

**APPLYING MACHINE LEARNING APPROACHES  
FOR STRUCTURED DATA EXTRACTION**

*Dissertation submitted to Jawaharlal Nehru University  
in partial fulfillment of the requirement  
for the award of the degree of*

**MASTER OF TECHNOLOGY  
In  
COMPUTER SCIENCE AND TECHNOLOGY**

**By**

**Anupama Pandey**

*Under the Supervision of*

**Dr. Aditi Sharan**



**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI – 110067  
JULY-2010**



**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI – 110067**

**DECLARATION**

I hereby declare that the dissertation work entitled “**Applying Machine Learning Approaches For Structured Data Extraction**” in partial fulfillment for the requirements for the degree of “**Master of Technology in Computer Science and Technology**” and submitted to School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, India. It is the authentic record of my own work carried out during the time of Master of Technology under the supervision of Dr. Aditi Sharan. This dissertation comprises only my original work.

The matter personified in the dissertation has not been submitted for the award of any other degree or diploma.

*Anupama*

**Anupama Pandey**

M.Tech (2008-2010)

School of Computer and System Sciences,

Jawaharlal Nehru University,

New Delhi 110067




**SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI – 110067**

**CERTIFICATE**

This is certify that the dissertation entitled “**Applying Machine Learning Approaches For Structured Data Extraction**” is being submitted by **Miss Anupama Pandey** to **School of Computer and Systems Sciences, Jawaharlal Nehru University New Delhi-110067, India** in the partial fulfillment of the requirements for the award of the degree of **Master of Technology in Computer Science and Technology**. This work carried out by herself in the School of Computer and Systems Sciences under the supervision of Dr. Aditi Sharan.

The matter personified in the dissertation has not been submitted for the award of any other degree or diploma.

  
**Supervisor**

**Dr. Aditi Sharan**

  
**Dean**

**Prof. Sonajharia Minz**

School of Computer & System Sciences  
Jawaharlal Nehru University  
New Delhi -110067

School of Computer & System Sciences  
Jawaharlal Nehru University  
New Delhi-110067

Prof. Sonajharia Minz  
Dean  
School of Computer & Systems Sciences  
Jawaharlal Nehru University  
New Delhi-110067

*Dedicated to my beloved parents*

## Acknowledgement

---

At the very outset I would like to thank Almighty God for all the favors He Showered upon me throughout my life.

With a deep sense of gratitude, I wish to express my sincere thanks to my guide/supervisor, **Dr. Aditi Sharan**, for her immense help in planning and executing this Master's Dissertation work in time. She consistently stood by me in all my difficult times, helping me to do my research fruitfully, giving valuable suggestions. In all respect, I am grateful to her for the time she has spent with me in discussions, or finding out the ways for me whenever I was stuck in understanding things by not only guiding me to find the way but also by searching all relevant materials for me. It would be impossible for me to come out successfully without her constant guidance.

I also express whole hearted thanks to my friends and classmates for their care and moral supports. The moments, I enjoyed with them during my M. Tech course will always remain as a happy memory throughout my life.

I owe a lot to my parents for their constant love and support. They always encouraged me to have positive and independent thinking, which really matter in my life. I would like to thank them very much and share this moment of happiness with them.

Last but not the least I am also thankful to entire faculty and staff of SC & SS for their unselfish help, I got whenever needed during the course of my work.

*Anupama*  
**Anupama Pandey**

## Abstract

---

The explosive growth and popularity of World Wide Web has resulted in a huge amount of information sources on the internet. Information Retrieval systems allow the users to retrieve relevant documents from the web. With the availability of popular search engines such as Yahoo, Google, MSN, one can easily conduct information search. However due to heterogeneity and lack of structure of Web information sources, access to this huge collection of information has been limited to browsing and searching. Web based browsing and searching paradigm is not convenient for locating and extracting specific information from Web documents. This is where information extraction comes into picture. *Information extraction may be defined as a type of concept extraction that automatically recognizes significant vocabulary items from specified documents.*

Structured data extraction is a type of information extraction in which semi-structured documents are transformed into a structured database (generally tabular form). This structured information in tabular form facilitates the user to extract the relevant information easily as compared to web based browsing and searching.

Due to large size and heterogeneous nature of web data, it is not possible to develop well defined rules for extracting relevant information. There is also a term called hidden web. 80% of the web is already in the hidden web form. The content of the hidden web is not part of the surface web, which is indexed by standard search engines. Hence we can not find the actual content of the hidden web through search engines. Web pages of the hidden web are generated on the fly from some databases based on user requests. Extracting information from such pages is dealt with SDE techniques.

In absence of well defined rules and hidden information, machine learning becomes a natural choice for extracting information from web documents. In this work I have explored the use of machine learning techniques for structured data extraction(SDE). These techniques include: inductive learning, active learning and automatic training approaches. Further I have proposed architecture for applying HMM for structured data extraction. Some experiments have been performed on the basis of

proposed model. The results are encouraging and indicate that use of HMM can be explored for SDE. But defining proper topology and providing proper training is a challenging task. This leaves a lot of scope exploring use of HMM for structured data extraction.

I observe that structured data extraction is still an open problem and is a very challenging task due to complexity of web pages, heterogeneity in data, lack of standards, dynamically changing resources and other related problems. There are many important issues that have to be taken care of, while applying machine learning for SDE. I have tried to highlight these issues. The work provides a base for applying machine learning techniques for structured data extraction.

## Table of Contents

	Page No
<b>Declaration</b>	<b>i</b>
<b>Certificate</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Abstract</b>	<b>v-vi</b>
<b>Table of Contents</b>	<b>vii-ix</b>
<b>List of Figures</b>	<b>x</b>
<b>Chapter 1: Introduction to Information Extraction</b>	<b>1-9</b>
1.1 Introduction	1
1.1.1 IR Versus IE	2
1.1.2 IE Definitions	2
1.1.3 IE Tasks	3
1.2 Evaluation Metrics for IE	3
1.3 IE Methodology	4
1.3.1 Knowledge Engineering Method	4
1.3.2 Automatic Training Method	5
1.4 Types of Inputs For IE	5
1.4.1 IE from Free Text	6
1.4.2 IE from Structured Text	7
1.4.3 IE from Semi-Structured Text	7
1.5 Outline Of Dissertation	9
<b>Chapter 2: Structured Data Extraction (SDE)</b>	<b>10-21</b>
2.1 Introduction	10
2.1.1 Need Of SDE	10
2.1.2 SDE Task	11
2.1.3 Types of Data Rich Pages	11
2.2 Basic Data Model	12
2.3 HTML Encoding Scheme	13



2.4	Web page Representation	14
2.4.1	EC Tree	14
2.4.2	DOM Tree	16
2.5	SDE Approaches	18
2.5.1	Knowledge Engineering Approach	18
2.5.1.1	Manual	
2.5.1.2	Wrapper Induction	20
2.5.2	Automatic Training Approach	20
2.5.2.1	Hidden Markov Model	
<b>Chapter 3: Wrapper Induction</b>		<b>22-32</b>
3.1	Introduction	22
3.2	Wrapper Induction	22
3.2.1	Web Page Representation	23
3.2.2	Types of Extraction Rules	23
3.2.3	Learning Methods	24
3.3	Inductive Learning	24
3.3.1	Example	25
3.3.2	Basic issues of Inductive Learning Approach	26
3.4	Instance-Based Learning (IBL)	26
3.4.1	Basic Approach	
3.4.2	IBL for Web Data Extraction	26
3.4.3	Example	28
3.5	Active Learning	29
3.5.1	Query Selection Strategy	30
3.5.2	Types of Active Learning	30
3.5.3	Advantage of Single View Over Multi-View Learning	31
3.6	Multi-View Active Learning for Wrapper Induction	31
<b>Chapter 4: Hidden Markov Model</b>		<b>33-45</b>
4.1	Basics of Hidden Markov Model(HMM)	33
4.1.1	HMM Definitions	33
4.1.2	Visible and Hidden Markov Model	34
4.1.3	General Architecture of a HMM	34
4.1.4	Markov property	35
4.1.5	HMM Parameters Definition	35

4.2	Three Canonical Problems associated with HMM	36
4.3	Solving Three Problems	37
4.3.1	Solving Problem 1	37
4.3.1.1	By Simple Calculation of Conditional Probability	37
4.3.1.2	Forward Backward Algorithm	38
4.3.2	Solving Problem 2	39
4.3.2.1	By Finding Most Likely State at any Time Step	39
4.3.2.2	Viterbi Algorithm	41
4.3.3	Solving Problem 3	42
4.3.3.1	Baum-Welch Algorithm	42
4.4	Selecting the Model Toplogy	44
4.4.1	Building Initial Model From Training Data	44
4.4.2	Generalization of the Model	44
<b>Chapter 5: Proposed Work and Experiment</b>		<b>46-58</b>
5.1	Problem Formulation and HMM for Solving Problem	46
5.1.1	Problem Formulation	46
5.1.2	Why HMM for Solving Problem	47
5.1.3	Topology Decision for HMM	47
5.2	Architecture of the Proposed System	50
5.3	Steps in Implementation	51
5.3.1	Data Extraction and Preprocessing	51
5.3.2	Training and Testing	52
5.3.2.1	Preparation of Training Data and Tagging	52
5.3.2.2	Training	53
5.3.2.3	Testing	54
5.4	Experiments and Results	54
<b>Chapter 6: Conclusion</b>		<b>59-60</b>
<b>Appendix A</b>		<b>61-63</b>
<b>Appendix B</b>		<b>64-65</b>
<b>References</b>		<b>66-69</b>

## List of Figures

Page No

---

<b>Figure 1.1 IE from Free Text (Input)</b>	<b>6</b>
<b>Figure 1.2 IE from Free Text (Output)</b>	<b>6</b>
<b>Figure 1.3 IE from semi-structured Text (Input)</b>	<b>8</b>
<b>Figure 1.4 IE from semi-structured Text (Output)</b>	<b>8</b>
<b>Figure 2.1 Sample Type Tree</b>	<b>12</b>
<b>Figure 2.2 Sample Web Page Content</b>	<b>15</b>
<b>Figure 2.3 HTML Source Code</b>	<b>15</b>
<b>Figure 2.4 Embedded Catalog Tree(EC tree)</b>	<b>15</b>
<b>Figure 2.5 Source Code for DOM Tree</b>	<b>17</b>
<b>Figure 2.6 Co-ordinate Representation for DOM Tree</b>	<b>17</b>
<b>Figure 2.7 Tree Representation for DOM Tree</b>	<b>17</b>
<b>Figure 2.8 DOM Tree Representation</b>	<b>18</b>
<b>Figure 2.9 Categorization of IE Techniques</b>	<b>19</b>
<b>Figure 3.1 Example of Wrapper Induction</b>	<b>25</b>
<b>Figure 5.1 Generalized transition diagram for the proposed system</b>	<b>49</b>
<b>Figure 5.2 Architecture of the Proposed System</b>	<b>50-51</b>
<b>Figure 5.3 Dictionary</b>	<b>53</b>

### Introduction to Information Extraction

---

#### 1.1 Introduction

With the expansion of Web, users have gained access to a large variety of comprehensive information repositories. Search engines have become most important tool for searching information from Web. Search engines are a type of information retrieval system, which allow relevant documents to be retrieved from the Web based on the user's query. Information Retrieval is comparatively mature field in Web mining, though there are still open issues in this field. With information retrieval as base many systems have been developed which allow the user's to search and browse the information from the Web. One can therefore easily conduct *information search* through the web search engines. However due to heterogeneity and lack of structure of Web information sources, access to this huge collection of information has been limited to browsing and searching. **Web based browsing and searching paradigm is not convenient for locating and extracting specific information from Web documents. This is where information extraction comes into picture.** Normally depending on user's requirement the manners of mining web information can mainly be categorized into *information retrieval(IR) and information extraction(IE)* respectively. With respect to information retrieval, it may be viewed as a process of selecting documents from a collection, according to the presence of keywords assigned by an indexer, *while information extraction may be defined as a type of concept extraction that automatically recognizes significant vocabulary items from specified documents.*

Currently the function of most of the web search engines is more close to fundamental information retrieval in which the user inputs keywords and obtains the output through word matching process. If the user is more interested in information extraction, results returned by web search engines will be too rough to

fulfill the user's need due to the limited mining capability of search engines. For instance, if a user wants to perform data mining within defense-related field for finding how much a sub-marine normally costs. The user then inputs keywords as 'submarine cost' into web search engine '*Google (www.google.com)*' to start survey. The results being dug out, in addition to minor number of URLs focusing on 'submarine cost' will contain lots of web pages which do not really fall into the user's interested domain. As a result one normally has to put considerable effort to further review the web search outcomes, so as to filter out the unnecessary data. Such tasks can be very time consuming. Also the obtained information can be *less of-completeness and accuracy* because the user may unintendedly lose certain information hidden within the text.

Therefore more precise and automatic information extraction techniques are clearly required in order to achieve accurate information extraction in a smarter way.

### 1.1.1 IR versus IE

While IR aims to select relevant documents IE aims to extract relevant facts from the documents. IR views the text in a document just as a bag of unordered words, whereas IE is interested in *structure or representation* of a document. Hence in general, IE works at a finer granularity level than IR does on the documents. In other words, IR retrieves relevant documents from collections, while IE extracts relevant information from documents. Therefore the two techniques are complementary [19].

### 1.1.2 IE Definitions

**General Definition:** An information extraction system takes as input an unrestricted text and summarizes the text with respect to a prespecified topic or domain of interest. It finds useful information about the domain and encodes that information in a structured form, suitable for populating databases [21].

**Formal Definition:** An information extraction system transforms text into a structured format and thereby reduces the information in a document to a structure, which is generally a tabular [4].

### 1.1.3 IE Task:

Information Extraction Task is defined by its input and its output(extraction target). The input can be *unstructured documents* like free text that are written in natural language or the *semi-structured documents* that are pervasive on the web, such as tables or itemized and enumerated lists and correspondingly result of the information extraction process could be in the form of a *compression or summary* of the original text/ documents or could be a *structured database* [34].

## 1.2 Evaluation Metrics for Information Extraction:

In the field of information extraction, there are several types of accuracy measurements that can be used for evaluation the extraction performance. The most widely measurements are Precision, Recall, and F-measures each hold specific characteristics. These measurements for information extraction define a correspondence between the extracted items and facts within the documents.

### Precision:

It answers the question that, for every item in the extracted outcomes, if there is a corresponding fact in the documents.

$$\text{Precision} = A / (A + B) \dots \dots \dots (1.1)$$

Where A= number of extracted items matching the facts

B= number of wrong extracted items

### Recall :

Recall corresponds to question that, for every fact in the documents, if there is a corresponding item shown in extracted outcomes.

$$\text{Recall} = A / (A + C) \dots \dots \dots (1.2)$$

Where A is the same as above

C=number of facts failing to be extracted

**F-measure:**

Low recall can be fixed by increasing the redundancy of the corpus, and low precision can be improved by adding more constraints in the system processing loop. If some one is interested in taking care of both recall and precision, F-measure can be used:

$$\text{F-measure} = \frac{2 * P * R}{(P + R)} \dots\dots\dots(1.3)$$

Where P=precision, R=recall-rate

### 1.3 IE Methodology (IE Techniques)

Information extraction techniques can broadly be divided into two main categories [1].

