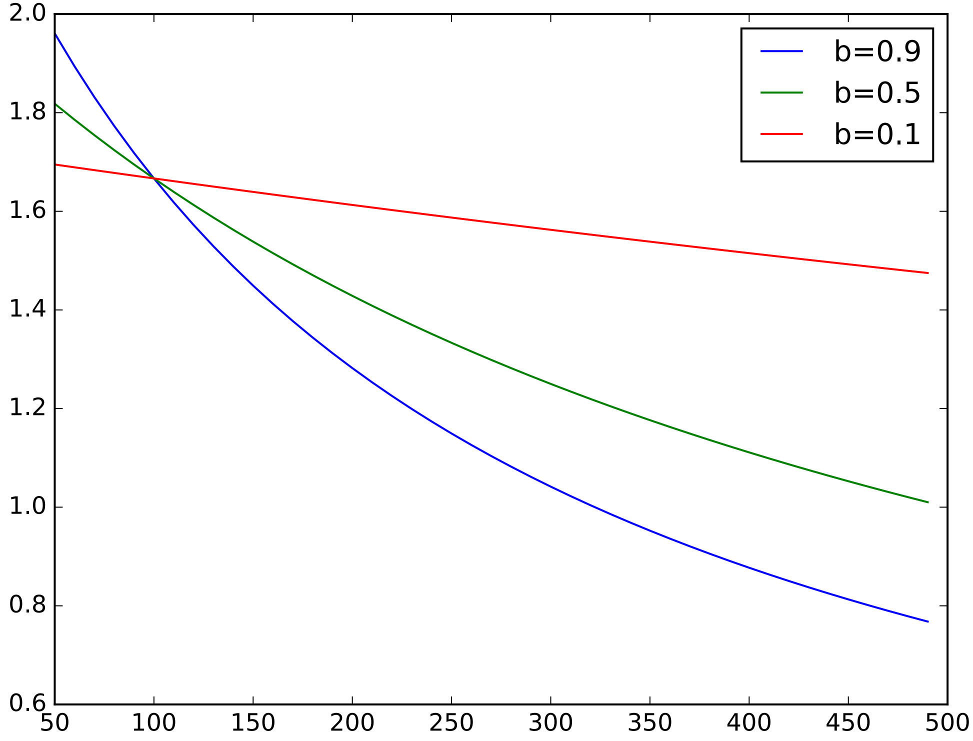
ROLE OF B

with k_1 = 1.5, l_{ave} = 100, $f_{t,d}$ = 3

BM25

score

(tf part)



L_d

LANGUAGE MODELS FOR IR

- Alternative probabilistic approach to IR
  - compelling theory
  - highly effective
- Probabilistic IR (motivating BM25)

$$P(R = 1|q, d)$$

- Language model

$$P(q|d)$$

LANGUAGE MODELS

- Recap: assigns a probability to a sequence of tokens
  - uses a Markov assumption
  - parameterised by simple token frequencies in training data
  - use careful smoothing / backoff to deal with low counts and unseen events
- Seen before for NLP, where we typically
  - train *high order* LM over large *corpus*
  - apply to *sentence* to find its probability, sample the next word, etc.

LANGUAGE MODELS IN IR

- Estimate the probability of a ***query*** given ***LM over document***

$$P(q|d) = \prod_{t \in q} P(t|d)$$

- where *P(t|d)* is a *unigram* language model trained on document *d*
- E.g., maximum likelihood estimate

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

  - where L_d is the length in words of document
- Finally, rank documents by decreasing *P(q|d)*

INTUITION

- *P(q|d)* asks:
  - *How likely is that the model that generated the document, also generated the query?*
- Understood as searcher behaviour:
  - Searcher told (or learns) to build queries using words likely to occur in relevant documents
  - Thus, their query attempts to approximate language of relevant documents
  - Testing against document language models then reasonable

PROBABILISTIC FORMULATION

- Consider probability of document, given q (query) and r (binary relevance)

