2-Layered HMMs for Search Interface Segmentation

Ritu Khare
(Under the Supervision of Dr Yuan An, Assistant Professor, iSchool)
Order of Presentation

- Background
  - Deep Web
  - What is Search Interface Understanding?
- What is Interface Segmentation?
- Why is Segmentation Challenging?
- Our Approach for Segmentation
- Interface Representation
- HMM: The Artificial Designer
- 2-Layered Approach
- Architecture
- Experimentation Parameters
- Result
- Contributions
- Future Work
- References
Background: Deep Web

- **What is Deep Web:**
  - The data that exists on the Web but is not returned by search engines through traditional crawling and indexing. The primary way to access this data is by filling up HTML forms on search interfaces.
  - Characteristics: A large proportion of structured databases; Diversity of domains; and its; Growing scale

- **Researchers have many goals for the deep Web:**
  - design intra-domain meta-search engines [22, 8, 15, 5, 21]
  - increase content visibility on existing search engines [17, 12]
  - derive ontologies from search interfaces [1].

- **A pre-requisite to attain these goals is an understanding of the search interfaces** (slide 4). In this project, we propose an approach to address the segmentation (slide 5) portion of the problem of search interface understanding.
Background: What is Search Interface Understanding?

Understanding semantics of a search interface (shown in figure) is an intricate process [4].

It involves 4 stages.

1. **Representation:** A suitable interface representation scheme is chosen; semantic labels (slide 8) to be assigned to interface components are decided. An interface component is any text or HTML form element (textbox, textarea, selection list, radiobutton, checkbox, file input) that exists inside an HTML form.

2. **Parsing:** Components are parsed into a suitable structure.

3. **Segmentation:** The interface components are assigned semantic labels, and related components are grouped together. The questions like “Which surrounding text is associated with which form element?” (In figure 2, “Gene ID” is associated with the textbox placed next) are also answered in this stage.

4. **Segment-processing:** Additional information, such as domain, constraints, and data type, about each segment component is extracted.
What is Interface Segmentation?

- This project focuses on **Segmentation**, the 3\textsuperscript{rd} stage of this process. Figure shows a segmented interface. The related components are grouped together.

- The left segment has 7 components. The right segment has 4 components (**cM Position:**, selection list, textbox, and **“e.g., “10.0-40.0””**).
Why is Segmentation Challenging?

- From a user’s (or designer’s) standpoint,
  - By looking at the visual arrangement of components, and based on past experiences, the user creates a logical boundary around the related components as they appear to belong to the same atomic query.

- On the other hand, a machine is unable to “see” a segment due to the following reasons:
  - The components that are visually close to each other might be located very far apart in the HTML source code,
  - A machine does not implicitly have any search experience that can be leveraged to identify a segment boundary.

- This project aims to investigate whether a machine can “learn” how to understand and segment an interface.

- Existing works have two shortcomings:
  - They [9, 13, 17] do not group all related components together i.e. do not create complete segments.
  - They [23, 7] use rules and heuristics to segment a search interface. These techniques have problems in handling scalability and heterogeneity [10].
Our Approach for Segmentation

- We incorporate the first-hand implicit knowledge using which a human designer is assumed to have designed an interface.

- This is accomplished by designing an artificial designer using **Hidden Markov Models** (refer to week 9’s slides on HMM introduction).

- We visualize segmentation as a two-folded problem
  - Identification of boundaries of **logical attributes** (slide 9)
  - Assignment of semantic labels (**attribute-name**, **operator**, and **operand** described in slide 9) to interface components.
In figure, each component of the lower segment is marked with a label, which we term as a *semantic label*. The semantic label for a particular component denotes the meaning of the component from a user’s or designer’s standpoint.
Interface Representation

- **Attribute-name**: Attribute-name denotes the criteria available for searching a particular entity, e.g. the entity “Genes” can be searched by “Gene ID” and by “Gene Name”.

- **Operand**: An attribute-name is usually associated with operand(s), the value(s) entered by the user that is(are) matched against the corresponding field value(s) in the underlying database.

- **Operator**: The user may also be given an option of specifying the operator that further qualifies an operand.

