Automatic Wrapper Generation and Data Extraction

Kristina Lerman
University of Southern California
Overview

- Methods for automatic wrapper creation and data extraction
  - Grammar Induction approach
    - *Towards Automatic Data Extraction from Large Web Sites*
  - Website structure-based approach
    - *AutoFeed: An Unsupervised Learning System for Generating Webfeeds*
    - *Using the Structure of Web Sites for Automatic Segmentation of Tables*
  - Rule-based extraction from natural language text
    - *KnowItAll*

- No hand-labeled training examples are required!
  - Scaled to the size of the Web
Data-intensive Web sites present results in dynamically generated pages.

Web sites are highly *structured* in terms of:
- Organization of the site
- Layout of pages
- Content of data

Exploit this structure for automatic information extraction.
Using the Structure of Web Sites for Automatic Segmentation of Tables
Structure of Web Sites

Entry page → List pages → Detail pages
Data in the same "column" is of the same type

- Each listing starts with NAME, followed by ADDRESS, CITY, STATE, etc.
Underlying Structure is not Always Clear

- Variability of real-world data may obscure the underlying structure
  - Missing columns
    - “List Price” and “You save”
  - Formatting
  - Content
Automatically, efficiently extract records from Web tables

Given a set of list and detail pages...
- Segment list data using information from detail pages
  - Logic based approach
    - Based on Constraint Satisfaction Problems (CSP)
    - Encode relations between data on list and detail pages as logical constraints and solve them
  - Probabilistic inference approach
    - Learns a model from data
    - Record segmentation is an assignment that maximizes the likelihood of data given the model
Identify Table and Extract Data

- Page template
  - Sequence of tokens shared by all pages
- Deduce page template
  - Given two or more example pages, derive the page template used to generate them
- Table data and formatting tags are not part of the template
- Find table
  - Extract contiguous sequences of tokens from the largest page slot
Record Segmentation Basics (1)

- List and detail pages present two views of the same record
  - Some overlapping fields
- Each detail page is a distinct record
- Assumption: Web tables are laid out horizontally
  - Each record is in a separate row
  - Order in which extracts appear in the text stream of list page is the same order they appear in the table
For each extract $E_i$, record all detail pages on which it appears:

- $E_1$: John Smith  
  Pages: r1, r2

- $E_2$: 221 Waterloo  
  Pages: r1

- $E_3$: New Holland  
  Pages: r1

- $E_4$: (740) 335-5555  
  Pages: r1, r2

- $E_5$: John Smith  
  Pages: r1, r2

- $E_6$: 221R Waterloo  
  Pages: r2

- $E_7$: Washington  
  Pages: r2

- $E_8$: (740) 335-5555  
  Pages: r1, r2

- $E_9$: George W. Smith  
  Pages: r3

- $E_{10}$: Findlay, OH 45840  
  Pages: r3

- $E_{11}$: (419) 423-1212  
  Pages: r3
Observations of extracts on detail pages add valuable information for record segmentation.

Second record can be

- $E_4E_5E_6E_7E_8$
- $E_4E_5E_6E_7$
- $E_5E_6E_7$
- $E_5E_6E_7E_8$
- $E_6E_7E_8$
- $E_6E_7$
CSP Approach to Record Segmentation

- In CSP, problems are stated as logical expressions over variables
  - Pseudo-boolean (PB) representation
    - Variables are 0-1, constraints can be inequalities
    - Solution is assignment that minimizes inequality constraints
- Encode record segmentation problem in PB representation
  - Assignment variable $x_{ij}$
    - $x_{ij}=1$ when $E_i$ is assigned to $r_j$
    - $x_{ij}=0$ when $E_i$ is no part of $r_j$
- Information from detail pages imposes constraints
  - Structure constraints
  - Position constraints
Structure Constraints

<table>
<thead>
<tr>
<th></th>
<th>$E_1$</th>
<th>$E_2$</th>
<th>$E_3$</th>
<th>$E_4$</th>
<th>$E_5$</th>
<th>$E_6$</th>
<th>$E_7$</th>
<th>$E_8$</th>
<th>$E_9$</th>
<th>$E_{10}$</th>
<th>$E_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_2$</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Uniqueness constraint**
  - Every extract $E_i$ belongs to exactly one record $r_j$

  \[ \sum_j x_{ij} = 1 \]

