

$$w_t = \left[\log \frac{N - f_t + 0.5}{f_t + 0.5} P(r) \right. \\
+ \frac{k_1 P(q, a, r) + 0.5 P(d/r)}{P(q, a, r) + 0.5 P(d/r)} \\
+ \frac{(k_3 + 1) P(q, b, r) + b P(d/r)}{P(q, b, r) + b P(d/r)} \\
\left. \frac{P(d/q, r) + f_{q,t}}{P(d/q, r) + f_{q,t}} \right] \\
\propto P(q/d, r) + f_{d,t} \\
\propto P(q/d, r) + f_{d,t}$$

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 $TF \cdot IDF$?
- ▶ Can we capture various aspects in simple formula
 - ▶ idf
 - ▶ tf
 - ▶ document length
 - ▶ query tf
- ▶ Then seek to tune each component

OKAPI BM25

$$w_t = \left[\log \frac{N - f_t + 0.5}{f_t + 0.5} \right] \quad (\text{idf})$$

$$\times \frac{(k_1 + 1) f_{d,t}}{k_1 \left((1 - b) + b \frac{L_d}{L_{ave}} \right) + f_{d,t}} \quad (\text{tf and doc. length})$$

$$\times \frac{(k_3 + 1) f_{q,t}}{k_3 + f_{q,t}} \quad (\text{query tf})$$

- ▶ 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 df 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, $P(R = 1 \mid d, q)$ 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 when
 - ▶ $k_1 = 0$ or limit as $k_1 \rightarrow \infty$
 - ▶ $b = 0 \dots b = 1$
- ▶ b controls length based term to *reward high frequency terms in shorter documents*

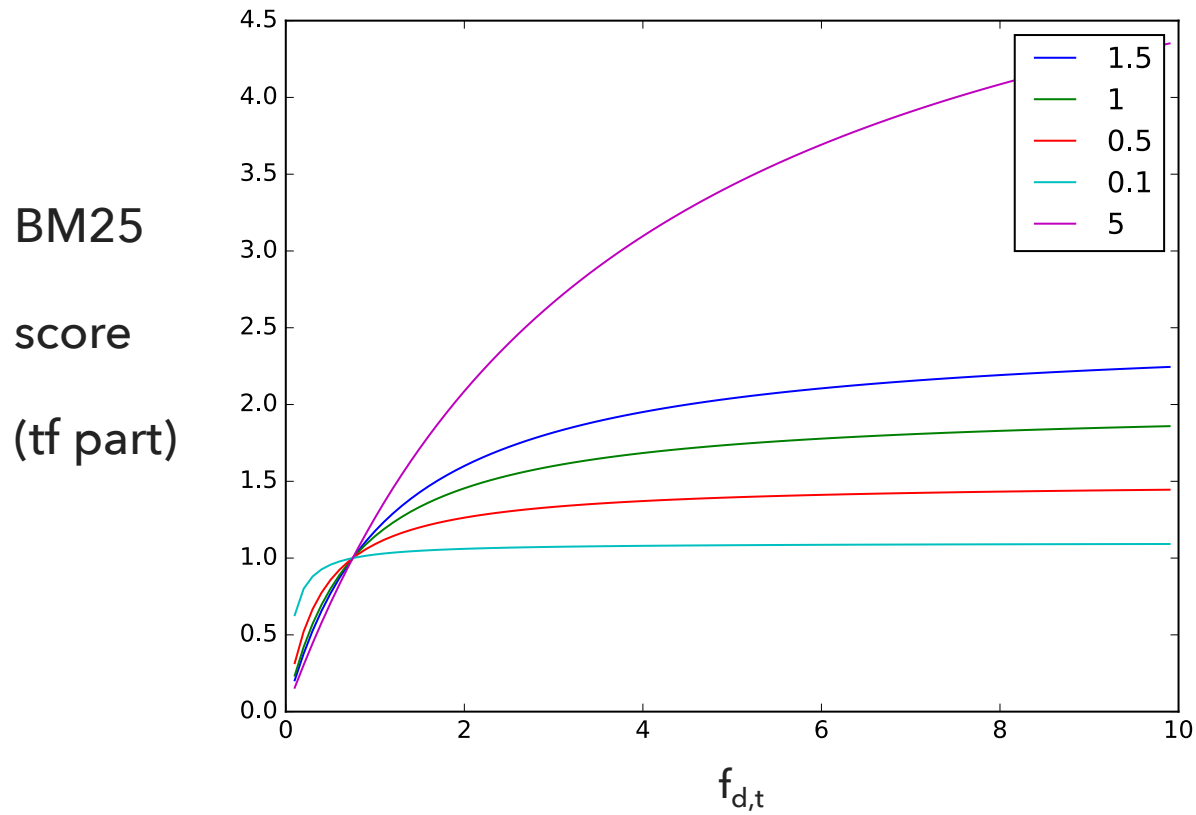
QUERY TF COMPONENT

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

- ▶ 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
 - ▶ consider $k_3 = 0$ and limit $k_3 \rightarrow \infty$

