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

Overview of the QryEval Software

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

1

Outline

- **QryEval software overview**
  - What it does (conceptually)
  - Applications
  - Classes
- **Query evaluation**
  - The Qry class
  - Iteration
  - Matching
  - Calculating scores
- **Overview of query parsing**

2

2

QryEval: What it Does

QryEval is a software application that conducts experiments

- Read a parameter file
  - The parameter file configures your system for an experiment
    - » On your laptop and in the homework testing service
  - Example parameter file for HW1

```
indexPath=someDirectory/index-gov2
retrievalAlgorithm=UnrankedBoolean
queryFilePath=queries.txt
trecEvalOutputPath=HW1-queries-UB.teIn
trecEvalOutputLength=100
```

  - Each homework will have additional parameters

3

3

QryEval: What it Does

QryEval is a software application that conducts experiments

- Read a parameter file
- Read a query file (one query per line)

```
10:#OR(cheap internet)
26:#AND(lower heart rate)
52:#AND (#OR (apple blueberry) pie)
71:living in india
```

4

4

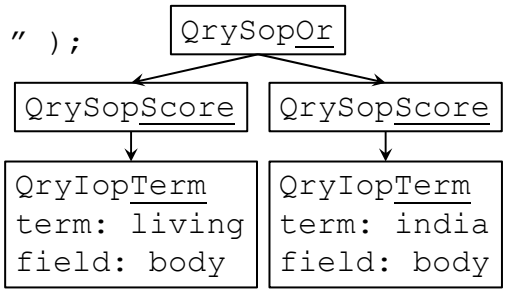
QryEval: What it Does

QryEval is a software application that conducts experiments

- Read a parameter file
- Read a query file (one query per line)
  - Parse the query

```
Qry q = QryParser.getQuery (
            "#or(living in india)" );
```

  - More on query parsing later in the lecture …



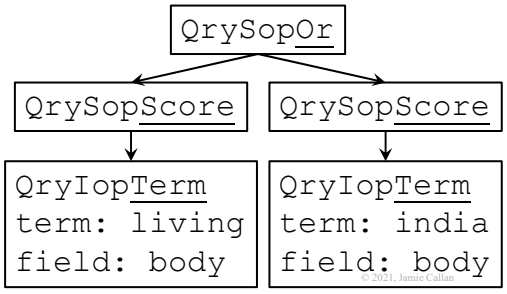
5

5

QryEval: What it Does

QryEval is a software application that conducts experiments

- Read a parameter file
- Read a query file (one query per line)
  - Parse the query ("living in india")
    - » Qry: Abstract class for all operators
    - » QrySop: Parent class of all score operators
    - » QrySopOr: A particular score operator
    - » Iop: Inverted list operator



6

6

QryEval: What it Does

QryEval is a software application that conducts experiments

- Read a parameter file
- Read a query file (one query per line)
  - Parse the query
  - Evaluate the query, using a DAAT architecture

```
model = new RetrievalModelUnrankedBoolean();
q.initialize (model);
while (q.docIteratorHasMatch (model)) {
  int docid = q.docIteratorGetMatch ();
  double score = ((QrySop) q).getScore (model);
  result.add (docid, score);
  q.docIteratorAdvancePast (docid);
}
```

Initialize the query

Each pass of this loop retrieves one document

7

7

QryEval: What it Does

QryEval is a software application that conducts experiments

- Read a parameter file
- Read a query file (one query per line)
  - Parse the query
  - Evaluate the query
  - Write the results for the query to a file

```
11 Q0 GX270-76-5299838 1   3.000  HW1-2a
11 Q0 GX000-25-2008761 2   2.000  HW1-2a
11 Q0 GX000-72-8784276 3   2.000  HW1-2a
```

Query Id | **Always Q0** | **Doc External Id** | **Doc Rank** | **Doc Score** | **Run ID (your choice)**

8

8

QryEval: What it Does

QryEval is a software application that conducts experiments

- Read a parameter file
- Read a query file (one query per line)
  - Parse the query
  - Evaluate the query
  - Write the results for the query to a file

Conceptual overview of HW1

Overview of the QryEval Software: Applications

QryEval: An application that conducts experiments

InspectIndex: An application for examining the index
- Helpful for debugging
- Helps you get a sense of what the index looks like
- You can ignore it if you don't need it

Overview of the QryEval Software: Classes

RetrievalModel: Define the model and its parameters (if any)

- RetrievalModelXxx: UnrankedBoolean, RankedBoolean, …

QryParser: Parse a text query into a query tree

Qry: Create and evaluate query operators

- QryIop: Parent class for inverted list operators
  - QryIopXxx: Xxx will be Syn, Near, or Window
- QrySop: Parent class for score list operators
  - QrySopXxx: Xxx will be And, Or, Sum, Wand, Wsum

11

11

Overview of the QryEval Software: Classes

Idx: Access the index

Data structures & utilities

- InvList: Create and access inverted lists
- ScoreList: Create and access score lists
- TermVector: Create and access term vectors (forward index)
- PopData: Tuples (you can ignore this)
- Timer: A simple timer that you may find useful

12

12

Overview of the QryEval Software: Classes

See the documentation on the website for more details

PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

Package <Unnamed>

Class Summary

Class	Description
Idx	Idx manages and provides access to Lucene indexes and auxiliary data structures.
InspectIndex	A simple commandline utility for inspecting Lucene 8 indexes.
InvList	This class implements the inverted list data structure and provides methods for accessing and manipulating inverted lists.
PopData<PopType,RemainingType>	PopData is a simple utility class for returning two items (a tuple of length two) from a method.
Qry	The root class in the query operator hierarchy.
QryEval	This software illustrates the architecture for the portion of a search engine that evaluates queries.
QryIop	All query operators that return inverted lists are subclasses of the QryIop class.
QryIopSyn	The SYN operator for all retrieval models.
QryIopTerm	The TERM operator for all retrieval models.

