“Deep Web” Databases

“Surface” Web
- Link structure
- Crawlable
- Documents indexed by search engines

“Deep” Web
- No link structure
- Documents “hidden” in databases
- Documents not indexed by search engines
- Need to query each collection individually
U.S. Patent Database: Example of a Deep Web Database

Query [wireless AND network] on USPTO database: **438,007 hits.**
USPTO database is at [http://patft.uspto.gov/netahtml/PTO/search-bool.html](http://patft.uspto.gov/netahtml/PTO/search-bool.html)

Query [site:patft.uspto.gov wireless network] on Google: **1 hit.**

[as of 9/2016]

• Search engines ignore deep web databases (cannot crawl inside), unless handled in case-by-case manner “manually.”
• Autonomous databases typically export no metadata.


Focus: Deep Web (Text) Databases

• Often sources of valuable information
• Often hidden behind search interfaces
• Often non-crawlable by traditional crawlers
Understanding and Interacting With Deep Web Databases

- **Classification** of deep web databases
  - Health > Publications > PubMed

- *(Distributed) search* of deep web databases

---

How to Classify Deep Web Databases *Automatically*

- Task definition
- Classification through query probing
Deep Web Database Classification: Two Possibilities

- **Coverage**-based classification:
  - Database contains *many documents* about category
  - **Coverage**: #docs about this category
- **Specificity**-based classification:
  - Database contains *mainly documents* about category
  - **Specificity**: #docs/|DB|

Text Database Classification: An Example

**Category: Basketball**

- **Coverage-based** classification
  - ESPN.com, NBA.com
- **Specificity-based** classification
  - NBA.com, but **not ESPN.com**
Text Database Classification: More Details

- Define two “editorial” thresholds:
  - $T_c$: coverage threshold (# docs in category)
  - $T_s$: specificity threshold (fraction docs in category)

- Assign a text database to a category $C$ if:
  - Database coverage for $C$ at least $T_c$
  - Database specificity for $C$ at least $T_s$

Brute-Force Database Classification “Strategy”

1. Extract all documents from database.
2. Classify documents.
3. Classify database accordingly.

**Problem:** No access to full contents of deep web databases!

**Solution:** Exploit database search interface to approximate document classification
Search-based Deep Web Database Classification

1. Train a rule-based document classifier.
2. Transform classifier rules into queries.
3. Adaptively issue queries to databases.
4. Categorize the databases based on adjusted number of query matches.

Training a Document Classifier

- **Feature Selection**: Zipf’s law pruning, followed by information theoretic feature selection
- **Classifier Learning**: RIPPER
  - Input: A set of pre-classified, labeled documents
  - Output: A set of classification rules
    - IF linux THEN **Computers**
    - IF jordan AND bulls THEN **Sports**
    - IF heart AND pressure THEN **Health**
Designing and Implementing Query Probes

• Transform each document classifier rule into query:
  \[
  \text{IF jordan AND bulls THEN Sports} \rightarrow [\text{jordan AND bulls}]
  \]
• Issue each query to database to obtain number of matches without retrieving any documents

Using Probe Results for Classification

We use probe results to estimate coverage and specificity

<table>
<thead>
<tr>
<th></th>
<th>ACM</th>
<th>NBA</th>
<th>PubMed</th>
<th></th>
<th>ACM</th>
<th>NBA</th>
<th>PubMed</th>
</tr>
</thead>
<tbody>
<tr>
<td>COV</td>
<td></td>
<td></td>
<td></td>
<td>SPEC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>comp</td>
<td>336</td>
<td>0</td>
<td>16</td>
<td>comp</td>
<td>0.95</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sports</td>
<td>0</td>
<td>6674</td>
<td>0</td>
<td>sports</td>
<td>0</td>
<td>0.985</td>
<td>0</td>
</tr>
<tr>
<td>health</td>
<td>18</td>
<td>103</td>
<td>81164</td>
<td>health</td>
<td>0.05</td>
<td>0.015</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Hierarchically Classifying the ACM DigLib ($T_c=100$, $T_s=0.5$)

Adjusting Query Results

- Search-based estimates of category distribution not perfect:
  - Queries for one category match documents from other categories
  - Queries might overlap
- Document classifiers not perfect:
  - Queries do not match all documents in a category

Can adjust estimates automatically based on the distribution of errors during training
(If interested, see discussion on “confusion matrix” adjustment in paper)
Experimental Results: Web Databases

- **F-measure above 0.7** for best $<Tc, Ts>$ combination found.
- **185 query probes per database on average** needed for choice of thresholds.
- Also, **probes are short**: 1.5 words on average; 4 words maximum.

