Lucidworks

LUCENE / SOLR REVOLUTION 2016

http://lucenerevolution.com

OCTOBER 11-14
BOSTON, MA

Wiring ML Models into the inner loop of a Lucene Scorer

Jake Mannix @pbrane

Lead Data Engineer, Lucidworks

previously: IR/RecSys @ LinkedIn,
Search & p13n @ Twitter,
Semantic Search @ Ai^2

(Apache Mahout committer)

Agenda

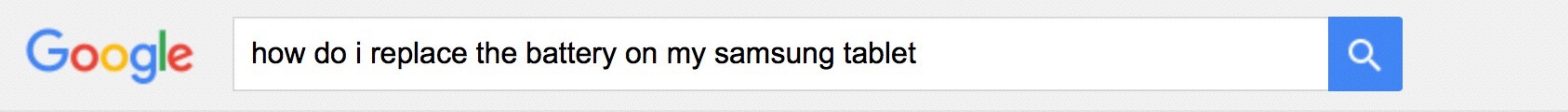
Lucidworks

- Search: State of the Industry
- Lucene OOTB Scoring / Similarity
- Trained models for Lucene
- DisjunctionScorer
- IdentifiedDisjunctionScorer

State of The (Search) Industry

Lucidworks

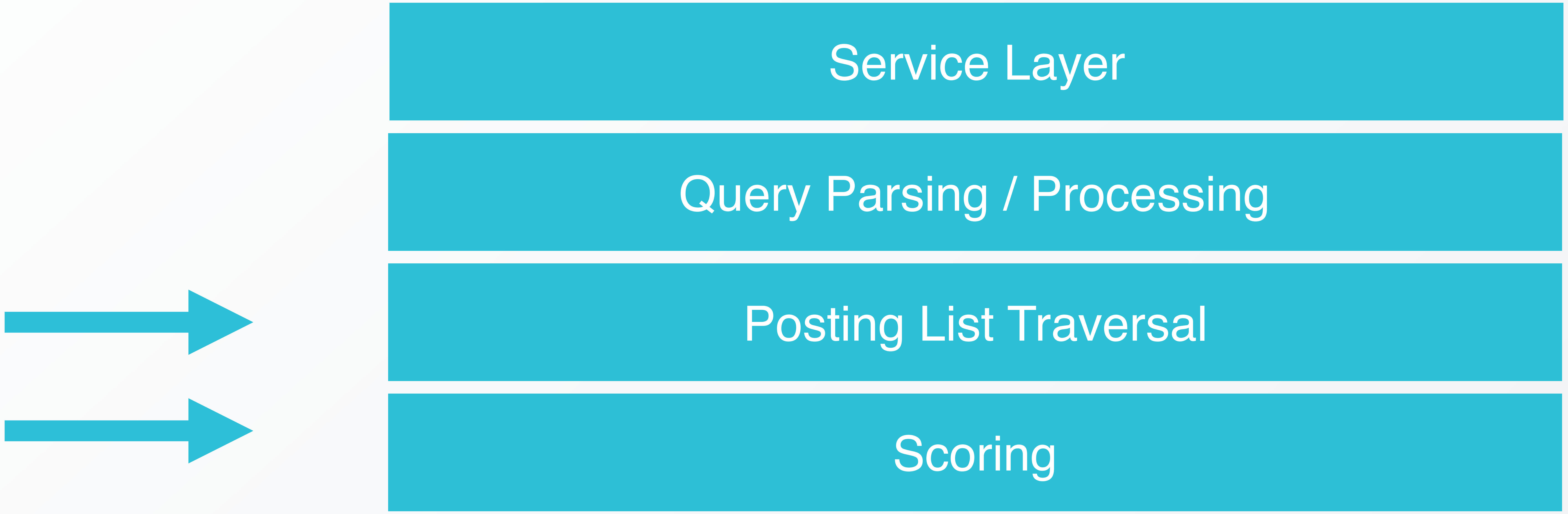
- Google has trained users too well. Queries like these are becoming commonplace:



how do i replace the battery on my samsung tablet

- NLP and query categorization / classification is often needed at query “parsing” stage
- Behind the scenes, previous queries of you (and everyone else!) factor into ranking

