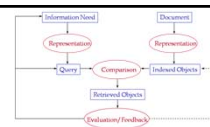
11-442 / 11-642:
Search Engines

Best-Match Retrieval: VSM, BM25

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

Introduction



Until now, we have focused on exact-match retrieval models

- Unranked Boolean and Ranked Boolean
- Used widely in industry until about 1990
- Still an important form of retrieval in many situations

Today's lecture introduces best-match retrieval models

- Easier for many people to use
- Often more accurate
- Considered the state-of-the-art today in many situations

2

© 2017, Jamie Callan

Introduction

Exact match retrieval models require that a document either match (1) or not match (0) a query

- Unranked Boolean and Ranked Boolean retrieval models

Best match retrieval models calculate how well a document d_i satisfies the information need I expressed by a query Q

- Ideally expressed as Satisfy (I, d_i) or $p\ (d_i \mid I)$
- More often expressed as Similarity (Q, d_i) or $p\ (d_i \mid Q)$
- Most documents match the query to some degree

The search result is a ranked list of documents

- "Best" first

© 2017, Jamie Callan

Introduction

Usually the document is modeled as a vector or distribution of term weights

- A set of index terms
  - The "bag of words"
- A weight for each index term
  - "term weights"

There is less agreement on …

- How to treat the query
- How to rank documents

Document

Term	Weight
camera	0.09551
image	0.07303
picture	0.06180
up	0.04494
movie	0.04494
like	0.03933
mode	0.03933
software	0.03933
red	0.03371
digital	0.02809
eye	0.02809
shutter	0.02809
sony	0.02809

© 2017, Jamie Callan

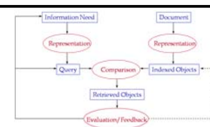
Best Match Retrieval Models

What distinguishes different best-match models?

- The <u>theory</u>
  - – Important to scientists, but maybe not to practitioners
- How <u>term weights</u> are calculated
  - – Most models use the same statistics (tf, df, doclen, numdocs)
- How <u>similarity</u> between is calculated
- Whether the retrieval model can handled <u>structured queries</u>

5

© 2017, Jamie Callan



Introduction

We will cover several best-match retrieval models

- Vector space retrieval model (VSM)
- Probabilistic retrieval models (BM25)
- Statistical language models (query likelihood)
- Inference networks (Indri)

Each retrieval model is based on a <u>different theory</u>

… however, they have (mostly) <u>similar architectures</u>

Disagreement about theory, but agreement about what works

- Don’t be confused by the theory – this stuff is simple

6

© 2017, Jamie Callan

Lecture Outline

Introduction

The vector space model (VSM)

- Standard VSM (lnc.ltc)
- Lucene

BM25

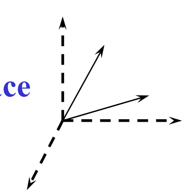
7

© 2017, Jamie Callan

The Vector Space Retrieval Model

Representation: Any text can be represented by a term vector

- **Examples:** Documents, queries, sentences, ….



Similarity is determined by distance in a vector space

- There are many ways of measuring this distance

Best known vector space systems

- SMART, developed by Gerard Salton, mostly at Cornell
- Lucene

8

© 2017, Jamie Callan

Vector Space Similarity

How similar is the text represented by vectors x and y?

There are many ways to measure the similarity of two vectors

- Overlap of vectors *x* and *y* can be determined by their inner product

  $\sum_{i=1}^{|V|} x_i \cdot y_i$

- *V* is the vocabulary
- Overlap measures the similarity of vector vocabularies

tf weights

Term	x	y	$x_i \bullet y_i$
apple	0	0	0
buy	1	0	0
camera	17	1	17
dog	0	0	0
image	13	1	13
like	7	0	0
mode	7	0	0
movie	8	0	0
up	8	0	0
…	…	…	…
zooms	1	1	1
Total			**31**

9

© 2017, Jamie Callan

Vector Space Similarity

Overlap measures the similarity of the vocabularies

- **Problem:** It doesn't normalize for vector length
- **Problem:** All terms are treated as equally important

Issue: Which is more significant?

