Ph.D. Defense

Toward Practical Relational Keyword Search Systems

Joel COFFMAN

University of Virginia

10 April 2012
Data Deluge

- Digital information growing 60% every year
  - 10-fold increase every 5 years
- 1200 exabytes of information created in 2010
Data Deluge

- Digital information growing 60% every year
  - 10-fold increase every 5 years
- 1200 exabytes of information created in 2010

How can we take advantage of this information?
Data Deluge

- Digital information growing 60% every year
  - 10-fold increase every 5 years
  - 1200 exabytes of information created in 2010

How can we take advantage of this information?

- Keyword search
  - Preferred means for data exploration and retrieval online
    - No specialized query language
    - No knowledge of data's representation
  - Difficult to support within relational databases

- Existing search engines cannot be used for
  - Social networking sites and microblogs
  - Confidential data (e.g., EMRs and intelligence databases)
Information Management Systems

- Information retrieval (IR)
  - Library users
  - Ranking (ordering by relevance) critical
    - Not all documents containing search terms are relevant
    - Inherent uncertainty regarding information need
    - Many documents may pertain to query

- Databases
  - Business applications
    - e.g., payroll, accounting, inventory
  - Consistency paramount
    - Precise query processing
Relational Data Model

Maintains consistency by eliminating redundancy

- **Foreign keys** refer to related information
  - e.g., id attributes of relations in figure
- **Join(s)** required to recover logical view of information

<table>
<thead>
<tr>
<th>Person</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harrison Ford</td>
<td>10</td>
</tr>
<tr>
<td>Sean Connery</td>
<td>11</td>
</tr>
<tr>
<td>Karen Allen</td>
<td>12</td>
</tr>
<tr>
<td>Steven Spielberg</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor</td>
<td>14</td>
</tr>
<tr>
<td>actress</td>
<td>15</td>
</tr>
<tr>
<td>director</td>
<td>16</td>
</tr>
<tr>
<td>composer</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Film</th>
<th>year</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raiders of the Lost Ark</td>
<td>1981</td>
<td>18</td>
</tr>
<tr>
<td>Indiana Jones and the Last Crusade</td>
<td>1989</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Character</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indiana Jones</td>
<td>7</td>
</tr>
<tr>
<td>Marion Ravenwood</td>
<td>8</td>
</tr>
<tr>
<td>Professor Henry Jones</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cast</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>filmId</td>
<td>personId</td>
<td>roled</td>
<td>characterId</td>
<td>id</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>10</td>
<td>14</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>13</td>
<td>16</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>14</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>11</td>
<td>14</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>16</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>15</td>
<td>8</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Relational Keyword Search

Definition (Intuitive)

Search technique that connects disparate data (i.e., tuples) via relationships present in relational database.
Relational Keyword Search

Definition (Intuitive)

Search technique that connects disparate data (i.e., tuples) via relationships present in relational database.

Definition (Technical)

A relational database is conceptually a graph $G = (V, E)$. A keyword query $Q$ is a set of terms. A result for $Q$ is a tree $T$ that is reduced with respect to $Q' \subseteq Q$ (i.e., $T$ has no proper subtree that contains all the terms in $Q'$).
Relational Keyword Search (continued)

Figure: Example IMDb data graph (left) and inverted index (right).

Postings

- actor → 14
- ark → 18
- composer → 17
- connery → 11
- crusade → 19
- director → 16
- ford → 10
- harrison → 10
- indiana → 7, 19
- jones → 7, 9, 19
- last → 19
- raiders → 18
- sean → 11
- ...
### Example

What is the relationship between **Harrison Ford** and **Sean Connery**?

---

#### Figure: Example IMDb data graph (left) and inverted index (right)
Relational Keyword Search (continued)

Example

What is the relationship between Harrison Ford and Sean Connery?

Algorithm

1. Scan inverted index

```
<table>
<thead>
<tr>
<th>17</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
```

Postings

- actor → 14
- ark → 18
- composer → 17
- connery → 11
- crusade → 19
- director → 16
- ford → 10
- harrison → 10
- indiana → 7, 19
- jones → 7, 9, 19
- last → 19
- raiders → 18
- sean → 11

...
Relational Keyword Search (continued)

Example

What is the relationship between Harrison Ford and Sean Connery?

Algorithm

1. Scan inverted index
2. Enumerate results

Figure: Example IMDb data graph (left) and inverted index (right).

Postings

| actor → 14 |
| ark → 18 |
| composer → 17 |
| connery → 11 |
| crusade → 19 |
| director → 16 |
| ford → 10 |
| harrison → 10 |
| indiana → 7, 19 |
| jones → 7, 9, 19 |
| last → 19 |
| raiders → 18 |
| sean → 11 |
| ... |
Relational Keyword Search (continued)

Example

What is the relationship between Harrison Ford and Sean Connery?

Algorithm

1. Scan inverted index
2. Enumerate results

Figure: Example IMDb data graph (left) and inverted index (right).
Relational Keyword Search (continued)

Example
What is the relationship between Harrison Ford and Sean Connery?

Algorithm
1. Scan inverted index
2. Enumerate results
3. Rank results

| 17 | 15 |
| 13 — 2 — 18 — 6 — 12 |
| 5 — 16 | 8 |
| 7 — 1 |
| 19 — 3 — 10 |
| 11 — 4 — 14 |
| 9 |

<table>
<thead>
<tr>
<th>Postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor → 14</td>
</tr>
<tr>
<td>ark → 18</td>
</tr>
<tr>
<td>composer → 17</td>
</tr>
<tr>
<td>connery → 11</td>
</tr>
<tr>
<td>crusade → 19</td>
</tr>
<tr>
<td>director → 16</td>
</tr>
<tr>
<td>ford → 10</td>
</tr>
<tr>
<td>harrison → 10</td>
</tr>
<tr>
<td>indiana → 7, 19</td>
</tr>
<tr>
<td>jones → 7, 9, 19</td>
</tr>
<tr>
<td>last → 19</td>
</tr>
<tr>
<td>raiders → 18</td>
</tr>
<tr>
<td>sean → 11</td>
</tr>
</tbody>
</table>

Figure: Example IMDb data graph (left) and inverted index (right).
Relational Keyword Search (continued)

Example

What is the relationship between Harrison Ford and Sean Connery?