13

13

Outline

- QryEval overview
  - What it does (conceptually)
  - Applications
  - Classes
- **Query evaluation**
  - The Qry class
  - Iteration
  - Matching
  - Calculating scores
- Overview of query parsing

14

14

Qry

Qry is the base class for all query operators

- Data that every query operator has (e.g., query arguments)
- Methods that work for all query operators (e.g., appendArg)
- Abstract methods that every query operator must define (e.g., HasMatch, toString)

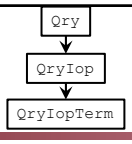
Qry has two subclasses: QryIop and QrySop

- QrySop: The base class for operators that return score lists
  - SCORE, OR, AND, SUM, WAND, WSUM, …
- QryIop: The base class for operators that return inverted lists
  - TERM, NEAR, WINDOW

15

15

Parts of an Object May Be Defined By Different Parts of the Class Hierarchy



QryIopTerm object: An object for accessing information about a term

```
String displayName;
ArrayList<Qry> args;
boolean matchStored;
int matchingDocid;
InvList invertedList;
int docIteratorIndex;
Int locIteratorIndex;
String field;
String term;
```

Defined by Qry

Defined by QryIop

Defined by QryIopTerm

Defined in multiple places, but stored in one place … the object

- Each object has its own args, invertedList, term string, …

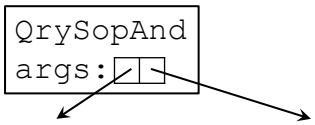
16

16

Qry

Qry is the base class for all query operators

- It defines a place to store query operator arguments & other data
  - Conceptually: #AND (apple pi)
  - Actually:



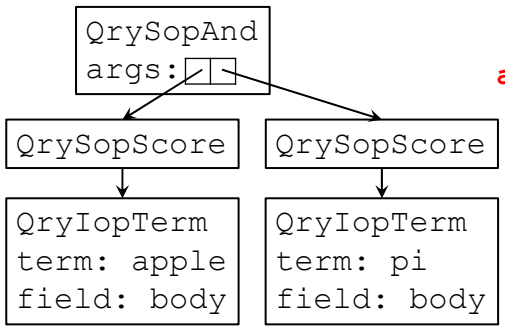
17

17

Qry

Qry is the base class for all query operators

- It controls how arguments are appended to the query operator
  - E.g., automatically insert a #SCORE operator between a scoring (Sop) and an inverted list (Iop) operator



`args` arrays for QrySopScore and QryIopTerm are not shown to save space

18

18

Qry

Qry is the base class for all query operators

- It defines docIterators to iterate over matching documents

```
docIteratorHasMatch     // Each subclass defines
      :     :     :
docIteratorGetMatch     // Get from HasMatch cache
docIteratorAdvancePast  // Advance past a docid
docIteratorAdvanceTo    // Advance to a docid
```

  – These are <u>not</u> Java-style iterators
    » <u>Really essential point</u>

19

19

Qry

Qry is the base class for all query operators

- It defines docIterators to iterate over matching documents

```
docIteratorHasMatch     // Each subclass defines
```

  – There are a few standard matching styles that are implemented as utility methods

```
docIteratorHasMatchAll  // Doc matches all args
docIteratorHasMatchMin  // Doc is min of all args
```

  – When you implement `docIteratorHasMatch` for a new [query operator, retrieval model] pair, consider whether one of the standard utility methods meets your needs
    » E.g., use `docIteratorHasMatchMin` for an OR operator

20

20

Iteration

Query evaluation is divided into three parts

1. Get all inverted lists from the disk
2. Get the docid of the next document that matches the query
3. Get the score of docid (which must be a matching document)

Iteration

Query evaluation is divided into three parts

1. Get all inverted lists from the disk
   - Done during <u>query initialization</u>
   - There are two ways of obtaining inverted lists
     - » Read from disk, e.g., "apple"
     - » Construct dynamically, e.g., "#NEAR/3 (lady gaga)"
   - Assumption: Everything fits into RAM
     - » This is a simple system for homework
     - » A production system might process inverted lists in blocks to control memory usage

Iteration

Query evaluation is divided into three parts

1. Get all inverted lists
2. Get the docid of the next document that matches the query
   - Iterate over (actual) inverted lists and (virtual) score lists
   - There are only a few types of matching strategies
     - » all query arguments match the document ("intersection")
     - » any query argument matches the document ("union")
   - The retrieval model determines what is considered a match
     - » E.g., Ranked Boolean: AND must match all arguments
     - » E.g., Indri: AND must match at least one argument

23

23

Iteration

Query evaluation is divided into three parts

1. Get all inverted lists
2. Get the docid of the next document that matches the query
3. Get the score of docid (which must be a matching document)
   - The retrieval model determines how the score is calculated
   - The QrySop operators do the calculation

24

24

Iteration

The retrieval model determines how a query operator iterates

- Ranked Boolean #AND and Indri #AND iterate differently

However, there are a few "typical" styles of iteration

- E.g., HasMatchFirst, HasMatchAll, HasMatchMin
- These are defined in Qry.java
- Often individual query operators just call one of the standard methods
  - Before you implement something, consider whether a standard method meets your needs

25

25

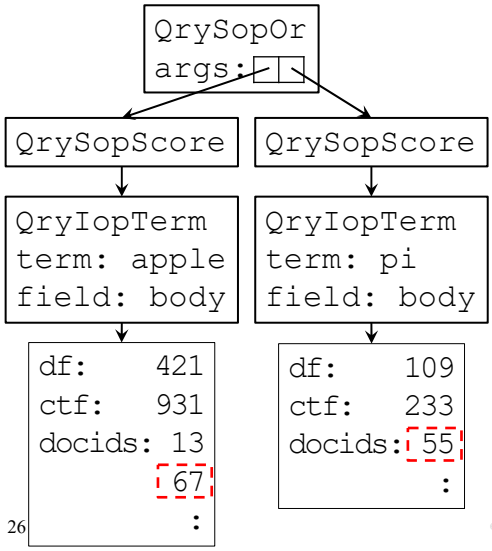
Method QrySopOr.docIteratorHasMatch

The OR operator matches if any argument matches

It uses docIteratorHasMatchMin

- Iterate over the arguments
- Ask each argument to return its current docid
- Return the minimum docid

Recursion handles complex queries naturally



26

26

Caching

When a docIteratorHasMatch matches a document, it caches the docid to improve efficiency

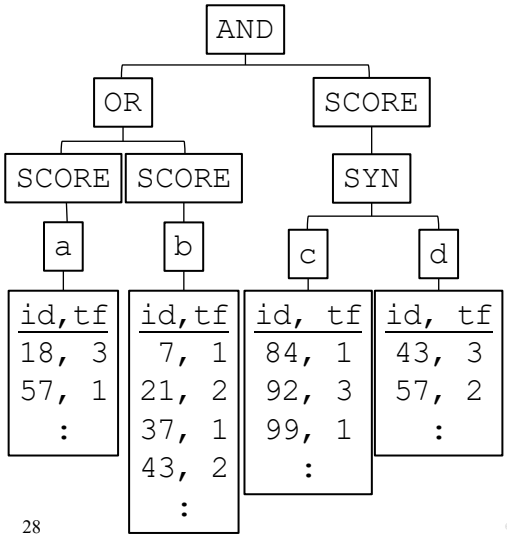
- docIteratorGetMatch just reads the docid from the cache
- Why not have HasMatch just return the docid?
  - If there is no match, it would need to return an invalid docid or throw an exception
  - Those seem messier to me than HasMatch + GetMatch

27

27

Matching and Scoring

What is the first document that matches this query?



