Efficient Information Extraction over Evolving Text Data

Fei Chen¹, AnHai Doan¹, Jun Yang², Raghu Ramakrishnan³

¹University of Wisconsin-Madison
²Duke University
³Yahoo! Research
Information Extraction (IE)

- Many solutions in database/Web/AI communities with significant progress

- But most solutions have considered only static text corpora

<table>
<thead>
<tr>
<th>Group Meeting Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 21: We’ll discuss CIM and IR in room <strong>CS 310</strong> at 4pm.</td>
</tr>
<tr>
<td>Jun 14: Meet in <strong>CS 105</strong> at 2pm.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Meetings</th>
</tr>
</thead>
<tbody>
<tr>
<td>room</td>
</tr>
<tr>
<td>CS 310</td>
</tr>
<tr>
<td>CS 105</td>
</tr>
</tbody>
</table>
IE over Evolving Text Data

Group Meeting Schedule

Jun 14: Meet in CS 105 at 2pm.

Jun 21: We’ll discuss CIM and IR in room CS 310 at 4pm.

Jun 28: No meeting this week.

Jun 14: Meet in CS 105 at 2pm.
Current Approach and Its Limitations

- Apply IE to each corpus snapshot in isolation, from scratch

- Limitations:
  - Inefficient: e.g., IE in DBLife
  - Unsuitable for time-sensitive applications: e.g., stock, auction
  - Unsuitable for interactive debugging over dynamic text corpora
Cyclex: Recycling Extraction

Group Meeting Schedule
Jun 14: Meet in CS 105 at 2pm.

Group Meeting Schedule
Jun 21: We’ll discuss CIM and IR in room CS 310 at 4pm.
Jun 14: Meet in CS 105 at 2pm.

Meetings_1

<table>
<thead>
<tr>
<th>room</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 105</td>
<td>2pm</td>
</tr>
</tbody>
</table>

Meetings_2

<table>
<thead>
<tr>
<th>room</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 105</td>
<td>2pm</td>
</tr>
<tr>
<td>CS 310</td>
<td>4pm</td>
</tr>
</tbody>
</table>
Challenges and Contributions

- **How to guarantee correctness?**
  - Model extractors using scope and context

- **How to choose a good way to match pages?**
  - Cost-based decisions using text specific cost model

- **How to efficiently execute the chosen plan given a large amount of disk-resident data?**
  - A way to scan data once
Why Guaranteeing Correctness Is Hard?

E extracts meetings only if a page has fewer than 4 lines
Extractor Properties: Scope

- Attribute mentions of an entity often appear in close proximity in data pages.

- An extractor \( E \) has scope \( \alpha \) iff any mention produced by \( E \) at most spans \( \alpha \) characters.

Example: \( E \) with scope \( \alpha = 50 \)

<table>
<thead>
<tr>
<th>Group Meeting Schedule</th>
<th>Meetings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 21: We’ll discuss CIM and IR in room CS 310 at 4pm.</td>
<td>CS 310</td>
</tr>
<tr>
<td>Jun 14: Meet in CS 105 at 2pm.</td>
<td>CS 105</td>
</tr>
</tbody>
</table>

room mentions | time mentions

< 50 characters
Extractor Properties: Context

- Many extractors only examine small “context windows” on both sides of a mention to extract the mention.

Example: E with context $\beta = 15$

<table>
<thead>
<tr>
<th>Project Meeting Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 21: We’ll discuss CIM and IR in room CS 310 at 4pm.</td>
</tr>
<tr>
<td>Jun 14: Meet in CS 105 at 2pm.</td>
</tr>
</tbody>
</table>

- The text outside the context of a mention $m$ is irrelevant for $E$ to extract $m$. 