Algorithm
1. Scan inverted index
2. Enumerate results
3. Rank results

3.1 *Indiana Jones and the Last Crusade*

<table>
<thead>
<tr>
<th>Postings</th>
</tr>
</thead>
<tbody>
<tr>
<td>actor → 14</td>
</tr>
<tr>
<td>ark → 18</td>
</tr>
<tr>
<td>composer → 17</td>
</tr>
<tr>
<td>connery → 11</td>
</tr>
<tr>
<td>crusade → 19</td>
</tr>
<tr>
<td>director → 16</td>
</tr>
<tr>
<td>ford → 10</td>
</tr>
<tr>
<td>harrison → 10</td>
</tr>
<tr>
<td>indiana → 7, 19</td>
</tr>
<tr>
<td>jones → 7, 9, 19</td>
</tr>
<tr>
<td>last → 19</td>
</tr>
<tr>
<td>raiders → 18</td>
</tr>
<tr>
<td>sean → 11</td>
</tr>
</tbody>
</table>

Figure: Example IMDb data graph (left) and inverted index (right).
Relational Keyword Search (continued)

Example

What is the relationship between Harrison Ford and Sean Connery?

Algorithm

1. Scan inverted index
2. Enumerate results
3. Rank results
   3.1 *Indiana Jones and the Last Crusade*
   3.2 actor

Figure: Example IMDb data graph (left) and inverted index (right).
Relational Keyword Search (continued)

Example
What is the relationship between Harrison Ford and Sean Connery?