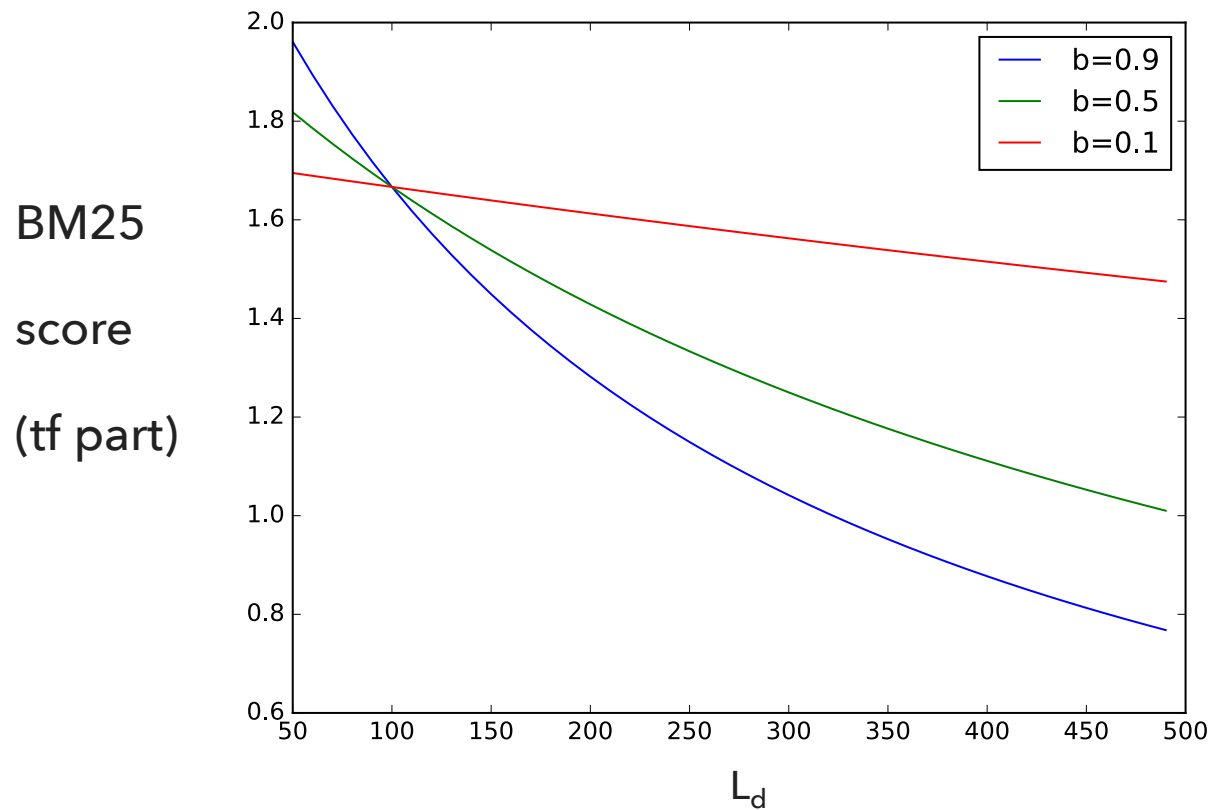
ROLE OF K_1

with $b=0.5$



ROLE OF B

with $k_1 = 1.5$, $l_{ave} = 200$, $f_{t,d} = 3$



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)

$$\begin{aligned}
 P(d|q, r) &= \frac{P(d, q, r)}{P(q, r)} \\
 &= \frac{P(q|d, r)P(d|r)P(r)}{P(q, r)} \\
 &\propto P(q|d, r)P(d|r) \\
 &\propto P(q|d, r)
 \end{aligned}$$

*Drop document
independent values
(irrelevant to ranking)*



*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., $tf=1$)
- ▶ Use smoothing to address these problems
 - ▶ combine document-specific LM with LM over whole corpus, $P(t)$, e.g., using interpolation or Dirichlet smoothing

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

INDEXING AND QUERYING WITH LM-IR¹⁸

- ▶ 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 as before
 - ▶ minor change to retrieval score computation

EXAMPLE

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

$p(\text{two})$	$p(\text{tea})$	$p(\text{me})$	$p(\text{you})$
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., $p(\text{two}|d) = (2 + 0.5 * 1/6) / (4 + 0.5)$
 $= 0.463$

	$p(\text{two} d)$	$p(\text{tea} d)$	$p(\text{me} d)$	$p(\text{you} d)$
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 $q = \text{"tea you"}$
 - ▶ $p(q|d=1) = p(\text{tea} | d=1) p(\text{you} | d=1)$
 $= 0.481 \times 0.028 = 0.014$
 - ▶ $p(q|d=2) = p(\text{tea} | d=2) p(\text{you} | d=2)$
 $= 0.481 \times 0.25 = \mathbf{0.120}$
 - ▶ $p(q|d=3) = p(\text{tea} | d=3) p(\text{you} | d=3)$
 $= 0.037 \times 0.472 = 0.017$

	$p(\text{two} d)$	$p(\text{tea} d)$	$p(\text{me} d)$	$p(\text{you} d)$
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 $\alpha P(t)$ 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)