Semantic annotation of unstructured and ungrammatical text

Matthew Michelson & Craig A. Knoblock

University of Southern California & Information Sciences Institute
User Entered Text (on the web)
Prevalent source of info on the web

- Craig’s list
- Ebay
- Bidding for Travel
- Internet Classifieds
- Bulletin Boards / Forums
- …
We want agents that search the Semantic Web
To search this data too!

Semantic Annotation

What we need …

Information Extraction! (label extracted pieces)

How to do it …
Information Extraction (IE)

What is IE on user entered text?

Example:

“1988 Honda Accord for sale! Only 80k miles, Runs Like New, V6, 2WD... $2,500 obo. SUPER DEAL.”
Information Extraction (IE)

- IE on user entered text is hard!
  - Unstructured
    - Can’t use Wrappers
  - Ungrammatical
    - Can’t use lexical information, such as Part of Speech Tagging or other NLP
  - Can’t rely on characteristics
    - Misspellings and errant capitalization
Information Extraction (IE)

Our 2 step solution:

1. Find match in Reference Set
2. Use match for extraction
REFERENCE SETS

Collection of known entities and their common attributes

Set of Reference Documents: CIA World Fact Book
Country, Economy, Government, etc.

Online database: Comics Price Guide
Title, Issue, Price, Description, etc.

Offline database: ZIP+4 database from USPS (street addresses)
Street Name, Street Number Range, City, etc.

Semantic Web: ONTOLOGIES!
REFERENCE SETS

Our Example: **CAR ONTOLOGY**

Attributes: Car Make, Car Model

<table>
<thead>
<tr>
<th>Car Make</th>
<th>Car Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
</tr>
<tr>
<td>Honda</td>
<td>Civic</td>
</tr>
<tr>
<td>Acura</td>
<td>Integra</td>
</tr>
<tr>
<td>Hyundai</td>
<td>Tiburon</td>
</tr>
</tbody>
</table>
Information Extraction (IE)

Our 2 step solution:

1. Find match in Reference Set (ONTOLOGIES)
2. Use match for extraction (LABEL FOR ANNOTATION)
Information Extraction (IE)

Our 2 step solution:

1. Find match in Reference Set (ONTOLOGIES)
2. Use match for extraction (LABEL FOR ANNOTATION)
Step 1: Find Ontology Match

“Record Linkage” (RL)

Algorithm:
1. Generate candidate matching tuples
2. Generate vector of scores for each candidate
3. Do binary rescoring for all vectors
4. Send rescored vectors to SVM to classify match
1: Generate candidate matches

“Blocking”

Reduce number of possible matches

Many proposed methods in RL community

Choice independent of our algorithm

Example:

<table>
<thead>
<tr>
<th>Car Make</th>
<th>Car Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
</tr>
<tr>
<td>Honda</td>
<td>Civic</td>
</tr>
</tbody>
</table>
2: Generate vector of scores

Vector of scores:

Text versus each attribute of the reference set

   Field level similarity

Text versus concatenation of all attributes of reference set

   Record Level Similarity

Example:

“1988 Honda Accrd for sale! Only 80k miles, Runs Like New, V6, 2WD... $2,500 obo. SUPER DEAL.” \( \rightarrow \) text

Candidate:  

| Honda | Accord |

Vector = \{ \text{Scores}(\text{text, Honda}) \cup \text{Scores}(\text{text, Accord}) \cup \text{Scores}(\text{text, Honda Accord}) \}
2: Generate vector of scores

Vector = \{ Scores(text, Honda) \cup Scores(text, Accord) \cup Scores(text, Honda Accord) \}

\{ Token(text, Honda) \cup Edit_Dist(text, Honda) \cup Other(text, Honda) \}

\{ Jensen-Shannon(text, Honda) \cup Jaccard-Sim(text, Honda) \}

\{ Smith-Waterman(text, Honda) \cup Levenstein(text, Honda) \cup Jaro-Winkler(text, Honda) \cup Jaccard-Character(text, Honda) \}

\{ Soundex(text, Honda) \cup Porter-Stemmer(text, Honda) \}
2: Generate vector of scores

Why use each attribute AND concatenation?

Possible for different records in ontology to have the same record level score, but different scores for the attributes. If one has higher score on a more discriminative attribute, we capture that.
3: Binary rescoring of vectors

Binary Rescoring –
If Max: score $\rightarrow 1$
Else: score $\rightarrow 0$
(All indices that have that max value for that score get a 1)

Example, 2 vectors:

Score(P,r1) = \{0.1, 2.0, 0.333, 36.0, 0.0, 8.0, 0.333, 48.0\}
BScore(P,r1) = \{1, 1, 1, 1, 1, 1, 1, 1\}

Score(P,r2) = \{0.0, 0.0, 0.2, 25.0, 0.0, 5.0, 0.154, 27.0\}
BScore(P,r2) = \{0,0,0,0,1,0,0,0\}

Why? Only one best match, differentiate it as much as possible.
4: Pass vector to SVM for match

{1, 1, 1, 0, 1, ...} → SVM
{0, 0, 0, 1, 0, ...} → SVM

Match
No Match
Information Extraction (IE)

Our 2 step solution:

1. Find match in Reference Set
   (ONTOLOGIES)

2. Use match for extraction
   (LABEL FOR ANNOTATION)
Step 2: Use Match to Extract

“IE / Labeling” step

Algorithm:

1. Break text into tokens
2. Generate vector of scores for each token versus the matching reference set member
3. Send vector of scores to SVM for labeling
Step 2: Use Match to Extract

Example:

“1988 Honda Accrd for sale! Only 80k miles, Runs Like New, V6, 2WD... $2,500 obo. SUPER DEAL.”

