Introduction to

Information Retrieval

Probabilistic Information Retrieval

Christopher Manning and Pandu Nayak

From Boolean to Ranked Retrieval

1. Why ranked retrieval?
2. Introduction to the classical probabilistic retrieval model and the probability ranking principle
3. The Binary Independence Model: BIM
4. Relevance feedback, briefly
5. The vector space model (VSM) (quick cameo)
6. BM25 model
7. Ranking with features: BM25F (if time allows …)

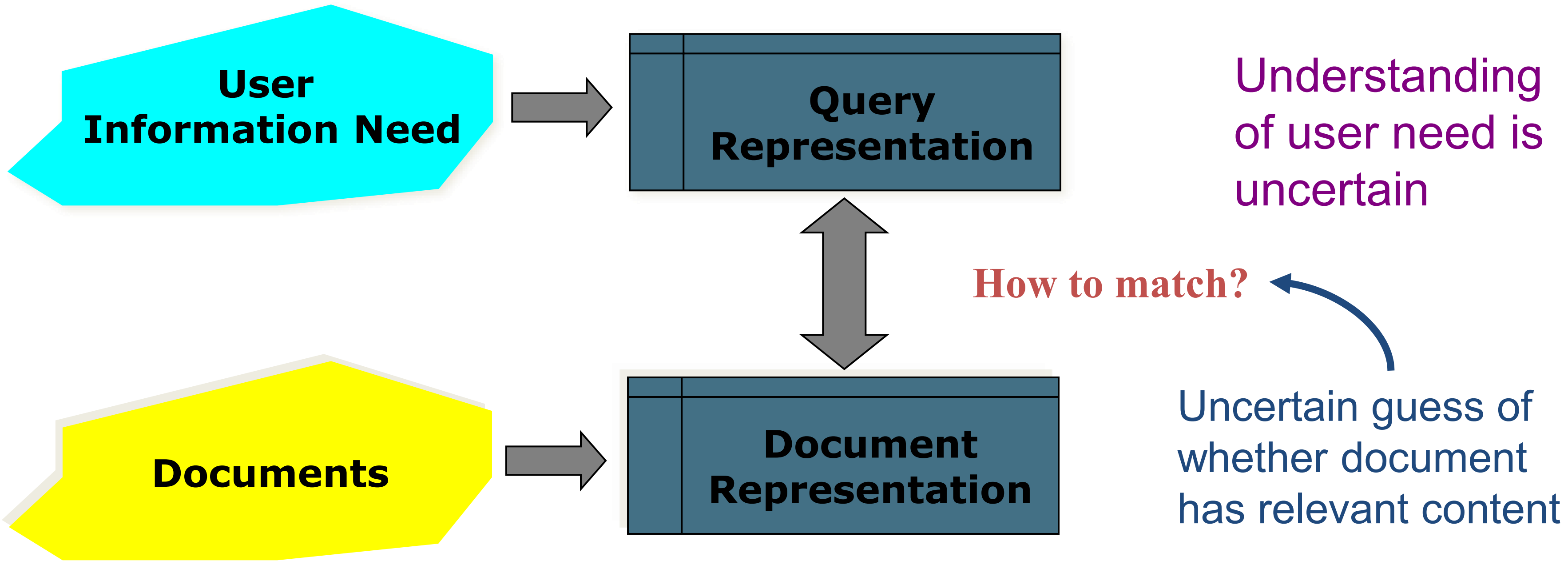
1. Ranked retrieval

- Thus far, our queries have all been Boolean
  - Documents either match or don’t
- Can be good for expert users with precise understanding of their needs and the collection
  - Can also be good for applications: Applications can easily consume 1000s of results
- Not good for the majority of users
  - Most users incapable of writing Boolean queries
    - Or they are, but they think it’s too much work
  - Most users don’t want to wade through 1000s of results
    - This is particularly true of web search

Problem with Boolean search: feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results
- Query 1: "*standard user dlink 650*" → 200,000 hits
- Query 2: "*standard user dlink 650 no card found*": 0 hits
- It takes a lot of skill to come up with a query that produces a manageable number of hits
  - AND gives too few; OR gives too many
- Suggested solution:
  - Rank documents by goodness – a sort of clever "soft AND"

2. Why probabilities in IR?


User Information Need
Query Representation
Understanding of user need is uncertain
How to match?
Documents
Document Representation
Uncertain guess of whether document has relevant content

In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our search uncertainties?

Probabilistic IR topics

1. Classical probabilistic retrieval model
   - Probability ranking principle, etc.
   - Binary independence model (≈ Naïve Bayes text cat)
   - (Okapi) BM25
2. Bayesian networks for text retrieval
3. Language model approach to IR (*IIR* ch. 12)
   - An important development in 2000s IR

Probabilistic methods are one of the oldest but also one of the currently hot topics in IR

- *Traditionally: neat ideas, but didn't win on performance*
- *It seems to be different now*

Who are these people?

Karen Spärck Jones

Stephen Robertson

Keith van Rijsbergen

The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is the core of modern IR systems:**
  - **In what order do we present documents to the user?**
  - We want the "best" document to be first, second best second, etc.
- **Idea: Rank by probability of relevance of the document w.r.t. information need**
  - $P(R=1|\text{document}_i, \text{query})$

The Probability Ranking Principle (PRP)

"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."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

Recall a few probability basics

- For events *A* and *B:*

$$p(A,B) = p(A \cap B) = p(A \mid B)p(B) = p(B \mid A)p(A)$$

- Bayes' Rule

Prior

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} = \frac{p(B \mid A)p(A)}{\sum_{X=A,\bar{A}} p(B \mid X)p(X)}$$

Posterior

- Odds:

$$O(A) = \frac{p(A)}{p(\bar{A})} = \frac{p(A)}{1 - p(A)}$$

The Probability Ranking Principle (PRP)

Let *x* represent a document in the collection.
Let *R* represent **relevance** of a document w.r.t. given (fixed) query and let **R=1** represent relevant and **R=0** not relevant.

Need to find $p(R=1|x)$ – probability that a document *x* is **relevant.**

$$p(R=1|x)=\frac{p(x|R=1)p(R=1)}{p(x)}$$

$p(R=1), p(R=0)$ - prior probability of retrieving a relevant or non-relevant document at random

$$p(R=0|x)=\frac{p(x|R=0)p(R=0)}{p(x)}$$

$p(x/R=1)$, $p(x/R=0)$ - probability that if a relevant (not relevant) document is retrieved, it is *x.*

$$p(R=0|x)+p(R=1|x)=1$$

Probabilistic Retrieval Strategy

- First, estimate how each term contributes to relevance
  - How do other things like term frequency and document length influence your judgments about document relevance?
    - Not at all in BIM
    - A more nuanced answer is given by BM25
- Combine to find document relevance probability
- Order documents by decreasing probability
- Theorem: Using the PRP is optimal, in that it minimizes the loss (Bayes risk) under 1/0 loss
  - Provable if all probabilities correct, etc. [e.g., Ripley 1996]

3. Binary Independence Model

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**: documents are represented as binary incidence vectors of terms (cf. IIR Chapter 1):
  - $\vec{x} = (x_1, \ldots, x_n)$
  - $x_i = 1$ iff term *i* is present in document *x*.
- **"Independence":** terms occur in documents independently
- Different documents can be modeled as the same vector

Binary Independence Model

- Queries: binary term incidence vectors
- Given query ***q***,
  - for each document ***d*** need to compute ***p*(*R*| *q*,*d*)**
  - replace with computing ***p*(*R*| *q*,*x*)** where ***x*** is binary term incidence vector representing ***d***
  - Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R \mid q, \vec{x}) = \frac{p(R=1 \mid q, \vec{x})}{p(R=0 \mid q, \vec{x})} = \frac{\dfrac{p(R=1 \mid q)\,p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid q)}}{\dfrac{p(R=0 \mid q)\,p(\vec{x} \mid R=0, q)}{p(\vec{x} \mid q)}}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = \frac{p(R=1 \mid q, \vec{x})}{p(R=0 \mid q, \vec{x})} = \underbrace{\frac{p(R=1 \mid q)}{p(R=0 \mid q)}}_{\text{Constant for a given query}} \cdot \underbrace{\frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)}}_{\text{Needs estimation}}$$

- Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)} = \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

- Since x_i is either 0 or 1:

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{x_i=1} \frac{p(x_i = 1 \mid R=1, q)}{p(x_i = 1 \mid R=0, q)} \cdot \prod_{x_i=0} \frac{p(x_i = 0 \mid R=1, q)}{p(x_i = 0 \mid R=0, q)}$$

- Let $p_i = p(x_i = 1 \mid R=1, q);\ r_i = p(x_i = 1 \mid R=0, q);$

- Assume, for all terms not occurring in the query $(q_i=0)$ $p_i = r_i$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{\substack{x_i=1 \\ q_i=1}} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i=0 \\ q_i=1}} \frac{(1-p_i)}{(1-r_i)}$$

	document	relevant (R=1)	not relevant (R=0)
term present	$x_i = 1$	p_i	r_i
term absent	$x_i = 0$	$(1 - p_i)$	$(1 - r_i)$

Binary Independence Model

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \underbrace{\prod_{x_i=q_i=1}}_{\text{All matching terms}} \frac{p_i}{r_i} \cdot \underbrace{\prod_{\substack{x_i=0\\q_i=1}}}_{\text{Non-matching query terms}} \frac{1-p_i}{1-r_i}$$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{\substack{x_i=1\\q_i=1}} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i=1\\q_i=1}} \left(\frac{1-r_i}{1-p_i} \cdot \frac{1-p_i}{1-r_i} \right) \cdot \prod_{\substack{x_i=0\\q_i=1}} \frac{1-p_i}{1-r_i}$$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \underbrace{\prod_{x_i=q_i=1}}_{\text{All matching terms}} \frac{p_i(1-r_i)}{r_i(1-p_i)} \cdot \underbrace{\prod_{q_i=1}}_{\text{All query terms}} \frac{1-p_i}{1-r_i}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = \boxed{O(R \mid q)} \cdot \boxed{\prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)}} \cdot \boxed{\prod_{q_i = 1} \frac{1-p_i}{1-r_i}}$$

Constant for each query

Only quantity to be estimated for rankings

Retrieval Status Value:

$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

Binary Independence Model
[Robertson & Spärck-Jones 1976]

All boils down to computing RSV.

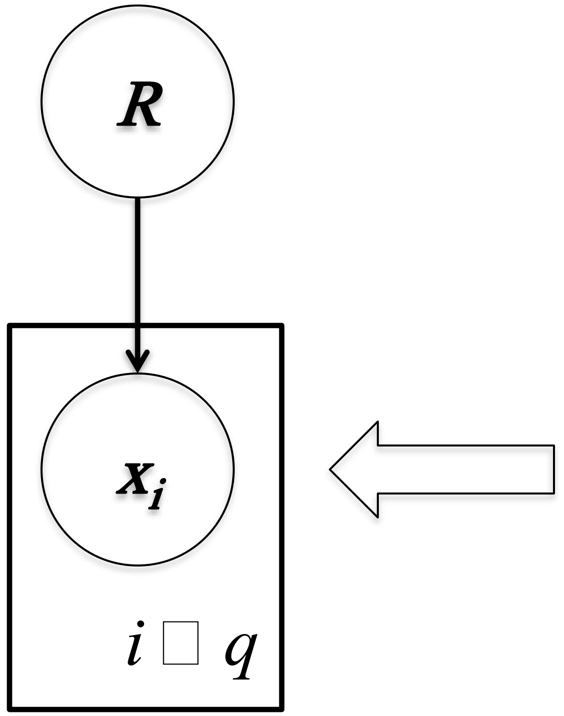
$$RSV = \log \prod_{x_i = q_i = 1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i = q_i = 1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

$$RSV = \sum_{x_i = q_i = 1} c_i; \qquad c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

The c_i are **log odds ratios** (of contingency table a few slides back)
They function as the term weights in this model

So, how do we compute c_i*'s* from our data?

Graphical model for BIM – Bernoulli NB



Binary variables

$$x_i = (tf_i \square 0)$$

Binary Independence Model

- Estimating RSV coefficients in theory
- For each term *i* look at this table of document counts:

$$c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

Documents	Relevant	Non-Relevant	Total
x_i=1	*s*	*n-s*	*n*
x_i=0	*S-s*	*N-n-S+s*	*N-n*
Total	*S*	*N-S*	*N*

- Estimates: $p_i \approx \frac{s}{S}$ $\quad r_i \approx \frac{(n-s)}{(N-S)}$

$$c_i \approx K(N,n,S,s) = \log \frac{s/(S-s)}{(n-s)/(N-n-S+s)}$$

For now, assume no zero terms. Remember smoothing.

Estimation – key challenge

- If non-relevant documents are approximated by the whole collection, then r_i (prob. of occurrence in non-relevant documents for query) *is n/N* and

$$\log \frac{1 - r_i}{r_i} = \log \frac{N - n - S + s}{n - s} \square \log \frac{N - n}{n} \square \log \frac{N}{n} = IDF!$$

- Inverse Document Frequency (IDF)
  - Spärck-Jones (1972)
  - A key, still-important term weighting concept

Collection vs. Document frequency

- Collection frequency of *t* is the total number of occurrences of *t* in the collection (incl. multiples)
- Document frequency is number of docs *t* is in
- Example:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- Which word is a better search term (and should get a higher weight)?

Estimation – key challenge

- p_i (probability of occurrence in relevant documents) cannot be approximated as easily
- p_i can be estimated in various ways:
  - from relevant documents if you know some
    - Relevance weighting can be used in a feedback loop
  - constant (Croft and Harper combination match) – then just get idf weighting of terms (with $p_i=0.5$)

$$RSV = \sum_{x_i=q_i=1} \log \frac{N}{n_i}$$

  - proportional to prob. of occurrence in collection
    - Greiff (SIGIR 1998) argues for 1/3 + 2/3 df_i/N

4. Probabilistic Relevance Feedback

1. Guess a preliminary probabilistic description of R=1 documents; use it to retrieve a set of documents
2. Interact with the user to refine the description: learn some definite members with $R = 1$ and $R = 0$
3. Re-estimate p_i and r_i on the basis of these
   - If i appears in V_i within set of documents V: $p_i = |V_i| / |V|$
   - Or can combine new information with original guess (use Bayesian prior):

$$p_i^{(2)} = \frac{|V_i| + \kappa p_i^{(1)}}{|V| + \kappa}$$

κ is prior weight

4. Repeat, thus generating a succession of approximations to relevant documents

Pseudo-relevance feedback (iteratively auto-estimate p_i and r_i)

1. Assume that p_i *is* constant over all x_i in query and r_i as before
   - $p_i = 0.5$ (even odds) for any given doc
2. Determine guess of relevant document set:
   - *V* is fixed size set of highest ranked documents on this model
3. We need to improve our guesses for p_i and r_i, so
   - Use distribution of x_i in docs in V. Let V_i be set of documents containing x_i
     - $p_i = |V_i| / |V|$
   - Assume if not retrieved then not relevant
     - $r_i = (n_i - |V_i|) / (N - |V|)$
4. Go to 2. until converges then return ranking

PRP and BIM

- It is possible to reasonably approximate probabilities
  - But either require partial relevance information or need to make do with somewhat inferior term weights
- Requires restrictive assumptions:
  - “Relevance” of each document is independent of others
    - Really, it’s bad to keep on returning **duplicates**
  - Term independence
  - Terms not in query don’t affect the outcome
  - Boolean representation of documents/queries
  - Boolean notion of relevance
- Some of these assumptions can be removed

Removing term independence

- In general, index terms aren't independent
  - "Hong Kong"
- Dependencies can be complex
- van Rijsbergen (1979) proposed simple model of dependencies as a tree
- Each term dependent on one other
  - Exactly Friedman and Goldszmidt's Tree Augmented Naive Bayes (AAAI 13, 1996)
- In 1970s, estimation problems held back success of this model