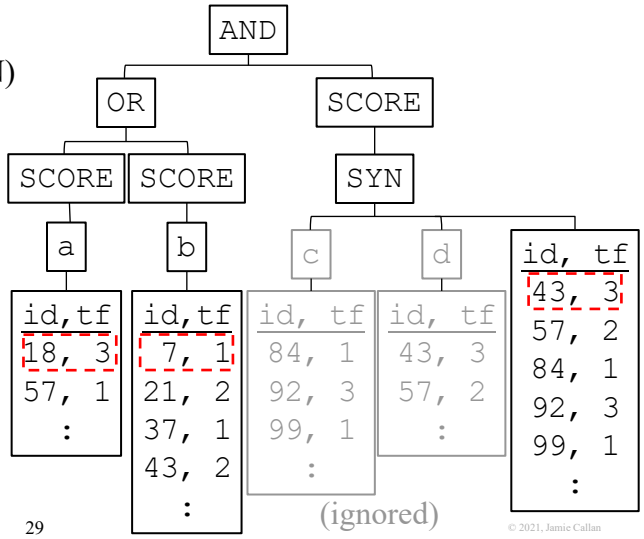
28

28

Matching and Scoring

Call q.initialize()

- Inverted list query operators (e.g., SYN) are materialized
  - Converted to inverted lists
- Iterators are initialized



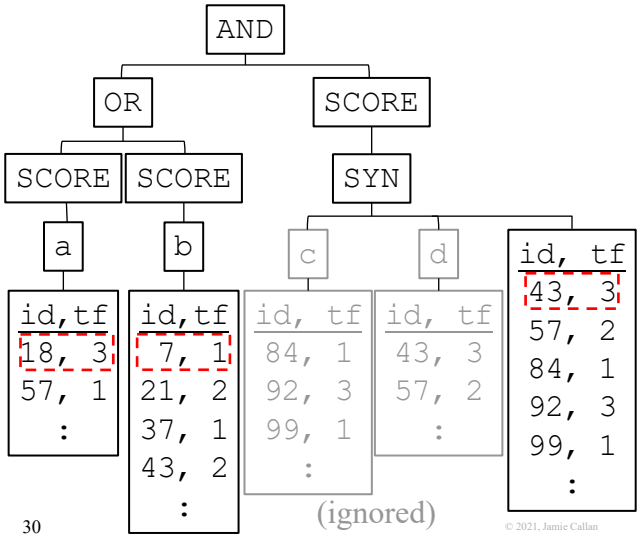
29

29

Matching and Scoring

Call q.docIteratorHasMatch on the AND operator

- The query is structured, so it passes the request to its children
- The children are query operators, so they pass it to their children
- Eventually the requests reach inverted lists



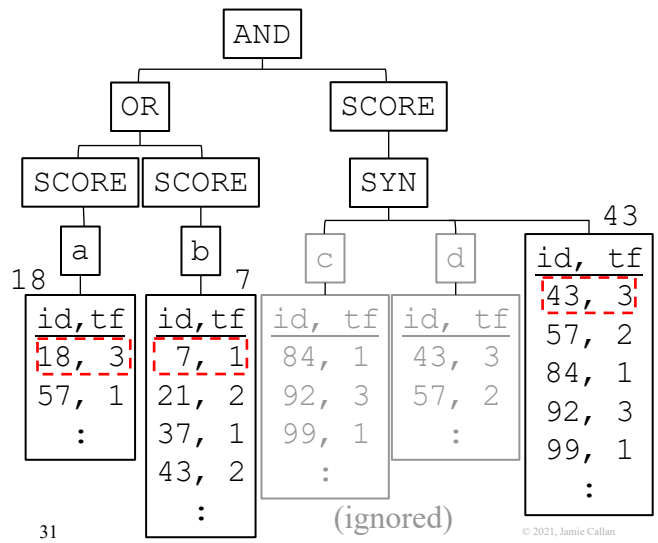
30

30

Matching and Scoring

The inverted lists return their next matching docid

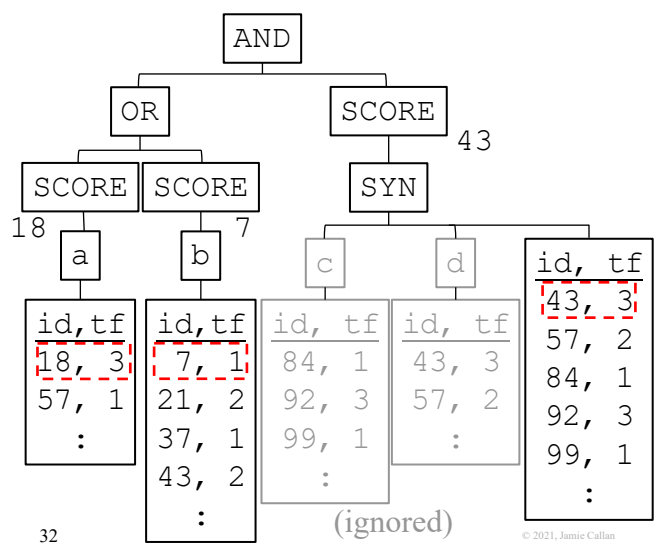
- The ids get passed upwards



31

Matching and Scoring

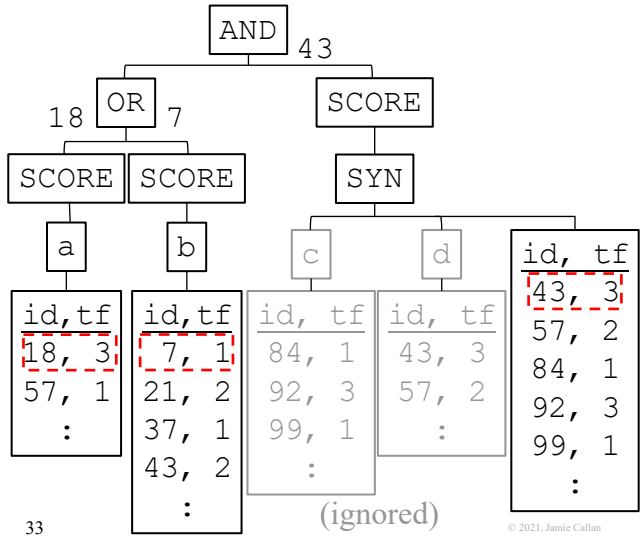
Each operator calculates its next matching document



