

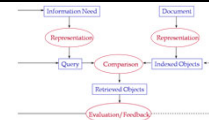
**11-442 / 11-642 / 11-742:
Search Engines**

**Best-Match Retrieval:
VSM, BM25**

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

1

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

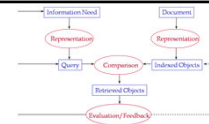
- Easier for many people to use
- Often more accurate
- Considered the state-of-the-art today in many situations

2

© 2021, Jamie Callan

2

Introduction



We started by examining exact match Boolean retrieval models

- Documents either match or do not match the query
- Unranked Boolean, Ranked Boolean

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 $p(d_i | I)$
- More often expressed as Similarity (Q, d_i) or $p(d_i | Q)$
- Many documents match the query to some degree (partial match)

The search result is a ranked list of documents

- “Best first”

3

© 2021, Jamie Callan

3

Introduction

Most best match retrieval models treat the document 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

4

© 2021, Jamie Callan

4

Best Match Retrieval Models

What distinguishes different best-match models?

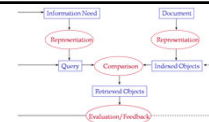
- The theory
 - Important to scientists, but maybe not to practitioners
- How term weights are calculated
 - Most models use the same statistics (tf, df, doclen, numdocs)
- How similarity between query and document is calculated
- Whether the retrieval model can handle structured queries

5

© 2021, Jamie Callan

5

Introduction



This unit covers 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 different theory
... however, they have (mostly) similar architectures

Disagreement about theory, but agreement about what works

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

6

© 2021, Jamie Callan

6

Lecture Outline

Introduction

The vector space model (VSM)

- Standard VSM (Inc.ltc)
- Lucene

BM25

7

© 2021, Jamie Callan

7

The Vector Space Retrieval Model

Representation: Any text can be represented by a term vector

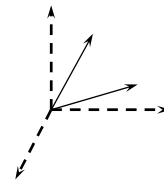
- **Examples:** Documents, queries, sentences,

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's ClassicSimilarity retrieval model



8

© 2021, Jamie Callan

8

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

tf weights

V	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

9

© 2021, Jamie Callan

9

Vector Space Similarity

Overlap measures the similarity of the vocabularies of x and y

- The number of terms they have in common

Overlap is a weak similarity metric

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

Does overlap favor any of these conditions?

- 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

10

© 2021, Jamie Callan

10

Cosine Similarity

Many similarity functions have been developed

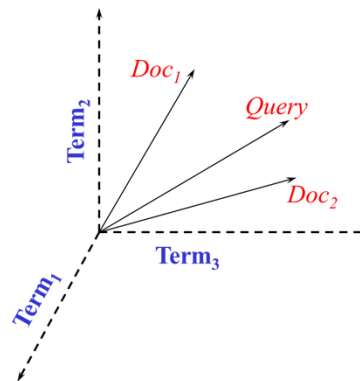
- Inner product, Dice coefficient, Jackard coefficient, ...

Cosine correlation is the most popular

The cosine of the angle between two vectors is 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**



11

© 2021, Jamie Callan

11

Cosine Similarity

tf weights

V	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:

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

Length of y:

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

Sim (x, y):

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

12

© 2021, Jamie Callan

12

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:** Inc.ltc

13

© 2021, Jamie Callan

13

Document Term Weights (D)

How should the importance of term t in document d be represented?

- How well does term t describe document d ?

Some common choices

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

14

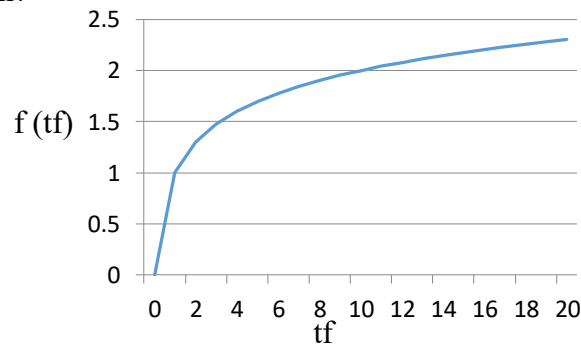
© 2021, Jamie Callan

14

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

15

© 2021, Jamie Callan

15

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

How should the importance of term t in the collection be represented?