- ✚ Knowledge Engineering Method
- ✚ Automatic Training Method

#### 1.3.1 Knowledge Engineering Method

This approach is more close to rule-based techniques. In this approach, a person (knowledge engineer) who is familiar with the IE system and the formalism for expressing rules for that system, writes rules for the IE system components that mark and extract the needed information. In this approach the *skills of the knowledge engineer* play a large factor in the level of performance that will be achieved by the overall system. In addition to *required skills* and *detailed knowledge* of a particular information extraction system this approach requires a *lot of labor* as well.

Building a high performance system is usually an iterative process, where a set of rules are written, system is run over a training corpus of texts and the system is examined to see whether the rules under and over-generate. The knowledge

engineers make appropriate modifications to the rules and iterate the process. *Wrapper Induction approach* [23] is one of the knowledge engineering approach.

### 1.3.2 Automatic Training Method

This approach is quite different. Following this approach it is not necessary to have someone on hand with the *detailed knowledge* of how the information extraction system works or how to write rules for it. *It is necessary only to have someone who knows enough about the domain and the task to take corpus of data and annotate the data appropriately for the information being extracted.*

Once a suitable training corpus has been annotated , a training algorithm is run resulting in information extraction that a system can employ in analyzing novel data. i.e rather than focusing on producing rules , it focuses on producing training data. *Corpus statistics* are then derived automatically from training data, and used to process novel data. Examples of Automatic Training includes *Pattern Matching using Hidden Markov Model, Tree Matching* [23]etc.

## 1.4 Types of Inputs for IE

Originally the aim of information extraction was to develop practical system which could take short natural language texts and extract a limited piece of information from them. But now the information extraction system are being developed not only for natural language text but for structured and semi-structured text also. Hence input to the information extraction system can be broadly divided into three main categories:

- ✚ IE from Free Text
- ✚ IE from Structured Text
- ✚ IE from Semi-Structured Text



### 1.4.1 IE from Free Text:

Information extraction for free text generally uses *natural language techniques*, and the extraction rules are typically based on patterns involving syntactic relations between words or semantic classes of words. Several steps are required including syntactic analysis, semantic tagging, constructing recognizers for domain objects such as person and company names and generating extraction rules. The rules are patterns that can be hand coded or generated from training examples annotated with the right label by a human expert [4]. Figure 1.1 represents the input in the form of free text and figure 1.2 represents its corresponding output in structured format.

#### Example:

**Input (Free Text)**  
**Posting from Newsgroup**  
Telecommunications, SOLARIS Systems  
Administrator, 38-44k, Immediate need  
Leading telecommunications y :firm in need of an energetic  
Individual to fill the following position in the Atlanta office:  
    SOLARIS SYSTEMS ADMINISTRATOR  
    Salary: 38-44k with full benefits  
    Location: Atlanta Georgia, no relocation assistance provided

**Figure 1.1**

**Output**  
computer\_science\_job  
title: SOLARIS Systems Administrator  
salary: 38-44k  
city: Atlanta  
platform: SOLARIS  
area: telecommunications

**Figure 1.2**

#### 1.4.2 IE from Structured Text:

Structured text is defined as textual information in a database or file following a predefined and strict format. Such information can easily be extracted using the format description. Usually quite simple techniques are sufficient for extracting information from text provided that the format is known, *otherwise the format must be learned*.

#### 1.4.3 IE from Semi-Structured Text:

Semi-structured data are an intermediate point between unstructured collection of textual documents and fully structured tuples of typed data. Natural language processing (NLP) techniques will usually not work for semi-structured text, which seldom contains full sentences and at the same time simple rules used for rigidly structured text will not be sufficient. *For semi-structured text extraction patterns are often based on tokens and delimiters like for instance HTML-tags*. Syntactic and semantic information can only be utilized to a limited extent.

#### Web Documents:

Web pages are generally considered as semi-structured as they all contain some structuring information concerning display styles. But web pages are also categorized of different types. A web page that provides itemized information is *structured* if each attribute in a tuple can be correctly extracted based on some uniform syntactic clues, such as delimiters or the order of attributes. *Semi-structured web pages* may contain tuples with missing attributes, attributes with multiple values, variant attribute permutations and exceptions. A web page is *unstructured* if linguistic knowledge is required to extract the attributes correctly.

When it comes to extract information from web pages the same applied to web documents as to other semi-structured documents. The *organization* and *hyperlinking* of documents are important aspects when extracting information from web pages. Extraction information from web pages or online documents having semi-struformat is

called *STRUCTURED DATA EXTRACTION(SDE)* [4,21]. Figure 1.3 represents a sample web page as input and when similar information are extracted from different web pages and database is populated with this information it is called SDE and output will look like as in figure 1.4 in the next page.

### Example: Input (web page)

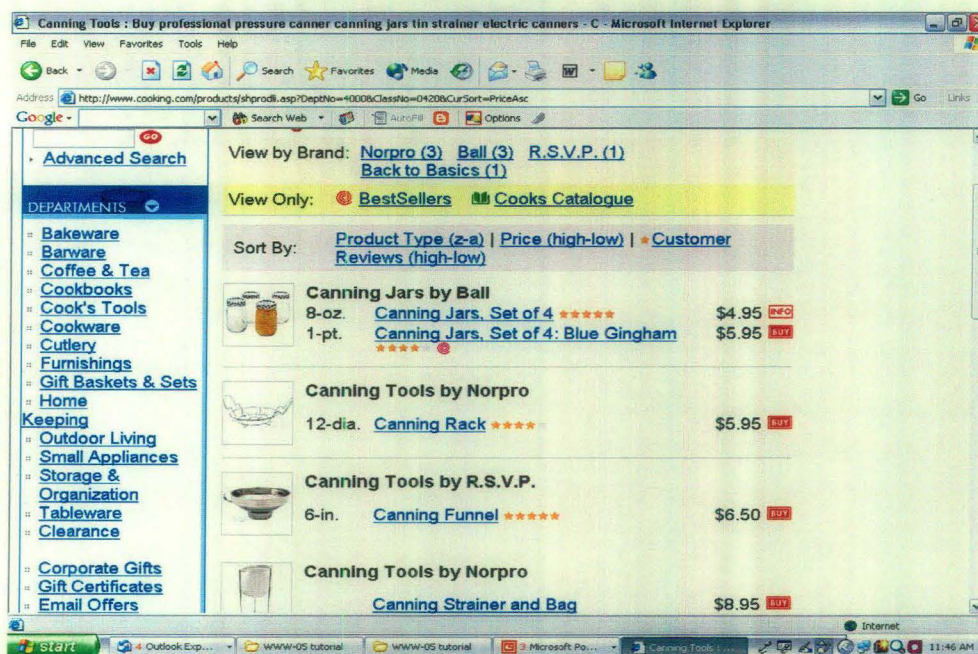


Figure 1.3

### Output:

IMAGE 1	Canning Jars By Balls	8-oz	*****	Canning Jars, Set of 4	\$ 4.95
IMAGE 1	Canning Jars By Balls	1-pt	***** @	Canning jars, Set 4:blue Gingham	\$5.95
IMAGE 2	Canning Tools by Norpro	12-dia	*****	Canning Rack	\$ 5.95

Figure 1.4

## **1.5 Outline of Dissertation**

This dissertation is organized as follows: In chapter 2 we have summarized various approaches that exist for structured data extraction and in chapter 3 various machine learning approaches that are suitable for structured data extraction are explained with examples. In chapter 4 all the basics and complete theory of Hidden Markov Model (HMM) is given because our work uses the approach of HMM. Basically Hidden Markov Model has been widely used for text information extraction but in chapter 5 in our proposed work and experiments we have explained how HMM can be applied for structured data extraction from semi-structured web pages. Lastly chapter 6 concludes the work.

## Structured Data Extraction

---

### 2.1 Introduction to Structured Data Extraction

Extracting information from web pages or online documents having semi-structured format is called Structured Data Extraction (SDE) [21]. In SDE an information extraction system transforms semi-structured documents into a structured database which generally as a tabular form. This chapter deals with the *various approaches that are used for structured data extraction* but before discussing these approaches first we need to explain what is structured data extraction task, its input and output, why we need SDE, which kind of web pages are used as input in SDE, basic data model and HTML encoding scheme of such input web pages and representation scheme of these web pages. After that we explain *why Machine Learning (ML) Approaches are suitable for SDE, over other approaches (manual, automatic)*. In the next chapter we explain these ML approaches in details.

#### 2.1.1 Need of SDE:

There is a term called the *Hidden Web or Deep Web*. 80% of the web is already in the hidden web. The content of the hidden web is not part of the surface, which is indexed by standard search engines. Hence we can not find the actual content of the hidden web through search engines. Web pages of the hidden web are generated on the fly from some databases based on user requests [13]. Extracting information from such pages is dealt with SDE techniques.

At the same time SDE enables to obtain and integrate data from multiple sources (websites and pages) to provide value-added services such as customizing Web Information Gathering, Comparative Shopping, and Meta search.

### 2.1.2 SDE Task:

Structured Data Extraction is a *Reverse Engineering Task* [4]. Generally before displaying data records of backend databases onto the web pages, some *data model and encoding schemes* are used to encode these data records. Reverse Engineering Task is that we need to develop the extraction system that recovers original data model and extract data from encoded data records. Input to the extraction system is Semi-structured web pages containing data records (*Structured Data*) formatted using HTML-tags. Extraction system extracts such data records and extraction target is a relation of tuples.

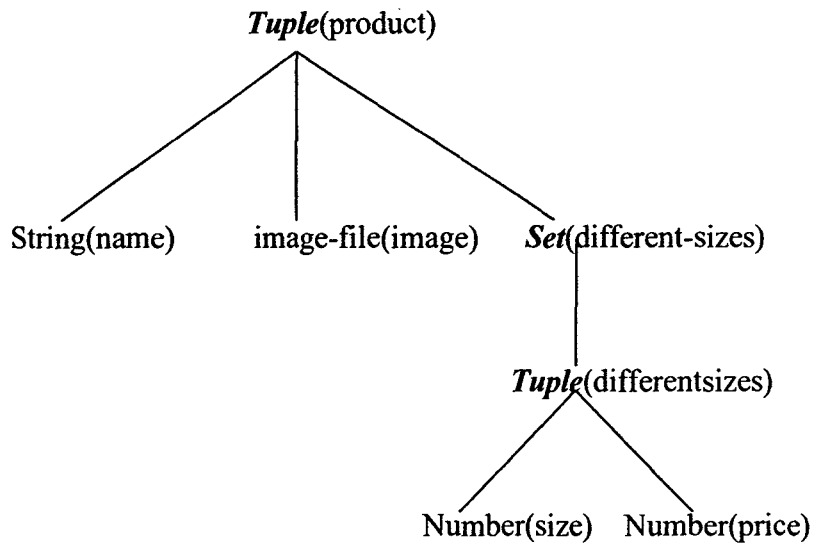
### 2.1.3 Types of Data-Rich Pages (input pages to SDE):

There are mainly two types of data rich pages that contain Structured Data (SD). Detailed pages and List pages [40]. A detailed page presents the detail of a single product like its name, image, price, purchasing information, specifications and customer rating etc. A list page usually contains a list objects. From a layout point of view we see list pages, as a contiguous region with more than one product formatted using the same template.

## 2.2 Basic Data Model:

Most of the web data can be modeled as nested relations that are typed objects allowing nested sets and tuples. These types are basic type, tuple type and set type. *Basic type* is similar to the type of an attribute in relational databases (string, integer etc). *tuple type* is similar to tuple and *set type* is similar to a set of tuples in context of a relational databases. These types can be represented as a tree (*type tree*).

Basic type is a leaf node; a tuple type is a tuple node with n-sub-trees in the tree. A set type is a tree rooted at a set node with one sub-tree. Attribute names are not included in the type tree.



**Figure 2.1 Sample Type Tree**

Labeling of a type tree given in the figure 2.1 is defined recursively:

- If a set node is labeled  $\emptyset$ , then its child node is labeled  $\emptyset.0$ , a tuple node.
- If a tuple node is labeled  $\emptyset$ , then its  $n$  children are labeled  $\emptyset.1$ ,  $\emptyset.2$ ,  $\emptyset.3$ ,  $\emptyset.4, \dots, \emptyset.n$ .

<b>Product</b> [ name: string; Image: image-file; Differentsizes: { [ size: string; Price: string; ]}]
---

Labels are generally considered as abstract names for types or attributes. In the above nested type the top-level tuple type is “product”, its three children are attributes: product.name, product.image, product.differentsizes. Here  $\emptyset.0$  labels a tuple node.

### 2.3 HTML Encoding Scheme:

Generally web pages are written in HTML consisting of plain texts, tags and links to image, audio and video files, and other pages. Most HTML tags work in pairs. Each pair consists of an **open tag** and a **close tag** indicated by  $< >$  and  $</>$  respectively. Within each corresponding tag-pair, there can be other pairs of tags, resulting in nested structures. Thus, HTML tags can naturally encode nested data. By nature HTML is used for formatting the data records. In order to include HTML encoding scheme for considering type information following points must be taken care of:

- There are no designated tags for each type as HTML was not designed as a data encoding language. Any HTML tag can be used for any type.
- For a tuple type, values (data items) of different attributes are usually encoded differently to distinguish them and to highlight important items.
- A tuple may be partitioned into several groups or sub-tuples. Each group covers a disjoint subset of attributes and may be encoded differently.

Based on these characteristics of the HTML language the HTML mark-up encoding of instances is defined recursively. *Encoding is generally done based on the type tree where each node of the tree is associated with an encoding function*, which will encode all the instances of the same type in the same way. *Encoding is done as follows:*

- For a leaf node of a basic type  $\emptyset$ , an instance  $c$  is encoded with

$$\text{Enc}(\emptyset, c) = \text{OPEN-TAGS } c \text{ CLOSE-TAGS.}$$

Where OPEN-TAGS is a sequence of open HTML tags and CLOSE-TAGS is a sequence of close HTML tags.

- A tuple node of  $n$  attributes or children,  $[\emptyset.1, \emptyset.2, \dots, \emptyset.n]$ , the attributes are partitioned into  $h (\geq 1)$  groups



$\langle \emptyset.1, \dots, \emptyset.e \rangle, \langle \emptyset.(e+1), \dots, \emptyset.g \rangle, \dots, \langle \emptyset.(k+1), \dots, \emptyset.n \rangle$

An instance  $[V_1, V_2, \dots, V_n]$  is encoded with

$$Enc(\emptyset: [V_1, V_2, \dots, V_n]) = OPEN-TAGS_1 enc(V_1) \dots enc(V_e) CLOSE-TAGS_1, \\ OPEN-TAGS_2 enc(V_{e+1}) \dots enc(V_g) CLOSE-TAGS_2, \\ \dots \\ OPEN-TAGS_h enc(V_{k+1}) \dots enc(V_n) CLOSE-TAGS_h$$

- For a set node labeled  $\emptyset$ , an non-empty set instance  $\{e_1, e_2, \dots, e_n\}$  is encoded with

$$Enc(\emptyset: \{e_1, \dots, e_n\}) = OPEN-TAGS enc(e_{j_1}) \dots enc(e_{j_n}) CLOSE-TAGS$$

This mark-up encoding does not cover all cases in web pages. But still they cover most of the representation.