Algorithm
1. Scan inverted index
2. Enumerate results
3. Rank results
   3.1 *Indiana Jones and the Last Crusade*
   3.2 actor
   3.3 ...

```
<table>
<thead>
<tr>
<th>17</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure: Example IMDb data graph (left) and inverted index (right).

Joel Coffman
Practical Relational Keyword Search 10 April 2012
Research Challenges

Answering relational keyword queries

- Enumerate search results efficiently
- Rank search results to maximize search effectiveness

Evaluating search techniques

- Standardized framework to evaluate systems
- Ensure evaluations are representative
Research Challenges

Answering relational keyword queries
- Enumerate search results efficiently
- Rank search results to maximize search effectiveness

Evaluating search techniques
- Standardized framework to evaluate systems
- Ensure evaluations are representative
Contributions

Answering relational keyword queries

▶ Novel ranking schemes that improve search effectiveness
  ▶ Structured cover density ranking
  ▶ SVM rank

Evaluating search techniques

▶ Benchmark to evaluate relational keyword search techniques
  ▶ Empirical evaluation of existing search techniques
▶ Analysis of real user queries to ensure evaluation query workloads are representative
Outline

Introduction
   Historical Perspective
   Keyword Search in Relational Databases
Evaluation Benchmark
   Benchmark
   Experiments
Ranking Relational Keyword Search Results
   Structured Cover Density Ranking
   SVM rank
   Experiments
Search Log Analysis
   Synthetic Queries
Conclusion
   Contributions
   Future Work
Research Challenges

Answering relational keyword queries

- Enumerate search results efficiently
- Rank search results to maximize search effectiveness

Evaluating search techniques

- Standardized framework to evaluate systems
- Ensure evaluations are representative
Why Start with Empirical Evaluation?

- Objective empirical evaluation is a cornerstone of the IR community (Singhal, 2001)

- Focus research on the right problems

  *If you don’t know where you are going, any road will get you there.*  
  ~ Lewis Carroll

- Prevent bias in evaluation

  *Self-authored queries have a strong potential for bias: it is too easy to formulate queries that are favourable to your own over other algorithms.*  
  (Webber, 2010)
Motivation

Relational keyword search has been a hot topic since 2002

- Numerous proposed search techniques
  - e.g., BANKS, DISCOVER, DPBF, BLINKS, EASE, STAR
  - all claim to outperform predecessors

⇒ Extensive survey of existing evaluations

- Datasets, queries, relevance assessment

- Claims (in aggregate) are overstated
  - ≈ 16-fold improvement in search effectiveness
  - orders of magnitude improvement in execution time

- Evaluations ad hoc, no standardization
  - different evaluations contradict each other
Outline

Introduction
   Historical Perspective
   Keyword Search in Relational Databases
Evaluation Benchmark
   Benchmark
   Experiments
Ranking Relational Keyword Search Results
   Structured Cover Density Ranking
   SVM rank
   Experiments
Search Log Analysis
   Synthetic Queries
Conclusion
   Contributions
   Future Work
First benchmark designed to evaluate keyword search in relational databases

- Satisfies calls by research community to standardize evaluations
- Incorporates best practices from IR community
- Considers both performance and search effectiveness
- Largest empirical evaluation to date in literature


Joel Coffman and Alfred C. Weaver. An Empirical Performance Evaluation of Relational Keyword Search Techniques. Under review in Transactions on Knowledge and Data Engineering (TKDE).
First benchmark designed to evaluate keyword search in relational databases

- Satisfies calls by research community to standardize evaluations
  
  Contributions from the research community are highly demanded for developing comprehensive frameworks for evaluating the retrieval and ranking strategies of keyword search on various structured data models. (Chen et al., 2009)

- Incorporates best practices from IR community
- Considers both performance and search effectiveness
- Largest empirical evaluation to date in literature
First benchmark designed to evaluate keyword search in relational databases

- Satisfies calls by research community to standardize evaluations
- Incorporates best practices from IR community
  
  [...] more needs to be done to set [keyword search on structured data] on a firm basis, to validate its results, and to inspire the confidence needed to convert this research technology into deployed tools. And most of these desiderata depend upon improvements in evaluation method. (Webber, 2010)

- Considers both performance and search effectiveness
- Largest empirical evaluation to date in literature
First benchmark designed to evaluate keyword search in relational databases

- Satisfies calls by research community to standardize evaluations
- Incorporates best practices from IR community
- Considers both performance and search effectiveness
- Largest empirical evaluation to date in literature


Joel Coffman and Alfred C. Weaver. An Empirical Performance Evaluation of Relational Keyword Search Techniques. Under review in Transactions on Knowledge and Data Engineering (TKDE).
## Datasets

| Dataset             | Size (MBs) | Relations | \(|V|\) | \(|E|\)     | \(|T|\) |
|---------------------|------------|-----------|-----|-----------|------|
| **MONDIAL**         | 16         | 28        | 17  | 28        | 12   |
| IMDb                | 9262       | 20        | 44,303 | 109,987  | 21,987 |
| IMDb subset         | 459        | 6         | 1,673 | 3,037     | 1,748 |
| **Wikipedia**       | 664        | 42        | 1,575 | 1,738     | 760  |
| Wikipedia subset    | 391        | 6         | 206  | 392       | 750  |

**Legend**

\(|V|\) number of vertices (i.e., tuples)

\(|E|\) number of edges in the data graph (i.e., foreign keys)

\(|T|\) number of terms in database
Followed accepted practice of IR community

- Individual creates information need (and query) for test collection
- Same individual judges relevance of results
- 50 information needs per dataset
  - queries realistic, not random terms
  - **minimum** for evaluating retrieval systems (but not met in previous evaluations)
    - only two prior evaluations include 50 representative queries for a dataset
- Binary relevance assessments