- **Consecutiveness constraint**
  - Only contiguous blocks of extracts can be assigned to the same record
Structure Constraints

- **Uniqueness constraint**
  - Every extract $E_i$ belongs to exactly one record $r_j$

- **Consecutiveness constraint**
  - Only contiguous blocks of extracts can be assigned to the same record

  \[ x_{ij} + x_{kj} \leq 1 \text{ when there is } n, \ k < n < i, \ \text{s.t. } x_{nj} = 0 \]
Position Constraints

- Position constraint
  - No two extracts assigned to same record can appear in the same position on the detail page
  - \( \text{pos}_j(E_i) = \text{pos}_j(E_k) \), then \( E_i \) and \( E_k \) cannot be assigned to same record \( j \)
- Constraints are expressed mathematically and solved using integer optimization

<table>
<thead>
<tr>
<th></th>
<th>( E_1 )</th>
<th>( E_2 )</th>
<th>( E_3 )</th>
<th>( E_4 )</th>
<th>( E_5 )</th>
<th>( E_6 )</th>
<th>( E_7 )</th>
<th>( E_8 )</th>
<th>( E_9 )</th>
<th>( E_{10} )</th>
<th>( E_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p^{730}_1 )</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p^{772}_1 )</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p^{812}_1 )</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p^{846}_1 )</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 1 )</td>
</tr>
<tr>
<td>( p^{536}_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p^{578}_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p^{608}_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( p^{642}_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Probabilistic Approach to Record Segmentation

- Record segmentation as probabilistic inference
  - No labeled training examples
  - **Factor** the problem for efficient learning
  - **Bootstrap** the learning algorithm with information from detail pages
  - **Structure** constrain the problem further with global parameters such as record length
Probabilistic Model for Record Extraction: Variables

- **Observed variables**
  - $T = \{T_1, \ldots, T_n\}$ token types of extract $E_i$
  - $D = \{D_1, \ldots, D_n\}$ detail pages on which $E_i$ was observed

- **Unobserved variables**
  - $R = \{R_1, \ldots, R_n\}$ record id
  - $C = \{C_1, \ldots, C_n\}$ column label
  - $S = \{S_1, \ldots, S_n\}: S_i = \text{true if } E_i \text{ is the start of a new record; false otherwise}$

- **Dependencies**
  - Given by arrows, eg, $P(C_i | C_{i-1})$

- **Segmentation**
  - find values for $R$ and $C$ given $T$, $D$ variables: $\arg\max P(R, C | T, D)$
Probabilistic Model for Record Extraction: Dependencies

- $P(T_i|C_i)$: token type of $E_i$ depends on column
- $P(C_i|C_{i-1})$: column label of $E_i$ depends on previous column label (e.g., NAME followed by ADDRESS, sometimes by STATE)
- $P(S_i|C_i)$: new record starts with a given column (e.g., NAME)
- $P(R_i|R_{i-1},D_i,S_i)$: record number of $E_i$ depends on record number of previous extract, whether it starts a new record, and detail pages on which it was observed.
Learning the Model

- **Constrain the problem further**
  - **Bootstrap**
    - Detail pages provide initial guesses for parameters
      - $P(R_i=r_i)$
      - Evidence about where records start: $P(S_i=true)=1$
      - Token types of columns $P(T_j|C_i)$
  - **Structure**
    - Table has $\pi$ columns specified by the underlying database schema
    - However, not every record will have an attribute for every field, i.e., not every record has $\pi$ fields
    - Number of fields in a record estimated from data
## Learning the Model

### Initial guess for record assignment \( P(R_i) \)

<table>
<thead>
<tr>
<th>( P(R_i=r_i) )</th>
<th>( E_1 )</th>
<th>( E_2 )</th>
<th>( E_3 )</th>
<th>( E_4 )</th>
<th>( E_5 )</th>
<th>( E_6 )</th>
<th>( E_7 )</th>
<th>( E_8 )</th>
<th>( E_9 )</th>
<th>( E_{10} )</th>
<th>( E_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_1 )</td>
<td>1/2</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( r_2 )</td>
<td>1/2</td>
<td></td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( r_3 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Initial guess for record start \( P(S_i) \)

