11-442 / 11-642:
Search Engines

Best-Match Retrieval: Statistical Language Models

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

Outline

HW2 implementation

- Indri default beliefs
- Window operator

Indri Implementation

Mostly the Indri query operators are easy to implement

- See the preceding slide for the score calculations
- **Calculate scores only for documents that contain a query term**
  - Use inverted or score lists – similar to HW1
- Use document length, ctf, and corpus length for smoothing
  - Lookup from the index – see the HW2 web page

But, one aspect is a little tricky to get right…

3

Indri Implementation

Query: #or (a #and (b c)) **Document:** a

Query terms b and c do not appear in this document
… what is the score of the #AND operator?

- $tf_b = 0$ $tf_c = 0$
- Do the usual Indri score calculation
  - So, only smoothing scores for b and c

This is simple conceptually, but how is it implemented?

- You don't want to calculate #AND scores for every document
  … just the documents that have at least one query term

4

Indri Implementation

Query: #or (a #and (b c)) **Document:** a

Add a new method to all QrySop operators

double getDefaultScore (RetrievalModel r, long docid)

When any QrySop operator calculates scores

If the i^{th} query argument contains document d
then call the i^{th} query argument's getScore method
else call the i^{th} query argument's getDefaultScore method

Indri Implementation

Query: #or (a #and (b c)) **Document:** a

QrySopScore.getDefaultScore (RetrievalModel r, long docid)

- The standard Indri SCORE calculation done with tf=0

If r == RetrievalModel.Indri

$$p_{scoreDefault}(t \mid \text{docid}) = (1-\lambda)\frac{0 + \mu\, p_{MLE}(t \mid C)}{length(\text{docid}) + \mu} + \lambda\, p_{MLE}(t \mid C)$$

This is the main difference.
Do the usual calculation, but with tf=0.

Indri Implementation

Query: #and (a #near/3 (b c))

QrySopScore.getDefaultScore (RetrievalModel r, long docid)

What happens if #near/3 (b c) doesn't occur in the collection?

- Its ctf == 0
  - … so $p_{MLE}(t|C)$ == 0 (i.e., no smoothing weight)
  - … so #AND returns 0 for all documents $p_{and}(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)^{\frac{1}{|q|}}$
- This behavior is exact match, not best match
  - We want Indri to be a best match model

Indri Implementation

Query: #and (a #near/3 (b c))

QrySopScore.getDefaultScore (RetrievalModel r, long docid)

What happens if #near/3 (b c) doesn't occur in the collection?

Solution: Extra smoothing for terms that have ctf = 0

If r == RetrievalModel.Indri
  If ctf (t) = 0
    calculate $p_{MLE}(t|C)$ using ctf (t) = 0.5

Undocumented behavior

$$p_{scoreDefault}(t \mid \text{docid}) = (1-\lambda)\frac{0 + \mu\ p_{MLE}(t \mid C)}{length(\text{docid}) + \mu} + \lambda\ p_{MLE}(t \mid C)$$

Indri Implementation

Query: #or (a #and (b c)) **Document:** a

QrySopAnd.getDefaultScore (RetrievalModel r, long docid)

- The standard Indri AND calculation done on the default score of each argument

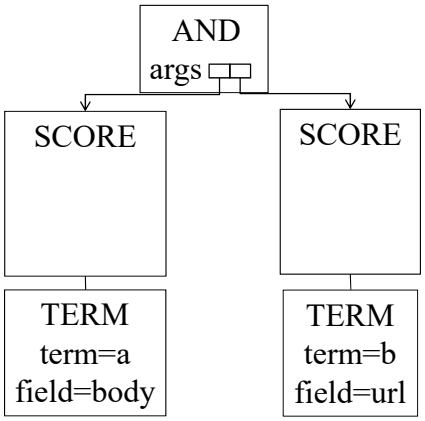
  If r == RetrievalModel.Indri

$$p_{andDefault}(q \mid d) = \prod_{q_i \in q} p_{q_i default}(q_i \mid d)^{\frac{1}{|q|}}$$