  - Relevant results identified by executing SQL queries
Outline

Introduction
  Historical Perspective
  Keyword Search in Relational Databases
Evaluation Benchmark
  Benchmark
  Experiments
Ranking Relational Keyword Search Results
  Structured Cover Density Ranking
  SVM rank
  Experiments
Search Log Analysis
  Synthetic Queries
Conclusion
  Contributions
  Future Work
Metrics

- Queries completed
  - not timeouts and no exhaustion of heap space
- Runtime performance
  - Execution time
  - Response time
- Search effectiveness
  - Precision $\equiv \frac{\text{number of relevant results retrieved}}{\text{number of relevant results}}$
    - Precision @ $k$
  - Mean reciprocal rank (MRR)
  - 11-point interpolated precision
  - Mean average precision (MAP)
  - Normalized discounted cumulative gain (nDCG)
Metrics

- Queries completed
  - not timeouts and no exhaustion of heap space
- Runtime performance
  - Execution time
  - Response time
- Search effectiveness
  - Precision \(\equiv \frac{\text{# relevant results retrieved}}{\text{# relevant results}}\)
    - Precision @ \(k\)
  - Mean reciprocal rank (MRR)
  - 11-point interpolated precision
  - MAP
  - NDCG
Execution Time

**Figure:** MONDIAL execution time: lower is better.

- BANKS
- DISCOVER
- DISCOVER-II
- BANKS-II
- DPBF
- BLINKS
- STAR
Execution Time (continued)

Figure: IMDb execution time: lower is better.
Execution Time (continued)

Figure: Wikipedia execution time: lower is better.
Search Effectiveness

MAP

Figure: Mean average precision (MAP). Higher is better (MAP ∈ [0, 1]).
Summary

- Abysmal runtime performance
- Poor scalability
- No ranking scheme outperforms all others
  ⇒ Previous evaluations overstate efficiency and effectiveness of search techniques
  - Baseline to assess proposed improvements
Outline

Introduction
  Historical Perspective
  Keyword Search in Relational Databases
Evaluation Benchmark
  Benchmark
  Experiments
Ranking Relational Keyword Search Results
  Structured Cover Density Ranking
  SVM rank
  Experiments
Search Log Analysis
  Synthetic Queries
Conclusion
  Contributions
  Future Work
Research Challenges

Answering relational keyword queries

- Enumerate search results efficiently
- Rank search results to maximize search effectiveness

Evaluating search techniques

- Standardized framework to evaluate systems
- Ensure evaluations are representative
Structured Cover Density Ranking

Generalization of (unstructured) cover density ranking

- Users prefer results that contain all search terms to appear first
- Ranking algorithm is trivially parallelizable

⇒ Demonstrates usefulness of adopting work from IR community

More details

Joel Coffman and Alfred C. Weaver. Structured Data Retrieval using Cover Density Ranking. In *Workshop on Keyword Search in Structured Data*, Indianapolis, IN, June 2010.
Outline

Introduction
- Historical Perspective
- Keyword Search in Relational Databases

Evaluation Benchmark
- Benchmark
- Experiments

Ranking Relational Keyword Search Results
- Structured Cover Density Ranking
- SVM rank
- Experiments

Search Log Analysis
- Synthetic Queries

Conclusion
- Contributions
- Future Work
Motivation

Existing ranking schemes

- Guided by intuition and anecdotal evidence
  - e.g., ranking by total edge weight
- Many different factors $\Rightarrow$ lots of parameters

Joel Coffman and Alfred C. Weaver. Learning to Rank Results in Relational Keyword Search. In *Conference on Information and Knowledge Management (CIKM)*, Glasgow, Scotland, October 2011.
Motivation

Existing ranking schemes

▶ Guided by intuition and anecdotal evidence
  ▶ e.g., ranking by total edge weight
  ▶ Many different factors ⇒ lots of parameters

No indication of why particular scoring functions outperform others

Joel Coffman and Alfred C. Weaver. Learning to Rank Results in Relational Keyword Search. In Conference on Information and Knowledge Management (CIKM), Glasgow, Scotland, October 2011.
Correlating Factors with Relevance

**Table:** Kendall $\tau$ rank correlation coefficient ($p < 0.01$)

<table>
<thead>
<tr>
<th>Factor</th>
<th>$\tau$</th>
<th>Factor</th>
<th>$\tau$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>N</td>
<td>$</td>
<td>-0.045</td>
</tr>
<tr>
<td>$</td>
<td>E</td>
<td>$</td>
<td>-0.045</td>
</tr>
<tr>
<td>$min\ weight(e)$</td>
<td>-0.036</td>
<td>$\sum weight(n)$</td>
<td>0.026</td>
</tr>
<tr>
<td>$max\ weight(e)$</td>
<td>0.215</td>
<td>$\frac{weight(n)}{weight(e)}$</td>
<td>0.024</td>
</tr>
<tr>
<td>$\sum weight(e)$</td>
<td>0.123</td>
<td>pivoted normalization</td>
<td>0.310</td>
</tr>
<tr>
<td>$\frac{weight(e)}$</td>
<td>0.154</td>
<td>Euclidean ($L^2$) distance</td>
<td>-0.207</td>
</tr>
</tbody>
</table>

- Most scoring factors not correlated with relevance
- IR scoring function most correlated with relevance
Training

- Machine learning $\implies$ ordinal regression
  - $SVM^{\text{rank}}$ (Joachims, *KDD ’06*)
  - 10-fold cross validation

Definition (SVM)

support vector machine
Outline

Introduction
   Historical Perspective
   Keyword Search in Relational Databases
Evaluation Benchmark
   Benchmark
   Experiments
Ranking Relational Keyword Search Results
   Structured Cover Density Ranking
   SVM rank
   Experiments
Search Log Analysis
   Synthetic Queries
Conclusion
   Contributions
   Future Work
Figure: Normalized discounted cumulative gain (nDCG) for each dataset. Higher is better (NDCG ∈ [0, 1]).
Comparison with Traditional Scoring

Figure: Normalized discounted cumulative gain (nDCG) for each dataset compared to traditional alternatives. Higher is better (NDCG ∈ [0, 1]).
Outline

Introduction
  Historical Perspective
  Keyword Search in Relational Databases
Evaluation Benchmark
  Benchmark
  Experiments
Ranking Relational Keyword Search Results
  Structured Cover Density Ranking
  SVM rank
  Experiments
Search Log Analysis
  Synthetic Queries
Conclusion
  Contributions
  Future Work
Research Challenges

Answering relational keyword queries
  ▶ Enumerate search results efficiently
  ▶ Rank search results to maximize search effectiveness

Evaluating search techniques
  ▶ Standardized framework to evaluate systems
  ▶ Ensure evaluations are representative
Motivation

What would users search for if relational keyword search systems were available?

- effective evaluation depends upon answer
Motivation

What would users search for if relational keyword search systems were available?

