Querying Web Data: The WebQA Approach

Sunny K.S. Lam and M. Tamer Özsu

Presented by E. Cem Sözgen
Outline

- Introduction
- Background and Literature
- WebQA Architecture
- Query Parser
- Summary Retriever
- Answer Extractor
- Evaluation
- References
- Comments
Outline

- Introduction
- Background and Literature
- WebQA Architecture
- Query Parser
- Summary Retriever
- Answer Extractor
- Evaluation
- References
- Comments
What do people want from a web query system?

• The ideal system for querying the web:
  (from the author’s point of view)
  • Accepts easy to pose query (possibly in natural language)
  • Searches all of the sources
  • Returns direct answers (not links)
• How about WebQA?
WebQA

- Factual query expressed in natural language
- Ranked list of short answers

e.g. Who invented the telephone?
1) Alexander Graham Bell (58.0)
2) Graham Bell (58.0)
3) Bell (58.0)
4) Alexander Graham (54.0)
Type of questions that WebQA do not deal?

- Who are the players of Toronto Raptors? (multiple results)
- Notify me whenever the temperature of Waterloo drops below zero. (continuous query)
- How do I make pancakes? (procedural query)
Which areas are involved?

- Question answering (QA) techniques
- Metasearch techniques
- Mediator/Wrapper techniques
- Information Retrieval (IR) techniques
- Extraction techniques
Outline

• Introduction
• Background and Literature
• WebQA Architecture
• Query Parser
• Summary Retriever
• Answer Extractor
• Evaluation
• References
• Comments
• Keyword Search approach
  • Search engines
  • Metasearchers
• Category Search approach
• Database view approach
• Semi-structured data querying approach
• Web Query Language approach
• Learning based approach
• Question answering approach
Mulder

- Very similar to WebQA
- Accepts short factual questions in NL
- Returns exact answers
- Similar main components
- Question types:
  - Nominal: Noun phrase
  - Numerical: Number
  - Temporal: Date
- Uses Google as a search engine
## Differences

<table>
<thead>
<tr>
<th>WebQA</th>
<th>Mulder</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Light NLP</td>
<td>• Heavy NLP</td>
</tr>
<tr>
<td>• 7 categories</td>
<td>• 3 categories</td>
</tr>
<tr>
<td>• Multiple sources</td>
<td>• Single search engine</td>
</tr>
<tr>
<td>• More fault tolerant</td>
<td></td>
</tr>
<tr>
<td>• More flexible and scalable</td>
<td></td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Background and Literature
• WebQA Architecture
• Query Parser
• Summary Retriever
• Answer Extractor
• Evaluation
• References
• Comments
Client-Server Architecture

Web Server

QA Server

Client #1

Client #2

... Client #N

Question / Answer (HTTP)

JSPs, HTMLs

QA Server Thread

QA Server Thread

QA Server Thread

QA Engine

Question / Answer (String)

Q / A
Interface

• Two types of interface
  • Textual Interface
    • Local access
    • Fast and provides debugging information
    • Need a copy of WebQA in local machine
  • Graphical User Interface
# WebQA: A Fast and Simple QA System

**About WebQA**
- What is WebQA?
- Features
- Result

**Categories**
- Name
- Place
- Time
- Quantity
- Abbreviation
- Weather
- Other

**Sample Questions**
- Name
- Place
- Time
- Quantity
- Abbreviation
- Weather
- Other

**WebQAL (Language for WebQA)**
- Syntax
- Examples

**Question Submission**
- Questions with good result
- Questions with bad result

**Help**
- Frequently Asked Questions
- Technical Support

**Contact**
- Sunny Lam: s6lam@uwaterloo.ca
- M. Tamer Ozsu

---

*Home page of WebQA*
Outline

• Introduction
• Background and Literature
• WebQA Architecture
• Query Parser
• Summary Retriever
• Answer Extractor
• Evaluation
• References
• Comments
Categories

• Defined to improve system accuracy

• Name
• Place
• Time
• Quantity
• Abbreviation
• Weather
• Other

• Who invented the telephone? (Name)
• Who was George Washington? (Other)
function categorize(String nlQuestion) : Category
    boolean hasWhat \equiv false
    for each word X in nlQuestion do
        if X = “what” or “which” then
            hasWhat \equiv true
        else if X = “where” then
            return “Place”
        else if X = “when” then
            return “Time”
        else if X = “how” then
            if the word after X is in howQuantityTermList then
                return “Quantity”
            end if
        else if X = “who” or “whom” then
            newQuestion \equiv nlQuestion without stopwords
            if every word in newQuestion starts with a upper letter then
                return “Other”
            else
                return “Name”
            end if
        end if
    end if
end for
return “Other”
Output Options

<Category>
[-output
.setOutput Option] 
-keywords
</Keyword List>
Outline

- Introduction
- Background and Literature
- WebQA Architecture
- Query Parser
- Summary Retriever
- Answer Extractor
- Evaluation
- References
- Comments
### List used by Source Ranker