- Two long pieces of text with overlapping vocabularies
- Two short pieces of text with overlapping vocabularies
- One long and one short piece of text with overlapping vocabularies

Which does overlap favor?

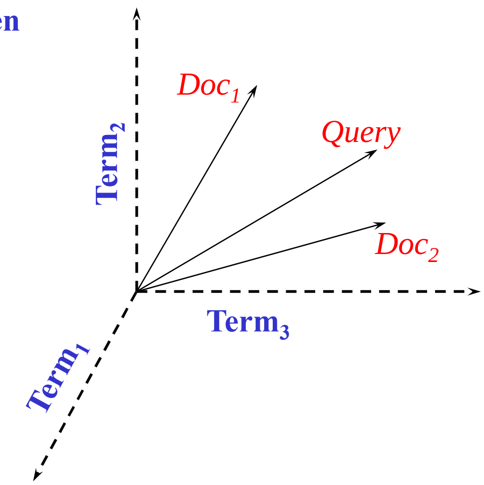
10

© 2017, Jamie Callan

Vector Space Similarity

Historically, there have been many popular similarity functions

- Inner product
- Dice coefficient
- Jackard coefficient
- Cosine correlation
- …



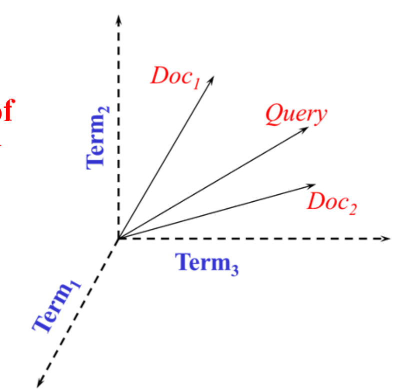
© 2017, Jamie Callan

Cosine Similarity

The <u>cosine of the angle between the vectors</u> can be determined by scaling for document length

$$\frac{\sum_{i=1}^{|V|} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{|V|} x_i^2} \cdot \sqrt{\sum_{i=1}^{|V|} y_i^2}}$$

Length of vector y



© 2017, Jamie Callan

Cosine Similarity

tf weights

Term	x	y	$x_i \cdot y_i$
apple	0	0	0
buy	1	0	0
camera	17	1	17
dog	0	0	0
image	13	1	13
like	7	0	0
mode	7	0	0
movie	8	0	0
up	8	0	0
zooms	1	1	1
Total			**31**

Length of x:

$$\sqrt{1^2 + 17^2 + 13^2 + \ldots + 1^2} = 26.19$$

Length of y:

$$\sqrt{1^2 + 1^2 + 1^2} = 1.73$$

Sim (x, y):

$$\frac{31}{26.19 \times 1.73} = \frac{31}{45.33} = 0.68$$

13

© 2017, Jamie Callan

Vector Coefficients

Vector coefficients (term weights) determine each term's effect

- The vector space model does not specify how to set term weights

Some common considerations:

- **Document term weight:** Importance of the term in this document
- **Collection term weight:** Importance of the term in this collection
- **Length normalization:** Compensate for varying document lengths

Naming convention for term weight functions: DCL.DCL

- First triple is document vector, second triple is query vector
- n=none (no weighting on that factor)
- **Example:** lnc.ltc

14

© 2017, Jamie Callan

Document Term Weights (D)

How should the importance of the term in this document be represented?

- Binary weight?
- Term frequency (tf)?
- Some function of term frequency?
- …

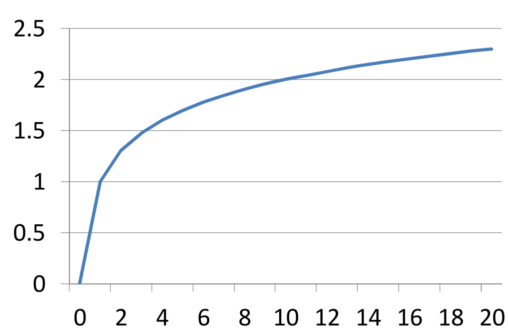
15

© 2017, Jamie Callan

Document Term Weights (D)

What characteristics are required in a term weighting function?