- effective evaluation depends upon answer

Existing evaluations

- use “queries [...] with different numbers of query keywords” (Li et al., 2011)
- “include a wide variety of keywords [...] in the query sets” (Luo et al., 2011)
- “independently derived a variety of information needs” (Coffman and Weaver, 2010)
Motivation

What would users search for if relational keyword search systems were available?

► effective evaluation depends upon answer

Existing evaluations

► use “queries [...] with different numbers of query keywords” (Li et al., 2011)

► “include a wide variety of keywords [...] in the query sets” (Luo et al., 2011)

► “independently derived a variety of information needs” (Coffman and Weaver, 2010)

Representativeness?
Motivation

What would users search for if relational keyword search systems were available?

- **effective evaluation depends upon answer**

Existing evaluations

- use “queries [...] with different numbers of query keywords” (Li *et al*., 2011)
- “include a wide variety of keywords [...] in the query sets” (Luo *et al*., 2011)
- “independently derived a variety of information needs” (Coffman and Weaver, 2010)

Representativeness?

### Dataset

| Dataset | $|Q|$ | $[q]$ | $\overline{[q]}$ |
|---------|------|-------|-----------------|
| IMDb    | 40   | 2–8   | 5.0             |
| IMDb    | 22   | 2–3   | 2.4             |
| IMDb    | 4    | 2–3   | 2.5             |
| IMDb    | 180  | 3–7   | 5.0             |
| IMDb    | 20   | 3–5   | 3.3             |
| IMDb    | 50   | 1–26  | 3.9             |
| IMDb    | 99234| 1–96  | 2.7             |

### Legend

- $|Q|$ total number of queries
- $[q]$ range in number of query terms
- $\overline{[q]}$ mean number of terms per query
Motivation (continued)

Synthetically generate representative workloads

- Labor-intensive for individuals to create representative queries
- Huge datasets $\Rightarrow$ subsets $\Rightarrow$ cannot use existing query log
  - e.g., the Internet Movie Database (IMDb)$^1 \approx 44$ million tuples

Joel Coffman and Alfred C. Weaver. What Are We Searching For? Analyzing User Objectives When Searching Relational Data. In Workshop on Web Search Click Data (WSCD), Seattle, WA, February 2012.

$^1$http://www.imdb.com/
Synthetic Queries

Query Templates

Abstraction of real queries that allow synthesizing representative workload

- replace substrings of real queries with types in data model
- synthetic workloads
  - create queries from tuples retained in subset

Example

<table>
<thead>
<tr>
<th>Original</th>
<th>Template</th>
<th>Synthesized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harrison Ford</td>
<td>⟨Person.name⟩</td>
<td>Al Pacino</td>
</tr>
<tr>
<td>The Parent Trap 1998</td>
<td>⟨Film.title⟩ ⟨Film.year⟩</td>
<td>Superman Returns 2006</td>
</tr>
<tr>
<td>Tom Hanks movies</td>
<td>⟨Person.name⟩ movies</td>
<td>Cassie Miller movies</td>
</tr>
</tbody>
</table>
## Synthetic Queries

### Query Templates (continued)

**Table:** Most common query templates

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Template</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDb</td>
<td>⟨Person.name⟩</td>
<td>40.8</td>
</tr>
<tr>
<td></td>
<td>⟨Film.title⟩</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td>⟨Film.title⟩ ⟨Film.type⟩</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>⟨Film.title⟩ ⟨Film.year⟩</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>⟨Character.name⟩</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>⟨Character.name⟩ ⟨Film.title⟩</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>⟨Film.title⟩ quotes</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>⟨Person.name⟩ ⟨Film.type⟩</td>
<td>1.2</td>
</tr>
</tbody>
</table>
## Synthetic Query Generation

**Table:** Query statistics for IMDb workloads generated synthetically.

| Source                    | \(|Q|\) | \([q]\) | \(\bar{q}\) |
|---------------------------|--------|--------|---------|
| BLINKS                    | 40     | 2–8    | 5.0     |
| SPARK                     | 22     | 2–3    | 2.4     |
| BANKS-III                 | 4      | 2–3    | 2.5     |
| STAR                      | 180    | 3–7    | 5.0     |
| Qin et al.                | 20     | 3–5    | 3.3     |
| Coffman and Weaver        | 50     | 1–26   | 3.9     |
| Synthetic                 | 80     | 1–11   | 2.73    |
| Synthetic                 | 160    | 1–9    | 2.76    |
| Synthetic                 | 400    | 1–11   | 2.73    |
| Synthetic                 | 800    | 1–10   | 2.62    |
| IMDb Log                  | 99234  | 1–96   | 2.75    |

**Legend**

- \(|Q|\): total number of queries
- \([q]\): range in number of query terms
- \(\bar{q}\): mean number of terms per query
Figure: Box plots of the total number of tuples containing search terms for each query. More similar to search log (far right) is better.
Outline

Introduction
  Historical Perspective
  Keyword Search in Relational Databases
Evaluation Benchmark
  Benchmark
  Experiments
Ranking Relational Keyword Search Results
  Structured Cover Density Ranking
  SVM rank
  Experiments
Search Log Analysis
  Synthetic Queries
Conclusion
  Contributions
  Future Work
Research Challenges

Answering relational keyword queries

- Enumerate search results efficiently
- Rank search results to maximize search effectiveness

Evaluating search techniques

- Standardized framework to evaluate systems
- Ensure evaluations are representative
Evaluation Benchmark

- Survey of existing evaluations that exposes their inadequacy
- Follows best practices established by IR community to evaluate retrieval systems
  - Satisfies call by research community to standardize evaluation
- Most extensive empirical evaluation of relational keyword search to date
- Publicly available for other researchers to use
  - e.g., Bicer et al. (CIKM ’11) and Mass and Sagiv (WSDM ’12)
Structured Cover Density Ranking

- Ranking adheres to users expectations
- Generalize unstructured scoring function to handle structured documents
- Outperforms previous attempts to adapt scoring functions from IR community
SVM rank

- Principled investigation of scoring functions
  - why some functions outperform others
  - what factors are most important when ranking
- Significantly improves search effectiveness as compared to previous work
  - simplified version easier to compute *and* more effective than previous work