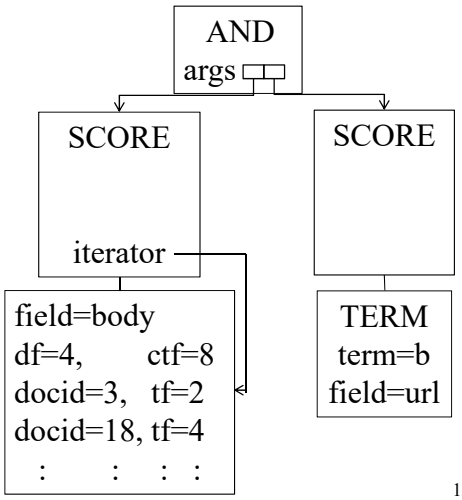
This is the only difference.
Call the i^{th} query argument's getDefaultScore method.

QryEval Example

The initial query:
#AND (a.body b.url)

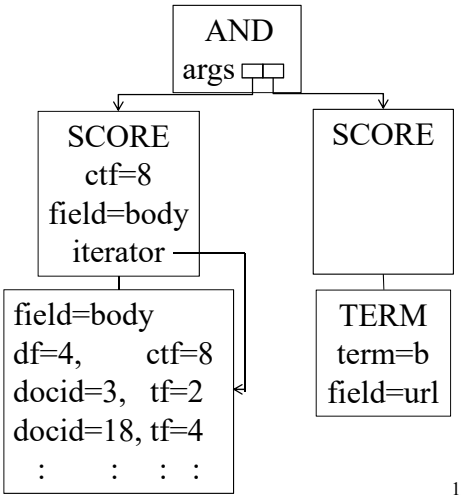


Query Initialization



AND operator initialization
AND initializes its first arg.
SCORE initializes its arg.
The result is an inverted list.

Query Initialization

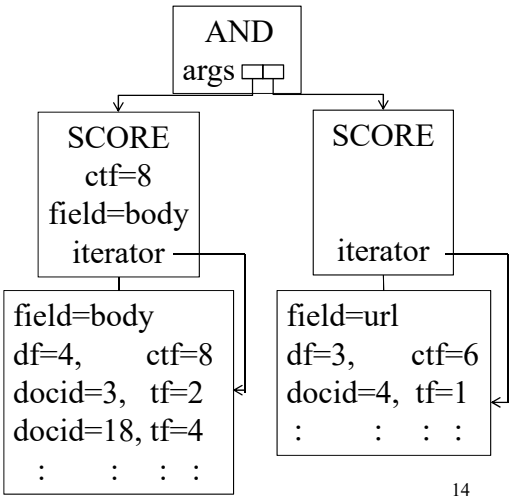


AND operator initialization
AND initializes its first arg.
SCORE initializes its arg.
The result is an inverted list.

The SCORE operator caches information that it will need later.

13

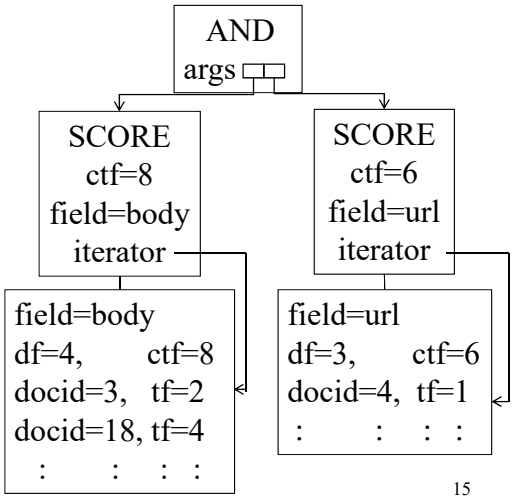
Query Initialization



AND operator initialization
AND initializes its second arg.
SCORE initializes its arg.
The result is an inverted list.