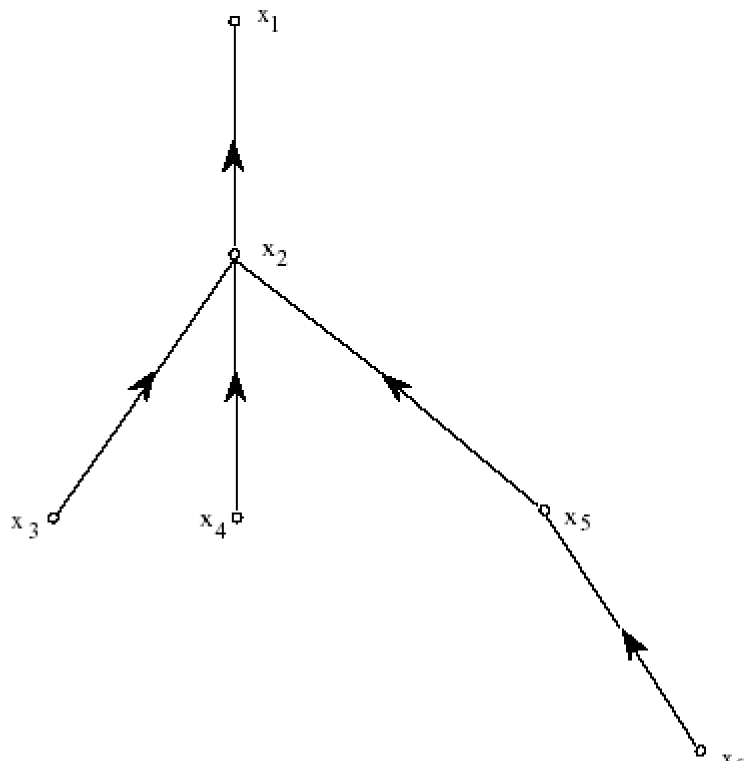


Figure 6.1.

5. Term frequency and the VSM

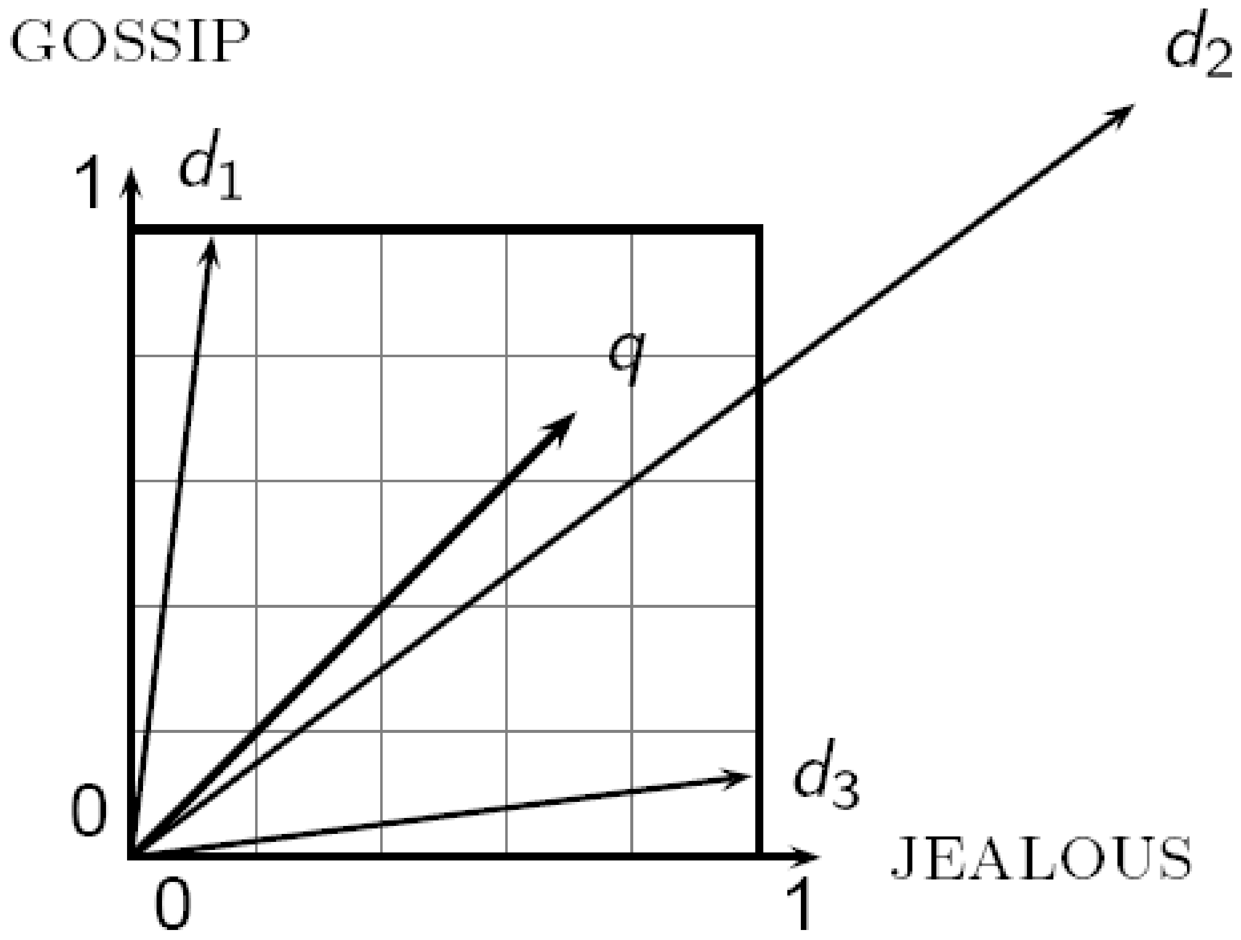
- Right in the first lecture, we said that a page should rank higher if it mentions a word more
  - Perhaps modulated by things like page length
- Why not in BIM? Much of early IR was designed for titles or abstracts, and not for modern full text search
- We now want a model with term frequency in it
- We’ll mainly look at a probabilistic model (BM25)
- First, a quick summary of vector space model

Summary – vector space ranking (ch. 6)

- Represent the query as a weighted term frequency/inverse document frequency (tf-idf) vector
  - (0, 0, 0, 0, 2.3, 0, 0, 0, 1.78, 0, 0, 0, …, 0, 8.17, 0, 0)
- Represent each document as a weighted tf-idf vector
  - (1.2, 0, 3.7, 1.5, 2.0, 0, 1.3, 0, 3.7, 1.4, 0, 0, …, 3.5, 5.1, 0, 0)

- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top K (e.g., $K = 10$) to the user