- **Observation:** Terms that occur in many documents in the collection are less useful for discriminating among documents (i.e., separating relevant and non-relevant sets)
- The most common choice is inverse document frequency (idf)

Some common forms of idf

$$idf = \log \frac{N}{df}$$

Standard form
(Know this)

$$idf = \log \frac{N+1}{df}$$

To avoid idf=0

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

To avoid idf=0

**N : # documents
in corpus
 df : # documents
containing t**

The three formulas produce similar (but not identical) rankings

16

© 2021, Jamie Callan

16

Inc.ltc

Inc.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:

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^2} \cdot \sqrt{\sum q_i^2}} = \frac{\overset{\text{“tf”}}{\sum (\log(tf) + 1)} \cdot \overset{\text{“user weight”}}{(\log(qtf) + 1)} \overset{\text{“idf”}}{\log \frac{N}{df}}}{\underset{\text{Cosine similarity metric}}{\sqrt{\sum d_i^2}} \cdot \underset{\text{“document length normalization”}}{\sqrt{\sum (\log(tf) + 1)^2}} \cdot \underset{\text{“query length normalization”}}{\sqrt{\sum \left((\log(qtf) + 1) \log \frac{N}{df} \right)^2}}}$$

17

© 2021, Jamie Callan

17

Inc.ltc

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

- Originally, idf represented the term’s importance to the query

Different vector space systems 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 (idf²)

There is no good theory to guide setting vector space weights

18

© 2021, Jamie Callan

18

Vector Space Implementation

It is easy to implement Inc.ltc

$$\frac{\sum d_i \cdot q_i}{\sqrt{\sum d_i^2} \cdot \sqrt{\sum q_i^2}} = \frac{\overbrace{\sum_{t \in d \cap q} (\log(tf_{t,d}) + 1) \cdot \left((\log(qtf_t) + 1) \log \frac{N}{df_t} \right)}^{\text{\#SCORE operator}}}{\underbrace{\sqrt{\sum_{t \in d} (\log(tf_{t,d}) + 1)^2}}_{\text{Document vector length (stored in index, look it up)}} \cdot \underbrace{\sqrt{\sum_{t \in q} \left((\log(qtf_t) + 1) \log \frac{N}{df_t} \right)^2}}_{\text{Query vector length (ignore – it is constant)}}$$

#SUM query operator (indicated by a red arrow pointing to the denominator's square root)

19

© 2021, Jamie Callan

19

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 rarely see anyone use them, so we don't cover them

What about proximity operators?

20

© 2021, Jamie Callan

20

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

21

© 2021, Jamie Callan

21

Lucene

Lucene's ClassicSimilarity uses a two-step retrieval process

1. Use a Boolean query to form a set of documents

- E.g., Unranked Boolean AND
- “Fuzzy” Boolean is also an option

2. Use a vector space retrieval algorithm to rank the set

22

© 2021, Jamie Callan

22

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
 - Field weights are assigned when the index is built
 - » Just look them up from the index when evaluating the query

23

© 2021, Jamie Callan

23

A Simplified View of Lucene's ClassicSimilarity Ranker

Lucene uses a modified vector space algorithm

- A simplified view is $RSV(d, q) = \text{weight}(d) \times \text{coordinationMatch}(d, q) \times \text{cosine}(d, q)$
 - $\text{weight}(d)$: Query-independent importance of d (e.g., PageRank)
 - $\text{coordinationMatch}(d, q)$: % of query terms that match d
 - $\text{cosine}(d, q)$:

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

24

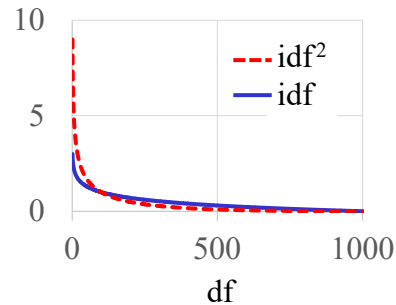
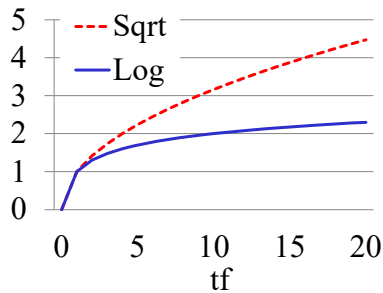
© 2021, Jamie Callan

24

A Simplified View of Lucene's ClassicSimilarity Ranker

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

- $\text{sqrt}(\text{tf})$ instead of $\log(\text{tf})+1$
- idf^2 instead of idf



What are the effects?