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

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

Drop document independent values (irrelevant to ranking)

$$\propto P(q|d,r)P(d|r)$$

$$\propto P(q|d,r)$$

Assume uniform prior for P(d|r)

LANGUAGE MODELS IN IR VS NLP

- For retrieval we have a language model **per document**
  - versus a single language model of a **corpus** in NLP
- Use simple **unigram** language model, i.e., bag-of-words
  - versus **high order** language models to capture word order
- Apply several different LMs to a **single query**
  - versus a single LM applied to **several sentences**

SMOOTHING

- Terms appear sparsely in documents
  - MLE for unseen terms results in $P(t|d) = 0$
  - LM gives query non-zero probability if all query terms appear in d; effectively a **conjunction** querying mechanism
  - also problems with poor probability estimates for low count terms (e.g., tf=1)

SMOOTHING (CONT.)

- Use smoothing to address these problems
- Combine document-specific LM with LM over whole corpus, *P(t), e.g.,*
  - using interpolation

$$P(q|d) = \kappa_q \prod_{t \in q} \left(\lambda \frac{f_{t,d}}{L_d} + (1-\lambda) P(t) \right)$$

  - or Dirichlet smoothing

$$P(q|d) = \kappa_q \prod_{t \in q} \frac{f_{t,d} + \alpha P(t)}{L_d + \alpha}$$

- N.b., Kappa is constant, and can be omitted

INDEXING AND QUERYING WITH LM-IR

- Need to index various values
  - term frequencies, $f_{d,t}$
  - document lengths, L_d (in words)
  - unigram language model over complete corpus, $P(t)$
- TF and lengths stored in inverted index, as in the VSM
- Querying can be performed similar to before
  - See Moffat & Zobel, 2006, "*Inverted files for text search engines*" for details

EXAMPLE

- Step 1: estimate corpus language model *P(t)*

p(two)	p(tea)	p(me)	p(you)
1/6	1/3	1/4	1/4

doc1	Two for tea and tea for two
doc2	Tea for me and tea for you
doc3	You for me and me for you

- Step 2: estimate document LMs *P(t|d)* (setting alpha = 0.5)
  - E.g., p(two|d) = (2 + 0.5 * 1/6) / (4 + 0.5)
    = 0.463

	p(two|d)	p(tea|d)	p(me|d)	p(you|d)
doc1	0.463	0.481	0.028	0.028
doc2	0.019	0.481	0.25	0.25
doc3	0.019	0.037	0.472	0.472

QUERYING

- For q="tea you"
  - p(q|d=1) = p(tea | d=1) p(you | d=1)
    = 0.481 x 0.028 = 0.014
  - p(q|d=2) = p(tea | d=2) p(you | d=2)
    = 0.481 x 0.25 = **0.120**
  - p(q|d=3) = p(tea | d=3) p(you | d=3)
    = 0.037 x 0.472 = 0.017

	p(two|d)	p(tea|d)	p(me|d)	p(you|d)
doc1	0.463	0.481	0.028	0.028
doc2	0.019	0.481	0.25	0.25
doc3	0.019	0.037	0.472	0.472

RELATION TO TF*IDF

- Consider log probability with Dirichlet smoothed LM

$$\log P(q|d) = \sum_{t \in q} \log(f_{t,d} + \alpha P(t)) - \log(L_d + \alpha)$$

- Components are
  - log term frequency; and
  - a form of document length normalisation
- For rare words in collection αP(t) is small, so value of $f_{t,d}$ becomes more important in ranking (similar effect to IDF)

SOFTWARE

- Various toolkits implement optimised versions of BM25 and LMs
  - Lemur http://www.lemurproject.org
  - Terrier http://terrier.org/
  - Apache Lucene http://lucene.apache.org/

SUMMARY

- BM25 scoring formula for VSM IR
- Language models for IR
- Reading
  - MRS 11.4.3 “Okapi BM25: A non-binary model”
  - MRS Ch12 “Language models for information retrieval” (mainly 12.2)