14

Query Initialization



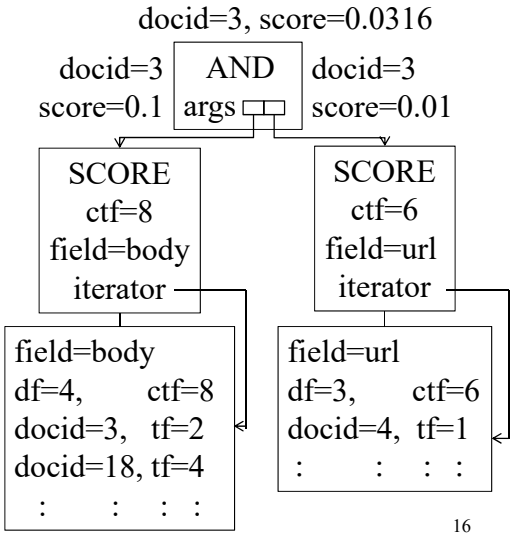
AND operator initialization

AND initializes its second arg.
SCORE initializes its arg.
The result is an inverted list.

The SCORE operator caches information that it will need later.

15

Call to docIteratorHasMatch & getScore (First Time)



AND Operator Evaluation

Min document is 3.

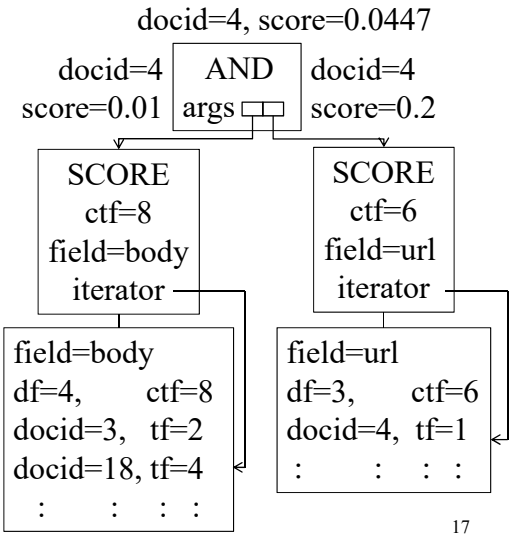
args[0] matches, so call
args[0].getScore ().
Suppose the result is 0.1.

args[1] does not match, so call
args[1].getDefaultScore(3).
Suppose the result is 0.01.

$Score_{AND}(3) = (0.1^{0.5} \times 0.01^{0.5})$

16

Call to docIteratorHasMatch & getScore (Second Time)



AND Operator Evaluation

Min document is 4.

args[0] does not match, so call args[0].getDefaultScore (4). Suppose the result is 0.01.

args[1] matches, so call args[1].getScore(). Suppose the result is 0.2.

$Score_{AND}(4) = (0.01^{0.5} \times 0.2^{0.5})$

17

Default Belief Scores Are Only a Small Complication

When evaluating a query argument

- If it matches the current document
  - Ask the query argument to calculate the document score for the current document
  - Else ask the query argument to calculate a default score for the current document
    - » E.g., a SCORE operator (the example given)
    - » E.g., an OR operator (similar logic)

18

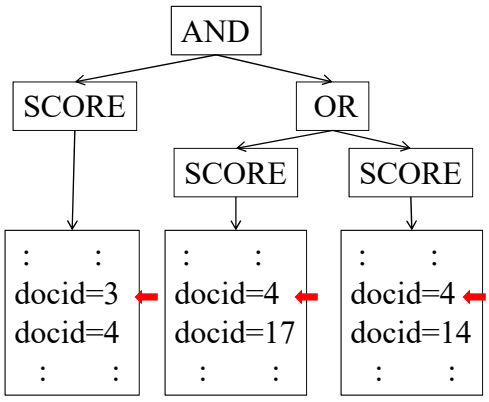
Default Belief Scores Are Only a Small Complication

Which types of query operators calculate default scores?