- A monotonic function
- Term frequency scaled by document length?
- Saturation?



One popular choice

tf	log(tf+1)
1	0.69
2	1.10
3	1.39
4	1.61
5	1.79
6	1.95

16

© 2017, Jamie Callan

Collection Term Weights (C): Inverse Document Frequency (idf)

Observation: Terms that occur in many documents in the collection are less useful for discriminating among documents

Document frequency (df): # of documents that contain the term

idf is often calculated as:

$$idf = \log\left(\frac{N}{df}\right) \qquad idf = \log\left(\frac{N+1}{df}\right) \qquad idf = \log\left(\frac{N}{df}\right)+1$$

Standard form (Know this) | **If we want to avoid idf=0** | **If we want to avoid idf=0**

The three formulas produce similar (but not identical) rankings

© 2017, Jamie Callan

lnc.ltc

lnc.ltc: A popular combination of term weights and similarity metric

- **"l":** document term weight = log (tf) + 1
- **"t":** collection term weight = log (N / df)
- **"c":** cosine length normalization $\sqrt{\sum w_i^2}$
- **"n":** weight = 1.0 (i.e., none)

Know this

- **For example:**

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^2} \cdot \sqrt{\sum q_i^2}} = \frac{\sum \underbrace{(\log(tf)+1)}_{\text{"tf"}} \cdot \left(\underbrace{(\log(qtf)+1)}_{\text{"user weight"}} \underbrace{\log\frac{N}{df}}_{\text{"idf"}}\right)}{\underbrace{\sqrt{\sum(\log(tf)+1)^2}}_{\text{"document length normalization"}} \cdot \underbrace{\sqrt{\sum\left((\log(qtf)+1)\log\frac{N}{df}\right)^2}}_{\text{"query length normalization"}}}$$

Cosine similarity metric

© 2017, Jamie Callan

lnc.ltc

Why does lnc.ltc put idf in the query term weight?

- Originally, idf represented the term's importance to query
- Today, vector space formulas may have it anywhere

Query	Document	
log (N / df)		**Query term weight**
	log (N / df)	**Document term weight**
sqrt (log (N / df))	sqrt (log (N / df))	**One policy for both vectors**
log (N / df)	log (N / df)	**Double idf weighting**

There is no good theory to guide setting vector space weights

© 2017, Jamie Callan

Vector Space Implementation

It is easy to implement lnc.ltc

- The query operator is #SUM
- Calculate scores only for documents that contain a query term
  – Use inverted lists – similar to HW1
- The document vector length is stored in the index – look it up
- The query vector length is determined when the query is created

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^2}\cdot\sqrt{\sum q_i^2}} = \frac{\sum_{t\in d\cap q}\left(\log\left(tf_{t,d}\right)+1\right)\cdot\left(\left(\log\left(qtf_t\right)+1\right)\ \log\frac{N}{df_t}\right)}{\sqrt{\sum_{t\in d}\left(\log\left(tf_{t,d}\right)+1\right)^2}\cdot\sqrt{\sum_{t\in q}\left(\left(\log\left(qtf_t\right)+1\right)\ \log\frac{N}{df_t}\right)^2}}$$

Query operator **Document vector length** **Query vector length**

© 2017, Jamie Callan

Boolean Queries

The vector space is based on the similarity of two vectors
… do query operators fit within the vector space framework?

AND, OR, and NOT have vector space implementations

- 'p-norm' operators
  - Score combinations that mimic the effects of Boolean operators
- I have never seen anyone use them, so we don't cover them

What about proximity operators?

© 2017, Jamie Callan

Vector Space Similarity: Query Operators Such As NEAR/n

Remember that some query operators can be viewed as dynamically creating indexing terms

- The operator produces an inverted list containing df, tf, …
- E.g., NEAR/n, UW/n, SYNONYM (…), ...
- Thus, the vector space can handle them just like other terms

How does this affect document length normalization?