32

Matching and Scoring

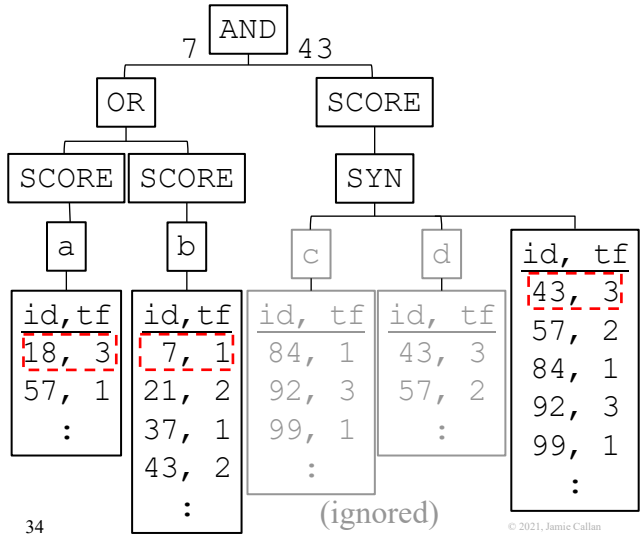
Each operator calculates its next matching document



33

Matching and Scoring

Each operator calculates its next matching document

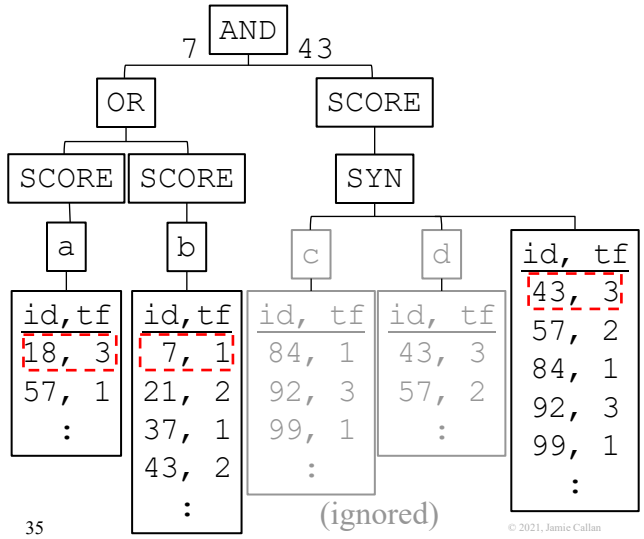


34

Matching and Scoring

Each operator calculates its next matching document

- AND doesn't have a match
- It knows that the next match must have docid ≥ 43
- It tells all children to advance their iterators to docid 43 (or the next docid after 43)

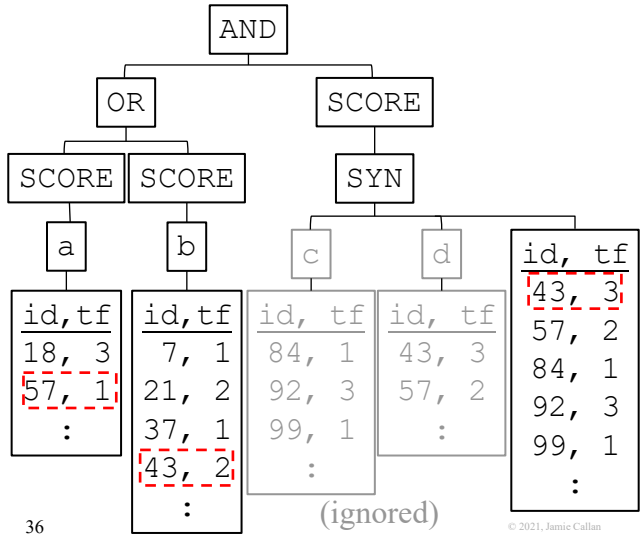


35

Matching and Scoring

AND polls its children again (this is a while loop)

- The children are query operators, so they pass it to their children
- Eventually the requests reach inverted lists

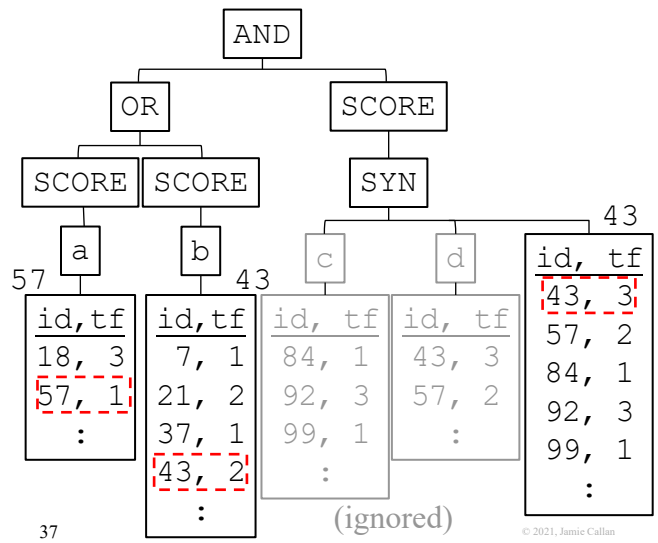


36

Matching and Scoring

The inverted lists return their next matching docid

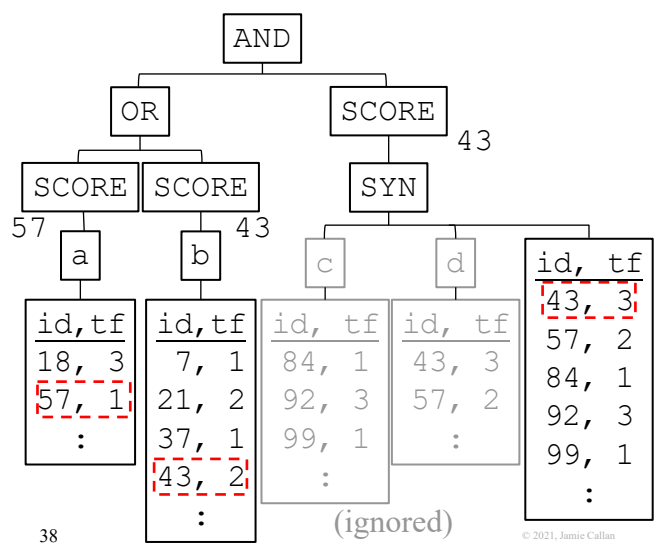
- The ids get passed upwards



37

Matching and Scoring

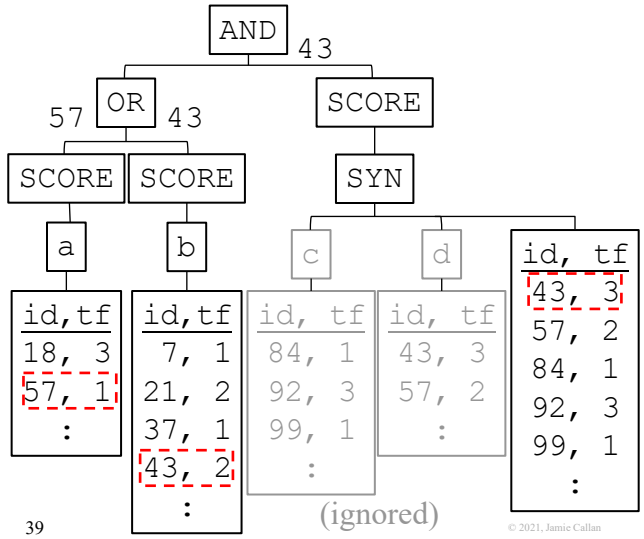
Each operator calculates its next matching document