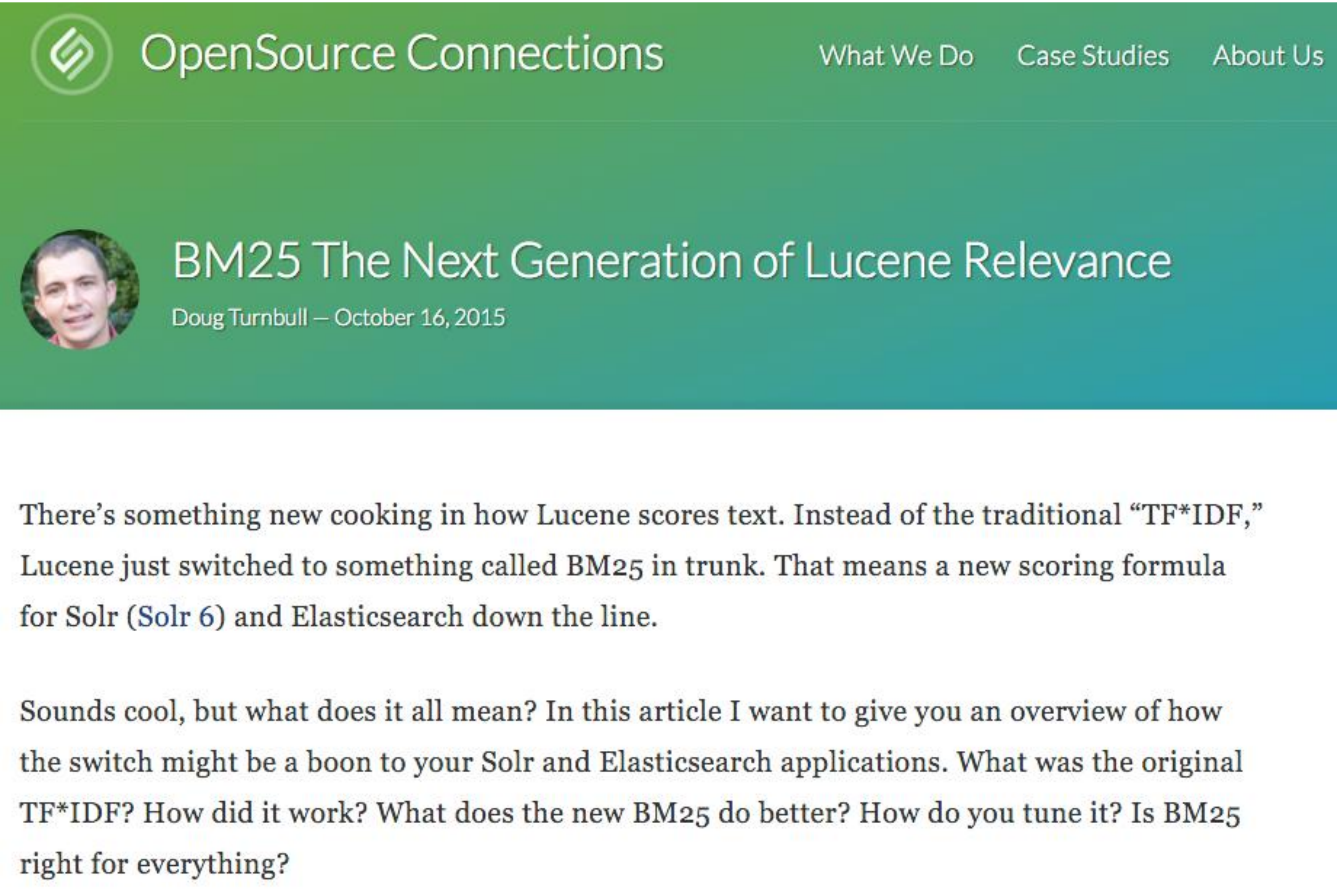
Cosine similarity



tf-idf weighting has many variants

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)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

6. BM25



OpenSource Connections

BM25 The Next Generation of Lucene Relevance

Doug Turnbull — October 16, 2015

There's something new cooking in how Lucene scores text. Instead of the traditional "TF*IDF," Lucene just switched to something called BM25 in trunk. That means a new scoring formula for Solr (Solr 6) and Elasticsearch down the line.

Sounds cool, but what does it all mean? In this article I want to give you an overview of how the switch might be a boon to your Solr and Elasticsearch applications. What was the original TF*IDF? How did it work? What does the new BM25 do better? How do you tune it? Is BM25 right for everything?

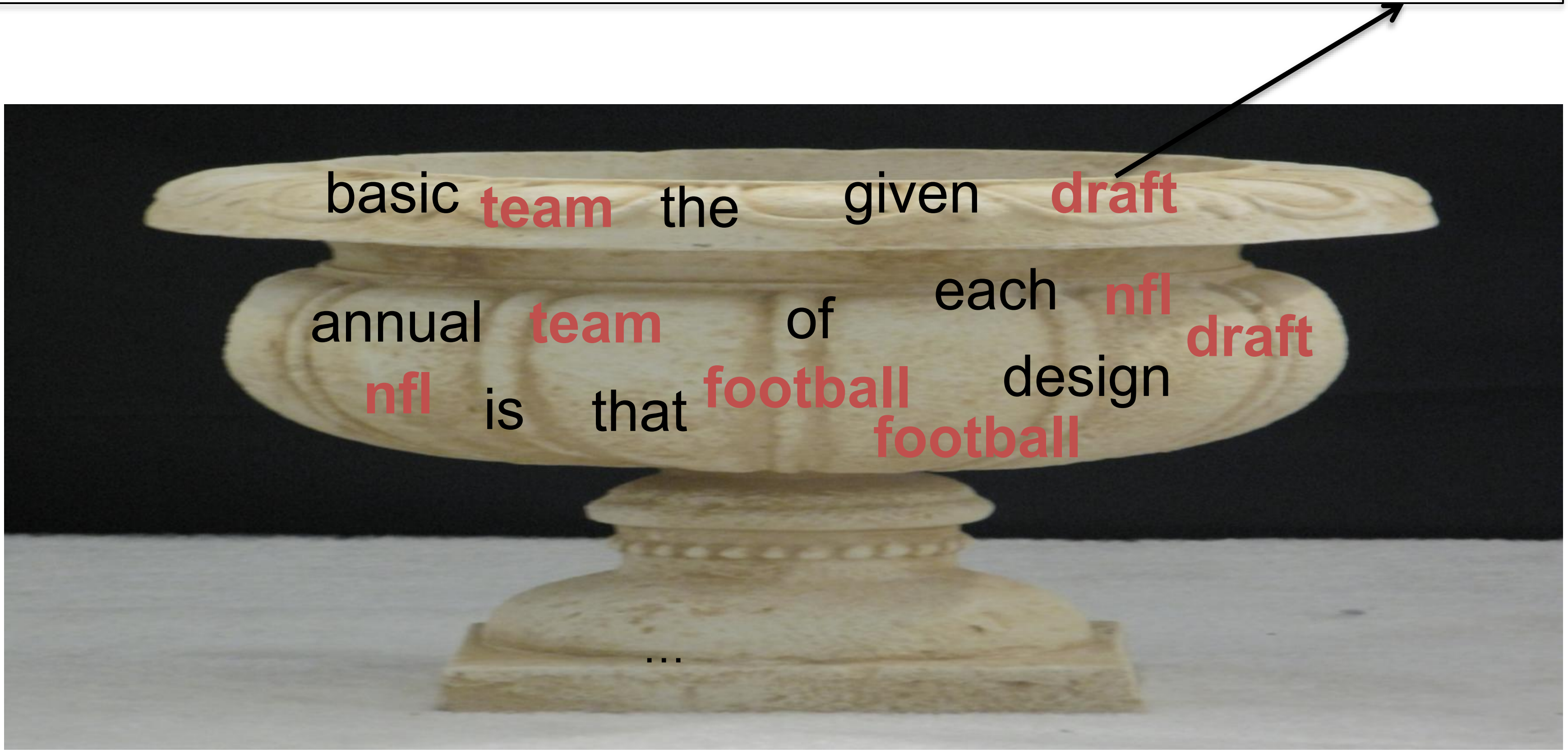
Okapi BM25 [Robertson et al. 1994, TREC City U.]

- BM25 “Best Match 25” (they had a bunch of tries!)
  - Developed in the context of the Okapi system
  - Started to be increasingly adopted by other teams during the TREC competitions
  - It works well
- Goal: be sensitive to term frequency and document length while not adding too many parameters
  - (Robertson and Zaragoza 2009; Spärck Jones et al. 2000)