- Standard practice is to ignore it
  - Just compute length over unigrams

© 2017, Jamie Callan

Lucene

Lucene uses a two-step retrieval process

1. **Use a Boolean query to form a set of documents**
   - E.g., Boolean AND
   - "Fuzzy" Boolean is also an option
2. **Use a vector space retrieval algorithm to rank the set**

© 2017, Jamie Callan

Lucene

Each document can have a query-independent weight

- E.g., PageRank

Documents are composed of fields

- Each field is an independent text representation
  - I.e., a distinct vector space or bag of words
- Each field can have a query-independent weight
  - E.g., so that Title is more important than Body

© 2017, Jamie Callan

A Simplified View of Lucene's tf.idf Ranker

Lucene uses a modified vector space algorithm

- **A simplified view is…**

  RSV (d,q) = weight (d) × coordination (d,q) × cosine (d,q)

  coordination: Percentage of query terms that match d

"tf" **"idf"** **"idf"**

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^{\,2}} \cdot \sqrt{\sum q_i^{\,2}}} = \frac{\sum\left(\sqrt{tf} \cdot \left(1+\log\frac{N}{df+1}\right)\right) \cdot \left(qtf \cdot \left(1+\log\frac{N}{df+1}\right)\right)}{\sqrt{\sum\left(\sqrt{tf} \cdot \left(1+\log\frac{N}{df+1}\right)\right)^2} \cdot \sqrt{\sum\left(qtf \cdot \left(1+\log\frac{N}{df+1}\right)\right)^2}}$$

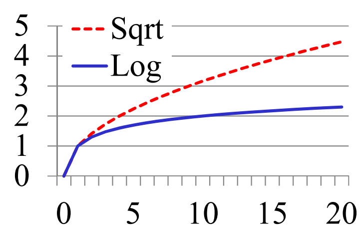
"document length normalization" **"query length normalization"**

© 2017, Jamie Callan

A Simplified View of Lucene's tf.idf Ranker

Lucene's vector space retrieval model has two important differences from lnc.ltc

- The tf weight is sqrt (tf) instead of log (tf)+1
- idf² instead of idf



What are the effects?

- <u>Stronger reward</u> for terms that are frequent in this document
- <u>Stronger penalty</u> for terms that are frequent across the corpus

© 2017, Jamie Callan

Vector Space Retrieval Model: Summary

Standard vector space

- Each dimension corresponds to a term in the vocabulary
- Vector elements are real-valued, reflecting term importance
- Any vector (document,query, ...) can be compared to any other
- Cosine correlation is the similarity metric used most often
- A best-match retrieval model
  - Unlike the Boolean retrieval model, which was exact-match

© 2017, Jamie Callan

Vector Space Retrieval Model

The key idea: Measure similarity among weighted term vectors

- Documents, queries, paragraphs, sentences, … anything

What is missing from the vector space model?

- No guidance about how to set term weights
- No guidance about how to determine similarity
- No method of supporting query-independent weights

You can do pretty much anything you want

- **Strength:** Very flexible, can absorb good ideas from anywhere
- **Weakness:** Everything is heuristic

© 2017, Jamie Callan

Lecture Outline

Introduction

The vector space model (VSM)

- Standard VSM (lnc.ltc)
- Lucene

BM25

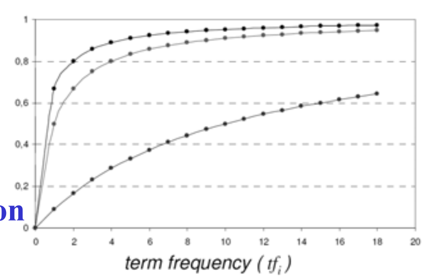
© 2017, Jamie Callan

Okapi BM25

Robertson used inspiration from another (unsuccessful) retrieval model to develop requirements for a tf function

- Zero when tf=0
- Increases monotonically with tf
- Saturates as tf increases
  - Shape depends on parameters

tf / (tf + k_1) is a close approximation



Document length normalization

- **Normalize by** $(1-b)+b\dfrac{doclen}{avg\_doclen}$

(Robertson & Zaragoza, 2007)