<table>
<thead>
<tr>
<th>Car Make</th>
<th>Car Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
</tr>
<tr>
<td>Honda</td>
<td>Civic</td>
</tr>
</tbody>
</table>
What if ???

Example:

“1988 Honda Accord for sale! Only 80k miles, Runs Like New, V6, 2WD... $2,500 obo. SUPER DEAL.”

<table>
<thead>
<tr>
<th>Car Make</th>
<th>Car Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
</tr>
<tr>
<td>Honda</td>
<td>Civic</td>
</tr>
</tbody>
</table>

Can still get some correct info!! Such as Honda
1: Break text into tokens

Example:

“1988 Honda Accrd for sale! Only 80k miles, Runs Like New, V6, 2WD... $2,500 obo. SUPER DEAL.”

{ “1998”, “Honda”, “Accrd”, “for” … }
2: Generate vector of scores

Vector of scores → “Feature Profile” (FP):

Score between each token and all attributes of reference set

Example:

“Accrd”

<table>
<thead>
<tr>
<th>Make</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honda</td>
<td>Accord</td>
</tr>
</tbody>
</table>

FP = \{ Scores(“Accrd”, Honda) U Scores(“Accrd”, Accord) \}

(sim. to Make) (sim. to Model)
Feature Profile

\[ FP = \{ \text{Scores(“Accrd”, Honda)} \cup \text{Scores(“Accrd”, Accord)} \} \]

\{ \text{Common(“Accrd”, Honda)} \cup \text{Edit-Dist(“Accrd”, Honda)} \cup \text{Other(“Accrd”, Honda)} \}\]

\{ \text{Smith-Waterman(“Accrd”, Honda)} \cup \text{Levenstein(“Accrd”, Honda)} \cup \text{Jaro-Winkler(“Accrd”, Honda)} \cup \text{Jaccard-Character(“Accrd”, Honda)} \} \]

\{ \text{Soundex(“Accrd”, Honda)} \cup \text{Porter-Stemmer(“Accrd”, Honda)} \} \]

No token based scores because use one token at a time…
Common Scores

- Functions that are user defined, may be domain specific
- Pick different common scores for each domain
- Examples:
  - Disambiguate competing attributes:
    - Street Name – 6th VS Street Num – 612
      - What if compare to reference attribute Street Num -- 600?
      - Same edit distance!
      - Common Score: Ratio of numbers to letters could solve this case
  - Scores for attributes not in reference set
    - Give positive score if match a regular expression for price or date
3: Send FP to SVM for Labeling

No binary rescoring $\rightarrow$ not picking a winner

$$FP = \{ \text{Scores(“Accrd”, Honda)} \cup \text{Scores(“Accrd”, Accord)} \}$$

FP’s not classified as an attribute type are labeled as Junk
Post Process

- Once extraction/labeling is done
  - Go backwards and group neighboring classes together as one class and remove junk labeling and make it correct XML

“…good <junk> Holiday <hotel> Inn <hotel> …”

“… good <hotel> Holiday Inn</hotel> …”
Experiments

- Domains:
  - COMICS:
    - Posts: Ebay Golden Age Incredible Hulk and Fan Four.
    - Ref Set: Comic Book Price Guide
  - HOTELS:
    - Posts: BiddingForTravel - Pitts, San Diego, Sacramento posts.
    - Ref Set: BFT Hotel Guide
Experiments

- **Domains:**
  - **COMICS:**
    - Attributes: price, date, title, issue, publisher, description, condition
  - **HOTELS:**
    - Attributes: price, date, name, area, star rating

Not in ref set
In ref set
Experiments

Precision = \( \frac{\text{# of Tokens Correctly Identified}}{\text{# of Total Tokens Given a Label}} \)

Recall = \( \frac{\text{# of Tokens Correctly Identified}}{\text{# of Total Possible Tokens with Labels}} \)

F-Measure = \( \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \)

Results reported as averaged over 10 trials
Baseline Comparisons

- **Simple Tagger**
  - From MALLET toolkit (http://mallet.cs.umass.edu/)
  - Uses Conditional Random Fields for labeling

- **Amilcare**
  - Uses Shallow NLP to do information extraction
  - (http://nlp.shef.ac.uk/amilcare/)
  - Included our reference sets as gazateers

- **Phoebus**
  - our implementation of extraction using reference sets
<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hotel</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phoebus</td>
<td>94.41</td>
<td>94.25</td>
<td>94.33</td>
</tr>
<tr>
<td>Simple Tagger</td>
<td>89.12</td>
<td>87.80</td>
<td>89.00</td>
</tr>
<tr>
<td>Amilcare</td>
<td>86.66</td>
<td>86.20</td>
<td>86.39</td>
</tr>
<tr>
<td><strong>Comic</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phoebus</td>
<td>96.19</td>
<td>92.5</td>
<td>94.19</td>
</tr>
<tr>
<td>Simple Tagger</td>
<td>84.54</td>
<td>86.33</td>
<td>85.42</td>
</tr>
<tr>
<td>Amilcare</td>
<td>87.62</td>
<td>81.15</td>
<td>84.23</td>
</tr>
</tbody>
</table>
Conclusion / Future Dir.

Solution:
- Perform IE on unstructured, ungrammatical text

Application:
- make user entered text searchable for agents on the Semantic Web

Future:
- Automatic discovery and querying of reference sets using a Mediator