38

Matching and Scoring

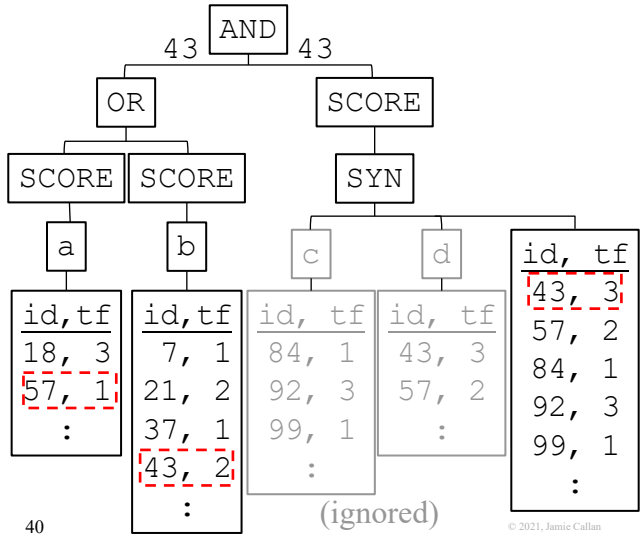
Each operator calculates its next matching document



39

Matching and Scoring

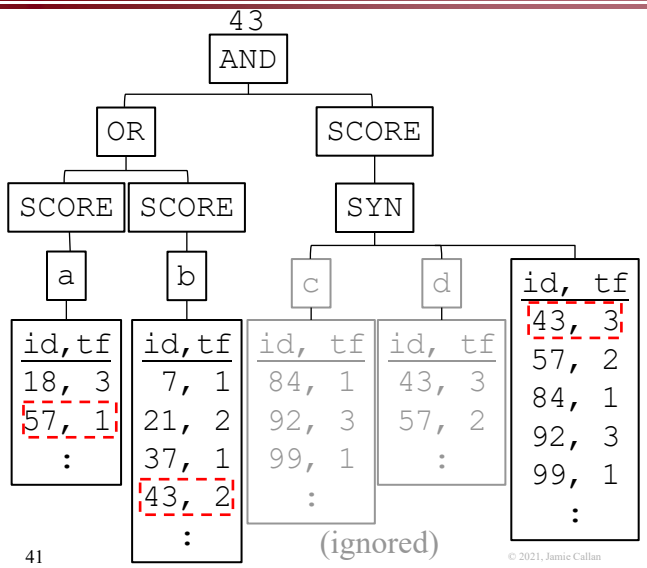
Each operator calculates its next matching document



40

Matching and Scoring

Each operator calculates its next matching document



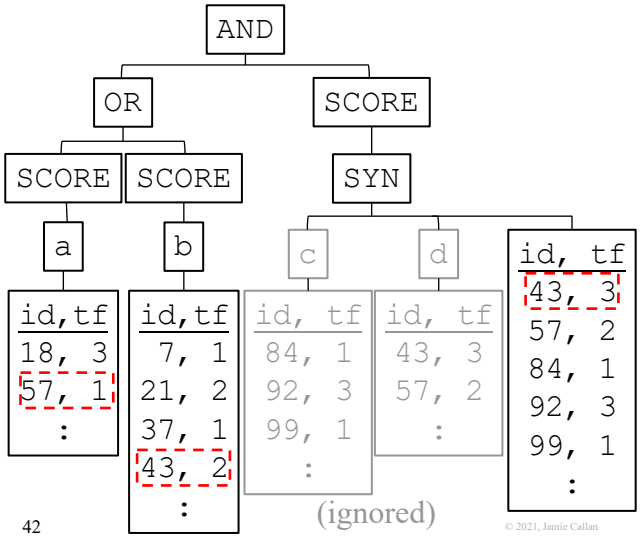
41

41

Matching and Scoring

Call q.getScore on the AND operator

- AND knows that it matches docid 43
  - q.docIteratorHasMatch cached it
- The query is structured, so AND passes the request to its children
- The children are query operators, so they pass it to their children
- Eventually the requests reach SCORE operators