- Stronger reward for terms that are frequent in this document
- Stronger penalty for terms that are frequent across the corpus

25

© 2021, Jamie Callan

25

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

26

© 2021, Jamie Callan

26

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

27

© 2021, Jamie Callan

27

Lecture Outline

Introduction

The vector space model (VSM)

- Standard VSM (Inc.ltc)
- Lucene

BM25

28

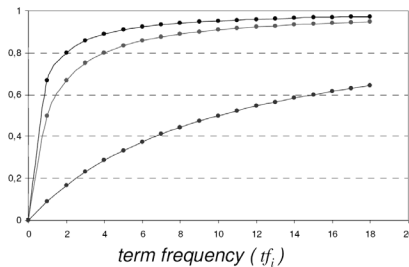
© 2021, Jamie Callan

28

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 good approximation



Document length normalization

- Normalize by $(1-b) + b \frac{doclen}{avg_doclen}$

(Robertson & Zaragoza, 2007)

29

© 2021, Jamie Callan

29

The BM25 Retrieval Model

The BM25 model

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

RSJ weight (idf)
tf weight
user weight

BM25 has been used with many different parameter configurations

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

(* Shane Culpepper, 2014, personal communication)

30

© 2021, Jamie Callan

30

The BM25 Retrieval Model

The BM25 model

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

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

31

© 2021, Jamie Callan

31

The 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”

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

32

© 2021, Jamie Callan

32

The 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

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

33

© 2021, Jamie Callan

33

The 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 b approaches 0?

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

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

34

© 2021, Jamie Callan

34

The BM25 Retrieval Model

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

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

Suppose $df_t = N/2$

- RSJ weight = $\log \frac{N - \frac{N}{2} + 0.5}{\frac{N}{2} + 0.5} = \log \frac{\frac{N}{2} + 0.5}{\frac{N}{2} + 0.5} = \log(1) = 0$
- Matching the term has no effect on the document score

35

© 2021, Jamie Callan

35

The BM25 Retrieval Model

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

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

Suppose $df_t > N/2$ (e.g., $N/2 + 1$)

- RSJ weight = $\log \frac{N - \left(\frac{N}{2} + 1\right) + 0.5}{\left(\frac{N}{2} + 1\right) + 0.5} = \log \frac{\frac{N}{2} + 0.5}{\frac{N}{2} + 1.5} = \log(\text{fraction}) < 0$
- Matching a frequent term lowers a document's score

36

© 2021, Jamie Callan

36

The BM25 Retrieval Model

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

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

Robertson's solution

- Change the RSJ weight to
 - Jamie's code does this

$$\text{Max} \left(0, \log \frac{N - df_t + 0.5}{df_t + 0.5} \right)$$

(Robertson & Zaragoza, 2007)

37

© 2021, Jamie Callan

37

Lucene's BM25 Ranker

Recently, Lucene added BM25 ranking

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

Modified RSJ weight

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

Equivalently

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

38

(Doug Turnbull — October 16, 2015)

© 2021, Jamie Callan

38

BM25 Implementation

BM25 is easy to implement

#SCORE operator

- If #SUM, $qtf_t = 1$
- If #WSUM, query parser sets qtf_t

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

#SUM or #WSUM query operator

- Constants (N, avg_doclen) are stored in the index – look them up
- doclen_d is stored in the index – look it up
- Parameters (b, k₁, k₃) are stored in the retrieval model – look them up

39

© 2021, Jamie Callan

39

BM25: Boolean Queries

BM25 doesn't support Boolean query operators

- Typically it is used only for unstructured queries

40

© 2021, Jamie Callan

40

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, 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

41

© 2021, Jamie Callan

41

BM25 Implementation: qtf

Query term frequency (qtf) seems to confuse many 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...

42

© 2021, Jamie Callan

42

BM25 Implementation: qtf

How is qtf implemented in a search engine?

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

Think of BM25 this way

$$\sum_{t \in q \cap d} w(t) f(t, d)$$

#WSUM query operator SCORE operator User weight

User weights are managed by the #WSUM query operator

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

43

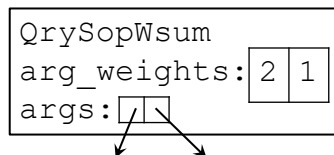
© 2021, Jamie Callan

43

BM25 Implementation: qtf

QrySopWsum implements a query operator that takes weights

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



- 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
- You will do this for Indri in HW2

44

© 2021, Jamie Callan

44

BM25 Implementation: qtf

Query term frequency (qtf) seems to confuse many 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...

45

© 2021, Jamie Callan

45

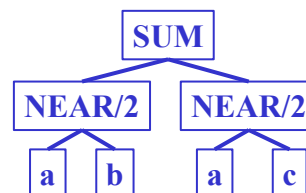
BM25 Implementation: qtf

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

- A query operator only sees its children
- It doesn't see its grandchildren



46

© 2021, Jamie Callan

46

The BM25 Retrieval Model: 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 – tune them for your dataset!

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

47

© 2021, Jamie Callan

47

Lecture Outline

Introduction

The vector space model (VSM)

- Standard VSM (Inc.ltc)
- Lucene

BM25

48

© 2021, Jamie Callan

48

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.