- If an operator calculates scores
  … it also calculates default scores
- QrySop operators calculate default scores
- QryIop operators do not calculate default scores

19

Call to docIteratorHasMatch & getScore (First Time)



AND Operator Evaluation

Min document is 3.

args[0] matches, so call args[0].getScore ().
Suppose the result is 0.3.

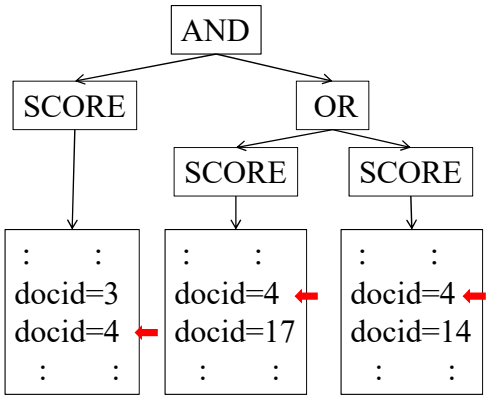
args[1] does not match, so call args[1].getDefaultScore(3).

OR calls getDefaultScore(3) for all of its args and computes a score. Suppose it is 0.01.

$Score_{AND}(3) = (0.3^{0.5} \times 0.01^{0.5})$

20

Call to docIteratorHasMatch & getScore (Second Time)



AND Operator Evaluation

Min document is 4.

args[0] matches, so call
args[0].getScore ().
Suppose the result is 0.3.

args[1] matches, so call
args[1].getScore().
Suppose the score is 0.2.

$Score_{AND}(4) = (0.3^{0.5} \times 0.2^{0.5})$

Using Default Belief Scores Properly Requires Two Components

Add a new method to all QrySop operators

double getDefaultScore (RetrievalModel r, long docid)

– QrySopScore.getDefaultScore calculates a score for a term

– QrySop <other>. getDefaultScore combines scores for *n* terms

When any QrySop operator calculates scores

If the i^{th} query argument contains document d

then read its score from the i^{th} score list

else call the i^{th} query argument's getDefaultScore method

It may sound complicated now, but actually it is very easy

Outline

Statistical language models

- Introduction
- Query likelihood model
- Kullback-Leibler (KL) Divergence
- Indri

HW2 implementation

- Indri default beliefs
- Window operator

Proximity Operators: The Window (or Unordered Window) Operator

The WINDOW/n operator is used to match related concepts

- Arguments can be in any order
- n specifies the maximum distance between any pair of terms

Examples

- WINDOW/100 (obama merkel putin)
  - We don't care which order they occur in

Typically proximity operators have complexity O(|C|)

- A single pass down each inverted list

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

25

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Initialize doc iterators

26

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Advance all doc iterators until they point to the same document

- This is a simple nested loop

27

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Same document
Initialize location iterators

28

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Find the min (47) and max (48) locations

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Is (max – min) < window?

48 – 47 < 20 (match)

Record match

- max location (48)

Proximity Operators: The Window (or Unordered Window) Operator

a	
df:	47
doc:	19
tf:	1
locs:	7
doc:	27
tf:	3
locs:	47
	98
	132
doc:	92
…	

b	
df:	95
doc:	23
tf:	1
locs:	99
doc:	27
tf:	4
locs:	48
	49
	133
	134
doc:	148
...	

Query: # WINDOW/20 (a b)

Increment all loc iterators

31

Proximity Operators: The Window (or Unordered Window) Operator

a	
df:	47
doc:	19
tf:	1
locs:	7
doc:	27
tf:	3
locs:	47
	98
	132
doc:	92
…	

b	
df:	95
doc:	23
tf:	1
locs:	99
doc:	27
tf:	4
locs:	48
	49
	133
	134
doc:	148
...	