<table>
<thead>
<tr>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>topic</td>
</tr>
<tr>
<td>CIM</td>
</tr>
<tr>
<td>IR</td>
</tr>
</tbody>
</table>

$< 15$ characters
Problem Definition

P1 \rightarrow E, \alpha, \beta
M1

P2 \rightarrow E, \alpha, \beta
M2

P3 \rightarrow E, \alpha, \beta
M3

P4 \rightarrow E, \alpha, \beta
M4

?
Match Pages To Find Overlapping Regions

- Consider 3 matchers (more can be added)
  - DN (Doing Nothing): immediately declares no overlapping regions are found
    - 0 runtime and no overlapping regions
  - UD (Unix Diff): a Unix-diff-command like algorithm
    - relatively fast runtime and some overlapping regions
  - ST (Suffix Tree): a novel suffix-tree based algorithm we developed
    - linear in the length of pages runtime and all overlapping regions

Matchers trade off runtime with result completeness

(See paper for more details)
Choose the Optimal Matcher

- Consider a plan space where plans differ in the matchers they use

- Use a cost model to estimate the completion time of each plan

- Text-specific cost model
  - e.g., change rate of the text corpus, cost of the extractor, size of matching results and IE results, etc.

- Collect statistics over past k snapshots
Challenge in Efficiently Executing the Chosen Plan

$P_1 \rightarrow E, \alpha, \beta$

$M_1$

$P_2 \rightarrow E, \alpha, \beta$

$M_2$

$P_3 \rightarrow E, \alpha, \beta$

$M_3$

$P_4 \rightarrow E, \alpha, \beta$

Matcher $M$

?
Interleave Matching, Extraction and Copy
Architecture

Last k snapshots and mentions extracted from these snapshots

Previous snapshot $P_3$, Current snapshot $P_4$
Experiment Setup

- **Datasets**

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>DBLife</th>
<th>Wikipedia</th>
</tr>
</thead>
<tbody>
<tr>
<td># Data Sources</td>
<td>980</td>
<td>925</td>
</tr>
<tr>
<td># Snapshots</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>Time between snapshots</td>
<td>1 day</td>
<td>21 days</td>
</tr>
<tr>
<td>Avg # Page per Snapshot</td>
<td>10155</td>
<td>3038</td>
</tr>
<tr>
<td>Avg Size per Snapshot</td>
<td>180M</td>
<td>35M</td>
</tr>
</tbody>
</table>

- **Extractors**

<table>
<thead>
<tr>
<th></th>
<th>DBLife</th>
<th>Wikipedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>researcher</td>
<td></td>
<td></td>
</tr>
<tr>
<td>affiliation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>talk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>actor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>play</td>
<td></td>
<td></td>
</tr>
<tr>
<td>award</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Scope α   | 32      | 93       | 400    | 35     | 96     | 250    |
| Context β | 3       | 7        | 10     | 3      | 4      | 10     |
Benefit of Recycling IE Results

- In all cases except “actor”, Cyclex drastically cut runtime of re-extraction from scratch by 50-90%
Importance of Optimization

None of the matchers is uniformly optimal.
(See paper for more details)
Conclusion and Future Work

- Proposed the first approach to speed up IE over evolving text data by recycling past IE results

- Defined challenges and provided initial solutions
  - Model properties of extractors
  - Cost-based decisions in choosing an optimal matcher
  - Efficiently interleave matching, extraction, and copying

- Future work
  - Handle multiple extractors
  - Handle extractors that extract mentions across multiple pages
Related Work

● **Much work on IE**
  – Improve accuracy and efficiency
  – Recent work on scalable IE [tutorial in KDD06, SIGMOD06]

● **Evolving text data**
  – Repair wrappers as page templates change [McCann VLDB05]
  – Incrementally update an inverted index [Lim WWW03]

● **Exploiting overlapping text data in a document collection to compress indices** [Herscovici ECIR07, Zhang WWW07]

● **Optimizing IE programs and developing text-centric cost models** [Ipeirotis SIGMOD06, Jain ICDE07, Shen VLDB07]