Search Log Analysis

- Taxonomy to classify queries by intent and expression
  - First analysis of real user queries in context of relational keyword search
- Analysis indicates that existing query workloads are skewed away from the queries that users tend to submit
- Synthetic query generation
  - Construct very large, representative query workloads
Future Work

- Update benchmark
  - Synthetically generate very large query workloads
  - Progression of database subsets
- Runtime performance improvements
  - Existing algorithms best-case performance not acceptable
  - Fast enumeration algorithms
- Visualization of results (in context of original graph)
This slide intentionally blank.
Backup Slides

Questions

Search Techniques

Evaluation Benchmark

Ranking
- Structured Cover Density Ranking
- SVM rank Analysis

Classifying Queries
- Analysis
Prepared Responses

- Why not precompute potential results?
- What about the cloud?
- Isn’t 3 GB of memory small for experiments?
Why not precompute potential results?

Approach has been suggested and implemented
▶ e.g., EKSO, SAINT, CSTree

▶ Advantages
▶ Efficient query processing using traditional inverted index(es)

▶ Disadvantages
▶ Difficult to support complex predicates
▶ High up-front cost before allowing any searches
▶ Expensive (or not practical!) to keep index consistent with underlying data
▶ What happens when distances among search terms exceeds indexing threshold?
Offline Indexing Cost

Number of result trees increases **exponentially** with size

- **Indexing time**
  - 92 minutes for 1340 MB of data (Baid et al., 2010)

- **Index size**
  - 2–8 times larger than size of database (Su and Widom, 2005)
  - 20 times larger than size of database (Baid et al., 2010)
Offline Index Consistency

How is an external database index kept up-to-date?

- **Hand waving** solution by graph-based approaches
  - Replay database log against graph (Kacholia et al., 2005)
    - not actually implemented
- **Ignored** by most offline approaches
  - EKSO updates virtual documents via database triggers (Su and Widom, 2005)
    - DB2 saturated at 10 inserts per second
    - DB2 Net Search Extender (supporting full text search) saturated at 0.75 inserts per second
- **Cannot** be supported efficiently by some search techniques
  - e.g., update could require recomputing all precomputed scores
What about the cloud?

Couldn’t existing performance problems be solved by moving search to the cloud?

- Not using existing graph-based search techniques
  - Existing algorithms are not parallel

- Non-trivial to improve both
  - execution time (partitioning search requires duplicate data graphs) and
  - memory consumption (partitioning data graph increases communication cost)

Definition (Cloud Computing)

elastic computing power and storage delivered on demand
# Best-Case Performance

**Table:** Response time for IMDb.

<table>
<thead>
<tr>
<th>System</th>
<th>query seg.</th>
<th>guided</th>
<th>oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>resp.</td>
<td>resp.</td>
<td>resp.</td>
</tr>
<tr>
<td>BANKS</td>
<td>2686.0</td>
<td>3451.3</td>
<td>236.6</td>
</tr>
<tr>
<td>DISCOVER</td>
<td>227.9</td>
<td>31.9</td>
<td>96.7</td>
</tr>
<tr>
<td>DISCOVER-II</td>
<td>201.8</td>
<td>30.9</td>
<td>98.6</td>
</tr>
<tr>
<td>BANKS-II</td>
<td>3604.3</td>
<td>3275.5</td>
<td>457.2</td>
</tr>
<tr>
<td>DPBF</td>
<td>2042.1</td>
<td>1640.5</td>
<td>40.4</td>
</tr>
<tr>
<td>BLINKS</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Isn’t 3 GB of memory small for experiments?

Recent experimental workloads use 2–4 GB of memory
  ▶ Memory consumption rarely reported in literature
What memory consumption is “reasonable?”
  ▶ Few lower bounds reported in literature
    ▶ Require data transformations ⇒ increase execution time and storage requirements
  ▶ Lower bound for entire IMDb database is 24 GB
  ▶ BLINKS uses ≈ 900 MB of memory for MONDIAL database (size = 16 MB)
    ▶ 3 GB insufficient to create index for IMDb subset with 44,000 tuples
Outline

Questions

Search Techniques

Evaluation Benchmark

Ranking
  Structured Cover Density Ranking
  SVM rank Analysis

Classifying Queries
  Analysis
Approaches to Relational Keyword Search

- Schema-based
  - Specific to relational databases
  - Enumerate all possible SQL expressions that might join tuples containing search terms

- Graph-based
  - Applicable to arbitrary data graphs
  - Use ordinary graph traversals (e.g., breadth-first search) to find relationships among tuples

- Precompute
  - Materialize possible results prior to any searches
  - Index results using traditional inverted index
### Comparison of Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Index</th>
<th>Efficiency</th>
<th>Ranking</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cons.</td>
<td>index</td>
<td>exec.</td>
<td>IR</td>
</tr>
<tr>
<td>Schema</td>
<td>⊖</td>
<td>⊕</td>
<td>⊖</td>
<td>⊖</td>
</tr>
<tr>
<td>Graph</td>
<td>⊖</td>
<td>⊖</td>
<td>⊖</td>
<td>⊕</td>
</tr>
<tr>
<td>Precompute</td>
<td>⊖</td>
<td>⊖</td>
<td>⊕</td>
<td>⊕</td>
</tr>
</tbody>
</table>

### Legend
- ☑ good
- ◯ fair
- ⊖ poor

- **cons.**: consistency
- **exec.**: execution time
- **IR**: IR
- **weight**: IR-style ranking
- **proximity search**
Outline

Questions

Search Techniques

Evaluation Benchmark

Ranking
  Structured Cover Density Ranking
  SVM rank Analysis

Classifying Queries
  Analysis
Survey of Existing Evaluations

- Existing experiments unrepeatable
  - Few details included in literature
  - Datasets, query workloads, and relevance assessments not released
- Query workloads vary widely
  - 12–1100 queries included in experiments
  - Too few representative queries
- Experiments
  - Performance-focus, less than half consider search effectiveness
  - Little comparison among different search techniques
Motivation

Example (Search Effectiveness)

DISCOVER  ≪  DISCOVER-II  ≪  Liu et al.  ≪  SPARK  ≪  Xu et al.
Motivation

Example (Search Effectiveness)

DISCOVER ≪ DISCOVER-II ≪ Liu et al. ≪ SPARK ≪ Xu et al.

► ≈ 16-fold improvement in search effectiveness during past decade
Motivation

Example (Search Effectiveness)

DISCOVER ≪ DISCOVER-II ≪ Liu et al. ≪ SPARK ≪ Xu et al.

- ≈ 16-fold improvement in search effectiveness during past decade
- Liu et al. claim to be better than Google
Motivation

Example (Search Effectiveness)

DISCOVER ≪ DISCOVER-II ≪ Liu et al. ≪ SPARK ≪ Xu et al.