© 2017, Jamie Callan

The Okapi BMxx Retrieval Model

The Okapi BMxx model

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1 \left((1-b) + b \frac{\mathrm{doclen}_d}{\mathrm{avg\_doclen}} \right)} \frac{(k_3 + 1)\, qtf_t}{k_3 + qtf_t}$$

RSJ weight (idf) | **tf weight** | **user weight**

BMxx indicates different parameter settings

- **Originally:** k_1=2, b=0.75, k_3=0 (also used in Inquery)
- **BM25:** k_1=1.2, b=0.75, k_3=0-1000
  k_1=0.9, b=0.40, k_3=0-1000 (large collections)*

(* Shane Culpepper, 2014, personal communication)

31

© 2017, Jamie Callan

The Okapi BM25 Retrieval Model

The Okapi BMxx model

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1 \left((1-b) + b \frac{\mathrm{doclen}_d}{\mathrm{avg\_doclen}} \right)} \frac{(k_3 + 1)\, qtf_t}{k_3 + qtf_t}$$

RSJ weight (idf) | **tf weight** | **user weight**

Note the similarities to the vector space

• A saturating tf function	tf
• idf	df, N
• Document length normalization	doclen, avg_doclen
• Summation of scores	

32

© 2017, Jamie Callan

The Okapi BM25 Retrieval Model

HW2 requires you to think about the effects of parameters

$$\sum_{t \in q \cap d} \left(\log \frac{N - df_t + 0.5}{df_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1 \left((1-b) + b \frac{doclen_d}{avg\_doclen} \right)} \frac{(k_3 + 1)\, qtf_t}{k_3 + qtf_t}$$

What happens when k_3 approaches 0?

- Query term frequency has no effect
  - "apple apple pie" is the same as "apple pie"

$$\sum_{t \in q \cap d} \left(\log \frac{N - df_t + 0.5}{df_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1 \left((1-b) + b \frac{doclen_d}{avg\_doclen} \right)}$$

33

© 2017, Jamie Callan

The Okapi BM25 Retrieval Model

HW2 requires you to think about the effects of parameters

$$\sum_{t \in q \cap d} \left(\log \frac{N - df_t + 0.5}{df_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1 \left((1-b) + b \frac{doclen_d}{avg\_doclen} \right)} \frac{(k_3 + 1)\, qtf_t}{k_3 + qtf_t}$$

What happens when k_1 approaches 0?

- Document term frequency has no effect
  - Rare words and repeated query terms dominate

$$\sum_{t \in q \cap d} \left(\log \frac{N - df_t + 0.5}{df_t + 0.5} \right) \frac{(k_3 + 1)\, qtf_t}{k_3 + qtf_t}$$

34

© 2017, Jamie Callan

The Okapi BM25 Retrieval Model

HW2 requires you to think about the effects of parameters

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1 \left((1-b) + b \frac{\mathrm{doclen}_d}{\mathrm{avg\_doclen}} \right)} \frac{(k_3+1)\, qtf_t}{k_3 + qtf_t}$$

What happens when b approaches 0?

- Document length is ignored
  - Long documents are more likely to be retrieved

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1} \frac{(k_3+1)\, qtf_t}{k_3 + qtf_t}$$

35

© 2017, Jamie Callan

The Okapi BM25 Retrieval Model

This is how BM25 is usually presented, but it contains a flaw

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5} \right) \frac{tf_{t,d}}{tf_{t,d} + k_1 \left((1-b) + b \frac{\mathrm{doclen}_d}{\mathrm{avg\_doclen}} \right)} \frac{(k_3+1)\, qtf_t}{k_3 + qtf_t}$$

Suppose $\mathrm{df}_t = \mathrm{N}/2$

- RSJ weight $= \log \frac{\mathrm{N} - \frac{\mathrm{N}}{2} + 0.5}{\frac{\mathrm{N}}{2} + 0.5} = \log \frac{\frac{\mathrm{N}}{2} + 0.5}{\frac{\mathrm{N}}{2} + 0.5} = \log(1) = 0$