Query-based Database Classification

- Easy classification using just a few queries
- No need for document retrieval
  - Only need to identify a line like: “82 matches found”
  - “Wrapper” needed is trivial
- Not limited to deep web databases, but useful over any text database with a search interface: query-based approach sometimes orders of magnitude more efficient than crawling
Understanding and Interacting With Deep Web Databases

- **Classification** of deep web databases
  Health > Publications > PubMed

- **(Distributed) search** of deep web databases
  - Key step to process a query: database selection (i.e., deciding where it’s worth evaluating the query and where it’s not)

Database Selection Step Needs Database “Content Summaries”

Typically the vocabulary of each database plus simple frequency statistics:

<table>
<thead>
<tr>
<th>Database</th>
<th>Content Summaries</th>
</tr>
</thead>
<tbody>
<tr>
<td>PubMed</td>
<td>(3,868,552 documents)</td>
</tr>
<tr>
<td></td>
<td>cancer</td>
</tr>
<tr>
<td></td>
<td>aids</td>
</tr>
<tr>
<td></td>
<td>heart</td>
</tr>
<tr>
<td></td>
<td>hepatitis</td>
</tr>
<tr>
<td></td>
<td>thrombopenia</td>
</tr>
<tr>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>
Database Selection for Distributed Search over Deep Web Databases

**Database selection** relies on simple **content summaries**:
- vocabulary, word frequencies

**Problem:** **Databases don’t export content summaries!**

**Observation:** Content summaries can be approximated from a **small document sample** extracted during classification.

- thrombopenia 24,348
- thrombopenia 0
- thrombopenia 18

---

**Sampling based on “Focused” Query Probing**

1. Automatically associate queries with categories
2. For each query:
   - Send to database
   - Record number of matches
   - Retrieve top-k matching documents
3. At end of round:
   - Analyze matches for each category
   - Choose category to focus on

**Output:**
- Representative document sample
- Actual frequencies for some “important” words
Adjusting Document Frequencies

- We know ranking \( r \) of words according to document frequency in sample
- We know absolute document frequency \( f \) of some words from one-word queries
- Mandelbrot’s formula connects empirically word frequency \( f \) and ranking \( r \)
- We use curve-fitting to estimate the absolute frequency of all words in sample

Actual PubMed Content Summary

**PubMed** (3,868,552 documents)
Categories: Health, Diseases

- Extracted automatically
- ~27,500 words in extracted content summary
- Fewer than 200 queries sent
- At most 4 documents retrieved per query

(heart, hepatitis, basketball not in 1-word probes)
Database Selection and Extracted Content Summaries

- Database selection algorithms assume complete content summaries
- Content summaries extracted by (small-scale) sampling are inherently incomplete (Zipf's law)
- Queries with undiscovered words are problematic

**Database Classification Helps:**

Similar topics ↔ Similar content summaries
Extracted content summaries complement each other

Content Summaries within Category Complement Each Other

- Cancerlit contains "thrombopenia", not found during sampling
- PubMed contains "chemotherapy", not found during sampling
- Health category content summary contains both

Database selection can proceed hierarchically: summaries of "sibling" databases help compensate for incomplete summaries
Hierarchical DB Selection: Example

To select D databases:
1. Use “standard” DB selection algorithm to score categories
2. Proceed to category with highest score
3. Repeat until category is a leaf, or category has fewer than D databases

Hierarchical Deep Web Database Sampling and Selection

- We extract content summaries efficiently from “uncooperative” deep web databases
- We estimate absolute word frequencies
- We improve effectiveness of hierarchical database selection by exploiting database classification