<table>
<thead>
<tr>
<th>( P(S_i) )</th>
<th>( E_1 )</th>
<th>( E_2 )</th>
<th>( E_3 )</th>
<th>( E_4 )</th>
<th>( E_5 )</th>
<th>( E_6 )</th>
<th>( E_7 )</th>
<th>( E_8 )</th>
<th>( E_9 )</th>
<th>( E_{10} )</th>
<th>( E_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

### Initial guess for length of records

<table>
<thead>
<tr>
<th>( \pi_k )</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(\pi_k) )</td>
<td>2/14</td>
<td>6/14</td>
<td>4/14</td>
<td>2/14</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Learning Algorithm

- Use EM to implement the inference algorithm
  1. Initial guess for $\pi_k$ for each record $j$
  2. For each potential record, update $P(C_i | T_i, C_{i-1})$
  3. Update $P(S_i | C_i)$
  4. Update $P(R_i | R_{i-1}, D_i, S_i)$

Result is the most likely assignment of data to $R$ and $C = \text{record segmentation}$. 
Validation

- Input data
  - list and detail pages from 12 sites in domains: book sellers, property tax, white pages, corrections

- Metrics
  - \[ P = \frac{Cor}{Cor + InCor + NonRecords} \]
  - \[ R = \frac{Cor}{Cor + UnsegRecords} \]
  - \[ F = \frac{2PR}{P + R} \]

- Results
  - CSP approach: P=0.85, R=0.84, F=0.84
  - Probabilistic approach: P=0.74, R=0.99, F=0.85
  - Good performance for an automatic algorithm!
Discussion of Results

- CSP approach is very reliable on clean data, but sensitive to errors in data source
  - Attribute has one value on list page and another on detail page
- Probabilistic approach tolerates inconsistencies and is more expressive
- Combination of two techniques may be more robust
Comparison with RoadRunner

- **RoadRunner System**
  - Automatically learns the page and table template by exploiting similarities in page layout (HTML tags)
  - Uses the template to automatically extract data
  - Does not allow for disjunctions
    - Disjunctions are necessary to represent alternative layout instructions for the same field
Discussion

- Domain-independent approach for automatically extracting and segmenting data from Web tables
  - Approach leverages additional information provided by Web site structure
    - Logic based approach
      - Information provided by detail pages encoded as constraints and solved to obtain record segmentation
    - Probabilistic inference approach
      - Information provided by detail pages and table structure represented as a probabilistic model
      - Use inference to learn proper segmentation
- Validated approach on 12 Web sites from diverse information domains
  - Efficient, accurate performance, F=0.85 and F=0.84
KnowI tAll: Methods for Domain-Independent Information Extraction from the Web
Automatic Data Extraction

- Extract data from the Web without hand-labeled training examples

Types of information extracted

- Data tuples

- Facts
  - Entities
    - ‘Los Angeles’, ‘Albert Einstein’
  - Classes and relations
    - Class instances:
      - ‘Los Angeles’ is a CITY
      - ‘Albert Einstein’ is a SCIENTIST
KnowItAll Approach

Two-stage approach to automatic data extraction

- Extraction patterns to generate candidate facts
  - Pattern “NP1 such as NP2”
  - “…tours in cities such as Paris and Berlin”
  - Extracts class CITY with instances Paris and Berlin

- Text candidate facts using Pointwise Mutual Information (PMI)
  - Statistics computed from all text on Web
    - Use existing Web search technology to efficiently compute statistics
  - Associates a probability with every fact it extracts
    - Automatically manage tradeoff between precision and recall
Extractor

- Extractor - natural language patterns to extract instances of classes
  NP1 "such as" NPList2
  & head(NP1) = plural(Class1)
  & properNoun(head(each(NPList2)))
  => instanceOf(Class1, head(each(NPList2)))
  keywords: "plural(Class1) such as"

- Uses part-of-speech tagger to identify Noun Phrases (NP)
Search Engine Interface

- Query search engine with phrases
  - “cities such as”
- Apply Extractor to all pages return by search
Assessor

- Uses statistics computed over all Web pages to assess the likelihood that extracted fact $I$ is correct
  - Pointwise Mutual Information (PMI)
    \[ PMI(I, D) = \frac{|\text{Hits}(D+I)|}{|\text{Hits}(I)|} \]
    - $D$ is discriminator phrase: e.g., “city of”
    - Hits($x$) = number of Web pages that contain $x$
  - PMI is a feature to Naïve Bayes Classifier
    - More likely classes get higher probabilities
    - Probability threshold tunable parameter to increase precision (at expense of recall)
Precision and Recall – a recap

