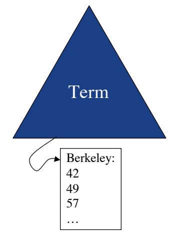
Ranking Results in IR Search

Review: Simple Relational Text Index

- **Create and populate a table**
  InvertedFile(term string, docID string)
- **Build a B+-tree or Hash index on InvertedFile.term**
  - Use something like "Alternative 3" index
    - Keep lists at the bottom sorted by docID
    - Typically called a "postings list"



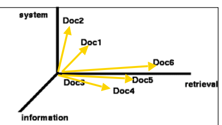
Boolean Search in SQL

"**Berkeley Database Research**"

```
SELECT IB.docID
  FROM InvertedFile IB, InvertedFile ID, InvertedFile IR
 WHERE IB.docID = ID.docID AND ID.docID = IR.docID
   AND IB.term = "Berkeley"
   AND ID.term = "Database"
   AND IR.term = "Research"
ORDER BY magic_rank()
```

- **This time we wrote it as a join**
  - Last time wrote it as an INTERSECT
- **Recall our query plan**
  - An indexscan on each table "instance" in FROM clause
  - A merge-join of the 3 indexscans (ordered by docID)
- **magic_rank() is the "secret sauce" in the search engines**
  - Will require rewriting this query somewhat…

Classical IR Ranking



- **Abstraction: Vector space model**
  - We'll think of every document as a "vector"
    - Imagine there are 10,000 possible terms
    - Each document (bag of words) can be represented as an array of 10,000 counts
    - This array can be thought of as a point in 10,000-dimensional space
  - Measure "distance" between two vectors: "similarity" of two documents
- **A query is just a short document**
  - Rank all docs by their distance to the query "document"!
- **What's the right distance metric?**
  - Problem 1: two long docs seem more similar to each other than to short docs
    - Solution: *normalize* each dimension by vector's (Euclidean) length
    - Now everything doc is a point on the unit sphere
  - Now: the *dot-product* (sum of products) of two normalized vectors happens to be cosine of the angle between them!
    - $(d_j \cdot d_k)/(|d_j||d_k|) = \cos(\theta)$ θ
  - BTW: for normalized vectors, cosine ranking is the same as ranking by Euclidean distance (prove this to yourself for 2-d)

TF × IDF

What is the tf × idf of a term that occurs in all of the docs?

- **Counting occurrences isn't a good way to weight each term**
  - Want to favor repeated terms in this doc
  - Want to favor unusual words in this doc
- **TF × IDF (Term Frequency × Inverse Doc Frequency)**
  - For each doc d
    - DocTermRank = #occurrences of t in d (TF)
      × log((total #docs)/(#docs with this term)) (IDF)
  - Instead of using counts in the vector, use DocTermRank
- **Let's add some more to our schema**
  - TermInfo(term string, numDocs int) -- used to compute IDF
    - This is a "materialized" view on the invertedFile table.
      - What's the SQL for the view?
  - InvertedFile (term string, docID int64, DocTermRank float)
    - Why not just store TF rather than DocTermRank?

In SQL Again…

–InvertedFile (term string, docID int64, DocTermRank float)

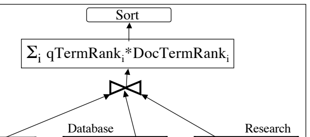
Simple Boolean Search

```
CREATE VIEW BooleanResult AS (
SELECT IB.docID, IB.DocTermRank as bTFIDF,
       ID.DocTermRank as dTFIDF,
       IR.DocTermRank as rTFIDF,
  FROM InvertedFile IB, InvertedFile ID, InvertedFile IR
 WHERE IB.docID = ID.docID AND ID.docID = IR.docID
   AND IB.term = "Berkeley"
   AND ID.term = "Database"
   AND IR.term = "Research");

SELECT docID,
       (<Berkeley-tfidf>*bTFIDF +
        <Database-tfidf>*dTFIDF +
        <Research-TFIDF>*rTFIDF>) AS magic_rank
  FROM BooleanResult
ORDER BY magic_rank;
```

Cosine similarity. Note that the query "doc" vector is a constant

Ranking



Berkeley

docID	DTRank
42	0.361
49	0.126
57	0.111

Database

docID	DTRank
16	0.137
49	0.654
57	0.321

Research

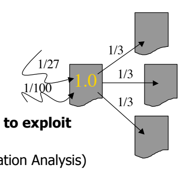
docID	DTRank
29	0.987
49	0.876
121	0.002