<table>
<thead>
<tr>
<th>Category</th>
<th>Rank</th>
<th>Sources</th>
<th>Number of Records Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>1</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td>Place</td>
<td>1</td>
<td>World Factbook</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Excite</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Overture</td>
<td>50</td>
</tr>
<tr>
<td>Time</td>
<td>1</td>
<td>World Factbook</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>All The Web</td>
<td>50</td>
</tr>
<tr>
<td>Quantity</td>
<td>1</td>
<td>World Factbook</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>1</td>
<td>World Factbook</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>All The Web</td>
<td>50</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>World Factbook</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Yahoo</td>
<td>50</td>
</tr>
</tbody>
</table>

### The structure of a record

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Name</td>
<td>String</td>
</tr>
<tr>
<td>Snippet</td>
<td>String</td>
</tr>
<tr>
<td>Local Rank</td>
<td>Integer</td>
</tr>
</tbody>
</table>
Mediator/Wrapper

- For information integration
- One wrapper for each data source
- Same Wrapper API
- One centralized mediator
- Different from data warehouse: integrated data is not materialized
Outline

- Introduction
- Background and Literature
- WebQA Architecture
- Query Parser
- Summary Retriever
- Answer Extractor
- Evaluation
- References
- Comments
Answer Extraction

List of Ranked Record

Candidate Identifier

List of Candidates

Best Candidate Retriever

List of Best Candidates

Rearranger

List of Rearranged Candidates

Output Converter

Best Answers
Candidate Identifier

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
</tr>
<tr>
<td>Score</td>
<td>double</td>
</tr>
</tbody>
</table>

Structure of a Candidate

- **Candidate list:** list of candidates
- **Four sub-identifiers**
  - Country sub-identifier
  - Abbreviation sub-identifier
  - Weather sub-identifier
  - Search engine sub-identifier
Rearranger

1) Bell (58.0)
2) Alexander Graham Bell (50.0)

↓

1) Alexander Graham Bell (58.0)
2) Bell (58.0)
Outline

- Introduction
- Background and Literature
- WebQA Architecture
- Query Parser
- Summary Retriever
- Answer Extractor
- Evaluation
- References
- Comments
Experiment 1

• To see the performance of categorizing questions
• TREC 9: 686/693 -> 98.99%
• TREC 10: 461/500 -> 92.2%
Experiment 2

- **To determine the best source ranking for each category**

<table>
<thead>
<tr>
<th>Category</th>
<th>Rank</th>
<th>Sources</th>
<th>Number of Records Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>1</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td>Place</td>
<td>1</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Excite</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Overture</td>
<td>50</td>
</tr>
<tr>
<td>Time</td>
<td>1</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>All The Web</td>
<td>50</td>
</tr>
<tr>
<td>Quantity</td>
<td>1</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>1</td>
<td>Yahoo</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>All The Web</td>
<td>50</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>Yahoo</td>
<td>50</td>
</tr>
</tbody>
</table>
Experiment 3

- To see how using secondary sources affects the results

<table>
<thead>
<tr>
<th></th>
<th>With Secondary Source</th>
<th>Without Secondary Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREC-9 Score</td>
<td>0.35</td>
<td>0.359</td>
</tr>
<tr>
<td>Response Time</td>
<td>2981</td>
<td>2991</td>
</tr>
</tbody>
</table>

![Bar chart showing comparison between With and Without Secondary Source]
Experiment 4
- Comparison of WebQA with other systems
Outline

- Introduction
- Background and Literature
- WebQA Architecture
- Query Parser
- Summary Retriever
- Answer Extractor
- Evaluation
- References
- Comments
References


Comments...
The followings shows a verb-to-noun conversion table.

<table>
<thead>
<tr>
<th>Verb</th>
<th>Noun</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>creator</td>
</tr>
<tr>
<td>created</td>
<td>creator</td>
</tr>
<tr>
<td>creates</td>
<td>creator</td>
</tr>
<tr>
<td>invent</td>
<td>inventor</td>
</tr>
<tr>
<td>invented</td>
<td>inventor</td>
</tr>
<tr>
<td>invents</td>
<td>inventor</td>
</tr>
<tr>
<td>locate</td>
<td>location</td>
</tr>
<tr>
<td>located</td>
<td>location</td>
</tr>
<tr>
<td>locates</td>
<td>location</td>
</tr>
<tr>
<td>own</td>
<td>owner</td>
</tr>
<tr>
<td>owned</td>
<td>owner</td>
</tr>
<tr>
<td>owns</td>
<td>owner</td>
</tr>
<tr>
<td>sang</td>
<td>singer</td>
</tr>
<tr>
<td>skate</td>
<td>skater</td>
</tr>
<tr>
<td>skated</td>
<td>skater</td>
</tr>
<tr>
<td>skates</td>
<td>skater</td>
</tr>
<tr>
<td>sing</td>
<td>singer</td>
</tr>
<tr>
<td>sings</td>
<td>singer</td>
</tr>
<tr>
<td>write</td>
<td>author</td>
</tr>
<tr>
<td>writes</td>
<td>author</td>
</tr>
<tr>
<td>wrote</td>
<td>author</td>
</tr>
</tbody>
</table>