Peerto-Peer Database Networks
Part 2
Wolf-Tilo Balke and Wolf Siberski
9.1.2008

Overview

1. Why Peer-to-Peer Databases?
   1. Federation
   2. Information integration
   3. Sensor networks
   4. ‘New’ internet

2. Distributed Databases

3. P2P Databases
   1. Challenges
   2. Design Dimensions

4. Existing P2P Database systems
   1. Edutella: focus on expressivity
   2. PIER: focus on scalability
   3. Piazza: focus on integration
   4. HiSbase: focus on scalability for spatial data
PIER

• P2P Relational Database

• Foundation: any DHT

• Extended hash interface
  ▶ put(namespace, key, value)
  ▶ get(namespace, key)
  ▶ namespace/key combination is used as hash value (DHT Key)

• Extended network capabilities
  ▶ Exploit DHT structure for broadcast
  ▶ Required for joins and aggregate queries

Application: Phi

• Phi: Public Health for the Internet
  ▶ Monitor ip network state world-wide
  ▶ Collect statistics
  ▶ Network traffic
  ▶ Latency
  ▶ ...
  ▶ Malware alerts
Storing and Indexing Tuples

- **Storing**
  - Every tuple needs a synthetic tuple key
  - Choose combination of table name and tuple key as DHT key
  - Insert complete tuple into DHT using this key

- **Indexing**
  - Additional attribute indexes are built by inserting attribute value/tuple key pairs into the DHT
  - Choose combination of attribute name and attribute value as DHT key
  - Insert tuple key as DHT value

**Example**

- **Sample Database**

<table>
<thead>
<tr>
<th>Doc</th>
<th>Author</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>DocId</td>
<td>PersonId</td>
</tr>
<tr>
<td>Title</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Surname</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Sample tuple: (456, ‘Critique of pure Reason’, 1781, ‘en’)

- **Storing**
  - put(Doc, 456, (456, ‘Critique...’, ‘en’, Philosophy))

- **Indexing on ‘Title’ and ‘Date’ attributes**
  - put(Doc.Title, ‘Critique...’, 456)
  - put(Doc.Date, ‘1781’, 456)
PIER Query Plans

- **DHT-Scan**
  1. Use index to retrieve tuple key(s)
  2. Use key(s) to retrieve data tuple(s)

- **Example**
  ```
  dhtScan(Doc, Date='1781')
  ```

- Each peer can create a query plan
- One DHT lookup per result tuple
- Filter has to be done on query originator

---

Aggregate and Range Queries

- **Example**
  ```
  SELECT COUNT(Id) FROM Doc WHERE Date>'1780' AND Date<'1790'
  ```

- Use spanning tree for broadcast
- Aggregate on return
Join Queries

- **Example**
  - Assume a Person tuple (789, ‘Kant’, ‘Immanuel’)
  - SELECT Id, Title FROM Doc WHERE Author.DocId = Doc.Id AND Author.PersonId = 789

- **Approach: Hierarchical Joins**
  - Use spanning tree for broadcast
  - Do local select on peer table fragments
  - Do local join on each peer
    - Improves load balancing
  - Forward table fragments and partial results to parent
  - Repeat until query originator has all fragments

Hierarchical Joins

- Assumption: Person tuple (789, ‘Kant’, ‘Immanuel’) in PersonTable
- Join Fragments: DocTable and PersonTable
- Join on Author.DocId = Doc.Id AND Author.PersonId = 789

Diagram:

- **T11**: Person Id 789
- **T12**: Document DocId
- **T21**: Document Title
- **T22**: Person Name
- **T23**: Person Name
- **D1**: Document Id
- **D2**: Document Title
- **D3**: Document Id
- **A1**: Author Name
- **A2**: Author Name
- **A3**: Person Name
- **A4**: Person Name

Load balancing achieved by forwarding partial results and table fragments to parent nodes.
PIER - Discussion

- Real query planning
- Very efficient access to individual tuples and small result sets
- Very good scalability in terms of network size

- Degrades to broadcast for many types of queries
  - Aggregate queries
  - Joins
- INSERT operation expensive (see P2P Inform. Retrieval)
- No load-balancing mechanisms

Overview

1. Why Peer-to-Peer Databases?
   1. Federation
   2. Information integration
   3. Sensor networks
   4. ‘New’ internet

2. Distributed Databases

3. P2P Databases
   1. Challenges
   2. Design Dimensions

4. Existing P2P Database systems
   1. Edutella: focus on expressivity
   2. PIER: focus on scalability
   3. Piazza: focus on integration
   4. HiSbase: focus on scalability for spatial data
Piazza