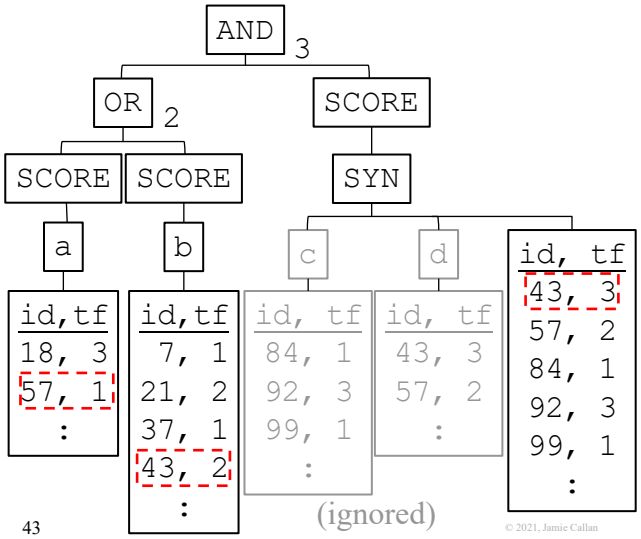
42

42

Matching and Scoring

The SCORE operators return scores

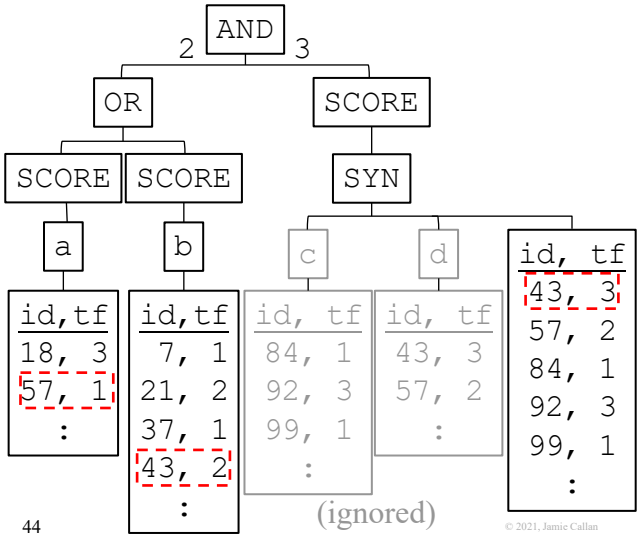
- Assume ranked Boolean



43

Matching and Scoring

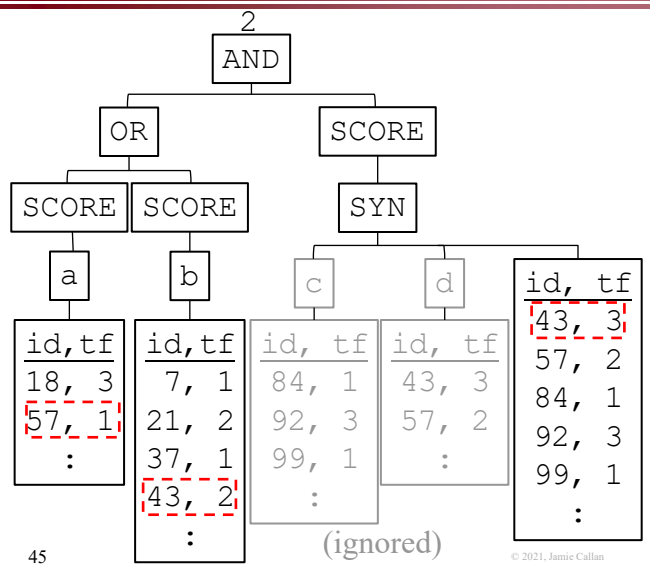
The OR operator combines scores



44

Matching and Scoring

The AND operator combines scores



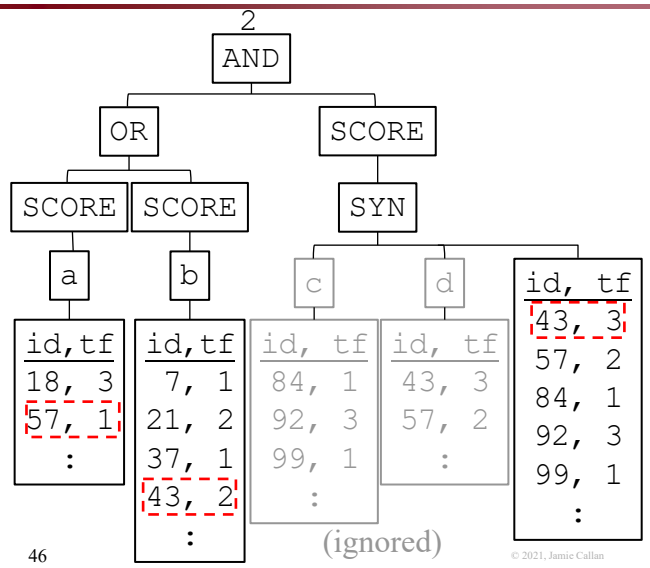
45

Matching and Scoring

Easy!

Advance past docid 43

Repeat for the next match



46

Calculating Scores

Each score operator (QrySopXxx) implements getScore (RetrievalModel r)

- Traverse the query to calculate a score for the current docid
  - We know that it matches, so just calculate a score
- The retrieval model tells the operator what strategy to use for calculating the score
  - For HW1, RankedBoolean and UnrankedBoolean
    - » Unranked Boolean: Score is 1.0 for all matches
    - » Ranked Boolean: Score is ≥ 1.0 for all matches
  - For HW2, BM25 and Indri
  - Retrieval models for HW2 will also store parameters

47

47

Qry Class Summary

The Qry class implements DAAT scoring

- Iterate over (actual) inverted lists and (virtual) score lists
- Several general ways to match a query operator to a document
  - Match all arguments, any argument, …
- Allows you to add different ways to calculate document scores
  - Unranked boolean, ranked boolean, …
- Much use of inheritance and recursion
  - Minimizes redundant implementation ☺
  - Requires greater understanding ☹

48

48

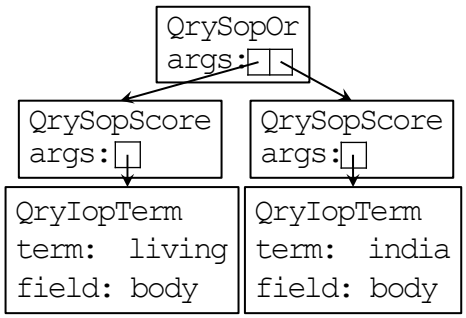
Outline

- **QryEval overview**
  - What it does (conceptually)
  - Applications
  - Classes
- **Query evaluation**
  - The Qry class
  - Iteration
  - Matching
  - Calculating scores
- **Overview of query parsing**

Query Parsing

QryParser is a simple query parser

```
Qry q = QryParser.getQuery (
            "#or(living in india)" );
```



Query Parsing

Pop the operator from the string

- Leftmost token, starts with '#'
- Create the operator (e.g., QrySopOr)

```
#OR(a b c)

#AND(a #OR(b c) d)
```

Find the list of query arguments

- Delimited by leftmost '(' and its matching ')'
- For each argument
  - – If it is not a stopword
    - » If it is a term, create a term object (i.e., QryIopTerm)
    - » Otherwise, recursively call the query parser
    - » Add the result (a Qry object) to the operator argument list
      - • A #SCORE operator may be inserted automatically

51

51

Query Parsing

```
#AND(a b c)
```

52

52

Query Parsing

`#AND(a b c)`

operator

Allocate an AND operator.

```
QrySopAnd
args:
```

53

53

Query Parsing

`#AND(a b c)`

Left '(' and right ')'

```
QrySopAnd
args:
```

54

54

Query Parsing

#AND(a b c)

arguments

```
QrySopAnd
args:
```

55

55

Query Parsing

#AND(a b c)

arguments

Allocate a TERM operator. Append it as an argument to the AND operator.

```
QrySopAnd
args:
```

```
QrySopScore
args:
```

```
QryIopTerm
term:  a
field: body
```

Note: Qry.appendArg automatically inserts a #SCORE operator

56

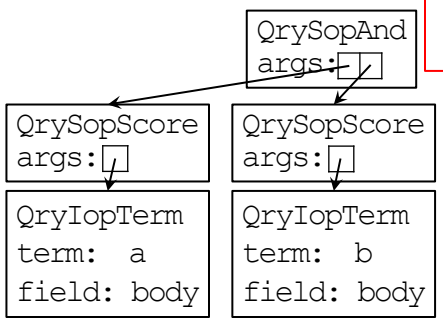
56

Query Parsing

#AND(a b c)

arguments

Allocate a TERM operator. Append it as an argument to the AND operator.