**2.4 Web Page Representation:**

Since web pages are intended to be human readable there are some *common conventions* for structuring HTML pages. For instance, the information on a web page often exhibits some *hierarchical structure*; semi-structured information is often presented in the form of list of tuples, with explicit separators used to distinguish different different elements. With these conventions in mind, generally two representations of web pages have been developed: *Embedded Catalog Tree (EC tree)* and *Document Object Model Tree(DOM Tree)*.

**2.4.1 Embedded Catalog Tree:**

Due to hierarchical structure of the web data embedded in an HTML page, a web page can be modeled using *hierarchical tree structure* called EC tree [17]. EC tree is based on the type tree. The root of the tree is the document containing the whole token sequence of the page, and the content of each child node is a subsequence of the sequence of its parent node. Basically the leaves are the items of interest for the

user(relevant data). One minor limitation of using the EC tree for web page representation is that EC-tree for a web page is to be specified by the user which is labor intensive task because user has to observe each web page and need to build EC tree for each web page manually. Figure 2.2 is a sample web page content and figure 2.3 represents its corresponding HTML source code and figure 2.4 represents EC tree for it.

**Example:** A sample web page content is

Restaurant Name : Good Noodles

- 205,Willow, Glen ,Phone 1-773-366-1987
- 25 oak, Forest ,Phone (800) 234-7903
- 324 Hallstead St.,Chicago,Phone 1-800-996-5023
- 700 Lake St., Oak Park , Phone (708) 798-0008

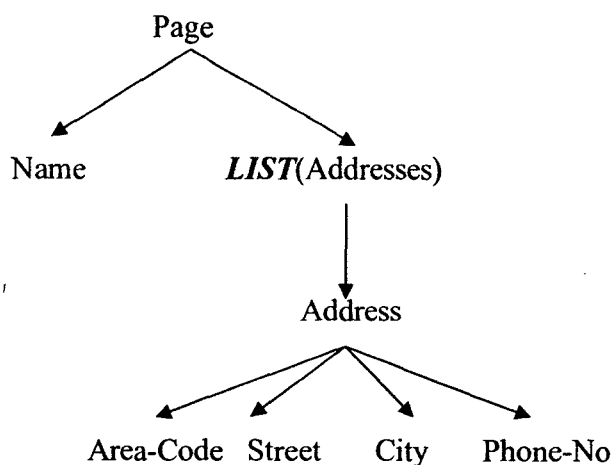
**Figure 2.2**

Its source HTML code

1: <p>Restaurant Name : <b> Good Noodles</b><br><br>  
 2: <li>205 Willow, <i> Glen </i>,Phone 1-<i> 773 </i> -366-1987</li>  
 3 :<li> 25 Oak,<i> Forest </i> ,Phone (800) 234-7903 </li>  
 4 :<li> 324 Halsted St.,<i> Chicago </i>,Phone 1-<i> 800 </i> -996- 5023</li>  
 5 :<li> 700 Lake St.,<i> Oak Park</i>, Phone : (708) 798-0008</li> </p>

**Figure 2.3**

Its EC tree



**Figure 2.4**

## 2.4.2 Document Object Model (DOM)Tree:

A DOM tree is an ordered tree, where each node is either an element node or a text node. An element node has an ordered list of zero or more child nodes, and contains a string-valued *tag* (such as table, h1 or li) and also zero or more string-valued *attributes* (such as href or src). A text node is normally defined to contain a single *text string*, and to have no children. DOM tree building from input pages is a necessary step for many data extraction algorithms [4,39]. There are generally two methods for building DOM tree :

### 1. Using Tags Alone:

Most HTML tags work in pairs. Each pair consists of an open tag and a close tag (indicated by `< >` and `</>` respectively). Within each corresponding tag-pair, there can be other pairs of tags, resulting in a nested structure. Building a DOM tree from a page using its HTML code is natural. In the tree, each pair of tags is a node, and the nested pairs of tags within it are the children of the node. Two tasks need to be performed:

**HTML code cleaning:** Some tags do not require close tags (e.g., `<li>`, `<hr>` and `<p>`) which create problems in identifying levels of hierarchy. Hence, additional close tags should be inserted to ensure all tags are balanced. Ill-formatted tags also need to be fixed. Such error tags are usually close tags that cross different nested blocks, e.g., `<tr> ... <td> ... </tr> ... </td>`, which can be hard to fix if multiple levels of nesting exist. There are open source programs that can be used to clean up HTML pages. One popular program is called **tidy** (available at <http://tidy.sourceforge.net/>).

**Tree building:** Nested blocks of the HTML tags in the page is followed to build the DOM tree.

### 2. Using Tags and Visual Cues:

Instead of analyzing the HTML code to fix errors, rendering or visual information (i.e., the locations on the screen at which tags are rendered) can be used to infer the structural

relationship among tags and to construct a DOM tree. This method leads to more robust tree construction due to the high error tolerance of the rendering engines of Web browsers (e.g., Internet Explorer). As long as the browser is able to render a page correctly, its tag tree can be built correctly.

In a Web browser, each HTML element is rendered as a rectangle. The visual information can be obtained after the HTML code is rendered by a Web browser. A DOM tree can then be constructed based on the nested rectangles.

- First the four boundaries of the rectangle of each HTML element are obtained by calling the rendering engine of a browser, e.g., Internet Explorer.
- Sequence of open tags are followed and containment checks are performed to build the tree (Containment check means checking if one rectangle is contained in another).

Figures 2.5,2.6,2.7 given below represent HTML code segment ,its corresponding resulting coordinate and resulting DOM tree.

```

<table>
  <tr>
    <td> data1
  </td>
    <td>data2 </td>
  </tr>
  <tr>
    <td> data3 </td>
    <td> data4</td>
  </tr>
</table>

```

Figure 2.5

left	right	top	bottom
100	300	200	400
100	300	200	300
100	200	200	300
200	300	200	300
100	300	300	400
100	200	300	400
200	300	300	400

Figure 2.6

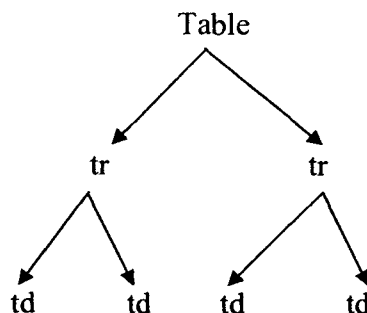


Figure 2.7

DOM tree given below of example given in section 2.4.1

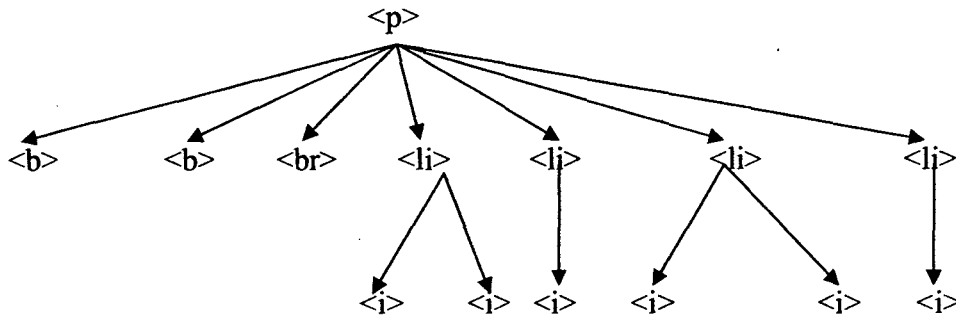


Figure 2.8

Advantage of building a DOM tree for a web page over building an EC tree is that no user effort is required to build the DOM tree because there are already available programs (Jscript, DHTML) and tools( tools take URL of the web page as input and create a DOM tree for the whole page and at the same time may create a DOM tree for a particular segment specified by the user)

## 2.5 Structured Data Extraction(SDE) Approaches:

In chapter one it is mentioned that there are two types of Information Extraction Methodology: Knowledge Engineering Method and Automatic Training Method [10] both of them are applied to structured data extraction also.

### 2.5.1 Knowledge Engineering Method:

It is more *closed to rule-based techniques* in which the experts in the designated knowledge domain write rules. This method has basically two approaches:

#### 2.5.1.1 Manual Approach

By observing the web-pages and their source codes (gaining the knowledge of the domain) the human programmer (knowledge engineer) writes program (rules) to extract the target data. However this approach is not scalable to a large number of sites because

manually observing large number of web pages for writing rules is very difficult and time-consuming.

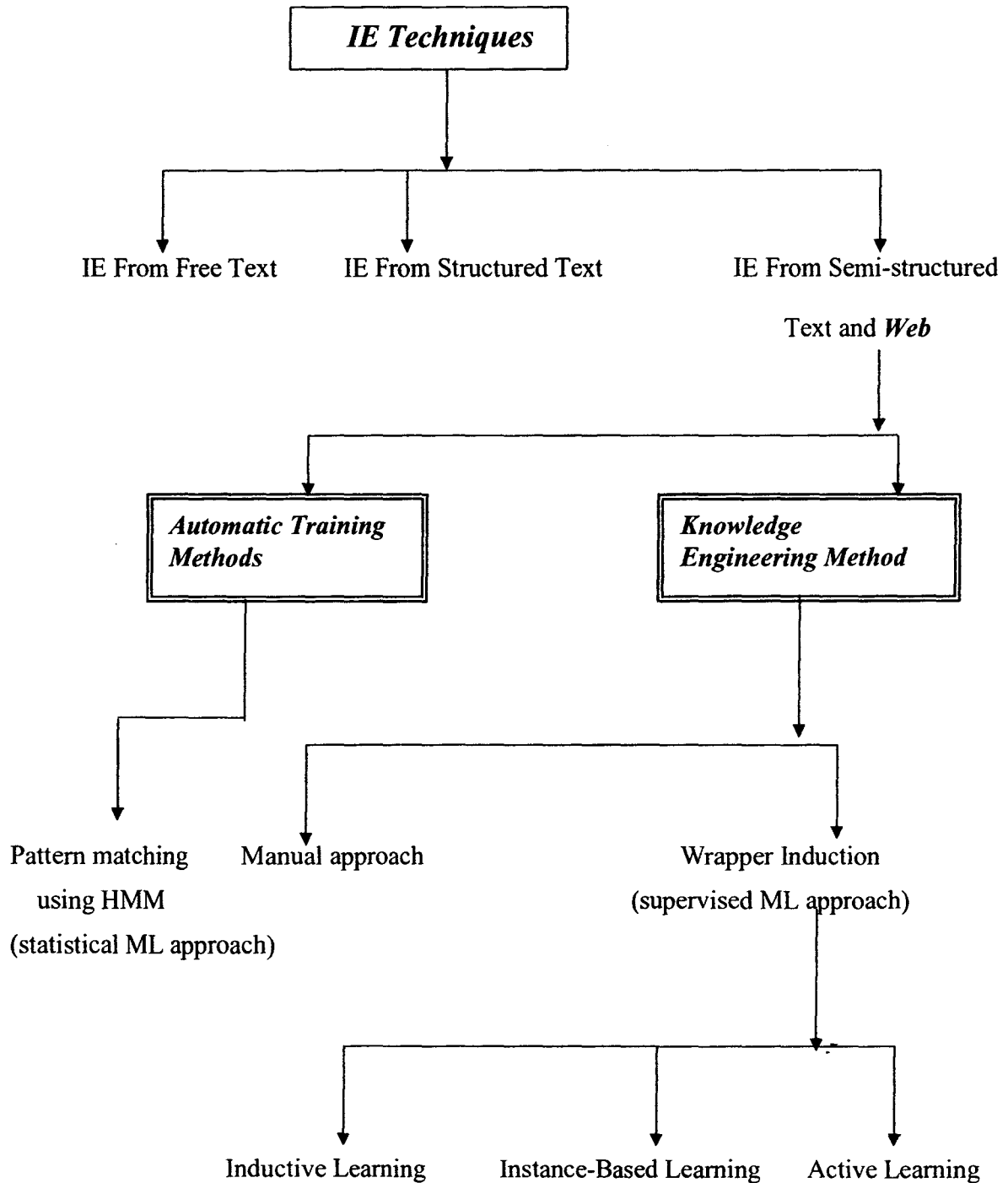


Figure : 2.9 categorization of IE techniques

### 2.5.1.2 Wrapper Induction Approach

This is the supervised learning approach and is semi-automatic. In this approach a set of extraction-rules is learnt from a collection of manually labeled pages. The rules are then employed to extract target data items from other similarly formatted pages.

There are different machine learning (ML) algorithms that can be used to learn wrappers (rules). In chapter 3 we will discuss these ML algorithms and advantages of one method over the other

### 2.5.2 Automatic Training Method:

Automatic training method requires someone who knows enough about the domain and in this method the task is to take corpus of data and annotate the data appropriately for the information being extracted. Once annotation of suitable training corpus has been done a training algorithm is run resulting in information extraction. This approach is more *flexible and robust* than knowledge engineering approach because it is too *time-consuming* in generating rules in rule-based approaches and when the information extraction domain changes rules may need to be *re-written*. *Hidden Markov Model (HMM)* is one of the most popular model in this method.

#### 2.5.2.1 Hidden Markov Model (HMM):

Hidden Markov model is a powerful *statistical machine learning technique* that is just beginning to gain use in information extraction tasks. When compared with many other techniques used in statistics-based methods, the HMM has a strong theoretical foundation with a well established training algorithm and HMM can process data quite robustly.

HMM for information extraction is the problem of locating textual sub-segments that answers a particular information need. Using HMM, the extraction system that needs to be modeled is assumed to be a *Markov Process* with unknown (hidden) parameters and the hidden parameters are found from the observable parameters. Estimating hidden parameters using various *parameter estimation*

*algorithm* is called training of HMM. Once HMM is trained, *Viterbi algorithm* is used for extracting information from web pages. This is called testing.

Details of HMM is explained in chapter 4 and its implementation details for structured data extraction is given in chapter 5.



TH-19207



# Wrapper Induction

---

### 3.1 Introduction

Wrapper Induction is one of the Knowledge Engineering Approach in which a set of extraction rules is learned from a collection of manually labeled pages and then learned rules are applied to extract target data from other pages. In this chapter first we describe what is wrapper and wrapper induction and then in subsequent sections web page representations and learning methods are described that are necessary for wrapper induction. Since inductive learning method is not suitable for wrapper induction therefore it is also explained how Active learning and Instance-based learning can improve learning performance.

#### **Wrapper:**

- ✦ A wrapper consists of a set of rules together with the code to apply those rules [20].
- ✦ It is a piece of software that enables a semi-structured web source to be queried as if it were a database. (as defined in database community) [21].
- ✦ In the web environment, its purpose should be to convert information implicitly stored as HTML document into information explicitly stored as a data structure for further processing [17].

### 3.2 Wrapper Induction:

In wrapper induction system, user marks the target items in a few training pages, system learns extraction rules from these pages and these rules are then applied to extract items from other similar pages. Following points needs to be considered in wrapper induction:

- Web page Representation
- Types of Extraction Rules
- Learning Methods

### 3.2.1 Web Page Representation:

In the wrapper induction approach web page is represented as embedded catalog tree( EC tree) as mentioned in chapter 1. It is a hierarchical tree structure and for each web page embedded catalog tree is specified by the user.