Runtime Retrieval



Service Layer

Query Parsing / Processing

Posting List Traversal

Scoring

State of The (Search) Industry

Lucidworks

- Internally, queries tend to hit semi-structured indexes in increasingly complicated ways:
  - recruiter searching on LinkedIn: “sr. SDE java scala spark big data”
  - becomes:
    - (title:(“sr sde” OR “senior software development engineer” OR sde^0.1)^3.5 AND (skills:(java OR scala OR spark OR “big data”)^1.5 OR skills:(hadoop OR pig OR hive)^0.1)
    - (and that’s without hitting multilingual fields, stemmed fields, cluster-label fields, synonym fields, etc…)

Relevance Factors which are Relevant in The Real World

- Why did Google succeed (at first)?
  - PageRank - crowdsourced page relevance by the global popularity structure of the web graph
- What are dominant factors for search at Google, Amazon,Facebook, Twitter, LinkedIn?
  - popularity (both globally, by demographic, context, and socially)
  - “social proof”
  - recency (sometimes as filter, sometimes as ranking factor)
  - past click/engagement history
  - user-model / preferences
- What is not a dominant factor, beyond the basics: textual similarity

Components of a Lucene Scoring Model

- **Query** -> **Weight** -> **Scorer**
  - Lucene **Query** is a (reusable) blueprint to tell an **IndexSearcher** how to build a **DocIdSetIterator**
  - **Weight** is an intermediate **IndexSearcher**-stateful object ready to build **Scorer**s.
    - it precomputes any query-global information (query normalization, separate sub-query parts which are for scoring, and which for just matching)
  - **Scorer**: both an iterator over posting list, and computes scores for each document

OOTB Lucene Queries

- Instantiable **Query** subclasses:
  - **TermQuery**
  - **BooleanQuery**, **DisjunctionMaxQuery**
  - **WildcardQuery**, **PrefixQuery**
  - **PhraseQuery**, **MultiPhraseQuery**
  - **FuzzyQuery**, **RegexpQuery**
  - **TermRangeQuery**, **PointRangeQuery**
  - **MatchAllDocsQuery**
  - Span queries: org.apache.lucene.search.spans.*

Lucene Similarity basics

- Implements the generic scoring model by comparing query <-> document similarity
- **CollectionStatistics**, **TermStatistics**
- Helper functions (e.g. what impl for “tf()” or “idf()” do you want to use?)
- queryNorm(float sumOfSquaredWeights) - make scores comparable across queries
- coord(int overlap, int maxOverlap) - score factor from #matching terms between doc/query

Lucene Similarity

- **TFIDFSimilarity**
- **BM25Similarity** (now default!)
- **LMSimilarity** (language models with different smoothing)
  - **LMDirichletSimilarity**
  - **LMJelinekMercerSimilarity**
- **DFISimilarity** (divergence from independence)
- **DFRSimilarity** (divergence from randomness)
- **IBSimilarity** (information-based models)
  - allows pluggable distributions modeling term occurrence
- combiners: **PerFieldSimilarity**, **MultiSimilarity**

Trained model insertion points?

- Fielded query model with trained weights applied to each field
  - (title:q_t)^title_w OR (desc:q_d)^desc_w OR (body:q_b)^body_w OR (_all:q_a)^all_w
- Rerank top-K results - see Diego's talk on LTR tomorrow!
- How to score a decision tree or neural net?
  - in particular, some models want to know things like:
    - if ((titleScore > 0.2 && skillsScore > 0.1) && (descScore > 0.6 || bodyScore > 0.95)) …

Nonlinear Models Difficult to include in a Scorer

- Decision Tree
  - CART
  - Random Forest, Bagging, Boosting, etc
  - even hand-crafted (not learned!) “artisanal” DTs (i.e. business rules)
- Neural Net
- ~~Deep Learning~~ (see line above)

Posting List Traversal and Scoring for Disjunction Queries

- Quiz (for non-Lucene developers): How would you efficiently traverse the posting list for:
  - q = “term1” OR “term2” OR “term3” OR … OR “termN” ?