Note: Qry.appendArg automatically inserts a #SCORE operator

57

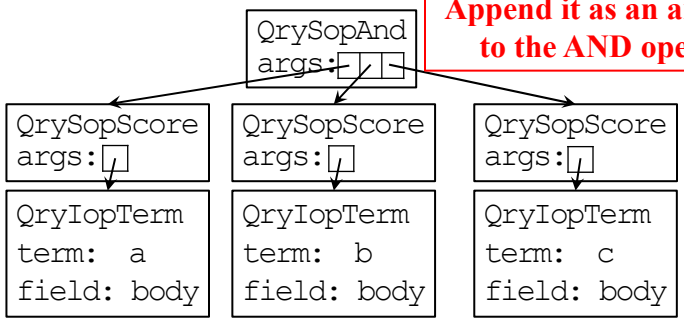
57

Query Parsing

#AND(a b c)

arguments

Allocate a TERM operator. Append it as an argument to the AND operator.



Note: Qry.appendArg automatically inserts a #SCORE operator

58

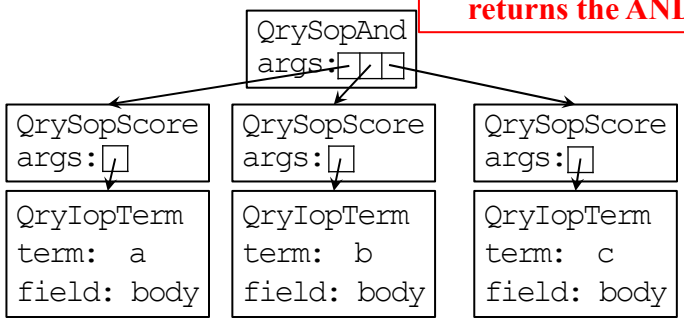
58

Query Parsing

#AND(a b c)

arguments

Query parsing completes & returns the AND node.



59

59

Query Parsing

#AND(a #OR(b c) d)

60

60

Query Parsing

#AND(a #OR(b c) d)

operator

Allocate an AND operator.

```
QrySopAnd
args:
```

61

61

Query Parsing

#AND(a #OR(b c) d)

Left '(' and right ')'

```
QrySopAnd
args:
```

62

62

Query Parsing

#AND(a #OR(b c) d)

arguments

```
QrySopAnd
args:
```


Query Parsing

#AND(a #OR(b c) d)

arguments

Allocate a TERM operator. Append it as an argument to the AND operator.

```
QrySopAnd
args:
```

```
QrySopScore
args:
```

```
QryIopTerm
term:  a
field: body
```

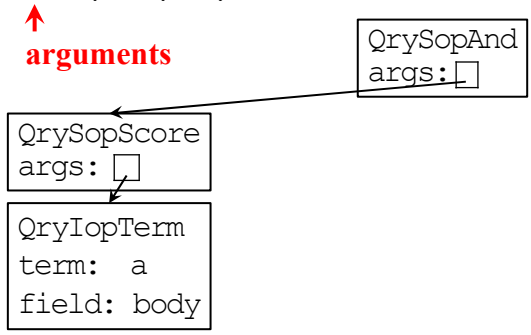
Note: Qry.appendArg automatically inserts a #SCORE operator

Query Parsing

#AND(a #OR(b c) d)

arguments

Operator detected. Recursive call on the argument.



65

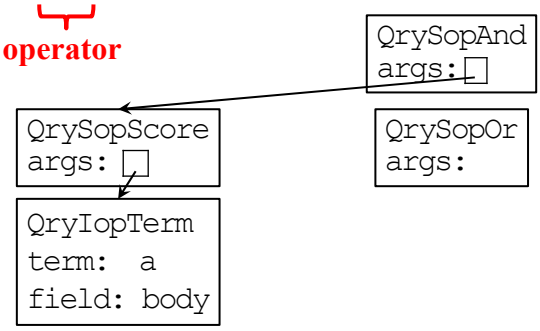
65

Query Parsing

#AND(a #OR(b c) d)

operator

Recursive call. Allocate an OR operator.



66

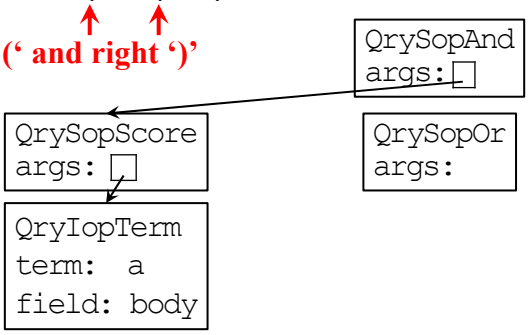
66

Query Parsing

#AND(a #OR(b c) d)

Left '(' and right ')'

Recursive call.



67

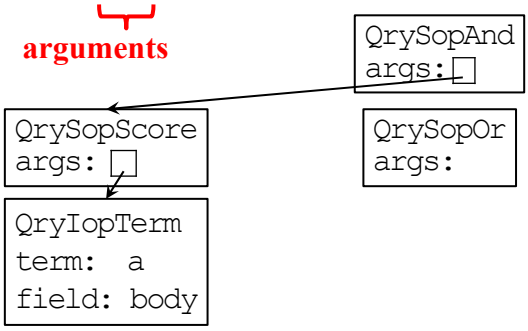
67

Query Parsing

#AND(a #OR(b c) d)

arguments

Recursive call.



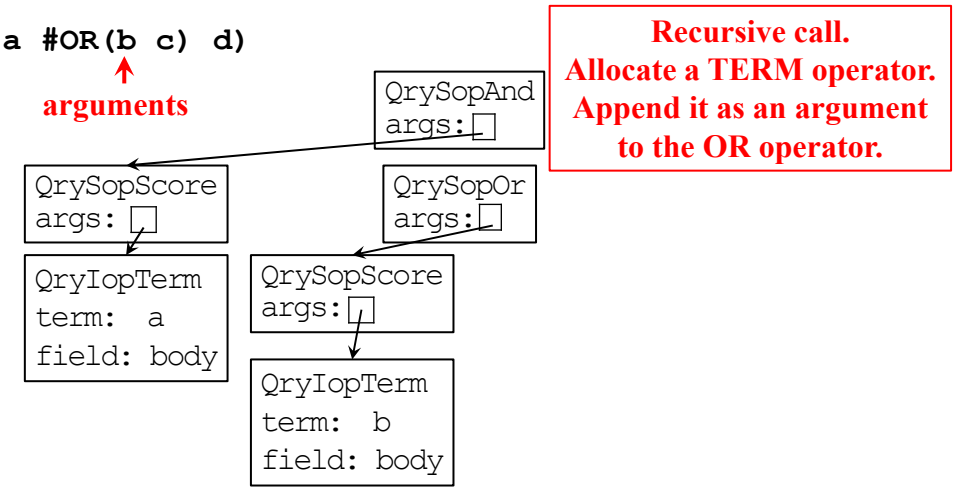
68

68

Query Parsing

#AND(a #OR(b c) d)

