Querying Probabilistic Information Extraction

Daisy Zhe Wang, Michael J. Franklin, Minos Garofalakis, Joseph M. Hellerstein

VLDB, Singapore, 16th September, 2010
Outline

• **Information Extraction Systems**
  • Information Extraction (IE)
  • “Extract-then-Query” – *Standard IE System*
  • “Query-Time-Extraction” – *BayesStore IE System*

• Primer on CRF

• **Query-Driven Extraction**
  • Select-over-Top1 Queries

• **Probabilistic SPJ Queries**
  • Probabilistic Join Queries

• **Experimental Results**

• **Conclusion**
Steve Jobs introduced the iPhone 4's videoconferencing feature FaceTime at WWDC 2010. Apple will hold a press conference Wednesday, where Steve Jobs is expected to announce the birth of new stars in his product galaxy, including (probably) new iPods and (possibly) a successor to Apple TV.

--- From WIRED August 30, 2010
Information Extraction (IE)

- Steve Jobs introduced the iPhone 4's videoconferencing feature FaceTime at WWDC 2010. Apple will hold a press conference Wednesday, where Steve Jobs is expected to announce the birth of new stars in his product galaxy, including (probably) new iPods and (possibly) a successor to Apple TV.

--- From WIRED August 30, 2010

Labels:
Person  Company  Product  Event  Other
“Extract-then-Query” – Standard IE Systems

Problems:
1. Exhaustive extraction for all entities over all in-coming documents
2. Loses uncertainties and probabilities which are inherent in IE
Exhaustive vs.
Query-Driven Extraction Example

Example Query:
SELECT persons FROM blog articles
WHERE company = "Apple"

• Steve Jobs introduced the iPhone 4's videoconferencing feature FaceTime at WWDC 2010. Apple will hold a press conference...

• The Big Apple lands '14 Super Bowl. Giants co-owner Jonathan Tisch said: “The greatest game will be played on the greatest stage!”...

• Apple Soufflé recipe by Julia Child: ... Pare, cut up, and stew ...
Exhaustive vs. Query-Driven Extraction Example

Example Query:
SELECT persons FROM blog articles
WHERE company = “Apple”

- **Steve Jobs** introduced the **iPhone** 4’s videoconferencing feature FaceTime at **WWDC 2010**. **Apple** will hold a press conference...
- The **Big Apple** lands ’14 **Super Bowl**. Giants co-owner **Jonathan Tisch** said: “The greatest game will be played on the greatest stage!”...
- **Apple Soufflé** recipe by **Julia Child**: ... Pare, cut up, and stew ...
Exhaustive vs. Query-Driven Extraction Example

Example Query:
SELECT persons FROM blog articles
WHERE company = "Apple"

- **Steve Jobs** introduced the **iPhone** 4's videoconferencing feature **FaceTime** at **WWDC 2010**. **Apple** will hold a press conference...
- The **Big Apple** lands  
- **Apple Soufflé** recipes

How to perform fast filtering without full inference? Challenge: Need to push condition **Label = ‘company’** into inference by deep integration of inference and relational ops.
“Extract-then-Query” – Storing Extractions and Probabilities

Still performs exhaustive extraction
Does not have the right representations to support IE probabilistic models inside of PDB [Gupta, VLDB2005]
“Query-Time-Extraction” – BayesStoreIE

Our Contributions:
• Deep Integration between Inference and Relational Operators
• Enable Query-Driven On-line Extraction
• Enable Probabilistic Queries over IE models
Outline

• Information Extraction Systems
  • Information Extraction (IE)
  • “Extract-then-Query” – *Standard* IE Approach
  • “Query-Time-Extraction” – *BayesStore IE Approach*

• Primer on CRF

• Query-Driven Extraction
  • Select-over-Top1 Queries

• Probabilistic SPJ Queries
  • Probabilistic Join Queries

• Experimental Results

• Conclusion
Conditional Random Fields (CRF)

Text (address string):
E.g., “2181 Shattuck North Berkeley CA USA”

CRF Model:

Possible Extraction Worlds:

<table>
<thead>
<tr>
<th>x</th>
<th>2181</th>
<th>Shattuck</th>
<th>North</th>
<th>Berkeley</th>
<th>CA</th>
<th>USA</th>
</tr>
</thead>
<tbody>
<tr>
<td>y1</td>
<td>apt. num</td>
<td>street name</td>
<td>city</td>
<td>city</td>
<td>state</td>
<td>country</td>
</tr>
<tr>
<td>y2</td>
<td>apt. num</td>
<td>street name</td>
<td>street name</td>
<td>city</td>
<td>state</td>
<td>country</td>
</tr>
</tbody>
</table>

(0.6)  

(0.1)
Two Query Families

Query Family 1: (SPJ-over-Top1)
Queries using only most-likely Extractions

Query Family 2: (Probabilistic SPJ)
Queries using probabilistic distributions
Query Family 1: Select-over-Top1

Example Query:
Select *
From Top-1 extractions of document set D
Where company like “%Apple%”
Viterbi Top-1 Inference on CRF

Viterbi Dynamic Programming Algorithm:

\[
V(i, y) = \begin{cases} 
\max_{y'}(V(i - 1, y')) & \text{if } i \geq 0 \\
0, & \text{if } i = -1.
\end{cases}
\]  

CRF Model:

Dynamic Programming V matrix:

<table>
<thead>
<tr>
<th>pos</th>
<th>street num</th>
<th>street name</th>
<th>city</th>
<th>state</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>15</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>24</td>
<td>21</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>32</td>
<td>32</td>
<td>30</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>40</td>
<td>38</td>
<td>42</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>47</td>
<td>46</td>
<td>46</td>
<td>50</td>
</tr>
</tbody>
</table>
Query Family 1: Select-over-Top1 – Viterbi Early-Stopping Algorithm

Example Query:
Select *
From Viterbi-Top1 extractions of document set D
Where company like "%Apple%"

<table>
<thead>
<tr>
<th>pos</th>
<th>event</th>
<th>city</th>
<th>company</th>
<th>state</th>
<th>other</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>15</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>24</td>
<td>21</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>24</td>
<td>21</td>
<td>18</td>
<td>17</td>
</tr>
</tbody>
</table>

Implemented in PostgreSQL using recursive queries and array functions
Example Query:
Select Top-1 results
From extraction distributions of documents in $D1, D2$
Where $D1.city = D2.city$
Query Family 2: Probabilistic Join

Example Query:
Select Top-1 results
From extraction distributions of documents in D1, D2
Where D1.city = D2.city

Naïve algorithm:
First compute top-k extractions for both input document sets, then compute join

Problem:
k needed to compute Top-1 results varies for different documents

Solution:
Probabilistic Rank-Join algorithm based on Incremental Ranked Access to the List of Possible Extractions
A novel variation of the Top-1 Viterbi algorithm, which computes the next highest-probability extraction \textit{incrementally} and \textit{more efficiently}.

<table>
<thead>
<tr>
<th></th>
<th>pos</th>
<th>street num</th>
<th>street name</th>
<th>city</th>
<th>state</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sacramento Avenue</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>San Francisco CA USA</td>
<td>1</td>
<td>2</td>
<td>15</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>12</td>
<td>24</td>
<td>21</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>21</td>
<td>32,</td>
<td>30</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>29</td>
<td>40</td>
<td>38</td>
<td>42</td>
<td>35</td>
</tr>
<tr>
<td>USA</td>
<td>5</td>
<td>39</td>
<td>47</td>
<td>46</td>
<td>46</td>
<td>50</td>
</tr>
</tbody>
</table>
A novel variation of the Top-1 Viterbi algorithm, which computes the next highest-probability extraction incrementally and more efficiently.

<table>
<thead>
<tr>
<th>pos</th>
<th>street num</th>
<th>street name</th>
<th>city</th>
<th>state</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>15,10</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>24,18</td>
<td>21</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>32</td>
<td>32,31</td>
<td>30</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>40</td>
<td>38</td>
<td>42,38</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>39</td>
<td>47</td>
<td>46</td>
<td>46</td>
<td>50,48</td>
</tr>
</tbody>
</table>

3rd highest-probability extraction can be computed by another call…
Probabilistic Rank-Join

Rank-join is applied to each pair of “joinable” document to compute Top-1 join results

<table>
<thead>
<tr>
<th>key</th>
<th>Ext.</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>.83</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>.12</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>.02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>key</th>
<th>Ext.</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td>.77</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>.15</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>.03</td>
</tr>
</tbody>
</table>

Outer Doc_i

Inner Doc_j
Probabilistic Rank-Join

A set of rank-joins are computed *simultaneously* for a set of outer documents and a set of inner documents.

<table>
<thead>
<tr>
<th>key</th>
<th>Ext.</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>.83</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>.12</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>.02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>key</th>
<th>Ext.</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
<td>.77</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>.15</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>.03</td>
</tr>
</tbody>
</table>

......

Outer Doc_1

<table>
<thead>
<tr>
<th>key</th>
<th>Ext.</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
<td>.95</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>.02</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>.01</td>
</tr>
</tbody>
</table>

......

Inner Doc_1

Inner Doc_n
Other Algorithms

- Probabilistic Selection
- Probabilistic Projection
- Query-Driven Join-over-Top1
Outline

• Information Extraction Systems
  • Information Extraction (IE)
  • “Extract-then-Query” — Standard IE Approach
  • “Query-Time-Extraction” — BayesStore IE Approach
• Primer on CRF
• Query-Driven Extraction
  • Select-over-Top1 Queries
• Probabilistic SPJ Queries
  • Probabilistic Join Queries
• Experimental Results
• Conclusion
Evaluation 1: [Efficiency Improvement] Exhaustive vs. Query-Driven Extraction with Inverted Index

Select-over-Top1 Queries
Evaluation 2: [Efficiency Improvement]
Query-Driven Extraction
Inverted Index vs. Early-Stopping

Take-away: Query-Driven Extraction improves Efficiency.
Evaluation 3: [Accuracy Improvement]
Probabilistic Join vs. Join-over-Top1

Take-away: Probabilistic SPJ improves accuracy at a computation cost
A Query Design Space: efficiency vs. accuracy
Conclusion

- Querying Probabilistic IE
  - BayesStoreIE framework
  - Deep Integration of Relational and Inference
  - Query-Driven Extraction
  - Probabilistic SPJ Queries

- Current & Future Work
  - MCMC inference in DB
  - Conditional and Aggregation Queries in IE
  - Optimizer for Inference Operators (cost-accuracy co-optimization)
Thank you! ... Questions?

BayesStore Project Page:
http://www.cs.berkeley.edu/~daisyw/BayesStore.html