- Tackles problem of “reconciling different models of the world” (A. Halevy)
- Goal: provide a uniform interface to a set of autonomous data sources.
- New abstraction layer over multiple sources

- Introduce mappings between ‘world views’
  - Mapping rules are specified manually by experts
  - Don’t need to be complete

Example – Publication Databases

UCSD

Data

Area(areaID, name, desc)
Project(projectID, name, sponsor)
ProfArea(projectID, areaID)
Pub(pubID, pubName, title, venue, year)
Author(authorID, author)
Member(projectName, member)

UPenn

Project(projectID, name, descr)
Student(studentID, name, status)
Faculty(facultyID, name, rank, office)
Advisor(advisorID, advisorID)
ProjMember(projID, memberId)

UW

Pub(pubID, title, venue, year)
PubAuthor(pubID, authorID)
PubProject(pubID, projID)

Member(memberID, projID, name, pos)
Alumnus(name, year, thesis)

Stanford

Data

Area(areaID, name, desc)
Project(projectID, areaID, name)
Pub(pubID, title, venue, year)
PubAuthor(pubID, authorID)
PubProject(pubID, projID)
Member(memberID, projID, name, pos)
Alumnus(name, year, thesis)
Mapping Rules

- **Datalog to specify mapping rules**
  
  UCSD : Member(projName; member) :
  UW : Member(pid; member; ;)
  UW : Project(pid; ; projName):
  
  UCSD : Member(projName; member) :
  UPenn : Student(sid; member; ;)
  UPenn : ProjMember(pid; sid);
  UPenn : Project(pid; projName; )
  
  UCSD : Member(projName; member) :
  UPenn : Faculty(sid; member; ;)
  UPenn : ProjMember(pid; sid);
  UPenn : Project(pid; projName; )

Storing and Indexing

- **Unstructured network (Gnutella-like)**
  - Peer keeps its database
    - No exchange of data between peers

- **Indexing**
  - Only on schema level
  - Each peer maintains schema catalog of its neighbors
  - Mappings Stored in central catalog (hybrid system)
    - could be replaced by DHT
  - Replication of mappings to all relevant peers
**Query Routing**

- **Query Flooding**
  - Peer translates query to schema of neighbor (if possible)
  - Result tuples are converted on way back

- **Queries answered by traversing semantic paths**

**Piazza - Discussion**

- Supports multiple schema world (more realistic)
- Very expressive mapping mechanism

- Not scalable
  - Gnutella-like topology and flooding

- Piazza mapping technique could be applied to other network infrastructures
Overview

1. Why Peer-to-Peer Databases?
   1. Federation
   2. Information integration
   3. Sensor networks
   4. 'New' internet

2. Distributed Databases

3. P2P Databases
   1. Challenges
   2. Design Dimensions

4. Existing P2P Database systems
   1. Edutella: focus on expressivity
   2. PIER: focus on scalability
   3. Piazza: focus on integration
   4. HiSbase: focus on scalability for spatial data

HiSbase

- Specialized on distributed spatial data

- Application: astronomy data
  - Huge amounts of data (terabyte scale)
  - Region-based queries
  - Skewed data distribution

- Main ideas
  - Distribute data on peers by region
  - Use DHT for data access
  - Use neighbor-preserving hash function (space-filling curve)
Load Distribution

- Use Quad-Tree structure to split data space into equally loaded regions

Data Hashing

- Use Z-Linearization for hashing coordinates
Insertion into DHT

Query Processing

- **Point query**: simple DHT access

- **Region query**
  - Route to arbitrary peer in range (e.g. using upper left region boundary)
  - This peer acts as coordinator
  - Forward query to peer region neighbors
    - Until whole area is covered
  - Collect results at coordinator
HiSbase - Discussion

- Very efficient for spatial queries

- Not completely self-organizing
  - Quad-Tree splitting needs central coordination

- Only spatial queries possible

Peer-to-Peer Database Networks – Summary (1)

- Challenges
  - Multi-Dimensional Search Space
  - Schema Heterogeneity
  - Potentially large result sets

- Design Dimensions
  - Network Properties (Data Placement, Topology and Routing)
  - Data Access (Data Model, Query Language)
  - Integration Mechanism (Mapping Representation/Creation/Usage)

- P2P Database Types
  - Focus on high network scalability (e.g., PIER)
  - Focus on high query expressivity (e.g., Edutella)
  - Focus on information integration (e.g., Piazza)
  - Focus on specific data structures (e.g., HiSbase)
Conclusion

- P2P Databases do already work
  - although immature compared to traditional database technology

- One size does not fit all
  - Choose P2P database approach according to application requirements

- Open problems
  - Load Balancing (Replication/Caching)
  - How to combine DHT and filtered flooding advantages
  - Reliability (probabilistic guarantees)
  - ...