Query: # WINDOW/20 (a b)

Find the min (49) and max (98) locations

32

Proximity Operators: The Window (or Unordered Window) Operator

a	
df:	47
doc:	19
tf:	1
locs:	7
doc:	27
tf:	3
locs:	47
	98
	132
doc:	92
…	

b	
df:	95
doc:	23
tf:	1
locs:	99
doc:	27
tf:	4
locs:	48
	49
	133
	134
doc:	148
...	

Query: # WINDOW/20 (a b)

Is (max – min) < window?

98 – 49 ≥ 20 (no match)

33

Proximity Operators: The Window (or Unordered Window) Operator

a	
df:	47
doc:	19
tf:	1
locs:	7
doc:	27
tf:	3
locs:	47
	98
	132
doc:	92
…	

b	
df:	95
doc:	23
tf:	1
locs:	99
doc:	27
tf:	4
locs:	48
	49
	133
	134
doc:	148
...	

Query: # WINDOW/20 (a b)

Increment the iterator for the min location

34

Proximity Operators: The Window (or Unordered Window) Operator

a

df: 47
doc: 19
tf: 1
locs: 7
doc: 27
tf: 3
locs: 47
98
132
doc: 92
…

b

df: 95
doc: 23
tf: 1
locs: 99
doc: 27
tf: 4
locs: 48
49
133
134
doc: 148
...

Query: # WINDOW/20 (a b)

Find the min (98) and max (133) locations

Proximity Operators: The Window (or Unordered Window) Operator

a

df: 47
doc: 19
tf: 1
locs: 7
doc: 27
tf: 3
locs: 47
98
132
doc: 92
…

b

df: 95
doc: 23
tf: 1
locs: 99
doc: 27
tf: 4
locs: 48
49
133
134
doc: 148
...

Query: # WINDOW/20 (a b)

Is (max – min) < window?

133 – 98 ≥ 20 (no match)

Proximity Operators: The Window (or Unordered Window) Operator

a	
df:	47
doc:	19
tf:	1
locs:	7
doc:	27
tf:	3
locs:	47
	98
	132
doc:	92
…	

b	
df:	95
doc:	23
tf:	1
locs:	99
doc:	27
tf:	4
locs:	48
	49
	133
	134
doc:	148
...	

Query: # WINDOW/20 (a b)

Increment the iterator for the min location

37

Proximity Operators: The Window (or Unordered Window) Operator

a	
df:	47
doc:	19
tf:	1
locs:	7
doc:	27
tf:	3
locs:	47
	98
	132
doc:	92
…	

b	
df:	95
doc:	23
tf:	1
locs:	99
doc:	27
tf:	4
locs:	48
	49
	133
	134
doc:	148
...	

Query: # WINDOW/20 (a b)

Find the min (132) and max (133) locations

38

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Is (max – min) < window?

133 – 132 < 20 (match)

Record match
- max location (133)

39

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Increment all loc iterators

q_0 locs are exhausted.
No more matches are possible in this document

40

Proximity Operators: The Window (or Unordered Window) Operator

a		b	
df:	47	df:	95
doc:	19	doc:	23
tf:	1	tf:	1
locs:	7	locs:	99
doc:	27	doc:	27
tf:	3	tf:	4
locs:	47	locs:	48
	98		49
	132		133
doc:	92		134
…		doc:	148
		...	

Query: # WINDOW/20 (a b)

Increment all doc iterators

…

Continue until the inverted lists are exhausted

41

Proximity Operators: The Window (or Unordered Window) Operator

Implementation note

- A document term can only match the query once
- **Query:** #WINDOW/100 (obama merkel putin)
- **Document**: obama … merkel … putin … merkel … obama
- There is just one match here

One can imagine other implementations, but this is the norm

- Usually more complicated matching doesn't improve accuracy

42