Generative model for documents

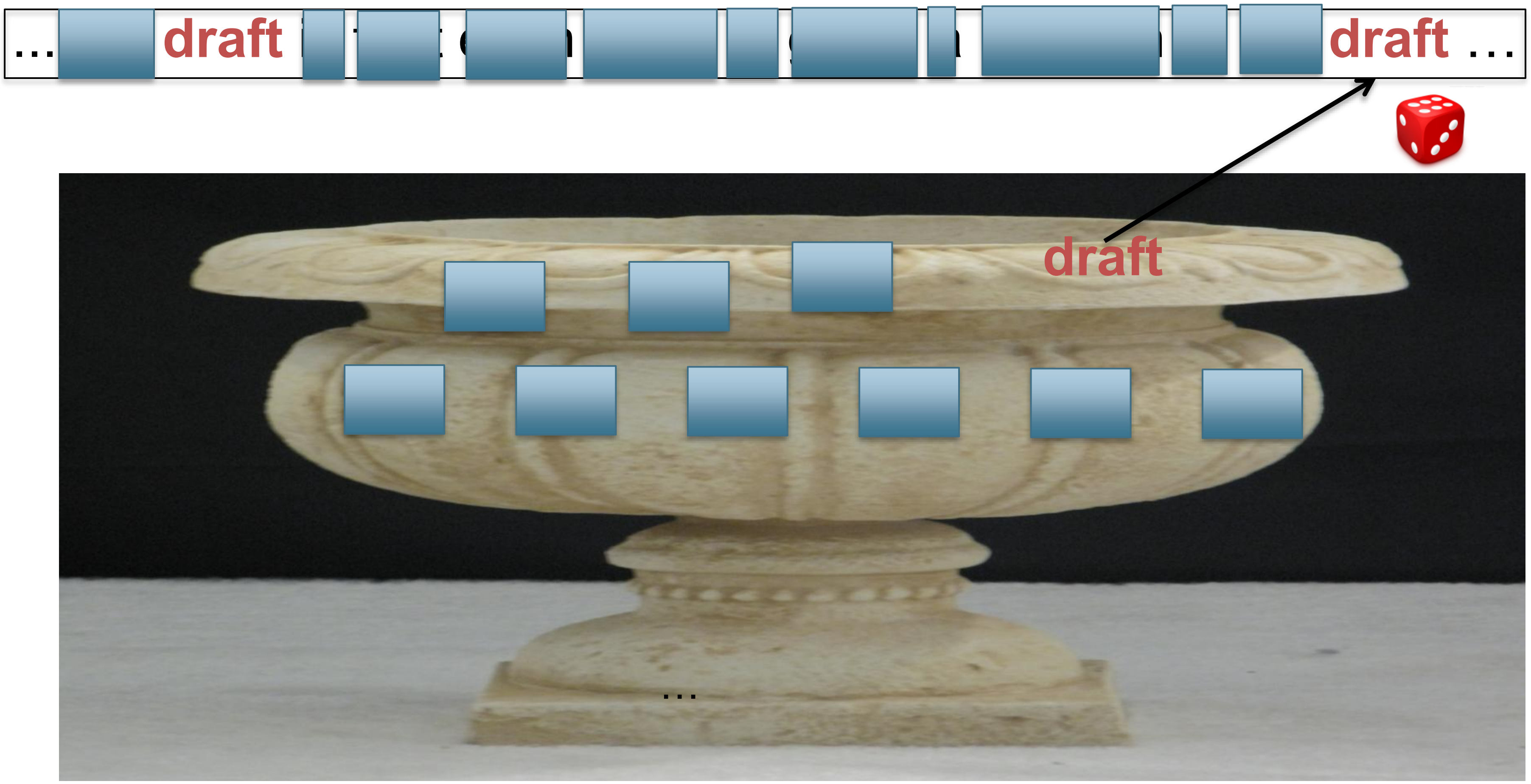
- Words are drawn independently from the vocabulary using a multinomial distribution

... the **draft** is that each **team** is given a position in the **draft** …



Generative model for documents

- Distribution of term frequencies (*tf*) follows a binomial distribution – approximated by a Poisson



Poisson distribution

- The Poisson distribution models the probability of *k*, the number of events occurring in a fixed interval of time/space, with known average rate λ (= cf/*T*), independent of the last event

$$p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

- Examples
  - Number of cars arriving at a toll booth per minute
  - Number of typos on a page

Poisson distribution

- If *T* is large and *p* is small, we can approximate a binomial distribution with a Poisson where λ = *Tp*

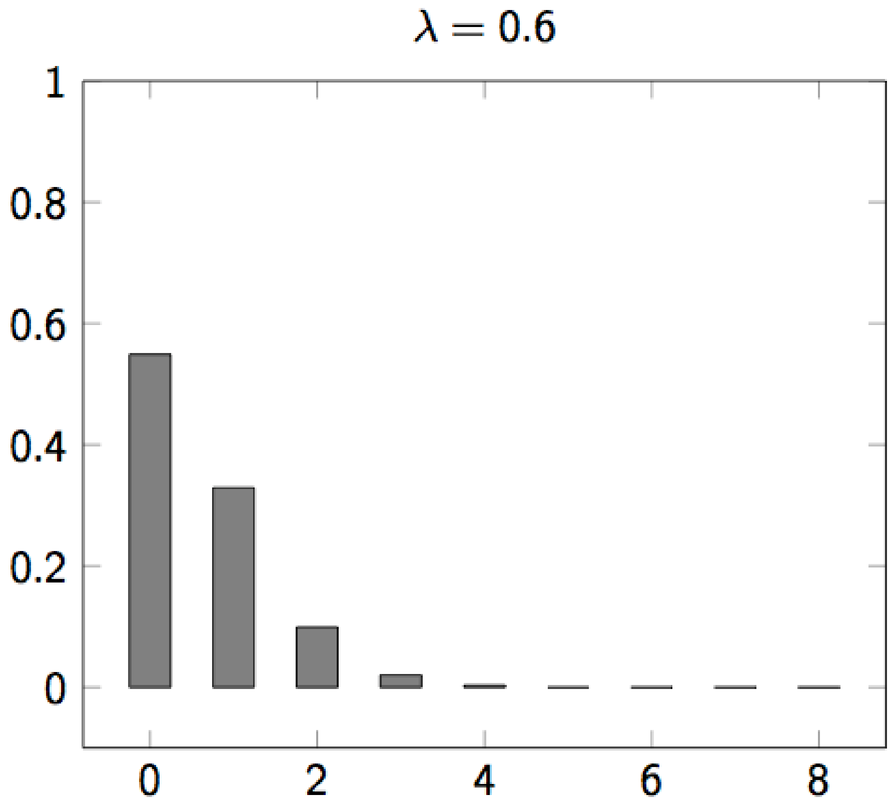
$$p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

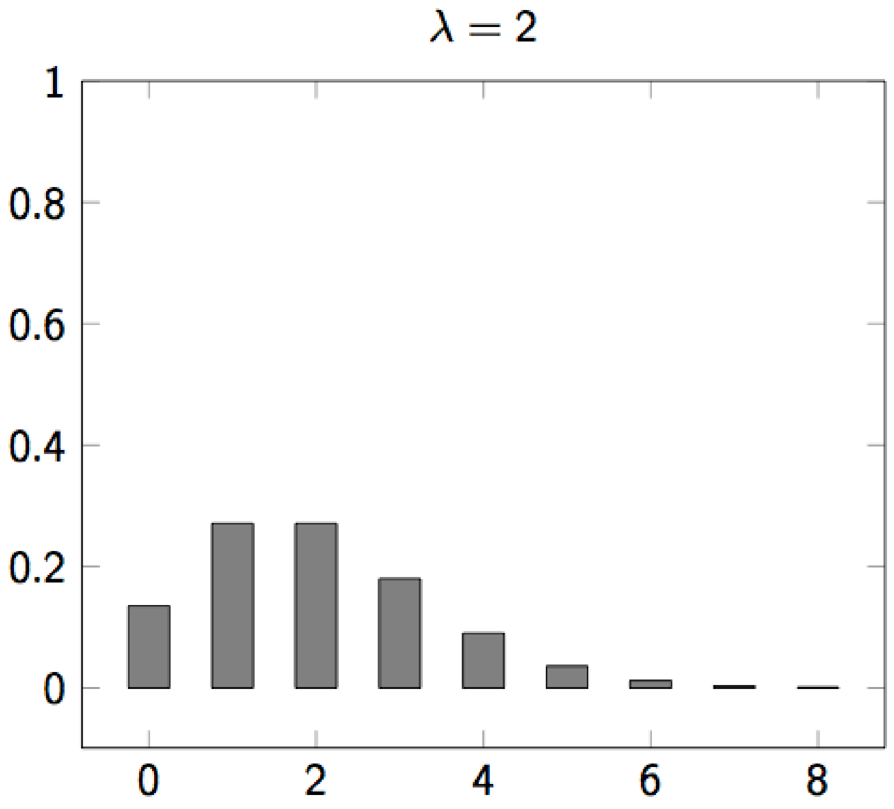
- Mean = Variance = λ = *Tp.*
- Example *p* = 0.08, *T* = 20. Chance of 1 occurrence is:
  - Binomial $P(1) = \binom{20}{1} (.08)^1 (.92)^{19} = .3282$
  - Poisson $P(1) = \frac{[(20)(.08)]^1}{1!} e^{-(20)(.08)} = \frac{1.6}{1} e^{-1.6} = 0.3230$ … already close