Peer-to-Peer Information Retrieval
Basics

Wolf-Tilo Balke and Wolf Siberski
Overview

- **Goal** = find documents *relevant* to an information need from a large document set

First Applications

- **Libraries (1950s)**
  - ISBN: 0-201-12227-8
  - Author: Salton, Gerard
  - Title: Automatic text processing: the transformation, analysis, and retrieval of information by computer
  - Editor: Addison-Wesley
  - Date: 1989
  - Content: <Text>

- **External attributes and internal attribute (= content)**
- **Search by external attributes = Search in DB**
- **IR: search by content**
## IR applications vs. Databases

<table>
<thead>
<tr>
<th>IR</th>
<th>DBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imprecise Semantics</td>
<td>Precise Semantics</td>
</tr>
<tr>
<td>Keyword search</td>
<td>SQL</td>
</tr>
<tr>
<td>Unstructured data format</td>
<td>Structured data</td>
</tr>
<tr>
<td>Read-Mostly, Add docs occasionally</td>
<td>Expect reasonable number of updates</td>
</tr>
<tr>
<td>Page through $\text{top } k$ results</td>
<td>Generate full answer</td>
</tr>
</tbody>
</table>

### The IR-Cycle

1. **Source Selection**
   - Resource

2. **Query Formulation**
   - Query

3. **Search**
   - Ranked List
   - Documents
   - query reformulation, vocabulary learning, relevance feedback
   - source reselection

4. **Selection**
   - Documents

5. **Examination**
   - Documents

6. **Delivery**
Supporting the search process

Information Hierarchy

More refined and abstract

Wisdom

Knowledge

Information

Data
Information?

- Data
  - The raw material of information

- Information
  - Data organized and presented in a particular manner

- Knowledge
  - “Justified true belief”
  - Information that can be acted upon

- Wisdom
  - Distilled and integrated knowledge
  - Demonstrative of high-level "understanding"

Information?

- Data
  - 98.6°F, 99.5°F, 100.3°F, 101°F, …

- Information
  - Hourly body temperature: 98.6°F, 99.5°F, 100.3°F, 101°F, …

- Knowledge
  - If you have a temperature above 100°F, you most likely have a fever

- Wisdom
  - If you don’t feel well, go see a doctor
What is the Retrieval-Task?

- “Fetch something” that’s been stored
- Recover a stored state of knowledge
- Search through stored messages to find some messages relevant to the task at hand

What is IR?

- Information retrieval is a problem-oriented discipline, concerned with the problem of the effective and efficient transfer of desired information between human generator and human user
What is IR?

Modern History

- The “information overload” problem is much older than you may think
- Origins in period immediately after World War II
  - Tremendous scientific progress during the war
  - Rapid growth in amount of scientific publications available
- The “Memex Machine”
  - Conceived by Vannevar Bush, President Roosevelt’s science advisor
  - Outlined in 1945 Atlantic Monthly article titled “As We May Think”
  - Foreshadows the development of hypertext (the Web) and information retrieval system
Memex

Memex in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Stun-ning translucent viewing screens magnify supermicrofilm filed by code markers. At left is a mechanism which automatically
photographs longhand notes, pictures and letters, then files them in the desk for future reference (LIFE 1941), p. 129.

Document View

Space of all documents

- Relevant
- Relevant + Retrieved
- Retrieved
- Not Relevant + Not Retrieved
What is a Model?

- A model is a construct designed to help us understand a complex system
  - A particular way of “looking at things”
- Models inevitably make simplifying assumptions
  - What are the limitations of the model?
- Different types of models:
  - Conceptual models
  - Physical analog models
  - Mathematical models
  - …

The central Problem in IR

Information Seeker

Authors

Concepts

Query Terms

Document Terms

Do these represent the same concepts?
**Representing Text**

- How do we represent the complexities of language?
  - Keeping in mind that computers don’t “understand” documents or queries
- Simple, yet effective approach: “bag of words”
  - Treat all the words in a document as index terms for that document
  - Assign a “weight” to each term based on its “importance”
  - Disregard order, structure, meaning, etc. of the words

---

**McDonald’s slims down spuds**

Fast-food chain to reduce certain types of fat in its french fries with new cooking oil.