Posting List Traversal and Scoring for Disjunction Queries

Lucidworks

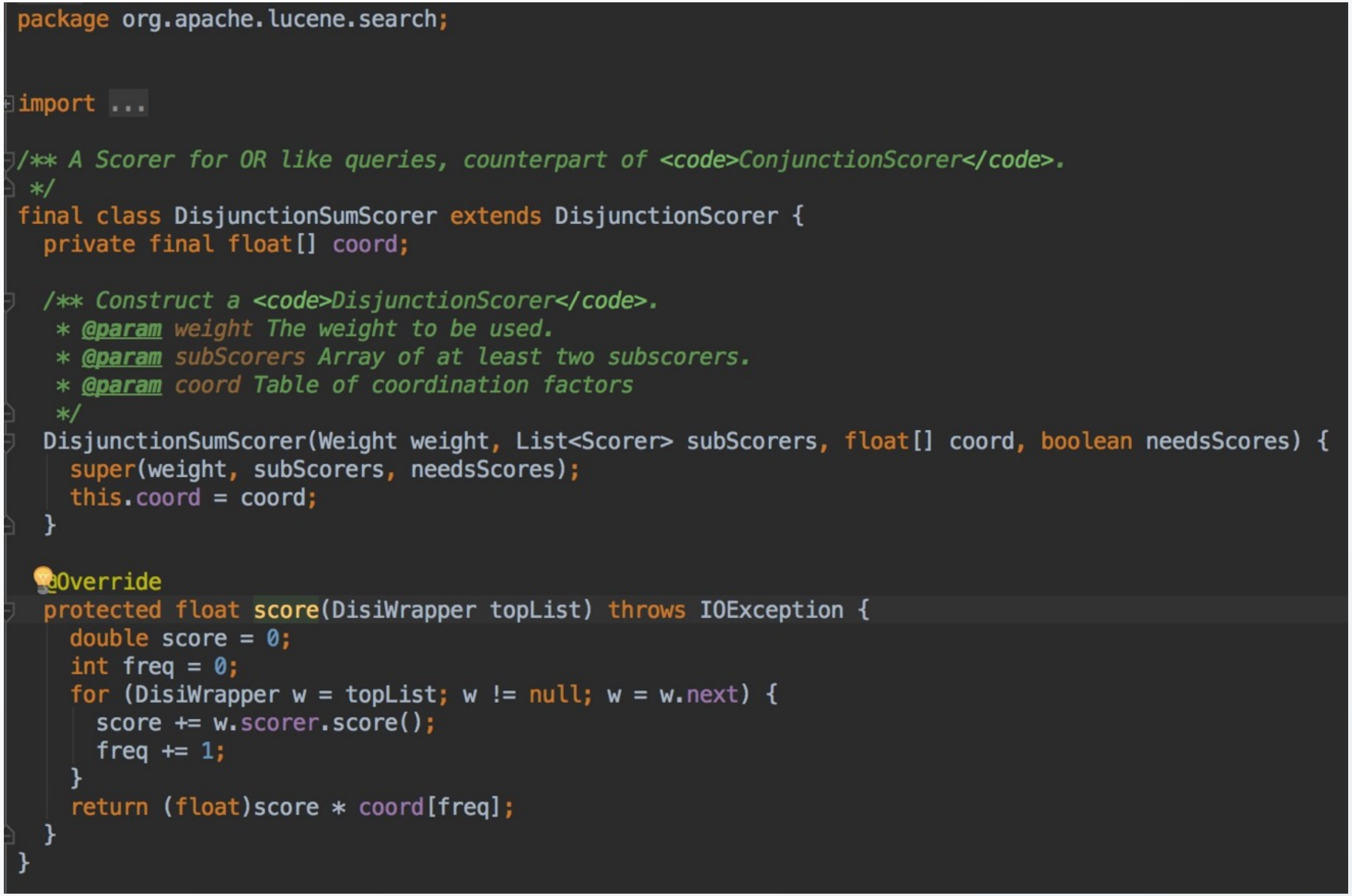
- Answer:
  - leaf query in the boolean tree <-> (docId-ordered) Iterator over posting list
  - put all these Iterators into a min-heap (PriorityQueue), ordered by the value of .currentDocId()
  - pull Iterators off of the heap, checking to see you're still on the same document, accumulate scores from each