### 3.2.2 Type of Extraction Rules:

If we consider the basic approach of learning the classifier, it learns to relate attribute values of data set to that of class labels. In our approach of wrapper induction, the wrapper needs to learn a set of extraction rules that enable the wrapper how to locate information on web pages. Since a web page is represented as a hierarchical tree structure which supports hierarchical information extraction hence instead of learning a single extraction rule that extract all the items of interest and become more complex when the depth of the tree increases several simpler rules are learnt that makes the hard problem of information extraction into a series of easier extraction tasks.

Some of the extraction rules are start rule, end rule, list iteration rule, and disjunctive rule. Extraction is done using two rules, **start rule** and **end rule**. The start rule identifies the beginning of the node and the end rule identifies the end of the node. For a list node, **list iteration rules** are needed to break the list into individual data records (tuple instances). To extract items from the data records, data extraction rules are applied to each record. All the rules are learned during wrapper induction. Extraction rules are based on the idea of **landmark** and **landmark automata**. Each landmark is a sequence of consecutive tokens and is used to locate the beginning or the end of target item and landmark automata can be collection of landmarks to locate the target item. Rules can be in the form of skipTo containing landmark within it, which means arguments to skipTo() is a landmark and group of

skipTo() defined that must be applied in a pre established order is called landmark automata. Class of landmark automata that corresponds to disjunctive rules are called simple landmark grammar (SLG).

### 3.2.3 Learning Methods:

Wrapper induction is a supervised learning approach. Supervised learning is also called inductive learning in machine learning. In standard supervised learning task, a standard dataset is used in the learning task which is generally represented as relational table having a fixed attribute-value pair representation. It has one special attribute called the class attribute. Class labels are provided in the data. The objective of supervised learning is given the value of the attributes and class labels, how to learn the classifier (learner) to relate values of attributes and classes. Once the classifier is learnt it is used to classify the unlabeled examples.

- Inductive learning
- Instance-Based Learning
- Active Learning

## 3.3 Inductive Learning

Inductive learning is the task of computing from a set of examples of some unknown target concept, a generalization that explains the observations. *The idea is that a generalization is good if it explains the observed examples and makes accurate predictions when unseen examples are encountered [36].*

Extraction rules are induced using *sequential covering algorithm*. The basic approach is that given a set of examples one rule is learnt, all the *positive examples*(examples that contains items of interest or target items are called positive examples and examples that do not contain items of interest are called negative examples) learnt by that rule are removed and again a rule is learnt for the remaining examples. This procedure is iterated as many times as desired to learn a *disjunctive set of rules (union of more than one rule)* that together cover any desired fraction of positive examples.

### 3.3.1 Example

For the web page example and its corresponding EC tree given in the chapter 2 section 2.4 the Wrapper need to perform the following steps:

1. Wrapper uses the start rule *SkipTo*(**<br><br>**), and the end rule *SkipTo*(**</p>**) to identify the entire list of addresses.
2. Iterate through the list to break it into four individual records. To identify the beginning of each address, the wrapper starts from the first token of the parent and repeatedly apply the start rule *SkipTo*(**<li>**) to the content of the list. Each successive identification of the beginning address starts from where the previous one ends. Similarly, to identify the end of each address, it starts from the last token of its parent and repeatedly apply the end rule *SkipTo*(**</li>**).

Once each address record is identified or extracted, wrapper extract the area code from it.

Due to variations in the format of area codes (some are in italic and some are not), wrapper need to use disjunctions. In this case, the disjunctive start and the end rules are respectively

**R1: either** *SkipTo*(**(**)  
or *SkipTo*(**-<i>**)

**R2: either** *SkipTo*(**)**)  
or *SkipTo*(**</i>**)

In a disjunctive rule, the disjuncts are applied sequentially until a disjunct can identify the target node.

Figure 3.1

### 3.3.2 Basic Issues of Inductive Learning Approach:

It has been found that there are many problems in applying sequential covering approach for wrapper extraction. Following are some major issues, which deal with application of inductive learning approaches for wrapper extraction:

- 1) The initial set of labeled training pages may **not** be **fully representative** of the **templates** of all other pages. For the pages that follow templates not covered by the labeled pages, learned rules will perform poorly. The usual solution to this problem is to label more pages as more pages should cover more templates.

However, manual labeling is labor intensive and time consuming which increases user involvement. Instead of labeling a large number of pages, **Active Learning** approach may be used. Active learning has the ability to detect **most informative examples** (having different format than already labeled examples). Third solution is to apply **Instance-based Learning**. Again instead of labeling a large number of pages only one page is labeled by the user in the beginning. Only when a new page can not be extracted it needs labeling. This avoids unnecessary labeling and also ensures that all different templates are covered.

- 2) Learning extraction rules becomes harder for complex pages because it increases the **complexity** of the rule.
- 3) Even if there are minor changes in the format of the data, this approach needs wrapper maintenance (wrapper verification and wrapper repair) which are very difficult. Active learning has the capability of **incremental learning** (instead of re-learning the rules, learnt rules are modified).

### **3.4 Instance-Based Learning:**

#### **3.4.1 Basic Approach:**

A set of labeled examples (more than one) is stored first (no inductive learning is applied). When a new instance is presented, it is compared with the stored instances to produce the results. It uses the **Euclidean Distance** or **text similarity measures** to compare the new example with the stored examples. Classical instance based learning approaches can not be directly applied for web data extraction because they still need an initial set of **many labeled examples**, thus have the same problem as inductive learning.

### 3.4.2 Instance-Based Learning For Web Data Extraction:

A random page (example) is selected for labeling by the user. The user labels the items of interest in the page. A sequence of **consecutive tokens before each** labeled items (prefix string of the item) and a sequence of consecutive tokens after the labeled item (suffix string of the item) are stored. (the prefix and suffix strings of all target items form a Template). The system then start to extract items from new page d. If some target items from d can not be extracted (**can not be uniquely identified**). It is sent to the user for labeling [40].

#### Sufficient match Technique:

In this approach instead of comparing the items itself their prefix (k-consecutive tokens right before the labeled item) and suffix (k-consecutive tokens right after the labeled item) strings are compared. It matches a **minimum number of tokens** in the new example with the prefix tokens and suffix tokens in the labeled example so that the target item in the new example is **uniquely identified**.

#### Prefix match score:

$P = \langle p_1, p_2, \dots, p_k \rangle$  be the prefix string of an item in a labeled page and A be the token string of page d (to be extracted). A sub-string of  $A (= \langle a_1, a_2, \dots, a_i, a_{i+1}, \dots, a_{i+h}, \dots, a_n \rangle)$  matches P with a match score of h ( $h \leq k$ ), if  $p_k = a_{i+h}$ ,  $p_{k-1} = a_{i+h-1}, \dots, p_{k-h+1} = a_{i+1}$  and ( $p_{k-h} \neq a_i$  or  $h=k$ ).

#### Suffix match score:

$P = \langle p_1, p_2, \dots, p_k \rangle$  be the suffix string of an item in a labeled page and A be the token string of page d (to be extracted). A sub-string of  $A (= \langle a_1, a_2, \dots, a_i, a_{i+1}, \dots, a_{i+h}, \dots, a_n \rangle)$  matches P with a match score of h ( $h \leq k$ ), if  $p_1 = a_{i+h}$ ,  $p_2 = a_{i+h-1}, \dots, p_h = a_{i+1}$  and ( $p_{h+1} \neq a_{i+h+1}$  or  $h=k$ ).

### 3.4.3 Example:

#### Source Code of a randomly selected page is

```

.....<b> name <br><br> .....
<table class="container id="head"> <tr>
<td valign="middle" class="frnavlink"><i><b> Price </td></td></td>
</td></td> <tr height="2" bgcolor="#046AA1">.....<br><u><b>
<img src='/ximages/products.jpg'> </b></u>.....

```

Figure 3.2

Suppose user selects a random page for labeling and its source code is as above. User is interested in extracting three items from each product, namely, name, image, and price. Then he labels these items in the above source code of the page. The template ( $T_j$ ) for a labeled example  $j$  is represented with:  $T_j = \langle \text{pat}_{\text{name}}, \text{pat}_{\text{price}}, \text{pat}_{\text{image}} \rangle$ . Each  $\text{pat}_i$  in  $T_j$  consists of a prefix string and a suffix string of the item  $i$ . For example, if the product price is embedded in the following source code `<table><tr><td><i><b> price </td><td></td></td></td>` (here  $k=3$  where  $k$  is the number of tokens in the prefix and suffix strings stored by the system so that it need not have to refer back to the original page during extraction). Then we have

```

pat_price = (price, prefix:(<table><tr><td><i><b>), suffix:(</td><td></td></td></td>))

```

Consider following prefix string of price in the labeled example

```

Prefix: <table> <tr> <td> <i> <b> price

```

Code Fragment of an Unlabeled Example  $d$  is

```

.....<td><font> <b>.....
.....<td><font> <i><b> .....
..... <table><tr><td><i><b> $25.00 ..
.....<br><font> <i><b> .....

```

We need to Extract Price( \$25.00) from the above unlabeled example

**After applying Sufficient Match technique:**

Prefix: <table> <tr> <td> <i> <b> price
---

	--	<b>		prefix match score=1	
	--	<i>	<b>	prefix match score=2	
<table>	<tr>	<td>	<i>	<b>	prefix match score=5( <i>best match score</i> )
	--	<i>	<b>	prefix match score=2	

We see that there are four sub-strings in d that have matches with the prefix string of price. These are shown in four rows. “-” means no match. The highest number of matches is 5, which is the best match score for this prefix string. We do not need the **best match score**. Instead, we only use the **sufficient match score** that can uniquely identify the target item in the new page. In the example, the sufficient match score is 3, which is from the match of <td><i><b>. This match can uniquely identify the item price. Additional item matches, i.e. <table> <tr> are not needed. We note that <b> is not sufficient match because there are 4 such matches in the unlabeled page. Similarly <i><b> match is not sufficient because there are 3 such matches in the new page. Thus, we cannot use <i><b> to identify a target item price because it is not unique. For a page, if the sufficient match score for the prefix or suffix match cannot be found for a particular target item, the extraction for the item fails, and the page is given to the user to label.

### 3.5 Active Learning:

Active learning is (**iterative supervised learning**) another learning approach. Active learning aims at minimizing the amount of labeled training data required to learn the target concept. Active learning detects the **most informative examples** in the domain and ask the user to label only them.



### 3.5.1 Query Selection strategies:

Informative examples that are chosen for labeling are called **queries**. There are basically two types of query selection strategy.

**a. Query Construction:** Queries an example that is constructed by setting the value of each attribute so that the resulting query is as informative as possible. *Artificially constructing* a query that is most informative may have high risk because user may not be able to label such query because it does not correspond to a real world entity [16].

**b. Selective Sampling:** Query must be chosen from a *given working set of unlabeled examples*. It is most popular active learning approach because in the real word domain it is possible to obtain a large number of unlabeled example than the query construction.

### 3.5.2 Type of Active Learning :

#### a. Single-view Active Learning :

In the single-view learning problem the learner have access to the entire set of features in the domain. In this, the problem is finding a query(an informative example) that splits the version-space into half(eliminate half of the hypotheses or rules consistent with the current training set).This is also called *Selective sampling approach*.

#### b. Multi-view Active Learning (Co-Testing):

In multi-view setting one can partition the domain's features in disjoint subsets (views), each of which is sufficient to learn the target concept. In multi-view domain, the problem of learning is to learn a classifier for each view and applies them to a pool of unlabeled examples and selects a query based on the *degree of disagreement among the classifiers*. If classifiers for each view are properly learnt the target concept in each view must agree. *More the degree of disagreement among the classifiers more the informative example is and vise-versa*. This approach is also called **Co-Testing approach**. Unlabeled examples for which classifiers of two views predict different

labels are called *contention points*. Contention points are extremely informative examples, because whenever the two classifiers disagree, at-least one of them must be wrong.

### 3.5.3 Advantages of Multi-view over Single-view Active learning:

- ✚ Co-testing reduce the hypothesis space faster than single-view active learning, that is *degree of convergence* of co-testing is more than its counterpart.
- ✚ Single-view active learning algorithms require more queries than Co-Testing to learn the target concept.
- ✚ By querying only contention points, co-testing always ask for the label of an example on which at-least one of the classifier is wrong (*detecting from mistakes*).

### 3.6 Multi-view Active Learning For Wrapper Induction:

It is two step iterative process that

1. Uses few labeled examples to learn a rule in each view.
2. Queries (ask the user to label) examples on which the rules predict a different label.

Such queries are highly informative because they correct mistakes made by one of the rule: whenever the rules disagree atleast one of them must be wrong.

#### 3.6.1 Example:

*Labeled example is*

Name: KFC <p> Phone: ( 310 ) 111-1111 <p> Review: Fried Chicken.....
--

Suppose given the above labeled example we have to learn the start rules (in the multi-view setting) for extracting the phone number. There are two views *forward view and backward view*. Forward view contains the sequence of tokens that precede the

beginning of the item(phone number) and backward view contains the sequence of tokens that follow the beginning of the item.

*R1 = SkipTo( Phone:) (Rule learnt in the forward view)*

*R2 = BackTo( Number ) (Rule learnt in the backward view)*

and unlabeled examples are

- |  |
|--|
| <ol style="list-style-type: none"><li>1. Name: Chez Jean &lt;p&gt; Phone: (310 ) 666-1111 &lt;p&gt; Review.....</li><li>2 .Name: Burger King &lt;p&gt; Phone: ( 810 ) 789-1111 &lt;p&gt; Review.....</li><li>3. Name: Café Del Rey &lt;p&gt; Phone: ( 800 ) 789-1111 &lt;p&gt; Review.....</li><li>4. Name: KFC &lt;p&gt; Phone: &lt;b&gt; ( 310 )789-1111 &lt;p&gt; Review.....</li></ol> |
|--|

Now when we apply these rules to the above set of unlabeled examples we find that the two rules agree for the first three examples. That is both of them identify the beginning ( ( ) of phone number for the first three examples but they disagree for the fourth example. R1 identify <b> whereas R2 identifies ( ( ). Hence fourth example is most informative and it is given to the user for labeling.

## Hidden Markov Model

---

Hidden Markov Model(HMM) is a statistical machine learning technique. When HMM is used for information extraction the only thing that needs to be done is that a suitable training corpus needs to be annotated and a training algorithm is run resulting in information extraction. It is flexible than automatic approach in the sense is that, if the domain changes we only needs to select the suitable training corpus and annotate them.

*In this chapter first we give all the Basics of HMM and in the next chapter we describe how HMM is implemented for information extraction.*

### 4.1 Basics of Hidden Markov Model:

In basics, first we need present the exact definitions of HMM with respect to different perspectives. Difference between visible and hidden Markov Models and its general architecture then we explain Markov property. Markov property is a property that suitably explains whether a model is Markov Model or not. Lastly HMM parameters are explained. Generally speaking, the success of a HMM lies in its dependence on parameter estimation algorithms that allow training of data-dependent parameters. If a parameter estimation algorithm is good it estimates hidden parameters accurately, resulting in better performance of data extraction.