NEW YORK (CNNMoney) - McDonald’s Corp. is cutting the amount of “bad” fat in its french fries nearly in half, the fast-food chain said Tuesday as it moves to make all its fried menu items healthier.

But does that mean the popular shoestring fries won’t taste the same? The company says no. “It’s a win-win for our customers because they are getting the same great french-fry taste along with an even healthier nutrition profile,” said Mike Roberts, president of McDonald’s USA.

But others are not so sure. McDonald’s will not specifically discuss the kind of oil it plans to use, but at least one nutrition expert says playing with the formula could mean a different taste.

Shares of Oak Brook, Ill.-based McDonald’s (MCD: down $0.54 to $23.22, Research, Estimates) were lower Tuesday afternoon. It was unclear Tuesday whether competitors Burger King and Wendy’s International (WEN: down $0.80 to $34.91, Research, Estimates) would follow suit. Neither company could immediately be reached for comment.

…

“Bag of Words”
Bag of Words

- Retrieving relevant information is hard!
  - Evolving, ambiguous user needs, context, etc.
  - Complexities of language

- To operationalize information retrieval, we must vastly simplify the picture

- Bag-of-words approach:
  - Information retrieval is *all* (and *only*) about matching words in documents with words in queries
  - Obviously, not true...
  - But it works pretty well!

Representing Documents as Vectors
Representing Text

How to compare documents and queries?

Boolean Retrieval

- Weights assigned to terms are either “0” or “1”
  - “0” represents “absence”: term isn't in the document
  - “1” represents “presence”: term is in the document
- Build queries by combining terms with Boolean operators
  - AND, OR, NOT
- The system returns all documents that satisfy the query
### Boolean View of a Document-Set (=Collection)

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
<th>Doc 5</th>
<th>Doc 6</th>
<th>Doc 7</th>
<th>Doc 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aid</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>all</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>back</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>brown</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>come</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>dog</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fox</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>good</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>jump</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>lazy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>men</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>now</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>party</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>quick</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>their</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>time</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Each column represents the view of a particular document: What terms are contained in this document?

Each row represents the view of a particular term: What documents contain this term?

To execute a query, pick out rows corresponding to query terms and then apply logic table of corresponding Boolean operator.

### Sample Queries

#### Term

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
<th>Doc 5</th>
<th>Doc 6</th>
<th>Doc 7</th>
<th>Doc 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fox</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>dog ∨ fox</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>dog ∨ fox</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>dog ¬ fox</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>fox ¬ dog</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Term

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
<th>Doc 5</th>
<th>Doc 6</th>
<th>Doc 7</th>
<th>Doc 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>party</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g ∨ p</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Term

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
<th>Doc 5</th>
<th>Doc 6</th>
<th>Doc 7</th>
<th>Doc 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>party</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g ∨ p</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Sample Queries:

- dog AND fox → Doc 3, Doc 5
- dog OR fox → Doc 3, Doc 5, Doc 7
- dog NOT fox → empty
- fox NOT dog → Doc 7
- good AND party → Doc 6, Doc 8
- good AND party NOT over → Doc 6
Why Boolean Retrieval works

- Boolean operators \textit{approximate} natural language
  - Find documents about a good party that is not over
- AND can discover relationships between concepts
  - good party
- OR can discover alternate terminology
  - excellent party, wild party, etc.
- NOT can discover alternate meanings
  - Democratic party

The Perfect Query Paradox

- Every information need has a perfect set of documents
  - If not, there would be no sense doing retrieval
- Every document set has a perfect query
  - AND every word in a document to get a query for it
  - Repeat for each document in the set
  - OR every document query to get the set query
- But can users realistically be expected to formulate this perfect query?
  - Boolean query formulation is hard!
Why Boolean Retrieval fails

- Natural language is way more complex
- AND “discovers” nonexistent relationships
  - Terms in different sentences, paragraphs, …
- Guessing terminology for OR is hard
  - good, nice, excellent, outstanding, awesome, …
- Guessing terms to exclude is even harder!
  - Democratic party, party to a lawsuit, …

Strengths and Weaknesses

- **Strengths**
  - Precise, if you know the right strategies
  - Precise, if you have an idea of what you’re looking for
  - Efficient for the computer
- **Weaknesses**
  - Users must learn Boolean logic
  - Boolean logic insufficient to capture the richness of language
  - No control over size of result set: either too many documents or none
  - When do you stop reading? All documents in the result set are considered “equally good”
  - What about partial matches? Documents that “don’t quite match” the query may be useful also
Ranked Retrieval

- Order documents by how likely they are to be relevant to the information need
  - Present hits one screen at a time
  - At any point, users can continue browsing through ranked list or reformulate query
- Attempts to retrieve relevant documents directly, not merely provide tools for doing so

Why Ranked Retrieval?