Extracted facts about cities

Recall = $E^R / R$
Precision = $E^R / E$

Goal: Make the blue circle overlap more of the yellow circle!
Enhancements to KnowItAll

- Enhancements to increase precision & recall
  - Rule Learning
    - Learns domain specific rules and validates accuracy of instances they extract
  - Subclass Extraction
    - Automatically identify subclasses
      - Learn that physicists, geologists, etc. are subclasses of scientists
      - Rule “physicists such as …” will extract more scientists
  - List Extraction
    - Locate lists of class instances
    - Learns a wrapper for the list to extract instances
Enhancements: Rule Learning

- Learn domain-specific rules to increase KnowItAll’s precision and recall
  E.g., “… headquartered in <CITY> …”

1. Start with instances extracted by generic patterns
2. Query search engine with instances → pages
3. From each page, extract context string for instance
   - 4 words before, and after
4. ‘Best’ substrings of the ‘best’ context strings are converted to new Extraction Rules that extract new instances with high precision
   - Heuristic: Prefer substrings that appear in multiple pages
   - Heuristic: Penalize substrings that lead to many false positives
Examples of Rule Learning

Most productive rules learn for each class, with number of correct extractions and precision

1. the cities of `<city>` 5215 0.80
2. headquartered in `<city>` 4837 0.79
3. for the city of `<city>` 3138 0.79
4. in the movie `<film>` 1841 0.61
5. `<film>` the movie starring 957 0.64
6. movie review of `<film>` 860 0.64
7. and physicist `<scientist>` 89 0.61
8. physicist `<scientist>`, 87 0.59
9. `<scientist>`, a British scientist 77 0.65
Subclass Extraction

- Identify subclasses and instantiate new generic patterns
  - PHYSICIST is a subclass of SCIENTIST → new rule “physicists such as …”
  - Increases KnowItAll coverage
- Subclasses of SCIENTIST found by KnowItAll
  - biologist
  - zoologist
  - astronomer
  - meteorologist
  - mathematicia
  - n
  - economist
  - geologist
  - sociologist
  - chemist
  - oceanographer
  - anthropologist
  - pharmacist
  - psychologist
  - climatologist
  - paleontologist
  - neuropsychologist
  - engineer
  - microbiologist
Subclass Extraction

1. Apply Subclass Extraction rules to extract candidate subclasses
   - “… such C₁ as CN …” → CN is subclass of C₁.
   - “… CN and other C₁ …” → CN is subclass C₁.

2. Assess validity of candidate
   - Is subclass in a reference taxonomy (WordNet)?
   - Check word morphology → “microbiologist” is a subclass of “biologist”
List Extraction

- Extract information from formatted lists
- Approach
  - Query search engine with $k$ random instances extracted by Knowl tAll
  - In each Web page, search for a list containing these keywords using HLRT-like wrapper induction algorithm*
    - Convert Web page to DOM tree
    - Select subtrees corresponding to positive examples
    - Finds greatest common prefixes (and suffixes) for these examples
    - Choose header and tail strings to limit extraction to good subtrees
  - *Can learn wrapper from few positive examples
  - Assess the likelihood of each extracted instance
    - Rank instances by the number of lists they appear in
Evaluation

- For classes CITY and FILM
  - Extracted >40k (compared to baseline 10k) at 90% precision
    - Most of the improvement due to List Extraction

- For class SCIENTIST
  - Extracted 40k instances (compared to baseline ~2k) at 90% precision
    - Most of the improvement due to Subclass Extraction
Discussion

- Automatic collection of large body of facts
- Extract facts (e.g., instances of classes) from text using generic NLP rules
- Heuristics added to KnowItAll (rule learning, subclass extraction, list extraction) greatly improve recall while maintaining high precision
Conclusion

- Covered method to automatically extract massive data sets from Web pages
  - Structured pages
  - Natural language text
- Extraction from structured Web pages
  - Exploit structure in pages (grammar)
  - Exploit structure of site
- Extraction from text
  - Exploit NLP rules and Web statistics to extract high quality facts