#### 4.1.1 HMM Definitions:

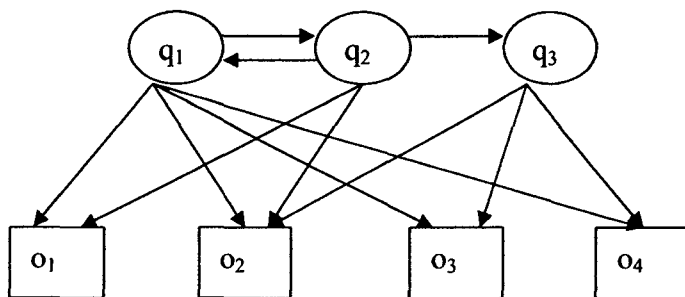
- ✚ HMM are a type of probabilistic finite state machine, and a well developed probabilistic tool for modeling sequences of observations [31].
- ✚ HMM are a principled and efficient approach to handle a sort of inherent uncertainty (for example suppose a phrase *will be held in* often precedes a seminar location, but a new document contains a *typographical error* ('will be held in') [41].

- ✦ HMM is a finite-state automaton with stochastic state-transitions and symbol emissions. The automaton models a probabilistic generative processes whereby a sequence of symbols is produced by starting in some state, transmitting to a new state, emitting a symbol selected by that state, transmitting again emitting another symbol and so-on until a designated final state is reached. States emits token according to a fixed and state specific distribution, and transition between states occur according to a fixed distribution [37].

#### 4.1.2 Visible Markov Model and Hidden Markov Model:

- In a regular Markov Model the state is directly visible to the observer and hence the state transition probabilities are the only parameters.
- Whereas in a Hidden Markov Model the state is not directly visible but, output dependent on the state is visible. Each state has a probability over the possible output tokens. Therefore the sequence of tokens generated by a HMM gives some information about the sequence of states. *Hidden* refers to the state sequence through which through which the model passes, not to the parameters of the model, *even if the model parameters are known exactly the model is still hidden [23]*.

#### 4.1.3 General Architecture of a HMM:



The random variable  $q(t)$  is the hidden state at time  $t$ .

$$q(t) \in \{q_1, q_2, q_3\}.$$

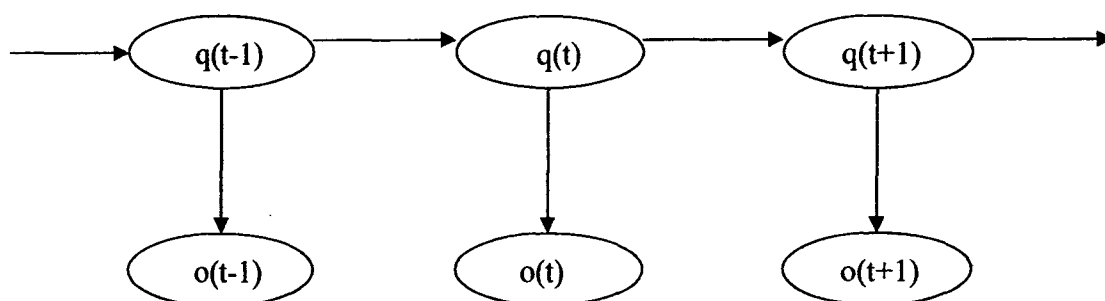
The random variable  $o(t)$  represents observation at time  $t$ .

$$o(t) \in \{o_1, o_2, o_3, o_4\}$$

Arrows denotes the conditional probability distribution.

#### 4.1.4 Markov Property:

Conditional probability distribution of the hidden state  $q(t)$  at time  $t$ , given the values of the hidden state  $q$  at all times, depends only on the values of the hidden state  $q(t-1)$ . The values at times  $(t-2)$  and before have no influence. This is called Markov property. Similarly the values of the observation  $o(t)$  depends on the value of the hidden state  $o(t)$ .



#### 4.1.5 HMM Parameters Definition:

Assume that the total number of states being  $N$ , and let  $q_t$  and  $o_t$  each denote the system state and the observation at time  $t$ . An HMM  $\lambda$  can be formally characterized by three types of parameters  $A$ ,  $B$ ,  $\Pi$  [37].

Where  $A$  is the matrix of transition probabilities between states

$$A = [a_{ij}] \quad a_{ij} = p(q_{t+1}=i/q_t=j) \quad 1 \leq i, j \leq N$$

Where  $a_{ij}$  are subjected to the constraints:

$$a_{ij} \geq 0 \quad \sum_{j=1}^N a_{ij} = 1, \quad \text{for } i=1,2,3,4,\dots,N$$

and  $B$  is the matrix of observation probability densities relating to states

$$B = [b_j(o_t)] \quad b_j(o_t) = p(o_t/q_t=j) \quad 1 \leq j \leq N$$

And  $\Pi$  is a matrix of initial state probabilities.

$$\Pi = [\Pi_i] \quad \Pi_i = p(q_1=i), \quad 1 \leq i \leq N \quad \text{and} \quad \sum_{i=1}^N \Pi_i = 1$$

## 4.2 Three Canonical Problems associated with HMM:

There are three basic problems to solve for a HMM. Problem 2 is used for *Extraction* and problem 3 for training model parameters. In our implementation work we have solved problem 2 by implementing Viterbi algorithm and problem 2 by manually calculating the parameters [29].

- 1) Given the parameters  $\lambda=(A,B,\Pi)$  of the model and given an observation sequence  $O=O_1O_2O_3\dots\dots O_T$  how to efficiently compute  $p(O/\lambda)$ . That is the probability of observation sequence(output sequence) given the model. This requires summation over all possible state sequences, but can be done efficiently using *Forward or Backward Algorithm*.
- 2) Given the parameters  $\lambda=(A,B,\Pi)$  of the model and a particular observation sequence  $O=O_1O_2O_3\dots\dots O_T$ . Find the state sequence that is most likely to have generated that output sequence. Again this requires finding a maximum over all possible state sequences, but similarly can be solved efficiently using *Viterbi Algorithm*.
- 3) Given an output sequence or a set of such sequences find the most likely set of state transition and output probabilities (that is how to adjust the parameters  $\lambda=(A,B,\Pi)$  to maximize the likelihood of all observation sequences this is also known as *Parameter Estimation*)

In other words derive the maximum likelihood estimate of the parameters of the HMM given a dataset of output sequences. No tractable algorithm is known for solving this problem exactly but a local maximum likelihood can be derived efficiently using the *Baum-Welch Algorithm*. Baum-Welch algorithm is an example of forward-backward algorithm, and is a special case of the Expectation-maximization algorithm.

### 4.3 Methods for Solving three Canonical Problems

In this section different algorithms and methods are explained that are used to solve the three canonical problems that are explained above.

#### 4.3.1 Solving Problem 1:

##### Two-state Markov Model:

Suppose that there are two states  $x$  and  $y$ . and a  $2 \times 2$  matrix  $A$  holds the transition probabilities  $a_{xx}$ ,  $a_{yx}$ ,  $a_{xy}$ ,  $a_{yy}$ . The array  $\Pi$  is a vector containing the probabilities of starting in state  $x$  or  $y$ .  $B$  is an observation matrix. If we have three observations

$O_1, O_2, O_3$  then  $B$  is a  $2 \times 3$  matrix. An element  $B_{ij}$  is the conditional probability of observing  $O_j$  given state  $i$ . The two matrices  $A$  and  $B$  with the probability vector  $\Pi$  define the parameters  $\lambda=(A,B,\Pi)$  for the HMM.

$$\Pi=[\Pi_x,\Pi_y]$$

$$A = \begin{array}{|c|c|} \hline a_{xx} & a_{xy} \\ \hline a_{yx} & a_{yy} \\ \hline \end{array} \quad B = \begin{array}{|c|c|c|} \hline b_x(O_1) & b_x(O_2) & b_x(O_3) \\ \hline b_y(O_1) & b_y(O_2) & b_y(O_3) \\ \hline \end{array}$$

For three observations and two states, we can build a trellis showing all possible paths. Basically there are 8-paths through the trellis to generate the observation sequence  $O_1, O_2, O_3$ . All observations need not be distinct from each other.

##### 4.3.1.1 By simple Calculation of Conditional probability:

Suppose the observation sequence  $O_1, O_2, O_3$  and parameter  $\lambda=(A,B,\Pi)$

Using Bayes's Theorem

$$p(O/\lambda) = \sum_x p(O/X,\lambda) * p(X/\lambda)$$



where  $X$ =some state sequence

if  $X=xyx$  then

$$\begin{aligned}
 p(X/\lambda) &= \Pi_x * a_{xy} * a_{yx} \\
 p(O/X,\lambda) &= b_x(O_1) * b_y(O_2) * b_x(O_3)
 \end{aligned}$$

and for these three observations  $O_1, O_2, O_3$  there are eight possible state sequence

$xxx, xxy, xyx, yxx, xyy, yxy, yyx, yyy$ . For a single state sequence, it takes 5 multiplications to compute the product  $p(O/X,\lambda) * p(X/\lambda)$ . For all 8 sequences it takes 40 multiplications and 7 additions.

In general, if we have  $N$  states and  $T$  observations, we would need  $N^T(2*T-1)$  multiplications and  $(N^T-1)$  additions. This is of order  $O(N^T)$  and is not computationally feasible for large  $N$  and  $T$ .

#### 4.3.1.2 Forward-Backward Algorithm:

Much of the above computation can be avoided by saving probability calculations in a trellis. These calculations are reused and it is not necessary to repeat earlier computations.

Consider a forward operator  $\alpha$  that gives the probability of being in a state and viewing a particular observation. If  $\alpha_1(x)$  is the probability of starting in state  $x$  and viewing  $O_1$ , we can say that

$$\begin{aligned}
 \alpha_1(x) &= \Pi_x * b_x(O_1) \\
 \alpha_1(y) &= \Pi_y * b_y(O_1)
 \end{aligned}$$

by induction

$$\begin{aligned}
 \alpha_2(x) &= (\alpha_1(x) * a_{xx} + \alpha_1(y) * a_{yx}) b_x(O_2) \\
 \alpha_2(y) &= (\alpha_1(x) * a_{xy} + \alpha_1(y) * a_{yy}) b_y(O_2)
 \end{aligned}$$

This inductive process continues until the last observation.

Similarly,

$$\alpha_3(x) = (\alpha_2(x) * a_{xx} + \alpha_2(y) * a_{yx}) b_x(O_3)$$

$$\alpha_3(y) = (\alpha_2(x) * a_{xy} + \alpha_2(y) * a_{yy}) b_y(O_3)$$

now,

$p(O/\lambda) = \alpha_3(x) + \alpha_3(y)$
--

it takes (12+2) multiplications and 4-additions to compute  $\alpha_3$  values. This computation is of order  $O(N^2T)$  and is a much smaller number. Similarly backward operator  $\beta$  is calculated from the backward direction i.e. from the last observation. The values of  $\beta_3(x)$ ,  $\beta_3(y)$  are initialized to 1. and values of  $\beta_2(x)$ ,  $\beta_2(y)$  are calculated and this process is repeated until the  $\beta_1(x)$ ,  $\beta_1(y)$  are calculated. The probability  $p(O/\lambda)$  is calculated using the sum ( $\beta_1(x) + \beta_1(y)$ ).

### 4.3.2 Solving Problem 2:

Second HMM problem is to uncover most likely state sequence that could have generated the observation sequence.

This problem arises in situations when a word is assigned to one or more classes. We are interested in the most likely class given context and prior knowledge. The context is either the class of neighboring words or the occurrence of particular words in the *neighborhood*. *Prior knowledge is stored in the form of the parameters for the HMM. Here in this second problem, the highest probability from a sequence of states is calculated.* Whereas in the first HMM problem, absolute probability for an observation sequence is calculated.

#### 4.3.2.1 By Finding Most Likely State at any time step:

Again for 3 observations  $O_1, O_2, O_3$  and 2 states  $x, y$ :

$$\alpha_2(x) = p(O_1 O_2 / q_2 = x)$$

$$\beta_2(x) = p(O_3 / q_2 = x)$$

The joint probability of seeing the observation sequence and being in state  $x$  at time step 2 is as follows:

$$p(O_1 O_2 O_3 / q_2 = x) = \alpha_2(x) \beta_2(x)$$

using Bayes's Theorem,

$$\begin{aligned} \gamma_2(x) &= p(q_2 = x / O) = p(O, q_2 = x) / p(O) \\ &= \alpha_2(x) \beta_2(x) / (\alpha_2(x) \beta_2(x) + \alpha_2(y) \beta_2(y)) \end{aligned}$$

the probability of being in state  $x$  at time step 2 given the observation sequence :

$$q_2 = \arg \max_{x,y} [\gamma_2(x), \gamma_2(y)]$$

Similarly, calculating  $\gamma_2(y)$ , and given  $\gamma_2(x)$ ,  $\gamma_2(y)$  we find the most likely state  $q_2$  at time step 2.

The values of  $q_1$ ,  $q_2$  are calculated similarly. The most likely state at any time step  $i$  is given by  $q_i$ . These  $q$  states are a solution to the second HMM problem of finding the most likely state sequence given an observation sequence.

#### **Problem with this Approach:**

In rare cases, a particular state transition  $ij$ , which occurs in the derived state sequence, may be disallowed ( $a_{ij} = 0$ ), leading to an invalid solution. *The reason for this error is that at each state, we are checking for the highest probability in the set of states without regard to the global trellis structure or neighboring states.*

#### 4.3.2.2 Viterbi Algorithm:

It is used to find the best state sequence. Similar to the above method with the addition of the state transition constraints:

Here, a new  $\delta$ -operator similar to  $\alpha$ -operator is introduced. The purpose of  $\delta$ -operator is to compute a state-sequence while that of  $\alpha$ -operator is used to compute a probability

$$\begin{aligned} \delta_1(x) &= \Pi_x * b_x(O_1) = \alpha_1(x) \\ \delta_1(y) &= \Pi_y * b_y(O_1) = \alpha_1(y) \end{aligned}$$

*( $\delta_1 = \alpha_1$  for the first time step)*

By induction step: induction step for the  $\delta$ -operator finds the maximum probability

$$\begin{aligned} \delta_2(x) &= \max \{ \delta_1(x) * a_{xx}, \delta_1(y) * a_{yx} \} b_x(O_2) \\ \delta_2(y) &= \max \{ \delta_1(x) * a_{xy}, \delta_1(y) * a_{yy} \} b_y(O_2) \end{aligned}$$

moving from left to right we can calculate  $\delta$ -values for the remaining observation. Here instead of searching for the most likely state at each time step, we look for the most probable path through the trellis that can generate the observation sequence. The most probable path can be found piecewise.

After calculating all the  $\delta$ -values, we find the most likely end state by comparing  $\delta_2(x), \delta_2(y)$ . the state  $x$  is chosen for time step 2 if  $\delta_2(x)$  is greater than  $\delta_2(y)$ . backtracking one time step, we compute:

$$\arg \max_{x,y} \{ \delta_2(x) * a_{xx}, \delta_2(y) * a_{yx} \}$$

This value gives us the most likely state for time step 2. the rest of the state sequence until time step 1 can be extracted in reverse order using the induction step above [12].

### 4.3.3 Solving problem 3 (Parameter Estimation):

Generally it is assumed that the parameters of the HMM  $\lambda=(A,B,\Pi)$  are known. When we initially begin to model a series of observation sequences, these parameters are not known. We need to set the parameters of our model such that the likelihood of the observation sequence is maximized.

Baum-Welch Algorithm is the most widely adopted training methodology for model parameter estimation. The purpose of such training process is to estimate the well-suited model parameters so as to make  $p(O/\lambda)$  maximized. Baum-Welch iteratively finds a set of HMM parameters.