- ≈ 16-fold improvement in search effectiveness during past decade
- Liu et al. claim to be better than Google
- SPARK achieves MRR of 1.0
Motivation

Example (Search Effectiveness)

DISCOVER ≪ DISCOVER-II ≪ Liu et al. ≪ SPARK ≪ Xu et al.

- ≈ 16-fold improvement in search effectiveness during past decade
- Liu et al. claim to be better than Google
- SPARK achieves MRR of 1.0
  - Best systems at TREC score ≈ 0.8 (Webber, 2010)
Motivation (continued)

Discrepancies among evaluations common
Motivation (continued)

Discrepancies among evaluations common

Example (Runtime Performance)

BANKS \gg \text{BANKS-II}
BANKS-II \gg \text{BLINKS}
Motivation (continued)

Discrepancies among evaluations common

Example (Runtime Performance)

BANKS $\gg$ BANKS-II
BANKS-II $\gg$ BLINKS
BANKS $\approx$ BANKS-II $\approx$ BLINKS $\gg$ STAR

Orders of magnitude improvement claimed by evaluations
Not validated by later experiments

Joel Coffman
Motivation (continued)

Discrepancies among evaluations common

Example (Runtime Performance)

BANKS $\gg$ BANKS-II

BANKS-II $\gg$ BLINKS

BANKS $\approx$ BANKS-II $\approx$ BLINKS $\gg$ STAR

- Orders of magnitude improvement claimed by evaluations
  - Not validated by later experiments
Mean average precision (MAP)

Definition (Average precision (AP))

Mean precision value computed at the position of each relevant result; relevant results not returned receive a score of 0.

\[ AP = \frac{1}{|R|} \sum_{i=1}^{|R|} \text{precision}(r_i) \]

where \(|R|\) is the set of relevant results and \(r_i\) is the position of the relevant result \(r\) in the list of results.

Definition (Mean average precision (MAP))

Mean of average precision (AP) across all queries in workload.
Normalized discounted cumulative gain (nDCG)

Definition (Discounted cumulative gain (DCG))

Relevance of results is penalized proportionally to its position in the list of results (because the user may abandon the search before seeing the result).

\[ \text{DCG}_p = \sum_{i=1}^{p} \frac{2^{\text{rel}_i} - 1}{\lg(i + 1)} \]

Definition (Normalized discounted cumulative gain (nDCG))

Discounted cumulative gain (DCG) normalized by dividing by the ideal DCG at that position.
Outline

Questions

Search Techniques

Evaluation Benchmark

Ranking

Structured Cover Density Ranking

Cover Density Ranking

Structured Cover Density Ranking

SVM rank Analysis

Classifying Queries

Analysis
Outline

Questions

Search Techniques

Evaluation Benchmark

Ranking
  Structured Cover Density Ranking
    Cover Density Ranking
    Structured Cover Density Ranking
    SVM rank Analysis

Classifying Queries
  Analysis
Cover Density Ranking

Definition (document)
sequence of terms, \( d = (t_1, t_2, \ldots, t_{|d|}) \)

Definition (extent)
sequence of terms, \((t_p, t_q)\), in document \(d\)

Definition (cover)
extent that 1) contains all terms from a set \(T\) and 2) does not contain a smaller extent that also satisfies \(T\)

Definition (cover set)
set of all covers for \(T\) in a document \(d\)
Cover Density Ranking (continued)

<table>
<thead>
<tr>
<th>id</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_7$</td>
<td>Indiana$^1$ Jones$^2$</td>
</tr>
<tr>
<td>$d_9$</td>
<td>Professor$^1$ Henry$^2$ Jones$^3$</td>
</tr>
<tr>
<td>$d_{19}$</td>
<td>Indiana$^1$ Jones$^2$ and$^3$ the$^4$ Last$^5$ Crusade$^6$</td>
</tr>
</tbody>
</table>

Example

Ranking by coordination level for “Indiana Jones Crusade” gives

$$C_3 = \{d_{19}\}, \quad C_2 = \{d_7\}, \quad C_1 = \{d_9\}$$

Cover sets are

- $d_7: \{(1, 2)\}$
- $d_9: \{(3, 3)\}$
- $d_{19}: \{(1, 6)\}$
Scoring

Intuitions:
1. the more covers contained within a document, the more likely the document is relevant
Scoring

Intuitions:

1. the more covers contained within a document, the more likely the document is relevant
2. the shorter the cover, the more likely the corresponding text is relevant
Scoring

Intuitions:
1. the more covers contained within a document, the more likely the document is relevant
2. the shorter the cover, the more likely the corresponding text is relevant

\[
\text{score}(C) = \sum_{j=1}^{n} \text{score}(p_j, q_j)
\]  

(1)

where

\[
\text{score}(p, q) = \begin{cases} 
\frac{\mathcal{H}}{q - p + 1} & \text{if } q - p + 1 > \mathcal{H} \\
1 & \text{otherwise}
\end{cases}
\]  

(2)

and \( \mathcal{H} \in [1, \infty) \) is a tuning parameter.
Example

Cover sets

\[
\begin{align*}
\text{id} & \quad \text{text} \\
\text{d}_7 & \quad \text{Indiana}^1 \text{ Jones}^2 \\
\text{d}_9 & \quad \text{Professor}^1 \text{ Henry}^2 \text{ Jones}^3 \\
\text{d}_{19} & \quad \text{Indiana}^1 \text{ Jones}^2 \text{ and}^3 \text{ the}^4 \text{ Last}^5 \text{ Crusade}^6
\end{align*}
\]

Scores (let $\mathcal{H} = 4$)

\[
\begin{align*}
\text{score}(\text{d}_7) & = 1 \\
\text{score}(\text{d}_9) & = 1 \\
\text{score}(\text{d}_{19}) & = \frac{4}{6} = \frac{2}{3}
\end{align*}
\]

Note: Abuse of notation

Final ranking (coordination level then by cover density): $d_{19}, d_7, d_9$
Structured Cover Density Ranking

Generalization of (unstructured) cover density ranking

▶ Must preserve properties of original ranking function, i.e.
  1. the more covers contained within a document, the more likely the document is relevant
  2. the shorter the cover, the more likely the corresponding text is relevant

▶ Term positions critical
  ▶ Previous generalizations preserve term frequency
Definitions

Definition (structured document)

set of fields, \( d = \{ f_1, f_2, \ldots, f_{|d|} \} \)

Definition (structured extent)

subset of document’s fields, \( E \subseteq d \)

Definition (structured cover)

structured extent that 1) contains all terms from a set \( T \) and 2) does not contain a smaller structured extent that also satisfies \( T \)

Definition (structured cover set)

set of all structured covers for \( T \) in a structured document \( d \)
Example

Ranking by coordination level for “Jones Ford” gives

\[ C_2 = \{d_{7,10}\}, \quad C_1 = \{d_7, d_{10}\} \]

Cover sets are

\[
\begin{align*}
  d_7 & : \quad \{d_7\} \\
  d_{10} & : \quad \{d_{10}\} \\
  d_{7,10} & : \quad \{d_7, d_{10}\}
\end{align*}
\]
Scoring

\[
score(d) = \sum_{E \in d} score(E) \tag{3}
\]

where

\[
score(E) = \begin{cases} 
\frac{H}{|E|} & \text{if } |E| > H \\
1 & \text{otherwise}
\end{cases} \tag{4}
\]
Outline

Questions

Search Techniques

Evaluation Benchmark

Ranking

Structured Cover Density Ranking

Cover Density Ranking