Delayed Scoring for Disjunction Queries

Lucidworks

- But what if you want to score the sub-queries other than by simply summing the results?
- Let’s look at the current implementation of DisjunctionScorer in Lucene

DisjunctionSumScorer



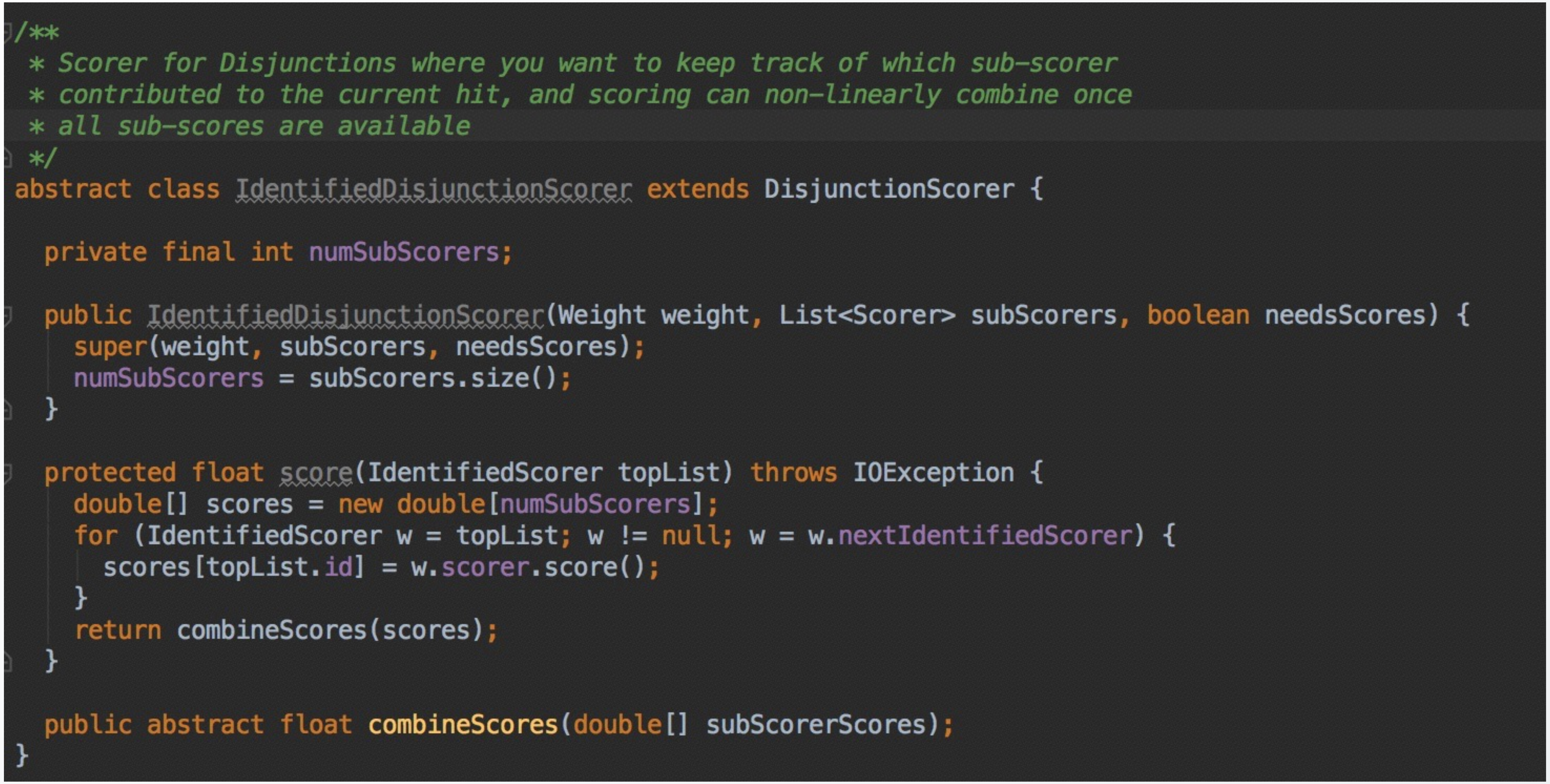
```
package org.apache.lucene.search;


import ...

/** A Scorer for OR like queries, counterpart of <code>ConjunctionScorer</code>.
 */
final class DisjunctionSumScorer extends DisjunctionScorer {
  private final float[] coord;

  /** Construct a <code>DisjunctionScorer</code>.
   * @param weight The weight to be used.
   * @param subScorers Array of at least two subscorers.
   * @param coord Table of coordination factors
   */
  DisjunctionSumScorer(Weight weight, List<Scorer> subScorers, float[] coord, boolean needsScores) {
    super(weight, subScorers, needsScores);
    this.coord = coord;
  }

  @Override
  protected float score(DisiWrapper topList) throws IOException {
    double score = 0;
    int freq = 0;
    for (DisiWrapper w = topList; w != null; w = w.next) {
      score += w.scorer.score();
      freq += 1;
    }
    return (float)score * coord[freq];
  }
}
```