*The observation sequences are called the training sequences to build the HMM model and are vital to build an accurate model.*

#### 4.3.3.1 Baum-Welch Algorithm:

A new variable  $\xi$  is defined

$$\xi_2(x,y) = p(q_2=x, q_3=y / O, \lambda)$$

the probability of being in state  $x$  at time step 2 and in state  $y$  at time step 3 given an observation sequence and HMM parameters.

$$\xi_2(x,y) = (\alpha_2(x) * a_{xy} * b_y(O_3) * \beta_2(y)) / (\alpha_2(x) \beta_2(x) + \alpha_2(y) \beta_2(y))$$

where  $\alpha$  and  $\beta$  are forward and backward operators and  $a$  and  $b$  are transition and observation probabilities.

Another variable  $\gamma_2$  is defined as

$$\gamma_2(x) = \xi_2(x,y) + \xi_2(x,x)$$

both  $\gamma$ ,  $\xi$  operators are used to find HMM model parameters.

**Steps in Baum-Welch Algorithm :**

- 1.) Assign initial values to the model parameters A, B,  $\Pi$ .
- 2.) Run the loop given below until the model parameters converge.
  - Compute the expected values of  $\xi$  and  $\gamma$  operators for all states and observed values using estimated values of A, B,  $\Pi$ .
  - Compute the maximized values of A, B,  $\Pi$  using the  $\xi$  and  $\gamma$  values.
- 3.) Return the model parameters.

The two parts of step 2 are repeated till the model parameters do not change significantly from one iteration to another. This algorithm will find better estimates of  $\lambda = \{ A, B, \Pi \}$  iteratively. For example with 3 observations  $O_1, O_2, O_3$  and 2 state x, y the expected number of transitions from state x to y is  $\xi_1(x, y) + \xi_2(x, y) + \xi_3(x, y)$ .

For *transition probability matrix A*: one element is calculated as

$$a'_{xy} = (\xi_1(x, y) + \xi_2(x, y)) / (\gamma_1(x) + \gamma_2(x))$$

For observation matrix B :

$$b'_x(O_1) = (\gamma_1(x/O_1) + \gamma_2(x/O_1) + \gamma_3(x/O_1)) / (\gamma_1(x) + \gamma_2(x) + \gamma_3(x))$$

and the prior probability vector  $\Pi$  is estimated using  $\gamma$  operator.

$\gamma_1(x)$  = expected number of times of being in state x at time-step 1 is  $\gamma_1(x)$ .

In the above training process, only one observation sequence is used to train the model , we can use more than one observation sequence to train and build the HMM model.

If we have N-training observation sequences, then the expected probability of starting in state x is

$$\gamma_1(x)' = (1/N) \sum_{i=1}^N \gamma_1^{(i)}(x)$$

#### **4.4 Selecting the Model Topology (Learning Model Structure from training Data):**

In order to build an HMM for information extraction, we must first decide how many states the model should contain, and what transitions are allowed between states [33].

##### **4.4.1 Building initial Model From training Data:**

Generally in an initial model one state per class is used and transitions are allowed from any state to any other state (a fully-connected model). However, this model may not be optimal in all cases.

An alternative to simply assigning one state per class is to learn the model structure from training data. Training data labeled with class information can be used to build a *maximally-specific model*. Each word in the training data is assigned its own state, which transitions to the state of the word that follow it. Each state is associated with the class label of its word token. A transition is placed from the start state to the first state of each training instance, as well as between the last state of each training instance and the end state.

##### **4.4.2 Generalizing Maximally-Specific Model(using state merging techniques):**

The initial model or maximally specific model can be used as the starting point of a variety of state-merging techniques. Two simple types of merging techniques that can be used are *Neighbor-Merging and V-Merging*.

###### **Neighbor-Merging:**

It combines all states that share a transition and have the same class label. For Example: the sequence of adjacent title states from a single header are merged into a single title state. As multiple neighbor states with the same class label are merged into one, a self transition loop is introduced, whose probability represents the expected state duration for that class.

**V-Merging:**

It merges any two states that have the same label and share transitions from or to a common state. V-merging reduces the branching factor of the maximally specific model. For example: instead of beginning in the start state and selecting from among many transitions into title states, the V-merged model would merge the children title states into one so that only one transition from the start state to the title state would remain.

Model structure can be learned automatically from data, starting with either a maximally-specific neighbor-merged or V-merged model, using a technique like Bayesian model merging. Bayesian model merging seeks to find the model structure that maximizes the probability of model M given some training data D, by iteratively merging states until an optimal tradeoff between fit to the data and model size has been reached. This relationship is expressed using Bayes's rule as :

$$P(M/D) \propto P(D/M)*P(M)$$

Where  $P(D/M)$  can be calculated with the forward algorithm, or approximated with the probability of the Viterbi path.



## Proposed Work and Experiment

---

### 5.1 Problem Formulation and HMM for Solving problem:

#### 5.1.1 Problem Formulation:

Our work is based on a well known Named-Entity Extraction problem. In a Named-entity Extraction, generally entities such as names of people, products, places, organizations, dates, dimensions and currency are located in the textual documents or semi-structured documents.

In Named-entity Extraction words of the text are assigned to one or more semantic classes (entity types). Words that do not belong to a semantic class are assigned to a default nonentity class. Before applying the entity extraction, first a training set is built that consists of tagged entities within the text. Sequence of entity types in the tagged entities are used to model the occurrence of entities in the text.

*In this work, our objective is to spot and extract named-entity (product name) from semi-structured documents (web pages).* In this work a set of web pages(containing product names) are collected from major websites(basically commercial websites containing information about different products) and their source code are analyzed for constructing model topology for HMM (states, observations, transitions of states). Tagged dataset is created and divided into two parts for training and testing. HMM parameters are learned (training) from training datasets. Sequence of observations from test data and estimated HMM parameters are then used to decide whether a particular observation in the testing observation sequence belongs to its correct state (testing). In other words given a testing sequence of observations (some of them are formatting tags, some are product names and rest are other than these two like product price and product description) we need to find out the corresponding state sequence and match how many of the token in the product name are correctly identified.

### 5.1.2 Why HMM for Solving Problem:

Named-Entity extraction is a classification problem. There are different machine learning methods viz: Decision trees, HMM, Rule-based methods for solving classification problems, which can be applied for solving NER problem. We have applied Hidden Markov Model for following reason:

- HMM has been applied successfully to problems characterized by a set of states, a set of observations, state transition probabilities, initial state probabilities and observation probabilities given a state. Named-Entity Extraction problem can be framed in a form suitable for an HMM because set of states of HMM can represent set of name classes (such as product name) and each word(token) is considered as an observation that belongs to at-least one state(entity type) and also a single word may belong to more than one state. Probability that a word belongs to a particular entity type is observation probability and probability that an entity type (state) appears after a particular entity type (transition of one entity type to other entity type) is state transition probabilities.

### 5.1.3 Topology Decision for HMM:

As mentioned in section 4 of chapter 4 we need to build an initial model from training data. i.e. we must first decide how many states the model should contain, and what transitions are allowed between states. After analyzing the training data we have the following description for the topology.

#### *States:*

- a. By observing the DOM tree of all the training segments we observe the pattern that the product name generally follows either HTML tag <A> or <B>. Most of the time <A> anchor tag occurs before the product name and at the same time it is followed

by its counterpart `</A>`. Hence two states AS and AE are used that precede and succeed the product name respectively. For all other HTML tags state *H* is created.

- b. All the segments always start with the state H.
- c. Each word in the product name is assigned to the state PN. All the words other than the product name is assigned to the state *O(others)*.
- d. Numeric values are assigned to state *O*.

*Hence total number of states are 5 H, AS, AE, O, PN.*

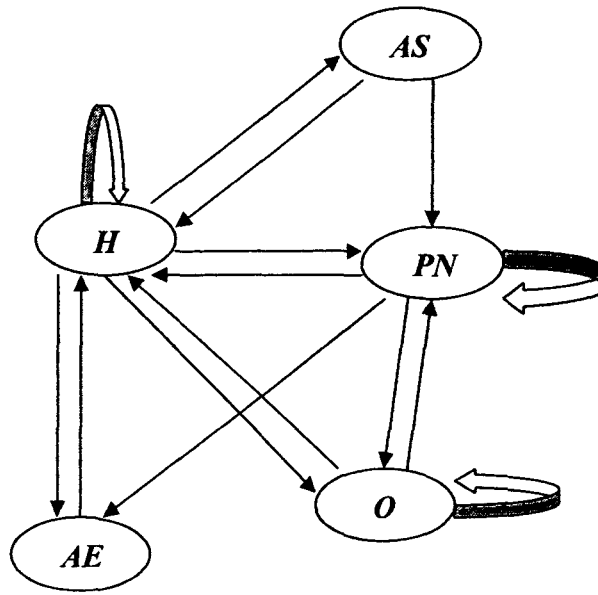
***Observations:***

- 1.) All the HTML observations except (except the anchor\_start and anchor\_end before and after the product name) are converted to *html* observations type (*observation type represents feature of an observation*).
- 2.) Anchor start observation is *a\_start* and Anchor end observation is *a\_end*.
- 3.) Each of the word in the product name and each of the word belonging to state *O* are converted to tokens of observation type *string*.
- 4.) Each of the numeric values is converted to a token of observation type *num*.

*Hence total number of observation types are 5 html, a\_start, a\_end, num, string.*

***Transitions:***

Observing the dataset a generalized transition diagram is given as below and depending upon the number of training data files used for parameter estimation some of these transitions may occur or may not occur. *Figure 5.1 gives the diagrammatic representation of generalized state transitions.*

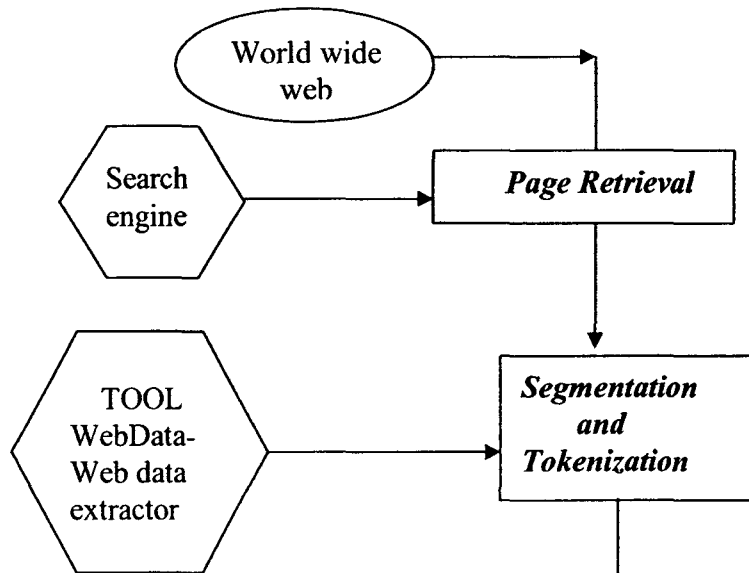


**Figure 5.1 Generalized Transition Diagram for the proposed HMM**

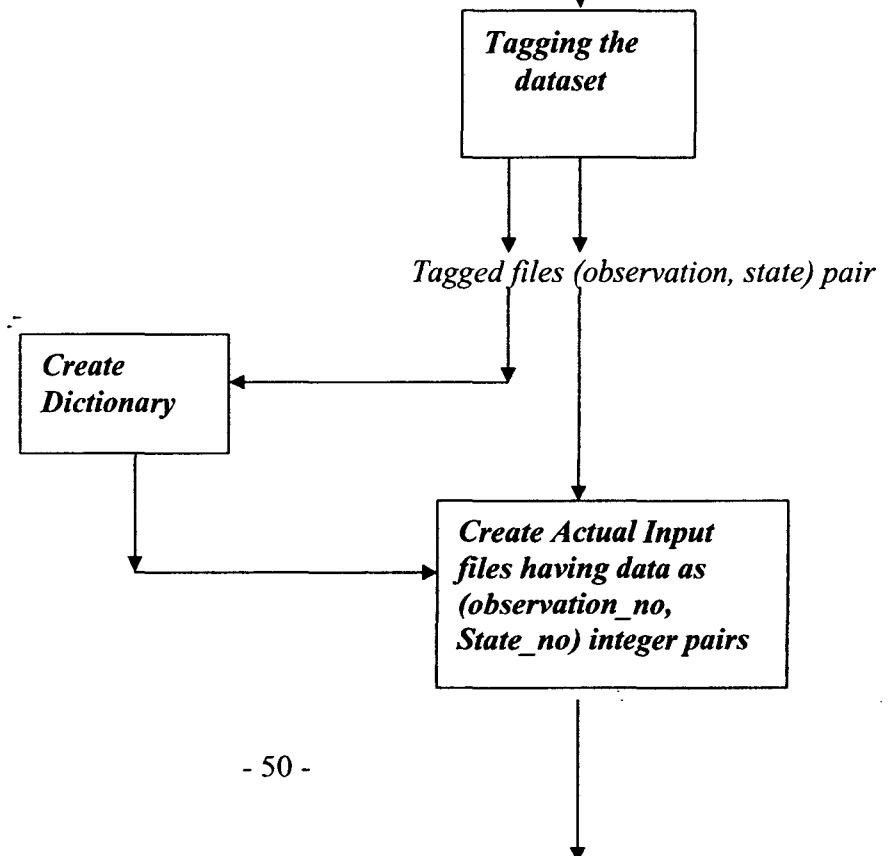
## 5.2 Architecture of The Proposed System:

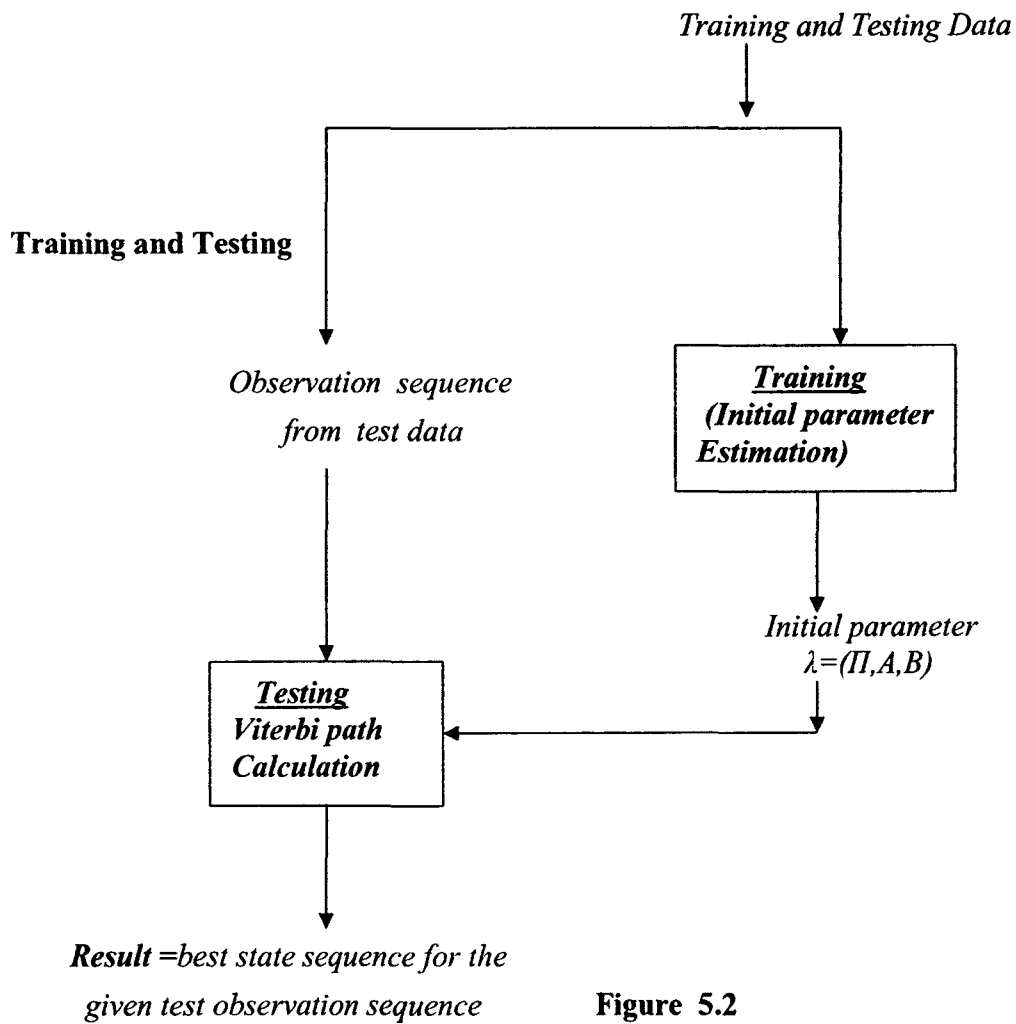
The figure 5.2 given below represents the architecture of our proposed system. The implementation details this system is given in the following section 5.3.