- Matching the term has <u>no effect</u> on the document score

36

© 2017, Jamie Callan

The Okapi BM25 Retrieval Model

This is how BM25 is usually presented, but it contains a flaw

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df_t} + 0.5}{\mathrm{df_t} + 0.5} \right) \frac{tf_{\mathrm{t,d}}}{tf_{\mathrm{t,d}} + k_1 \left((1-b) + b \frac{\mathrm{doclen_d}}{\mathrm{avg\_doclen}} \right)} \frac{(k_3+1)\, qtf_{\mathrm{t}}}{k_3 + qtf_{\mathrm{t}}}$$

Suppose $\mathrm{df_t}$ > N/2 (e.g., N/2 + 1)

- $\text{RSJ weight} = \log \frac{\mathrm{N} - \left(\frac{\mathrm{N}}{2} + 1 \right) + 0.5}{\left(\frac{\mathrm{N}}{2} + 1 \right) + 0.5} = \log \frac{\frac{\mathrm{N}}{2} + 0.5}{\frac{\mathrm{N}}{2} + 1.5} = \log(\mathit{fraction}) < 0$

- Matching a frequent term <u>lowers</u> a document's score

© 2017, Jamie Callan

The Okapi BM25 Retrieval Model

This is how BM25 is usually presented, but it contains a flaw

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df_t} + 0.5}{\mathrm{df_t} + 0.5} \right) \frac{tf_{\mathrm{t,d}}}{tf_{\mathrm{t,d}} + k_1 \left((1-b) + b \frac{\mathrm{doclen_d}}{\mathrm{avg\_doclen}} \right)} \frac{(k_3+1)\, qtf_{\mathrm{t}}}{k_3 + qtf_{\mathrm{t}}}$$

A common solution

- Change the RSJ weight to $\mathit{Max}\left(0, \log \frac{\mathrm{N} - \mathrm{df_t} + 0.5}{\mathrm{df_t} + 0.5} \right)$
  – Jamie's code does this

© 2017, Jamie Callan

Lucene's BM25 Ranker

Recently, Lucene switched to BM25 ranking

- Standard BM25, except for the RSJ weight
- Lucene adds +1 to prevent negative RSJ weights

Modified RSJ weight

$$\sum_{t \in q \cap d} \left(\underbrace{\log \left(1 + \frac{\mathrm{N} - \mathrm{df}_\mathrm{t} + 0.5}{\mathrm{df}_\mathrm{t} + 0.5} \right)}_{} \right) \frac{tf_{\mathrm{t,d}}}{tf_{\mathrm{t,d}} + k_1 \left((1-b) + b \frac{\mathrm{doclen}_\mathrm{d}}{\mathrm{avg\_doclen}} \right)} \frac{(k_3 + 1)\, qtf_\mathrm{t}}{k_3 + qtf_\mathrm{t}}$$

(Doug Turnbull — October 16, 2015)

39

© 2017, Jamie Callan

Okapi BM25 Implementation

Okapi BM25 is easy to implement

- The query operator is #SUM
- Calculate scores only for documents that contain a query term
  - Use inverted lists – similar to HW1
- Constants (N, avg_doclen) are stored in the index – look them up
- $\mathrm{doclen}_\mathrm{d}$ is stored in the index – look it up
- Parameters (b, k_1, k_3) are stored in the retrieval model – look them up

Query operator **Query weight**

$$\sum_{t \in q \cap d} \left(\log \frac{\mathrm{N} - \mathrm{df}_\mathrm{t} + 0.5}{\mathrm{df}_\mathrm{t} + 0.5} \right) \frac{tf_{\mathrm{t,d}}}{tf_{\mathrm{t,d}} + k_1 \left((1-b) + b \frac{\mathrm{doclen}_\mathrm{d}}{\mathrm{avg\_doclen}} \right)} \frac{(k_3 + 1)\, qtf_\mathrm{t}}{k_3 + qtf_\mathrm{t}}$$

40

© 2017, Jamie Callan

Okapi BM25:
Boolean Queries

Okapi BM25 doesn't support Boolean query operators

- Typically it is used only for unstructured queries

41

© 2017, Jamie Callan

Okapi BM25:
Query Operators Such As NEAR/n

Remember that some query operators can be viewed as dynamically creating indexing terms

- The operator produces an inverted list containing df, tf, …
- E.g., NEAR/n, UW/n, SYNONYM (…), ...
- Thus, Okapi BM25 can handle them just like other terms

How does this affect document length normalization (avg_doclen)?

