Web Information Extraction using Visual Regions

Ramakrishna Varadarajan
(5th Year PhD from Florida Int’l University)

IBM India Summer Research Internship 2008

Research group members:
- Raghuram Krishnapuram
- Prasad M Deshpande
- Vijil E Chenthamarakshan
Motivation

• Information in Web pages are designed for human consumption and hence exhibit some visual pattern.

• The web page author communicates this abstract visual pattern to the web browser using a specification language (HTML/CSS/Javascript Code, etc.)

• Humans typically don’t look at the specification code to understand the data. They look at the rendered version of the page through a browser.

• However, current rule based information extraction systems don’t deal with visual representations of the page. They look for patterns in the specification language code.
Issues in Pure Text Based Information Extraction Systems

• The specification languages are becoming more and more complex and difficult to analyze.

• Visualization logic is in Javascript and CSS, preventing text based analysis.

• The same visual pattern can be implemented using multiple ways (E.g. A single column tabular structure can be implemented using <table>, <div>, <li> etc.) Hence rules based on DOM trees and HTML tags are tied to specific implementation of the pages.

• If a rule is tied too much to the implementation, it performs poorly with manually authored web pages (as opposed to machine generated web pages)

• Rules become sensitive to even minor modifications of the web page and rule maintenance is an issue.
Our Contributions

- A rule based information extraction system that can exploit visual information in a rendered web page.

- We describe a visual operator algebra, for extracting regions from a web page based on their relative locations. This algebra is rich enough to specify most of the commonly occurring patterns.

- We implement the visual operator algebra using a geographical database (DB2 Spatial Extender). The GIS database is used to store, query and retrieve visual regions of a web page. This provides efficient indexing, retrieval and query optimization.

- We show how such a system can combined with a traditional text based information retrieval system.

- Some preliminary experiments on the usefulness of the system.
Example of a visual pattern
Visual Regions on Web Pages

• A box (rectangle) with the following properties:
  – $x, y$ coordinates of the top left of the box.
  – Width and Height of the box.
  – A Tag.
  – Text contained in the region.
  – HTML code contained in the region.

• How can we find all visual regions of a page?
  – The visual regions are retrieved from the browser using APIs provided by the Firefox browser.
  – Visual regions can be found for all the HTML tags as well as for data annotated with meta tags.
Visual Operator Algebra

Directional Predicates

- **NorthOf** (Region r1, Region r2) (returns *Boolean*)
- **StrictNorthOf** (Region r1, Region r2)
- **SouthOf** (Region r1, Region r2)
- **StrictSouthOf** (Region r1, Region r2)
- **EastOf** (Region r1, Region r2)
- **StrictEastOf** (Region r1, Region r2)
- **WestOf** (Region r1, Region r2)
- **StrictWestOf** (Region r1, Region r2)
- **NorthWestOf** (Region r1, Region r2)
- **SouthWestOf** (Region r1, Region r2)
- **NorthEastOf** (Region r1, Region r2)
- **SouthEastOf** (Region r1, Region r2)

All above functions Implemented as a database User-Defined Function

Our Goal: Design a rule-based system for Information Extraction using visual regions.

Framework: A formal algebra of useful operations over visual regions.

Rules potentially vary with dataset but the underlying basic operations would remain the same.
Visual Operator Algebra (Contd)

**Grouping Operators**

- *HorizontallyAligned* (Region Set $R$) returns horizontally aligned subgroups of region set $R$.

- *VerticallyAligned* (Region Set $R$) returns vertically aligned subgroups of region set $R$.

**Generalization/Specialization Operators**

- *MinimalSuperRegion* (Region set $R$) returns the smallest region that contains all regions in set $R$.

- *MaximalRegion* (Region $r$) returns the largest region, $s$ that contains $r$ and also has the same text content as $r$.

- *MinimalRegion* (Region $r$) returns the smallest region, $s$ that is contained in $r$ and also has the same text content as $r$. 
Visual Operator Algebra (Contd)

**Containment Operators**
- **Contains** (Region $r_1$, Region $r_2$)
- **Within** (Region $r_1$, Region $r_2$)
- **Touches** (Region $r_1$, Region $r_2$)
- **Intersects** (Region $r_1$, Region $r_2$)

**Geometrical Operators**
- **Area** (Region $R$)
- **Centroid** (Region $R$)

**Distance Operators**
- **GapBetween** (Region $r_1$, Region $r_2$) returns the distance between 2 regions that are either horizontally or vertically aligned. Returns 0 when the 2 regions touch each other in their boundaries.