- Filling up an HTML form is similar to writing SQL queries. Assuming the underlying database table name is “Gene”, the SQL queries for figure would be:
  - SELECT * FROM Gene WHERE Gene_ID = ‘PF11_0344’;
  - SELECT * FROM Gene WHERE Gene_Name LIKE ‘maggie’;

- **Logical Attribute**: The predicate in the WHERE clause of each query is created by a group of related components. We combine the semantic roles (attribute-name, operator(s), and operand(s)) of these components to create a composite semantic label called logical attribute. Our approach assumes that a segment corresponds to a logical attribute.
HMM: The Artificial Designer

- We assume that an HMM can act like a human designer who has the ability to design an interface using acquired knowledge and to determine (decode) the segment boundaries and semantic labels of components.

- The designing process is similar to statistically choosing one component from a bag of components (a superset of all possible components) and placing it on the Web page while keeping the semantic role (attribute-name, operand, or operator) of the component in mind.
HMM: The Artificial Designer

- While the components are observable, their semantic roles appear hidden to a machine. The proceeding of one semantic label by another is similar to the transitioning of HMM states. In the figure, Ovals=states (semantic labels); Rectangles= emitted symbols (components).

- The designing ability is provided by training the HMM with suitable algorithms. Once an HMM is trained, it can be used for the decoding process i.e. for explaining the design of a given search interface.
2-Layered HMM

- The problem of decoding that we address in this paper is two-folded involving segmentation as well as assignment of semantic labels to components. Hence, we employ a layered HMM [14] with 2 layers.
  - The first layer T-HMM tags each component with appropriate semantic labels (attribute-name, operator, and operand).
  - The second layer S-HMM segments the interface into logical attributes.
Experimentation Parameters

- **Data-Set**: 200 interfaces (NAR collection)  
  http://www3.oup.co.uk/nar/database/c/

- **Parsing**: DOM-trees [3] of components. Trees were traversed in the depth-first search order.

- **Testing and Training Data**: The examples were randomly divided into 20 equal-sized sets. We conducted 20 experiments each having 190 training and 10 testing examples.

- **Testing and Training Algorithms**: In both layers, training and testing were performed using Maximum Likelihood method and Viterbi algorithm respectively.
## Results

<table>
<thead>
<tr>
<th>Semantic label</th>
<th>Total</th>
<th>Correctly Identified</th>
<th>Accuracy</th>
<th>Reason for Misidentification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment</td>
<td>459</td>
<td>395</td>
<td>86.05</td>
<td>misidentification of one of the states by T-HMM.</td>
</tr>
<tr>
<td>Operator</td>
<td>47</td>
<td>40</td>
<td>85.10</td>
<td>because an operator state was mistaken as one of the other 3 states.</td>
</tr>
<tr>
<td>Operand</td>
<td>429</td>
<td>423</td>
<td>98.60</td>
<td>misidentified as an operator state</td>
</tr>
<tr>
<td>Attribute-name</td>
<td>546</td>
<td>492</td>
<td>90.109</td>
<td>In the rest of the cases, attribute-name was misidentified as text-trivial</td>
</tr>
</tbody>
</table>
Contributions

- We studied a challenging stage (segmentation) of the process of search interface understanding. In the context of deep Web, this is the third formal empirical study (after [23] and [7]) that groups components belonging to the same logical attribute together.
- We incorporated the first-hand knowledge of the designer for interface segmentation and component tagging. To the best of our knowledge, this is the first work to apply HMMs on deep Web search interfaces.
- The interface has been represented in terms of the underlying database. This helped in extracting database querying semantics.
- Moreover, we tested our method on a less-explored domain (biology), and found promising results.
Future Work

- To recover the schema of deep Web databases by extraction of finer details such as data type and constraints of logical attribute.

- To do justice to the balanced domain distribution of the deep Web [6], we want to test this method on interfaces from other less-explored domains.

- To improve the degree of automation we want to investigate the use of Baum Welch training algorithm.

- To minimize the zero emission probabilities, we want to investigate the use of Synset-HMM [20].
References


References


Thank You

Questions, Comments, Ideas???