### Data Selection and Preprocessing



### Preparation of Training Data and Tagging





**Figure 5.2**

### 5.3 Steps In Implementation

#### 5.3.1 Data Selection and Preprocessing

1. List the major websites that contain different product information.
2. Select few sample pages from each website and save each of them to a folder.
3. Pass the URL of each web page to a tool called *TotalData-Web Data Extractor*.

- Select a region/segment that contains the information about the product [must contain at least product name (this is considered as one segment of interest)].
- Save the source code of this region to a file.

If there are more than one segment of interest in this selected page then repeat the step 3 for all the segments. Also repeat the step 3 for all the saved web pages as in step 2.

4. In each of the saved file in step 3 each word and html tags are tokenized. For example there may be different html tags like <B>, <U>, <I> and so on. All html tags except <A> and </A> before and after the product names are converted into a single token *html*. <A> is converted to the token *a\_start* and </A> is converted to the token *a\_end*. Each of the word is converted to the token *string* and each of the numeric values is converted to the token *num*. Once the file of tokens is created, with the help of the topology given in subsection 5.1.2 (topology decision) tagging of data is done as described below.

### 5.3.2 Training and Testing

#### a. Preparation of training data and Tagging

1. After tokenization is html token is assigned to state H. Each *a\_start* and *a\_end* tokens are assigned to states AS and AE states respectively. All the numeric value token are assigned to state O. All the tokens in the product name are assigned to state PN and all the words other than product name are assigned to state O.
2. Once the entire training and testing data have been tagged we need to create a dictionary of observations and states. Dictionary is created by observing each tagged data file and assigning unique integers to each states and observations. Figure 5.3 gives states, observation types and their corresponding integer values respectively.

State		Observation	
H	1	html	1
AS	2	a_start	2
AE	3	a_end	3
PN	4	String	4
O	5	num	5

Figure 5.3

3. Using the dictionary, the tagged data files are converted into the actual input files where each pair of <Value, tag> or <observation, state> is converted into the <observation\_no, state\_no>. These files are actual input files and they can be used for initial parameter estimation.

**b. Parameter Estimation (Training):**

HMM parameters  $\lambda=(\Pi,A,B)$  are estimated as follows:

**Initial probability:**

$$\Pi_i = I(i) / \sum_{j=1}^N I(j); \quad 1 \leq i \leq N \quad \dots\dots\dots(5.1)$$

Where I(.) is the initial probability that a sequence starts from a specific state and N is the number of states included in the model.

Each *element*  $a_{ij}$  in the *transition probability matrix*  $A$  is defined as follows:

$$a_{ij} = C_{ij} / \sum_{k=1}^N C_{i,k} \quad 1 \leq i,j \leq N \quad \dots\dots\dots(5.2)$$

where  $C_{i,j}$  is the number of transitions that transmit from state  $S(i)$  to state  $S(j)$ .

The *emission probability matrix element*  $b_j(V_k)$  is defined as follows:

$$b_j(V_k) = E_j(V_k) / \sum_{i=1}^N E_j(V_i) \quad 1 \leq j \leq N, \quad 1 \leq k \leq M \quad \dots\dots(5.3)$$



**c. Viterbi Path Calculation (Testing):**

Once the parameters are estimated user inputs the testing observation sequence for which he wants to calculate the most probable (best) state sequence. This sequence and parameters  $\lambda=(\Pi,A,B)$  are then passed to Viterbi Path Calculation module and most probable state sequence is generated.

- d. In order to calculate accuracy of the method, we need to compare the result obtained for the testing sequence with the actual states of the testing observation sequence and analyze how many observations are identified by their correct states. We have used the precision and recall measures for evaluation of the result formulae for these measures are used as given in equations 1.1,1.2 respectively as in chapter 1.

## 5.4 Experiments and Results:

**Objective:** *To extract Product Name from semi-structured web pages containing information about various products.*

**Input :**

Sample Pages : as given in appendix A

**Segment 1:** (This the first segment of first page)

**HP**  
**HP Pavilion p6213w 2.6GHz Intel Pentium Desktop PC - NY639AAR#ABA**  
2.6GHz Intel Pentium E5300, 6GB RAM, 500GB Hard Drive, SuperMulti DVD+RW, High-Definition Audio, 15-in-1 Integrated Digital Media Reader, 10/100/1000 Network LAN, Windows 7 Home Premium 64-Bit - Refurbished / Recertified  
eCOST.com Part #54780647  
Mfg. Part #NY639AAR#ABA

### Source Code:

```
<TD style="BORDER-BOTTOM: #0000ff thick solid; BORDER-LEFT: #0000ff thick solid; BORDER-TOP: #0000ff thick solid; BORDER-RIGHT: #0000ff thick solid" class=prd><DIV style="FONT-WEIGHT: bold">HP </DIV>
<DIV><A
href="http://www.ecost.com/Detail/Desktops/HP/NY639AARABA/54780647.aspx?
navid=155438718" cmImpressionSent="1">HP Pavilion p6213w 2.6GHz Intel
Pentium Desktop PC - NY639AAR#ABA </A></DIV>
<DIV>2.6GHz Intel Pentium E5300, 6GB RAM, 500GB Hard Drive, SuperMulti
DVD+RW, High-Definition Audio, 15-in-1 Integrated Digital Media Reader,
10/100/1000 Network LAN, Windows 7 Home Premium 64-Bit - Refurbished /
Recertified <BR>eCOST.com Part #54780647 <BR>Mfg. Part #NY639AAR#ABA
</DIV></TD>
```

### Preparation of training data and Tagging

#### DOM Tree (for above segment)

```
<TD class=prd>
  <DIV>HP </DIV>
  <DIV>
    <A >
      HP Pavilion p6213w 2.6GHz Intel Pentium
      Desktop PC - NY639AAR#ABA
    </A>
  </DIV>
  <DIV>
    2.6GHz Intel Pentium E5300, 6GB RAM, 500GB Hard
    Drive, SuperMulti DVD+RW, High-Definition Audio,
    15-in-1 Integrated Digital Media Reader, 10/100/1000
    Network LAN, Windows 7 Home Premium 64-Bit -
    Refurbished / Recertified
    <BR> eCOST.com Part #54780647
    <BR>Mfg. Part #NY639AAR#ABA
  </DIV>
</TD>
```

**Tokenization (actual tokens are preprocessed)**

Html html string html html a\_start string string string string string string string string  
string string a\_end html html string string string string string string string string  
string string string string string string string string string string string string string  
string string string num string string string string string string string string html  
string string string html string string string html html

**Tagging:**

<html,H> <html,H> <string,PN> <html,H> <html,H> <a\_start,AS> <string,PN>  
<string,PN> <string,PN> <string,PN> <string,PN> <string,PN> <string,PN>  
<string,PN><string,PN> <a\_end,AE> <html,H> <html,H> <string,O> <string,PN>  
<string,PN> <string,O><string,O> <string,O> <string,O> <string,O> <string,O>  
<string,O> <string,O> <string,O> <string,O> <string,O> <string,O> <string,O>  
<string,O> <string,O> <string,O> <string,O> <string,O> <string,O> <num,O>  
<string,O> <string,O> <string,O> <string,PN> <string,O> <string,O> <string,O>  
<html,H> <string,O> <string,O> <string,O> <html,H> <string,O> <string,O>  
<string,O> <html,H> <html,H>

**Actual Input file (observation\_no,state\_no) pair:**

<1,1> <1,1> <4,4> <1,1> <1,1> <2,2> <4,4> <4,4> <4,4> <4,4> <4,4>  
<4,4> <4,4> <4,4> <4,4> <3,3> <1,1> <1,1> <4,5> <4,4> <4,4> <4,5>  
<4,5> <4,5> <4,5> <4,5> <4,5> <4,5> <4,5> <4,5> <4,5> <4,5> <4,5>  
<4,5> <4,5> <4,5> <4,5> <4,5> <4,5> <4,5> <5,5> <4,5> <4,5> <4,5>  
<4,5> <4,5> <4,5> <4,5> <1,1> <4,5> <4,5> <4,5> <1,1> <4,5> <4,5>  
<4,5> <1,1> <1,1>

## Training:

### Parameter estimation

Initial probability is  $[1,0,0,0,0]$  because first token of all the data files belong to state H .

Hence initial probability of state H is 1 and all others are zeros.

<b>1.0000</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>0</b>	<b>1.0000</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>0</b>	<b>0</b>	<b>1.0000</b>	<b>0</b>	<b>0</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>1.0000</b>	<b>0</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>0.9706</b>	<b>0.0294</b>

<b>0.4444</b>	<b>0.1111</b>	<b>0</b>	<b>0.1111</b>	<b>0.3333</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>1.0000</b>	<b>0</b>
<b>1.0000</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
<b>0.0833</b>	<b>0</b>	<b>0.0833</b>	<b>0.7500</b>	<b>0.0833</b>
<b>0.0882</b>	<b>0</b>	<b>0</b>	<b>0.0294</b>	<b>0.8824</b>

## Testing:

After estimating the parameters as given in equations 5.1, 5.2, 5.3, product names were extracted for the observation sequences belonging to tested data. We have taken 40 segments as training segments and trained the HMM model to estimate parameters then by applying Viterbi algorithm we have tested the model by giving various testing observation sequences(not present in the training data) and then calculating the precision and recall values. After doing experiments on different sets of testing data we have found average values of precision and recall as 90% and 75% respectively. The results for different testing observation sequences are given as n appendix B.

### **Analysis of result:**

Analysis of result (sequences given in appendix B) shows that in most of the test cases, states identified as product name are correct which is indicated by a precision of 90%. It was observed that most of these states are following a pattern of HTML tags. The wrongly identified states are not following these patterns. This may be because of lack of proper training examples or presence of web pages. Which are not following some standard pattern. The performance may be improved by incorporating variety of web pages in the training dataset. The recall of our result is comparatively less than precision, which leaves a scope for improving parameter estimation techniques as well as experimenting with different model topologies. However results are encouraging which indicates that HMM can be explored for Structured Data Extraction.

### Conclusion

---

Generally information extraction can be useful for any collection of documents from which we want to extract specific facts. World Wide Web is such a collection of documents. Here, information on a subject is often scattered among different web servers and hosts, each having different formats, and it would be useful if this information could be extracted and integrated into a structured form. Different approaches for information extraction exist. These approaches have been divided into two broad categories: Knowledge Engineering approach and Automatic Training approach. Automatic training approach is found to be more suitable for information extraction. We have explored and analyzed various Knowledge Engineering methods for information extraction and found that two of them are more general, Manual and Wrapper Induction. But each of them has their own limitations. When we apply wrapper induction for information extraction, basically three machine learning methods are used. Inductive learning method has major issues that can be solved by Instance-based learning and Active learning. When we explored Automatic learning method we found that Hidden Markov Model is a powerful statistical machine learning technique that is just beginning to gain use in information extraction tasks. When compared with many other techniques used in statistics-based methods, HMM has a strong theoretical foundation. In this dissertation, we have tried to provide a general architecture for extracting named-entities from semi-structured web pages using Hidden Markov Model. In the previous works Named-Entity Recognition has been applied for extracting names, people, places, organizations etc. from textual information. In our work, the problem is to spot and extract named entity (product name) from semi-structured web pages. The main reason for using HMM for named entity extraction was that entity extraction problem can be framed in a form suitable for an HMM. Finally we can conclude that this work is an attempt to explore new emerging areas for Structured data extraction by applying various machine learning approaches because Structured data extraction is still

an open problem. There are many problems while applying inductive learning approaches for SDE such as labeling of web pages is very difficult, rules become very large and complicated, due to dynamic nature of web incremental learning is required. All these problems can be reduced by using Instance-based learning and Active learning. Although we have explored Hidden Markov Model but Active learning and Instance based learning methods can also be explored further for Structured Data Extraction.