- **We'll only rank Boolean results**
  - Note: this is just a heuristic! (Why?)
    - What's a fix? Is it feasible?
  - Recall: a merge-join of the postings-lists from each term, sorted by docID
- **While merging postings lists…**
  - For each docID that matches on all terms (Bool)
    - Compute cosine distance to query
      - I.e. For all terms, Sum of (product of query-term-rank and DocTermRank)
    - This collapses the view in the previous slide
- **What's wrong with this picture??**

Some Additional Ranking Tricks

- **Phrases/Proximity**
  - Bump exact phrase matches up the ranking
  - Give extra weight to proximate occurrences
- **Query expansion, suggestions**
  - Can keep a similarity matrix on terms, and expand/modify people's queries
- **Fix misspellings**
  - E.g. via an inverted index on n-grams
  - Trigrams for "misspelling" are {mis, iss, ssp, spe, pel, ell, lli, lin, ing}
- **Document expansion**
  - Can add terms to a doc before inserting into inverted file
    - E.g. in "anchor text" of refs to the doc
- **Not all occurrences are created equal**
  - Mess with DocTermRank based on:
    - Fonts, position in doc (title, etc.)
    - Don't forget to normalize: "tugs" doc in direction of heavier weighted terms

Hypertext Ranking



- **On the web, we have more information to exploit**
  - The hyperlinks (and their anchor text)
  - Comes from Social Network Theory (Citation Analysis)
  - "Hubs and Authorities" (Clever), "PageRank" (Google)
- **Intuition (Google's PageRank)**
  - If you are important, and you link to me, then I'm important
  - Recursive definition --> recursive computation
    1. Everybody starts with weight 1.0
    2. Share your weight among all your outlinks
    3. Repeat (2) until things converge
  - Note: computes the principal eigenvector of the adjacency matrix
    - And you thought linear algebra was boring :-)
  - Leaving out some details here …
- **PageRank sure seems to help**
  - But rumor says that other factors matter as much or more
    - Anchor text, title/bold text, etc. --> much tweaking over time

Random Notes from the Real World

- **The web's dictionary of terms is HUGE. Includes:**
  - numerals: "1", "2", "3", … "987364903", …
  - codes: "transValueIsNull", "palloc", …
  - misspellings: "teh", "quik", "browne", "focs"
  - multiple languages: "hola", "bonjour", "こ こんんににちちはは" (Japanese), etc.
- **Web spam**
  - Try to get top-rated. Companies will help you with this!
  - Imagine how to spam TF x IDF
    - "Stanford Stanford Stanford Stanford Stanford Stanford Stanford Stanford Stanford … Stanford lost The Big Game"
    - And use white text on a white background :-)
  - Imagine spamming PageRank…?!
- **Some "real world" stuff makes life easier**
  - Terms in queries are Zipfian! Can cache answers in memory effectively.
  - Queries are usually little (1-2 words)
  - Users don't notice minor inconsistencies in answers
- **Big challenges in running thousands of machines, 24x7 service!**

Parallelism 101: Hardware

- **Shared Memory**
- **Shared Disk**
- **Shared Nothing (Clusters)**

Parallelism 101: Metrics

- **Speedup**
  - Same task, more resources
- **Scaleup**
  - Bigger task, bigger resources
- **Transaction scaleup**
  - More tasks, bigger resources

Parallelism 101: Types of Parallelism

- **Pipelined Parallelism**
- **Partition Parallelism**

Barriers to Perfect Parallelization

- **Startup**
- **Interference**
- **Skew**

Relational Stuff Parallelizes Beautifully

- **Dataflow: Single Instruction Multiple Data (SIMD)**
- **Relational, so order-independent!**
- **Pipelines AND Parallelizes beautifully**

Data Layouts

- **How to partition a table?**
  - Round-robin
  - Range-partition
  - Hash partition
- **Secondary indexes?**
  - Partitioned with data
    - Broadcast and fetch
    - Expensive to maintain DISTINCT
  - Partitioned by key
    - Two-step lookup (latency)

Parallel Aggregation

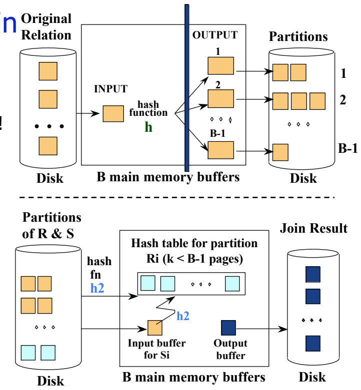
- **SUM**
- **AVERAGE**
- **MEDIAN**

Parallel Sort

- **Read data and range-partition it on the fly**
- **Sort locally**
- **Pipelining too!**
  - Reading from disk
  - Sending over NW
  - Receiving and sorting
  - Writing runs

|| Hash Join

- **Piece of cake!**



How about our text search query?