- Arranging documents by relevance is
  - Closer to how humans think: some documents are “better” than others
  - Closer to user behavior: users can decide when to stop reading
- Best (partial) match: documents need not have all query terms
  - Although documents with more query terms should be “better”
- Easier said than done!
Similiarity-based Retrieval?

- Let’s replace relevance with “similarity”
  - Rank documents by their similarity with the query
- Treat the query as if it were a document
  - Create a query bag-of-words
- Find its similarity to each document
- Rank order the documents by similarity
- Surprisingly, this works pretty well!

Vector Space Model

Postulate: Documents that are “close together” in vector space “talk about” the same things.

Therefore, retrieve documents based on how close the document is to the query (i.e., similarity ~ “closeness”)
How to Weight Terms?

- Idea: Hans Peter Luhn 1958, IBM

Here’s the intuition:

- Terms that appear often in a document should get high weights
  The more often a document contains the term “dog”, the more likely that the document is “about” dogs.

- Terms that appear in many documents should get low weights
  Words like “the”, “a”, “of” appear in (nearly) all documents.

How do we capture this mathematically?

- Term frequency
- Inverse document frequency

TFxIDF

TFxIDF [Gerald Salton, 1961]

$$TFxIDF(t_i, d_j) = \frac{n(t_i, d_j) \cdot \log \frac{|T_r|}{n(t_i)}}{|D|}$$

- Term Frequency (TF)
  How often a term appears in a document

- Document Frequency (DF)
  Number of documents, which contain a specific term

- Inverse Document Frequency (IDF)
  Discriminator for the importance of a term regarding the number of occurrences in all documents
Working on Indices

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
<th>Doc 5</th>
<th>Doc 6</th>
<th>Doc 7</th>
<th>Doc 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>aid</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>all</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>back</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>brown</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>come</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>dog</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fox</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>good</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>jump</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lazy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>men</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>now</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>over</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>party</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>quick</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>their</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>time</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The term-document matrix has “bag of words” information about the collection.

Small yet Fast?

- Can we make this data structure smaller, keeping in mind the need for fast retrieval?

- Observations:
  - The nature of the search problem requires us to quickly find which documents contain a term
  - The term-document matrix is very sparse
  - Some terms are more useful than others
### Inverted Document Index

<table>
<thead>
<tr>
<th>Term</th>
<th>Postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>aid</td>
<td>4, 8</td>
</tr>
<tr>
<td>all</td>
<td>2, 4, 6</td>
</tr>
<tr>
<td>back</td>
<td>1, 3, 7</td>
</tr>
<tr>
<td>brown</td>
<td>1, 3, 5, 7</td>
</tr>
<tr>
<td>come</td>
<td>2, 4, 6, 8</td>
</tr>
<tr>
<td>dog</td>
<td>3, 5</td>
</tr>
<tr>
<td>fox</td>
<td>3, 5, 7</td>
</tr>
<tr>
<td>good</td>
<td>2, 4, 6, 8</td>
</tr>
<tr>
<td>jump</td>
<td>3</td>
</tr>
<tr>
<td>lazy</td>
<td>1, 3, 5, 7</td>
</tr>
<tr>
<td>men</td>
<td>2, 4, 8</td>
</tr>
<tr>
<td>now</td>
<td>2, 6, 8</td>
</tr>
<tr>
<td>over</td>
<td>1, 3, 5, 7, 8</td>
</tr>
<tr>
<td>party</td>
<td>6, 8</td>
</tr>
<tr>
<td>quick</td>
<td>1, 3</td>
</tr>
<tr>
<td>their</td>
<td>1, 5, 7</td>
</tr>
<tr>
<td>time</td>
<td>2, 4, 6</td>
</tr>
</tbody>
</table>
What goes in the Postings?

- **Boolean retrieval**
  - Just the document number

- **Ranked Retrieval**
  - Document number and term weight ($tf.idf$, ...)

- **Proximity operators**
  - Word offsets for each occurrence of the term

- ...

Summary

- **Information retrieval needs techniques different from database style retrieval**
  - Ranked query model instead of simple look-ups
  - Global statistics about the collection may be needed (e.g., IDFs)
  - Inverted indices are main datastructures

- **Problem: How to perform IR-style retrieval in P2P systems?**
  - How does the distributed setting affect rankings?
  - How to collect global statistics over autonomous peers?
  - How to deal with unstable collections due to network churn?