**Other Operators**
- **FindRegions** (Text Pattern $p$) returns regions with text pattern $p$. 
Architecture (Implementation)

- List of URLs
- webpage dump (firefox extension)
- Local copies of the web pages
- GATE (Annotation & Tagging)
- JavaScript DOM + ChickenFoot (firefox extension)
- Visual Regions

- SQL (visual Information extraction rules)
- Access/Manipulate Visual regions
- JDBC
- Insert Structured Data
- DB2 & DB2 spatial Extender (Visual Operator Algebra implemented as UDFs)
- JDBC
- Java (inserting visual regions)
Implementation Details

- Visual Operator Algebra implemented in DB2 Spatial Extender (a geographic database) using a combination of built-in and user defined functions.
- New high level operators can be implemented as UDFs.
- Information Extraction Queries written in SQL.
- Mozilla Firefox Extensions (to retrieve visual regions rendered in browser)
  - WebPageDump (to save local copies of web pages).
  - ChickenFoot (Execute javascript in the context of Browser).
- Any Traditional Text Based Annotation System (GATE/System-T/UIMA).
- XRAY for web developers to visualize web page regions.
Dewey Encoding of visual region IDs [1850s]

We can differentiate regions across pages by adding the document node as parent to root.

Taken from: XRANK, SIGMOD 2003

Slide 11
Advantages of rule based extraction using visual regions:

- Rules are easier to write as they are at a higher level of abstraction.
- Not implementation specific (not tied up to the page implementation) and hence will work better across service providers (Makemytrip.com/Cleartrip)
- Rule maintenance is simpler as the rules are more robust to page changes.
- Can be easily combined with traditional information extraction systems
- Rules exploit indexing, retrieval and query optimization capabilities of the geographical database. Scalable to a large number of regions.
- Extensible: Write your own high level abstractions as user defined functions in the geographical database and use them in your rules.
Problems & Heuristic solutions

Problem:

Given a system requirements page \( sp \), identify software/hardware requirements of the product.

- Sub Task 1 – Filtering the navigational links of a page
- Sub Task 2 – Extracting the requirements

Sub Task 1: Filter out navigational regions of a page, \( p \):

1. Region set \( R \leftarrow \text{FindRegions}(["A"] \ ^ \text{Page}[p]); \)
2. Main Content Region \( m \leftarrow \text{EstimateMainContentRegion}(p); // \) next slide
3. For each \( r \) in \( R \) do {
   while(db2gse.ST_Intersects(r,m)=0 || db2gse.ST_Touches(r,m)=1)
   {
      \( r \leftarrow \text{MinimalSuperRegion}(r); \)
   }
   Output(r); }
4. Step 3 is done since HTML tags are nested. Hence parent navigational regions should also be eliminated. Hence we grow these regions until they intersect with the main content region.
Filtering Navigational Regions

Filtering navigational parts of the page.

Pattern:
- Groups of Links that are Horizontally or Vertically Aligned.
- Appear in the corners of the page far from the main content
- Primary purpose is intra-site navigation
Problem & Heuristic Solutions

EstimateMainContentRegion (IBM product Page p):

1. Find region that contains the product name (also appearing in the title) surrounded by H1 tag.
   - Header_region $\leftarrow$ FindRegions(Tag[‘H1’]);
   - This defines the $x$, $y$ and $width$ of the main content region.

2. The height of main content region can be defined as 85% of the height of entire page region. This heuristic works for most of the IBM product pages.

3. Find page boundaries using the following:
   - Region bodyregion $\leftarrow$ FindRegions(Tag[‘BODY’]);
   - Region pageregion $\leftarrow$ MinimalRegion(bodyregion);
Sub Task 2 – Extracting Product Requirements

1. Find *maximal region set* $R$ that contains phrases that relate to some requirements like “software requirements”, “hardware requirements”, “operating systems” and so on.

2. \( hgrps \leftarrow \text{HorizontallyAligned}(R) \)

3. \( vgrps \leftarrow \text{VerticallyAligned}(R) \).

4. Identify the pattern that is the most significant in the page. [**Heuristic**: Subgroup with maximum size]

5. Extract the requirements accordingly.
Sub Task 2 Examples

Horizontally aligned regions (with no gaps)

Vertically aligned regions (with no gaps)
Vertically aligned regions (with large gaps)

Functions used for information extraction:

- **SouthOf**
- **NorthOf**

Notice that it is important to discard navigational regions before information extraction.