Structured Cover Density Ranking

SVM rank Analysis

Classifying Queries

Analysis
Feature Selection

Algorithm
- Identify factor that contributes least to ranking
- Train new model without that factor
- \{ Repeat \}

Removal order
1. $\max weight(n)$
2. $\sum weight(n)$
3. $\overline{weight(n)}$
4. $L^2$ norm
5. $\sum weight(e)$
6. $|N|$
7. $\max weight(e)$
8. $\overline{weight(e)}$
9. $|E|$
10. $\min weight(e)$
Feature Selection

Algorithm

- Identify factor that contributes least to ranking
- Train new model without that factor
- Repeat

SVM rank (minimal)

- \( \min weight(n) \)
- pivoted normalization

\( \Rightarrow \) edge weight not indicative of relevance

Removal order

1. \( \max weight(n) \)
2. \( \sum weight(n) \)
3. \( weight(n) \)
4. \( L^2 \) norm
5. \( \sum weight(e) \)
6. \( |N| \)
7. \( \max weight(e) \)
8. \( weight(e) \)
9. \( |E| \)
10. \( \min weight(e) \)
Sensitivity

Table: Comparison of SVM rank trained using different cross folds.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>nDCG baseline</th>
<th>µ</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONDIAL</td>
<td>0.849</td>
<td>0.862</td>
<td>0.006</td>
</tr>
<tr>
<td>IMDb</td>
<td>0.527</td>
<td>0.525</td>
<td>0.002</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>0.618</td>
<td>0.621</td>
<td>0.007</td>
</tr>
</tbody>
</table>

- Cross folds have little effect on results
Table: Mean number of training instances and mean training time for different percentages of training

<table>
<thead>
<tr>
<th>%</th>
<th>Instances</th>
<th>Time (s)</th>
<th>MAP</th>
<th>nDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>2279</td>
<td>298</td>
<td>0.457</td>
<td>0.657</td>
</tr>
<tr>
<td>1.0</td>
<td>3430</td>
<td>321</td>
<td>0.489</td>
<td>0.663</td>
</tr>
<tr>
<td>2.0</td>
<td>5909</td>
<td>550</td>
<td>0.509</td>
<td>0.664</td>
</tr>
<tr>
<td>3.0</td>
<td>8453</td>
<td>1024</td>
<td>0.516</td>
<td>0.667</td>
</tr>
</tbody>
</table>

Increasing number of training instances has minimal effect on effectiveness
Threats to Validity

- Different factors might produce different results
  - e.g., edge weights in data graph
- Marginally relevant results must overlap completely
  - Larger results more likely to be judged marginally relevant
  - Ranking by size may be good at identifying best result but not marginally relevant results
Outline

Questions

Search Techniques

Evaluation Benchmark

Ranking
  Structured Cover Density Ranking
  SVM rank Analysis

Classifying Queries
  Analysis
Intent

User’s objective when performing search

- **Navigational**: locate a particular database entity
  - e.g., Indiana Jones, Sean Connery, *The Parent Trap* 1998

- **Informational**: obtain information from the database
  - e.g., *The Lord of the Rings* Oscars, Tom Hanks 2004

- **Resource**: retrieve a resource from the database
  - e.g., battle *Gettysburg* Devil’s Den images

Similar to previous taxonomies for web search
Expression

How is the user’s intent expressed?

▶ entity
  ▶ e.g., Tom Hanks, Gettysburg

▶ entity-weak entity
  ▶ e.g., The Hunt for Red October quotes

▶ entity-relationship
  ▶ e.g., Star Wars cast

▶ multi-entity
  ▶ e.g., Sean Connery James Bond

⇒ Need model of database for classification
Methodology

- Rank sites in AOL search log by number of click-throughs
  - Determine sites that **might** have database backend
  - Amazon, IMDb, Wikipedia, and lyrics datasets
- Manually classify 250 queries for each site by
  - intent
  - expression (if data model available)
## Lyrics

**Table:** Contingency table of intent and expression.

<table>
<thead>
<tr>
<th>Intent</th>
<th>Navigational</th>
<th>Informational</th>
<th>Resource</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>11.2</td>
<td>3.2</td>
<td>30.0</td>
<td>44.4</td>
</tr>
<tr>
<td>Entity-weak entity</td>
<td>0.4</td>
<td>4.0</td>
<td>51.2</td>
<td>55.6</td>
</tr>
<tr>
<td>Entity-relationship</td>
<td>0.4</td>
<td>4.0</td>
<td>51.2</td>
<td>55.6</td>
</tr>
<tr>
<td>Multi-entity</td>
<td>0.4</td>
<td>4.0</td>
<td>51.2</td>
<td>55.6</td>
</tr>
<tr>
<td>Total</td>
<td>11.6</td>
<td>7.2</td>
<td>81.2</td>
<td>100.0</td>
</tr>
</tbody>
</table>
### IMDb

**Table:** Contingency table of intent and expression.

<table>
<thead>
<tr>
<th></th>
<th>Navigational</th>
<th>Informational</th>
<th>Resource</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>75.2</td>
<td>6.0</td>
<td>4.8</td>
<td>86.0</td>
</tr>
<tr>
<td>Entity-weak entity</td>
<td>0.4</td>
<td>3.2</td>
<td>0.8</td>
<td>4.4</td>
</tr>
<tr>
<td>Entity-relationship</td>
<td>0.4</td>
<td>3.2</td>
<td>0.8</td>
<td>4.4</td>
</tr>
<tr>
<td>Multi-entity</td>
<td>1.6</td>
<td>3.2</td>
<td>1.6</td>
<td>6.4</td>
</tr>
<tr>
<td>Total</td>
<td>77.2</td>
<td>12.4</td>
<td>7.2</td>
<td>96.8</td>
</tr>
</tbody>
</table>
## Intent

**Table:** User intent (% of total) in previous evaluations.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Navigational</th>
<th>Informational</th>
<th>Resource</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>lyrics</td>
<td>2.0</td>
<td>2.0</td>
<td>90.0</td>
<td>94.0</td>
</tr>
<tr>
<td>lyrics</td>
<td>11.6</td>
<td>7.2</td>
<td>81.2</td>
<td>100.0</td>
</tr>
<tr>
<td>IMDb</td>
<td>23.0</td>
<td>77.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>IMDb</td>
<td>25.0</td>
<td>75.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>IMDb</td>
<td>40.0</td>
<td>60.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>IMDb</td>
<td>77.2</td>
<td>12.4</td>
<td>7.2</td>
<td>96.8</td>
</tr>
</tbody>
</table>
## Expression

**Table:** Query expression (% of total) in previous evaluations.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Entity</th>
<th>Entity-weak entity</th>
<th>Entity-relationship</th>
<th>Multi-entity</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>lyrics</td>
<td>34.0</td>
<td></td>
<td>60.0</td>
<td>94.0</td>
<td></td>
</tr>
<tr>
<td>lyrics</td>
<td>44.4</td>
<td></td>
<td>55.6</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>IMDb</td>
<td>14.0</td>
<td>9.0</td>
<td>77.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>IMDb</td>
<td>25.0</td>
<td></td>
<td>75.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>IMDb</td>
<td>40.0</td>
<td>10.0</td>
<td>50.0</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>IMDb</td>
<td>86.0</td>
<td>4.4</td>
<td>6.4</td>
<td>96.8</td>
<td></td>
</tr>
</tbody>
</table>
This slide intentionally blank.