Generalized Linear Scoring

- (title:q_t)^title_w OR (desc:q_d)^desc_w OR (body:q_b)^body_wOR (_all:q_a)^all_w
- field boosts can encode the model weights
  - summing: linear regression scoring
  - summing then apply logit(): logistic regression scoring

Nonlinear Scoring

- IdentifiedScorer*, IdentifiedDisjunctionScorer*



```java
/**
 * Scorer for Disjunctions where you want to keep track of which sub-scorer
 * contributed to the current hit, and scoring can non-linearly combine once
 * all sub-scores are available
 */
abstract class IdentifiedDisjunctionScorer extends DisjunctionScorer {

  private final int numSubScorers;

  public IdentifiedDisjunctionScorer(Weight weight, List<Scorer> subScorers, boolean needsScores) {
    super(weight, subScorers, needsScores);
    numSubScorers = subScorers.size();
  }

  protected float score(IdentifiedScorer topList) throws IOException {
    double[] scores = new double[numSubScorers];
    for (IdentifiedScorer w = topList; w != null; w = w.nextIdentifiedScorer) {
      scores[topList.id] = w.scorer.score();
    }
    return combineScores(scores);
  }

  public abstract float combineScores(double[] subScorerScores);
}
```

- (do not exist yet! hack with me tomorrow?)

Not covered: The Devil in the Details

- Latency:
  - complex scoring at every hit - will it work on your real data?
    - (early termination of scoring function [return 0 early when you know it won’t score well])
- True API compatibility
  - there are many ways **Query** and **Scorer** objects can be used (as a filter with no scoring, scoring in bulk, nesting hierarchy: **IdentifiedQuery** containing a **BooleanQuery**, or vice versa?)
- Truly comparable query norms
  - Most query scores are “kinda comparable” - but are <u>not</u> strictly bounded!
  - So training things like logistic regression models often get confused
- Collecting training data from implicit information (clicks and related engagement signals)
- Model I/O

Resources

- Fusion: http://www.lucidworks.com/products/fusion
- Search Hub: http://searchhub.lucidworks.com
- Company: http://www.lucidworks.com
  - Our blog: http://www.lucidworks.com/blog
- Twitter: @pbrane

Thanks!

Lucidworks

this page left intentionally blank