This reduces errors in extraction.
Sub Task 2 Examples

Vertically aligned regions (with compact subgroups with gaps in between)

Functions used for information extraction:

- **StrictEastOf**

- Notice that it is important to discard navigational regions before information extraction.

- This reduces errors in extraction.
Preliminary Results

• **Dataset** – A sample of 22 IBM product pages of varying formats.

• Algorithms evaluated:
  – **Sub Task 1** : Filtering out the navigational parts of a page
  – **Sub Task 2**: Extracting product requirements

*Evaluation Metrics* used:

• For Sub Task 1:
  - Precision = \( \frac{\text{No. of navigational regions returned}}{\text{Total no. of regions returned}} \)
  - Recall = \( \frac{\text{No. of navigational regions returned}}{\text{Total no. of navigational regions}} \)

• For Sub Task 2:
  - Precision = \( \frac{\text{No. of requirement regions returned}}{\text{Total no. of regions returned}} \)
  - Recall = \( \frac{\text{No. of requirement regions returned}}{\text{Total no. of requirement regions}} \)
Preliminary Results

Results for Sub Task 1

- Total number of navigational regions – 5046.
- Number of navigational regions returned – 4993.
- **Recall** = 4993/5046 = 98.95%.
- Number of navigational regions returned – 4993.
- Total number of regions returned – 4993.
- **Precision** = 4993/4993 = 100.00%.
- **Major Reasons:**
  - Can estimate the main content region fairly accurately.
    - Can precisely find the top left x, top left y and width.
    - Cannot precisely estimate the height of the main content region (heuristic used is 85% of the total page height).
  - Can accurately find the LEFT, TOP, RIGHT and BOTTOMMOST navigation regions.
  - Rules fine tuned for IBM product pages.
Preliminary Results

Results for Sub Task 2

• Number of requirement regions returned –1748.
• Total number of regions returned –2037.
• \textbf{Precision} = \frac{1748}{2037} = 85.81\%.
• Total number of requirement regions –1748.
• Number of requirement regions returned –1748.
• \textbf{Recall} = \frac{1748}{1748} = 100.00\%.

• \textbf{Major Reasons:}
  – Cannot figure out boundaries of regions for extraction!
  – Rules fine tuned for the dataset of 22 pages.
  – In some pages it is difficult to find the most significant pattern.
Preliminary Results

Results on a Held Out Dataset

Dataset of 10 randomly selected IBM Product requirement pages.

Results for Sub Task 1:

- Number of navigational regions returned – 2282.
- Total number of regions returned – 2289.
- **Precision** = \( \frac{2282}{2289} = 99.69\% \).
- Total number of navigational regions – 2282.
- Number of navigational regions returned – 2282.
- **Recall** = \( \frac{2282}{2282} = 100\% \).
- **Major Reasons:**
  - We can precisely compute the header of the main content region.
  - The Heuristic – “85% of height of the page comprises of the main content” fails in some cases.
Preliminary Results

Results on a Held Out Dataset

Dataset of 10 randomly selected IBM Product requirement pages.

Results for Sub Task 2:

- Total number of requirement regions – 258.
- Number of requirement regions returned – 190.
- \[ \text{Recall} = \frac{190}{258} = 73.64\% \]
- Number of requirement regions returned – 190.
- Total number of regions returned – 190.
- \[ \text{Precision} = \frac{190}{190} = 100\% \]
- Major Reasons:
  - The meta phrases that predict the requirement regions varied.
Subtle complications & Disadvantages

• Visual regions of a page depend on the HTML DOM.

• Some interesting (useful) regions may be absent because those regions don’t have a corresponding node in the HTML DOM. *(Annotating useful text fragments can alleviate this problem).*

• Visual regions returned are always in box form (rectangle). If a visual region cannot be represented as a rectangle, then the browser returns incorrect boundaries for those regions.

• Sensitive to screen size changes & browser size changes.

• Estimating the exact boundaries of a page could be tricky.

• The HTML document should have some structure or at least organized in some manner. This technique cannot be useful on plain text.
Future Directions

- More experimentation on different service providers
- Performance evaluation to demonstrate the scalability of the system
- Tighter integration with text based annotators like System-T
- Ability to add Visual Annotators in a UIMA pipeline