- Standard practice is to ignore it
  - Just compute length over unigrams

42

© 2017, Jamie Callan

Okapi BM25 Implementation: qtf

Query term frequency (qtf) seems to confuse a lot of students

- You don't need to worry about it in your homework
  - We will not give you duplicate query terms
  - Your BM25 queries will always have qtf=1
- But, if you want to know how it works…

© 2017, Jamie Callan

Okapi BM25 Implementation: qtf

How is qtf implemented in a search engine?

$$\sum_{t\in q\cap d}\left(\log\frac{\mathrm{N}-\mathrm{df}_{\mathrm{t}}+0.5}{\mathrm{df}_{\mathrm{t}}+0.5}\right)\frac{tf_{\mathrm{t,d}}}{tf_{\mathrm{t,d}}+k_1\left((1-b)+b\frac{\mathrm{doclen}_{\mathrm{d}}}{\mathrm{avg\_doclen}}\right)}\;\frac{(k_3+1)\,qtf_{\mathrm{t}}}{k_3+qtf_{\mathrm{t}}}$$

Think of BM25 this way

$$\underbrace{\sum_{t\in q\cap d}}_{\text{WSUM query operator}}\underbrace{w(t)}_{\text{User weight}}\,\underbrace{f(t,d)}_{\text{SCORE operator}}$$

User weights are managed by the WSUM query operator

- E.g., "apple apple pie" → #WSUM (2 apple 1 pie)

© 2017, Jamie Callan

Okapi BM25 Implementation: qtf

QrySopWsum implements a query operator that takes weights

- Query: #WSUM (2 apple 1 pi)
- Object:

```
QrySopWsum
arg_weights: | 2 | 1 |
args: | | |
```

- For the BM25 retrieval model, treat the input values as qtf

The Qry and QrySop ancestor classes don't provide arg_weights

- But, a subclass can add elements

© 2017, Jamie Callan

Okapi BM25 Implementation: qtf

Query term frequency (qtf) seems to confuse a lot of people, so…

Query: #SUM (a b c)

- a: qtf=1
- b: qtf=1
- c: qtf=1

Query: #SUM (a b a)

- a: qtf=2
- b: qtf=1

Easy cases…

© 2017, Jamie Callan

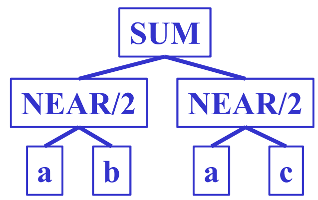
Okapi BM25 Implementation: qtf

Query: #SUM (#NEAR/2 (a b) #NEAR/2 (a c)

- #NEAR/2 (a b): qtf=1
- #NEAR/2 (a c): qtf=1
- a, b, and c don't have qtf because they are arguments to NEAR
  - The SUM operator doesn't see them

If confused, think about the parse tree

- Operators see only their children
- They don't see grandchildren



© 2017, Jamie Callan

The Okapi BMxx Retrieval Models: Summary

Advantages

- Motivated by sound probabilistic theory
- Parameters allow it to be tuned to new environments
- Very effective in a wide variety of evaluations

Disadvantages

- Heuristic tf weighting and document length normalization
- Effects of parameters not immediately obvious

One of the most popular retrieval models for the last 15 years

© 2017, Jamie Callan

Lecture Outline

Introduction

The vector space model (VSM)

- Standard VSM (lnc.ltc)
- Lucene

BM25

49 © 2017, Jamie Callan

For Additional Information

- S. E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. SIGIR 1994.
- S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4). 2009.
- G. Salton. *Automatic Information Organization and Retrieval*. McGraw-Hill. 1968.
- G. Salton. *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice-Hall. 1971.
- G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill. 1983.
- G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley. 1989.

50 © 2017, Jamie Callan