Poisson model

- Assume that term frequencies in a document (tf_i) follow a Poisson distribution
  - “Fixed interval” implies fixed document length … think roughly constant-sized document abstracts
    - … will fix later

Poisson distributions





(One) Poisson Model flaw

- Is a reasonable fit for "general" words
- Is a poor fit for topic-specific words
  - get higher *p(k)* than predicted too often

		Documents containing *k* occurrences of word (λ = 53/650)												
Freq	**Word**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
53	**expected**	**599**	**49**	**2**										
52	*based*	600	48	2										
53	*conditions*	604	39	7										
55	*cathexis*	619	22	3	2	1	2	0	1					
51	*comic*	642	3	0	1	0	0	0	0	0	0	1	1	2

Harter, "A Probabilistic Approach to Automatic Keyword Indexing", JASIST, 1975

Eliteness (“aboutness”)

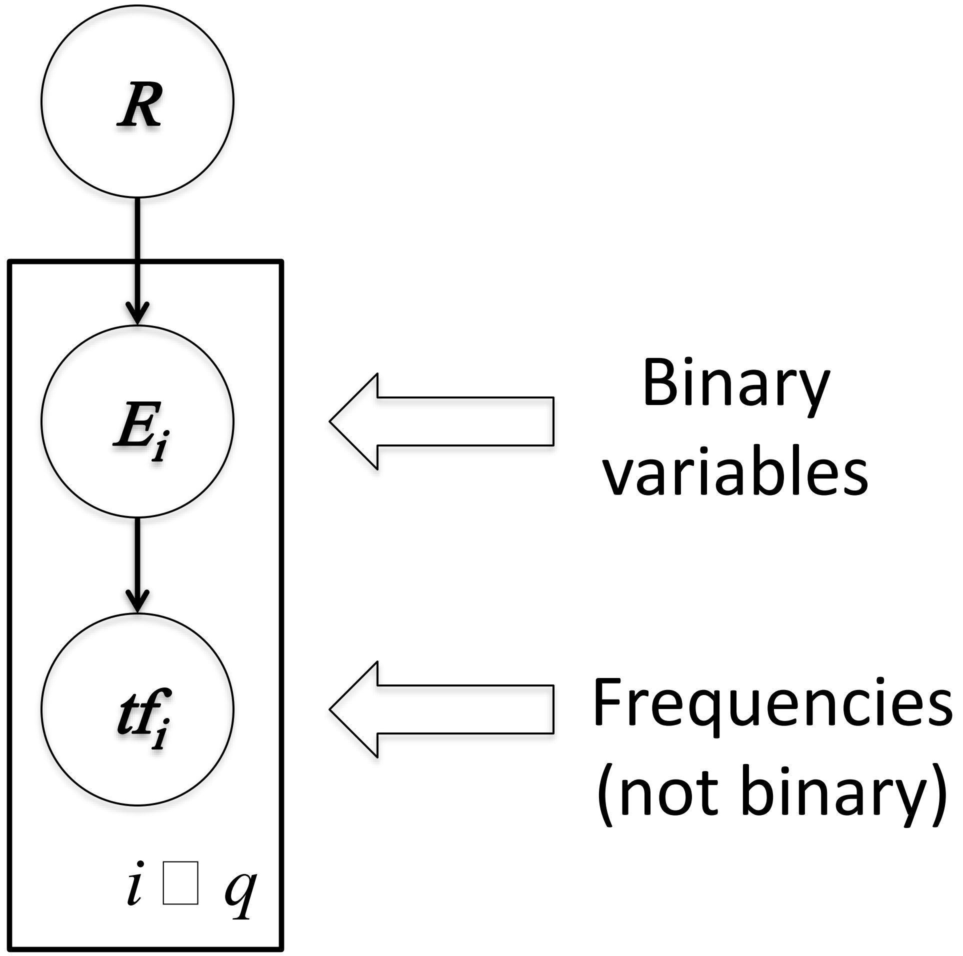
- Model term frequencies using *eliteness*
- What is eliteness?
  - Hidden variable for each document-term pair, denoted as E_i for term i
  - Represents *aboutness*: a term is elite in a document if, in some sense, the document is about the concept denoted by the term
  - Eliteness is binary
  - Term occurrences depend only on eliteness…
  - … but eliteness depends on relevance

Elite terms

Text from the Wikipedia page on the NFL draft showing **elite terms**

The **National Football League Draft** is an annual event in which the **National Football League** (**NFL**) **teams select eligible college football players**. It serves as the **league's** most common source of **player recruitment**. The basic design of the **draft** is that each **team** is given a **position** in the **draft order** in **reverse order** relative to its **record** …

Graphical model with eliteness



Retrieval Status Value

- Similar to the BIM derivation, we have

$$RSV^{elite} = \sum_{i \in q, tf_i > 0} c_i^{elite}(tf_i);$$

where

$$c_i^{elite}(tf_i) = \log \frac{p(TF_i = tf_i \mid R=1)\, p(TF_i = 0 \mid R=0)}{p(TF_i = 0 \mid R=1)\, p(TF_i = tf_i \mid R=0)}$$

and using eliteness, we have:

$$\begin{aligned} p(TF_i = tf_i \mid R) = {} & p(TF_i = tf_i \mid E_i = elite)\, p(E_i = elite \mid R) \\ & + p(TF_i = tf_i \mid E_i = \overline{elite})(1 - p(E_i = elite \mid R)) \end{aligned}$$