arguments



Note: Qry.appendArg automatically inserts a #SCORE operator

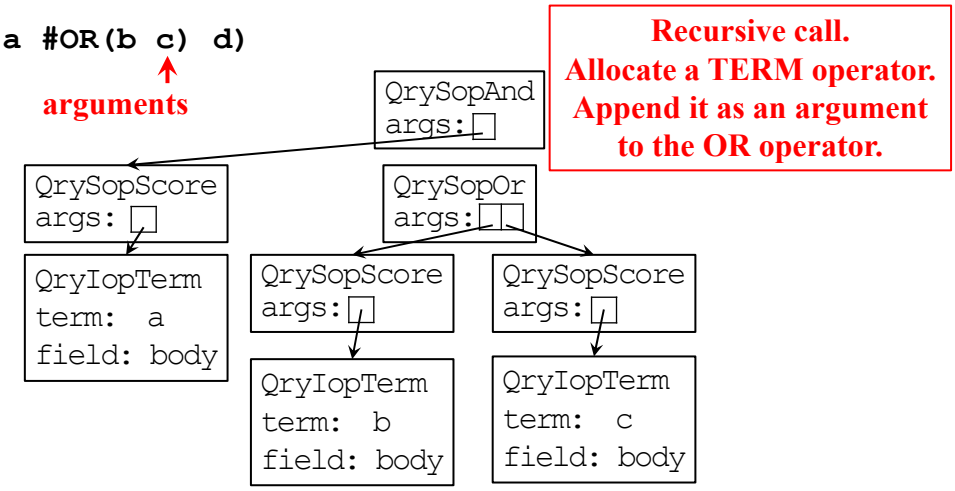
69

69

Query Parsing

#AND(a #OR(b c) d)

arguments



Note: Qry.appendArg automatically inserts a #SCORE operator

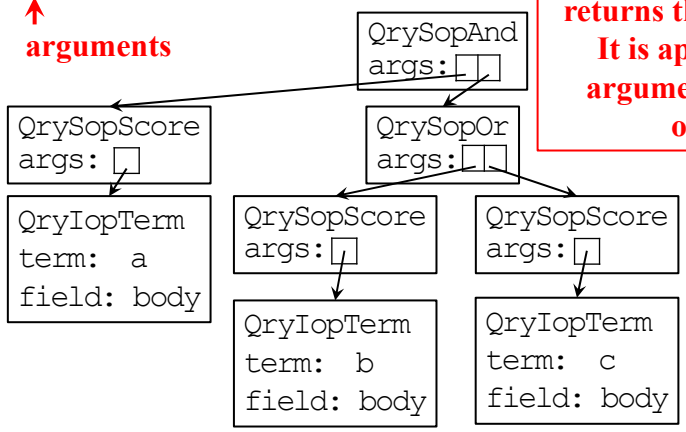
70

70

Query Parsing

#AND(a #OR(b c) d)

arguments



Recursive call completes & returns the OR operator. It is appended as an argument to the AND operator.

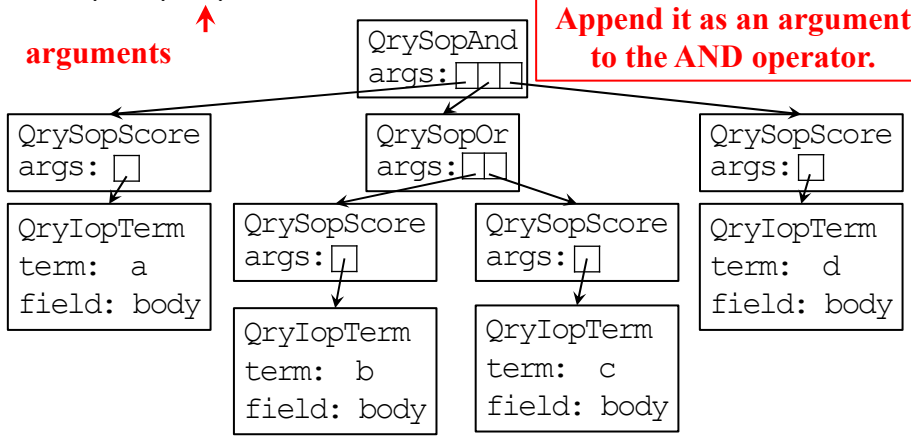
71

71

Query Parsing

#AND(a #OR(b c) d)

arguments



Allocate a TERM operator. Append it as an argument to the AND operator.

Note: Qry.appendArg automatically inserts a #SCORE operator

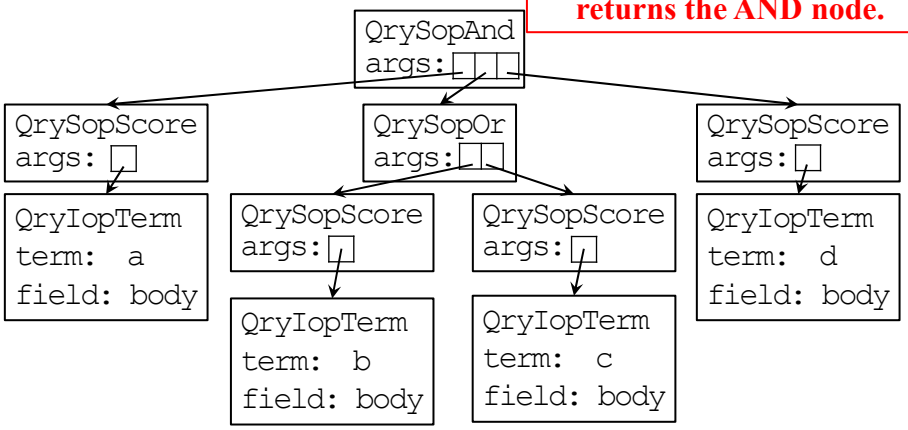
72

72

Query Parsing

#AND(a #OR(b c) d)

Query parsing completes & returns the AND node.



Query Parsing

a b c

Query Parsing

`a b c`

This is a syntax error because there is no query operator

We want the search engine to support unstructured queries

- **Solution:** The caller must add the default query operator
- E.g., **`a b c`** → **`#OR(a b c)`**

Every retrieval model has a default query operator

- It isn't the same for every retrieval model
- The query parser doesn't know which retrieval model will be used
- So, the query parser can't apply the default query operator for you
- Your code must do it

75

75

Query Parsing

You will need to modify the query parser

- HW1: Add new query operators (e.g., #AND, #NEAR/n)
- HW2: Add support for query operators that require weights
  - E.g., #WSUM (0.5 barack 1.0 obama)

76

76

Outline

- **QryEval overview**
  - What it does (conceptually)
  - Applications
  - Classes
- **Query evaluation**
  - The Qry class
  - Iteration
  - Matching
  - Calculating scores
- **Overview of query parsing**