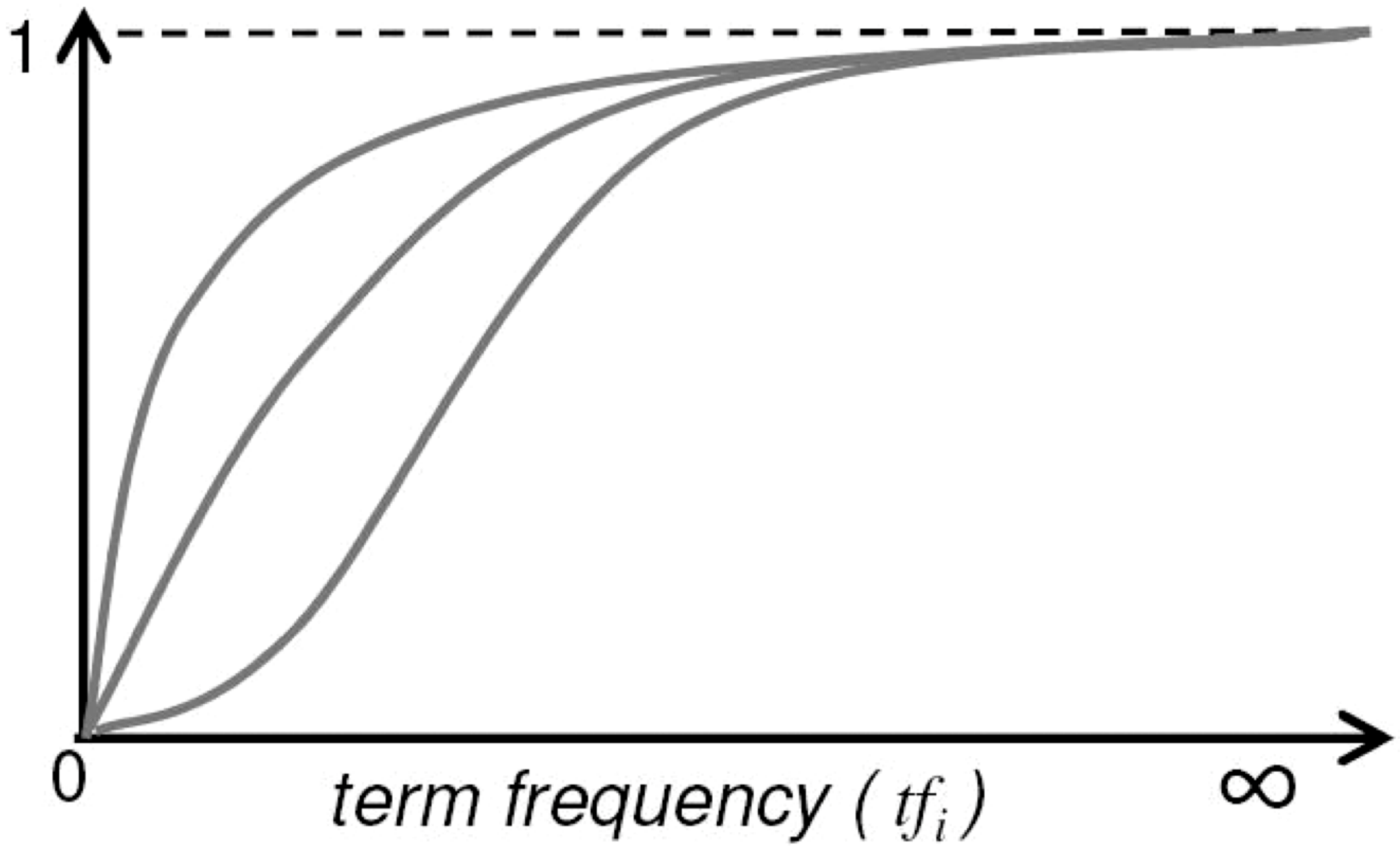
2-Poisson model

- The problems with the 1-Poisson model suggests fitting two Poisson distributions
- In the “2-Poisson model”, the distribution is different depending on whether the term is elite or not

$$p(TF_i = k_i | R) = \pi \frac{\lambda^k}{k!} e^{-\lambda} + (1-\pi) \frac{\mu^k}{k!} e^{-\mu}$$

- where π is probability that document is elite for term
- but, unfortunately, we don’t know π, λ, μ

Let's get an idea: Graphing $c_i^{elite}(tf_i)$ for different parameter values of the 2-Poisson



Qualitative properties

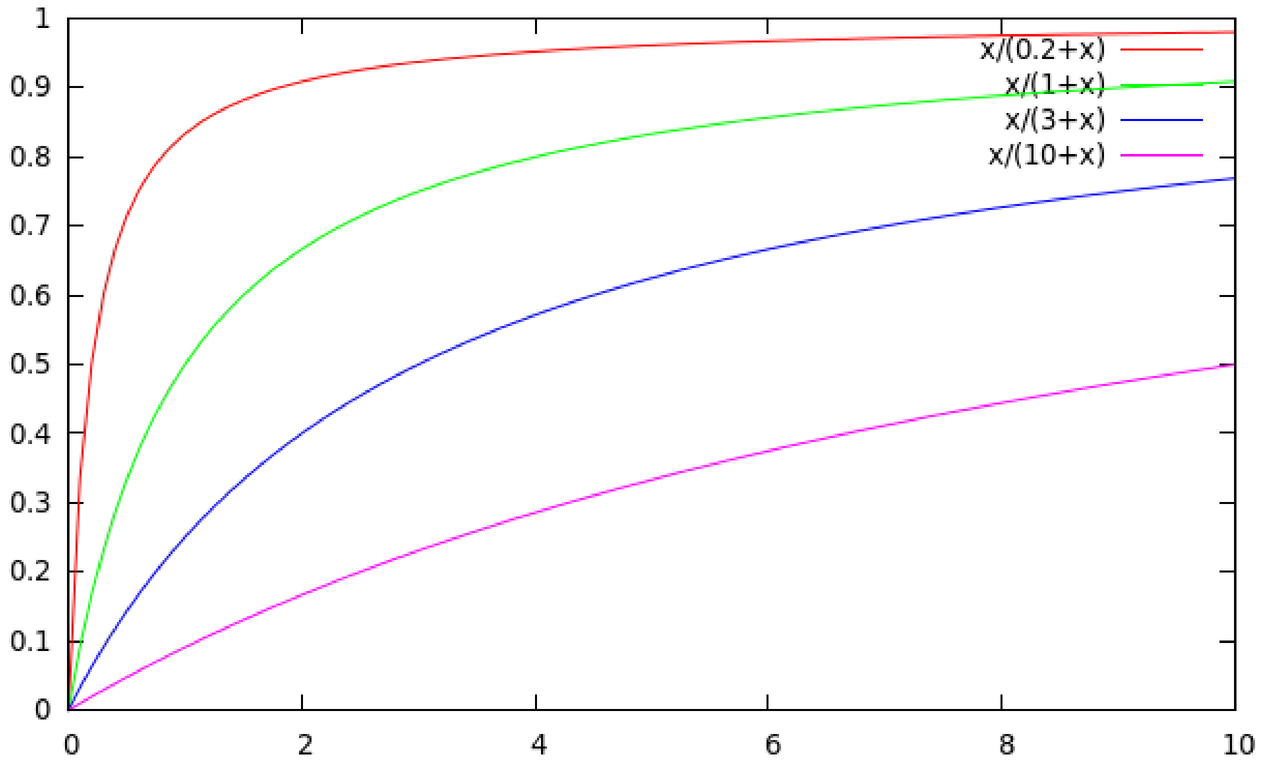
- $c_i^{elite}(0) = 0$
- $c_i^{elite}(tf_i)$ increases monotonically with tf_i
- … but asymptotically approaches a maximum value as $tf_i \to \infty$ [not true for simple scaling of tf]
- … with the asymptotic limit being c_i^{BIM} $\Leftarrow$ Weight of eliteness feature

Approximating the saturation function

- Estimating parameters for the 2-Poisson model is not easy
- … So approximate it with a simple parametric curve that has the same qualitative properties

$$\frac{tf}{k_1 + tf}$$

Saturation function



- For high values of k_1, increments in tf_i continue to contribute significantly to the score
- Contributions tail off quickly for low values of k_1

"Early" versions of BM25

- Version 1: using the saturation function

$$c_i^{BM25v1}(tf_i) = c_i^{BIM} \frac{tf_i}{k_1 + tf_i}$$

- Version 2: BIM simplification to IDF

$$c_i^{BM25v2}(tf_i) = \log \frac{N}{df_i} \square \frac{(k_1+1)tf_i}{k_1 + tf_i}$$

  - (k_1+1) factor doesn't change ranking, but makes term score 1 when $tf_i = 1$
  - Similar to *tf-idf*, but term scores are bounded

Document length normalization

- Longer documents are likely to have larger tf_i values

- Why might documents be longer?
  - Verbosity: suggests observed tf_i too high
  - Larger scope: suggests observed tf_i may be right

- A real document collection probably has both effects
- … so should apply some kind of partial normalization

Document length normalization

- Document length:

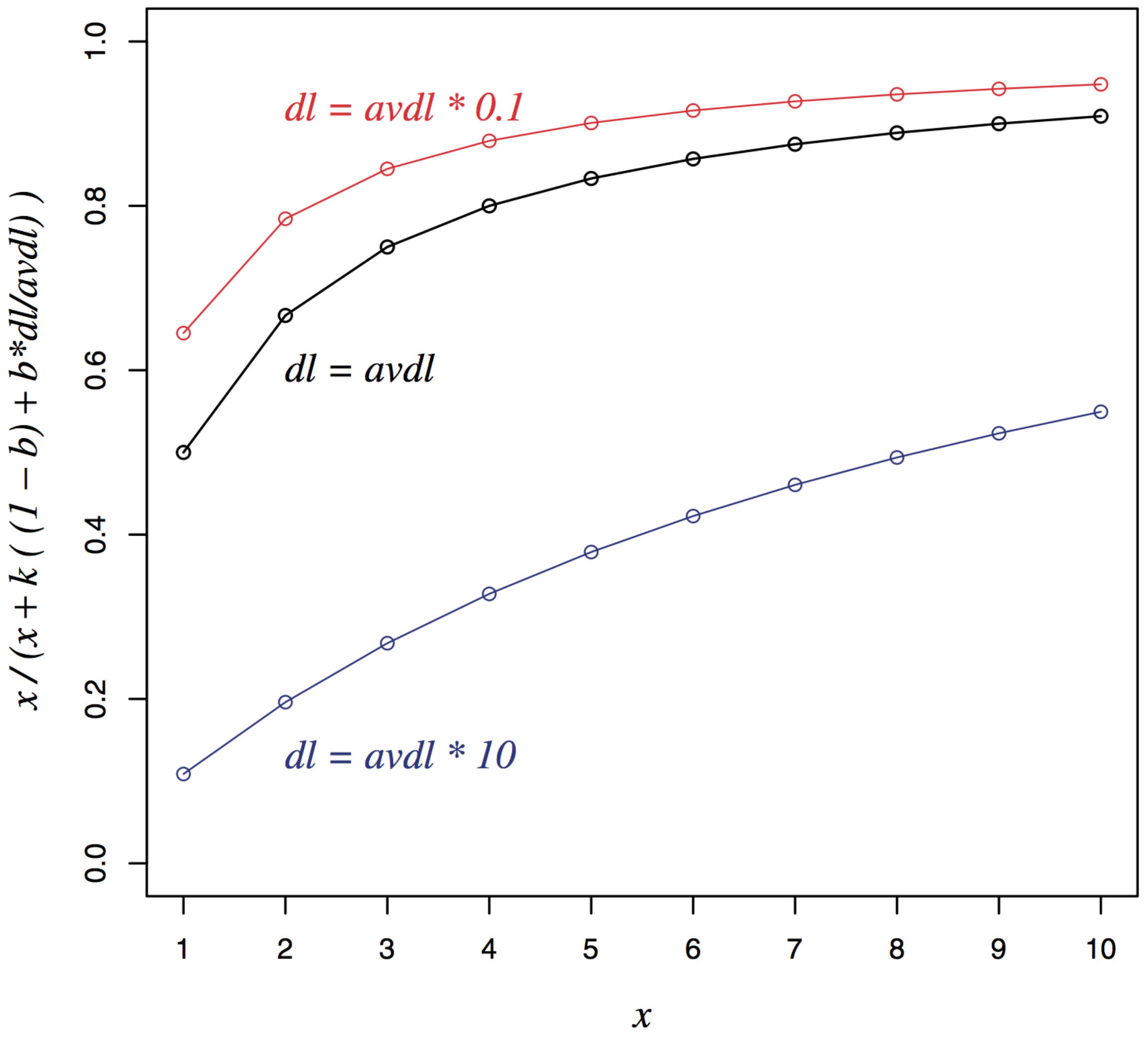
$$dl = \sum_{i \in V} tf_i$$

- *avdl*: Average document length over collection
- Length normalization component

$$B = \left((1-b) + b \frac{dl}{avdl} \right), \qquad 0 \le b \le 1$$

  - $b = 1$ full document length normalization
  - $b = 0$ no document length normalization

Document length normalization



Okapi BM25

- Normalize *tf* using document length

$$tf_i' = \frac{tf_i}{B}$$

$$\begin{aligned} c_i^{BM25}(tf_i) &= \log \frac{N}{df_i} \cdot \frac{(k_1+1)tf_i'}{k_1 + tf_i'} \\ &= \log \frac{N}{df_i} \cdot \frac{(k_1+1)tf_i}{k_1((1-b)+b\frac{dl}{avdl}) + tf_i} \end{aligned}$$

- BM25 ranking function

$$RSV^{BM25} = \sum_{i \in q} c_i^{BM25}(tf_i);$$

Okapi BM25

$$RSV^{BM25} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1) tf_i}{k_1((1-b) + b\frac{dl}{avdl}) + tf_i}$$

- k_1 controls term frequency scaling
  - $k_1 = 0$ is binary model; k_1 large is raw term frequency
- b controls document length normalization
  - $b = 0$ is no length normalization; $b = 1$ is relative frequency (fully scale by document length)
- Typically, k_1 is set around 1.2–2 and b around 0.75
- *IIR* sec. 11.4.3 discusses incorporating query term weighting and (pseudo) relevance feedback

Why is BM25 better than VSM tf-idf?

- Suppose your query is [machine learning]
- Suppose you have 2 documents with term counts:
  - doc1: learning 1024; machine 1
  - doc2: learning 16; machine 8
- tf-idf: $\log_2$ tf * $\log_2$ (N/df)
  - doc1: 11 * 7 + 1 * 10 = **87**
  - doc2: 5 * 7 + 4 * 10 = **75**
- BM25: $k_1 = 2$
  - doc1: 7 * 3 + 10 * 1 = **31**
  - doc2: 7 * 2.67 + 10 * 2.4 = **42.7**

7. Ranking with features

- Textual features
  - Zones: Title, author, abstract, body, anchors, …
  - Proximity
  - …
- Non-textual features
  - File type
  - File age
  - Page rank
  - …

Ranking with zones

- Straightforward idea:
  - Apply your favorite ranking function (BM25) to each zone separately
  - Combine zone scores using a weighted linear combination
- But that seems to imply that the eliteness properties of different zones are different and independent of each other
  - …which seems unreasonable

Ranking with zones

- Alternate idea
  - Assume eliteness is a term/document property shared across zones
  - … but the relationship between eliteness and term frequencies are zone-dependent
    - e.g., denser use of elite topic words in title
- Consequence
  - First combine evidence across zones for each term
  - Then combine evidence across terms

BM25F with zones

- Calculate a weighted variant of total term frequency
- … and a weighted variant of document length

$$\tilde{tf}_i = \sum_{z=1}^{Z} v_z tf_{zi} \qquad \tilde{dl} = \sum_{z=1}^{Z} v_z len_z \qquad av\tilde{dl} = \text{Average } \tilde{dl} \text{ across all documents}$$

where

v_z is zone weight

tf_{zi} is term frequency in zone z

len_z is length of zone z

Z is the number of zones

Simple BM25F with zones

$$RSV^{SimpleBM25F} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1+1)\tilde{tf}_i}{k_1((1-b)+b\frac{\tilde{dl}}{av\tilde{dl}})+\tilde{tf}_i}$$

- Simple interpretation: zone z is "replicated" v_z times
- But we may want zone-specific parameters (k_1, b, IDF)

BM25F

- Empirically, zone-specific length normalization (i.e., zone-specific *b*) has been found to be useful

$$\tilde{tf}_i = \sum_{z=1}^{Z} v_z \frac{tf_{zi}}{B_z}$$

$$B_z = \left((1 - b_z) + b_z \frac{len_z}{avlen_z} \right), \qquad 0 \le b_z \le 1$$

$$RSV^{BM25F} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1)\tilde{tf}_i}{k_1 + \tilde{tf}_i}$$

See Robertson and Zaragoza (2009: 364)

Ranking with non-textual features

- Assumptions
  - Usual independence assumption
    - Independent of each other and of the textual features
    - Allows us to factor out $\frac{p(F_j = f_j | R=1)}{p(F_j = f_j | R=0)}$ in BIM-style derivation
  - Relevance information is ***query independent***
    - Usually true for features like page rank, age, type, …
    - Allows us to keep all non-textual features in the BIM-style derivation where we drop non-query terms

Ranking with non-textual features

$$RSV = \sum_{i \in q} c_i(tf_i) + \sum_{j=1}^{F} \lambda_j V_j(f_j)$$

where

$$V_j(f_j) = \log \frac{p(F_j = f_j \mid R = 1)}{p(F_j = f_j \mid R = 0)}$$

and λ_j is an artificially added free parameter to account for rescalings in the approximations

- Care must be taken in selecting V_j depending on F_j. E.g.

$$\log(\lambda'_j + f_j) \qquad \frac{f_j}{\lambda'_j + f_j} \qquad \frac{1}{\lambda'_j + \exp(-f_j \lambda''_j)}$$

- Explains why $RSV^{BM25} + \log(pagerank)$ works well

Resources

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

C. J. van Rijsbergen. 1979. *Information Retrieval.* 2nd ed. London: Butterworths, chapter 6. http://www.dcs.gla.ac.uk/Keith/Preface.html

K. Spärck Jones, S. Walker, and S. E. Robertson. 2000. A probabilistic model of information retrieval: Development and comparative experiments. Part 1. *Information Processing and Management* 779–808.

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