**APPENDIX A**  
**(Sample Web Pages)**





Flash needed

Flash needed

[Rebate Center](#)
[Helpdesk](#)

[Search](#)

**No Payments for 6 Months**  
 **BillMeLater** on orders over \$250 [details](#) subject to credit approval.

Flash plugin not installed

**Shop by Price**

- Below \$25 (2)
- \$25 - \$50 (5)
- \$75 - \$100 (8)
- \$100 - \$200 (37)
- \$200 - \$300 (36)
- [+ More](#)

**Manufacturer**

- HP (746)
- Lenovo (513)
- Wyse (218)
- Acer (79)
- Dell (60)
- [+ More](#)

**Operating System**

- Windows Vista Business (487)
- Windows XP Professional (68)
- Windows Vista Home Premium (18)
- Windows Vista Home Premium x64 (15)
- Windows Vista Business x64 (9)

[Systems](#) > [Desktops](#)

Sort By: Most Popular  25 items per page  Page 1 of 71 >>

1



**HP HP Pavilion p6213w 2.6GHz Intel Pentium Desktop PC - NY639AAR#ABA**  
 2.6GHz Intel Pentium E5300, 6GB RAM, 500GB Hard Drive, SuperMulti DVD+RW, High-Definition Audio, 15-in-1 Integrated Digital Media Reader, 10/100/1000 Network LAN, Windows 7 Home Premium 64-Bit - Refurbished / Recertified  
 eCOST.com Part #54780647  
 Mfg. Part #NY639AAR#ABA

**In Stock**

**\$329.99**  
 You Save: 27%

[+ ADD TO CART](#)  
[BARGAIN](#)

2




**HP HP Pavilion Elite e9270f 2.8GHz Intel Core i7 Desktop PC - AU917AAR#ABA**  
 2.8GHz Intel Core i7-860, 8GB RAM, 1TB Hard Drive, ATI Radeon 4650 w/ 1GB RAM, Blu-ray and SuperMulti DVD+RW, Wireless 802.11a/b/g/n, HDMI, Windows 7 Home Premium 64-Bit - Refurbished / Recertified  
 eCOST.com Part #54780621  
 Mfg. Part #AU917AAR#ABA

**In Stock**

**\$767.99**  
 You Save: 23%

[+ ADD TO CART](#)  
[BARGAIN](#)

TVs Digital Cameras Noteb  s Networking Storage GPS Receivers MP3 Play

**COMPARE PRICES FROM 100'S OF ONLINE STORES: Notebooks & Laptops**

SHOP FOR:

**CATEGORY NAVIGATOR**

- ⋮ Computers & Equipment
  - ⋮ Notebooks, Laptops & Accessories
    - ⋮ **Notebooks & Laptops**
    - ⋮ Tablet PCs
    - ⋮ Docking Station
    - ⋮ Port Replicators
    - ⋮ Network/Modem Cards
    - ⋮ Notebook, Laptop Accessories
    - ⋮ Carrying Cases
    - ⋮ Batteries & Adapters

**BRAND FILTERS**

- Toshiba (670)
- Asus (340)
- Giga-Byte (2)
- Intel (1)
- Sony (834)

[See More Brands...](#)



**ATTRIBUTE FILTERS**

Screen Size

- < 11.1" (55)
- 12.1" - 13.4" (44)
- 14" - 15.6" (79)
- 16" - 17.3" (51)

Processor

- AMD Athlon X2 (4)
- AMD Sempron (1)
- AMD Turion X2 (11)
- Intel Atom (176)
- Intel Core 2 Duo (366)

Intel Pentium M (15)

Processor

**MOST POPULAR**

LOW PRICED

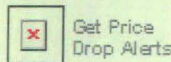
TOP RATED

USER FI



**Asus EPC1005HAPU1XBK Eee PC 10.1-Inch Black Netbook - 10.5 Hour Battery Life**

Intel Atom | 1.66GHz | 10.1" | 1 GB RAM | 160 GB HDD | 1005HA-PU1X-BK | EPC1005HAPU1XBK



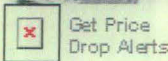
**\$315 - \$465** (14 stores)

3.9 out of 5.0 (858 reviews)



**Dell Latitude Notebook Pentium M Processor 1.40GHz with 14.1in XGA Display (D600)**

D600



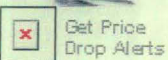
**\$217** (1 store)

3.8 out of 5.0 (39 reviews)



**Asus Eee Box Nettop PC - Black (EB1501-B0167)**

EB1501-B0167



**\$465 - \$580** (16 stores)

3.7 out of 5.0 (6 reviews)



**HP TouchSmart 12.1 tx2z Tablet PC - 2.2 GHz; 320GB HD; 4GB Memory**

<a href="#">+ More</a>
<b>Optical Drive</b>
DVD-Writer (334)
DVD-Reader (145)
Combo Drive (60)
BD-Reader/DVD-Writer (4)
BD-Writer (2)
<a href="#">+ More</a>
<b>Processor</b>
<b>Manufacturer</b>
Intel (161)
AMD (37)
<b>Processor</b>
<b>Speed</b>
3GHz (146)
2.66GHz (60)
2.6GHz (54)
2.8GHz (50)
3.16GHz (49)
<a href="#">+ More</a>
<b>Processor Type</b>
Core 2 Duo (122)
Pentium Dual-core (27)
Athlon 64 X2 (15)
Core 2 Quad (13)
Phenom X3 (9)
<a href="#">+ More</a>
<b>Product Line</b>
ThinkCentre (331)
Business Desktop (229)
Pavilion (18)

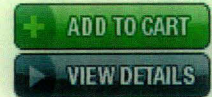
3



**HP**  
**HP Pavilion p6140f 2.33GHz Intel Core 2 Quad Desktop PC w/ 8GB RAM and 1TB HD - NP192AAR#ABA**  
 2.33GHz Intel Core 2 Quad Q8200 Processor, 8GB RAM, 1TB Hard Drive, SuperMulti DVD Burner w/ LightScribe, Wireless 802.11a/b/g/n, Intel GMA 3100 IGP, Windows Vista Home Premium 64-bit - HP Refurbished / Recertified eCOST.com Part #51673424 Mfg. Part #NP192AAR#ABA

**In Stock**

**\$465.99**  
 You Save: 48%



4



**Dell**  
**Dell Optiplex SX280 3.2GHz Pentium 4 Desktop PC - SX280-P4-500G-R**  
 3.2GHz Intel Pentium 4 Processor, 2GB RAM, 500GB Hard Drive, DVD / CD-RW, Windows XP Professional - Refurbished / Recertified eCOST.com Part #52551568 Mfg. Part #SX280-P4-500G-R

**In Stock**

**\$268.99**  
 You Save: 33%



5



**HP**  
**HP TouchSmart 600-1055 2.13GHz Intel Core 2 Duo All-in-One Desktop PC w/ Blu-ray Player - Windows 7 - NY539AA#ABA**  
 2.13GHz Intel Core 2 Duo, 4GB RAM, 750GB HD, DVD+RW, Blu-ray Player, 23" Touchscreen LCD, nVidia GeForce GT230, AVerMedia TV Tuner, Wireless 802.11 b/g/n, Webcam,

**In Stock**

**\$999.99**  
 You Save: 33%



**APPENDIX B**

**(Result of Testing Observation Sequences)**

Testing Sequences for the calculation of precision and recall values (sequence 1)


	(observation,state) pair	Actual state sequence	observed state sequence		(observation,state) pair	Actual state sequence	observed state sequence
1	<1,1>	1	1	31	<4,5>	5	5
2	<1,1>	1	1	32	<4,5>	5	5
3	<4,4>	4	5	33	<4,5>	5	5
4	<1,1>	1	1	34	<4,5>	5	5
5	<1,1>	1	1	35	<4,5>	5	5
6	<2,2>	2	2	36	<4,5>	5	5
7	<4,4>	4	4	37	<4,5>	5	5
8	<4,4>	4	4	38	<4,5>	5	5
9	<4,4>	4	4	39	<4,5>	5	5
10	<4,4>	4	4	40	<4,5>	5	5
11	<4,4>	4	4	41	<5,5>	5	5
12	<4,4>	4	4	42	<4,5>	5	5
13	<4,4>	4	4	43	<4,5>	5	5
14	<4,4>	4	4	44	<4,5>	5	5
15	<4,4>	4	4	45	<4,5>	5	5
16	<3,3>	3	3	46	<4,5>	5	5
17	<1,1>	1	1	47	<4,5>	5	5
18	<1,1>	1	1	48	<4,5>	5	5
19	<4,5>	5	5	49	<1,1>	1	1
20	<4,4>	4	5	50	<4,5>	5	5
21	<4,4>	4	5	51	<4,5>	5	5
22	<4,5>	5	5	52	<4,5>	5	5
23	<4,5>	5	5	53	<1,1>	1	1
24	<4,5>	5	5	54	<4,5>	5	5
25	<4,5>	5	5	55	<4,5>	5	5
26	<4,5>	5	5	56	<4,5>	5	5
27	<4,5>	5	5	57	<1,1>	1	1
28	<4,5>	5	5	58	<1,1>	1	1
29	<4,5>	5	5				
30	<4,5>	5	5				

means wrongly identified

Testing Sequences for the calculation of precision and recall values

(sequence 2)

	(observation,state) pair	Actual state sequence	observed state sequence		(observation,state) pair	Actual state sequence	observed state sequence
1	<1,1>	1	1	31	<4,5>	5	5
2	<1,1>	1	1	32	<4,5>	5	5
3	<5,5>	5	5	33	<4,4>	4	5
4	<1,1>	1	1	34	<4,5>	5	5
5	<4,4>	4	5	35	<4,5>	5	4
6	<1,1>	1	1	36	<4,4>	4	4
7	<1,1>	1	1	37	<4,5>	5	4
8	<2,2>	2	2	38	<4,5>	5	5
9	<4,4>	4	4	39	<4,5>	5	4
10	<4,4>	4	4	40	<1,1>	1	1
11	<4,4>	4	4	41	<1,1>	1	1
12	<4,4>	4	4	42	<4,5>	5	5
13	<4,4>	4	4	43	<4,5>	5	5
14	<5,5>	5	5	44	<1,1>	1	1
15	<4,4>	4	4	45	<5,5>	5	5
16	<4,4>	4	4	46	<1,1>	1	1
17	<4,4>	4	5	47	<4,5>	5	5
18	<4,4>	4	5	48	<4,5>	5	5
19	<3,3>	3	3	49	<5,5>	5	5
20	<1,1>	1	1	50	<1,1>	1	1
21	<1,1>	1	1	51	<1,1>	1	1
22	<4,4>	4	5				
23	<5,5>	5	5				
24	<4,5>	5	5				
25	<4,5>	5	5				
26	<4,4>	4	5				
27	<4,5>	5	5				
28	<4,5>	5	5				
29	<4,5>	5	5				
30	<4,5>	5	5				

 means wrongly identified

## References

---

- [1] Applet, D., & Israel, D. (1999). Introduction to information extraction technology. In Proceedings of the 16th International Joint Conference on Artificial Intelligence. Stockholm, Sweden.
- [2] Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In:SIGMOD'03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 337-348, ACM, New York(2003).
- [3] Baluja, S., Mittal, V. and Sukthankar, R. 2000. Applying Machine Learning for High-Performance Named-Entity Extraction. *Computat Intell.* 16,586-595.
- [4] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data.* Springer, 2007
- [5] Chang, C.-H., Kayed, M., Girgis, M. R., Shaalan, K., A Survey of Web Information Extraction Systems, *IEEE TKDE (SCI, EI)*, Vol. 18, No. 10, pp. 1411-1428, Oct. 2006.
- [6] D. Applet, J.Hobbs, D. Israel, and M. Tyson. FASTUS: A finite-state processor for Information Extraction from Real World text. In Proceedings IJCAI -93,1993.
- [7] D. Bikel , S. Miller , R. Schwartz , R. Weischedel. NYMBLE: A High-Performance learning name-finder. In Proceedings of the fifth Conference on Applied and Natural Language Processing, pages 194-201, Washington, D.C., 1997. ACL.
- [8] D. Freitag. Information Extraction from HTML: Application of a General Machine Learning Approach. Proceedings of the 15'th National Conference on Artificial Intelligence(AAAI-98) Madison, Wisconsin, July 1998.
- [9] Erik Schlyter(2007). Structured Data Extraction. Master's Thesis.
- [10] Feldman , R., Aumann, Y., Finkelstein -Landau, M., Hurvitz , E., Reger , Y., Yaroshevich, A.: A comparative study of information extraction strategies. In CICLing'02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent text processing, pp. 349-359.Springer , Berlin Heidelberg New York (2002).

- [11] Flesca, S., Manco, G., Masciari, E., Rende, E., & Tagarelli, A. (2004). Web wrapper induction: a brief survey. *AI Communications*, 17(2), 57–61.
- [12] Forney, G. D. (1978). The Viterbi algorithm. In *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)* (Vol. 61, pp. 268-278).
- [13] Freitag, D. 1999. *Machine Learning for Information Extraction in Informal Domains*. Ph.D. Dissertation, Carnegie Mellon University.
- [14] Freitag, D., McCallum, A. : *Information Extraction With HMMs and Shrinkage*. In: *Proc. Of AAI Workshop on Machine Learning for IE* , pp. 31-36(1999).
- [15] Grishman, R. (1997). *Information extraction: techniques and challenges*. In *Information Extraction International Summer School SCIE-97* (pp. 10-27). Frascati, Italy.
- [16] I. Muslea, S. Minton, and C. A. Knoblock. *Active Learning with Multiple Views*. *Journal of Artificial Intelligence Research*, 1, pp. 1–31, 2006.
- [17] I. Muslea, S. Minton, and C. Knoblock, “A Hierarchical Approach to Wrapper Induction,” *Proc. Ann. Conf. Autonomous Agents*, pp. 190-197, 1999.
- [18] Knoblock , C.A., Lerman , K., Minton , S., Muslea , I.: *Accurately and reliably extracting data from the web: a machine learning approach* , pp. 275-287(2003).
- [19] Kosala R. and Blockeel, H., “Web Mining Research : A Survey”, *SIGKDD Explorations*, 2(1):1-15, 2000.
- [20] Kushmerick, N. (2000). *Wrapper induction: Efficiency and expressiveness*. *Artificial Intelligence Journal*, 118 (1-2), 15-68.
- [21] L. Eikvil. *Information extraction from world wide web - a survey*. Technical report 945, [http://www.nr.no/documents/samba/research\\_areas-/BAMG/Publications/webIE\\_rep945.ps](http://www.nr.no/documents/samba/research_areas-/BAMG/Publications/webIE_rep945.ps), 1999.
- [22] Leek , T.R. 1997 , *Information Extraction using Hidden Markov Models*, Masters Thesis , UC San Diego.
- [23] Manu Konchady. *Text Mining Application Programming*[Programming Series], Charles River Media,2006.



- [24] Muslea, I., Minton, S., & Knoblock, C. (2003). Active learning with strong and weak views:a case study on wrapper induction. In Proceedings of International Joint Conference on Artificial Intelligence(IJCAI-2003), pp. 415-420.
- [25] Muslea, I., Minton, S., & Knoblock, C. (2001). Hierarchical wrapper induction for semistructured sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4, 93-114.
- [26] Muslea, I. Extraction patterns for Information Extraction Tasks: A Survey. In *AAAI-99 Workshop on Machine Learning for Information*, 1999.
- [27] N. Ashish , C. A. Knoblock . Wrapper Generation for Semi-Structured Internet Sources. *SIGMOD Record*, Vol 26, No.4, pp 8-15, December 1997.
- [28] Nicholas Kushmerick(1997). Wrapper Induction for Information Extraction . Ph.d. Thesis.
- [29] Park, D.,C., Huong, V., Woo, D. N.(2009). Information Extraction System Based on Hidden Markov Model. *Lecture Notes in Computer Science*, pp 52-59.
- [30] Rabiner , L.R: A Tutorial on Hidden Markov Models and Selected Applications in speech Recognition. *Proc. Of IEEE* 77(2), 57-286(1989).
- [31] Rabiner ,L.R: An Introduction to Hidden Markov Model. *IEEE ASSP Magazine*, January 1986.
- [32] S. Chakrabarti., "Mining The Web: Discovering Knowledge from HyperText Data", Morgan Kaufmann Publishers, 2005.
- [33] Seymore, K., McCallum, A., & Rosenfeld, R. (1999). Learning hidden Markov model structure for information extraction. In *Proceedings of 16th national conference on artificial intelligence (AAAI-99)*, Orlando, Florida, USA, pp. 37–42.
- [34] Simoes, G., Galhardas, H., & Coheur, L. (2009). Information extraction tasks:a survey (INESC-ID technical report No. 37/2009). Lisbon, Portugal.
- [35] Soderland, S. (1999). Learning extraction rules for semi-structured and free text. *Machine Learning*, 34, 233-272.
- [36] Tom M. Mitchell. *Machine learning*, McGraw Hill,1997

- [37] Tso B, Chang PY (2007). Mining Free-Structured Information based on Hidden Markov Models. *Expert Syst Appl* 32(1):97-102.
- [38] U. Irmak, and T. Suel. Interactive Wrapper Generation with Minimal User Effort. *InProc. of the 15th Intl. Conf. on World Wide Web (WWW'06)*,2006.
- [39] W. W. Cohen, M. Hurst, and L. S. Jensen. A Flexible Learning System for Wrapping Tables and Lists in Html Documents. *In Proc. of the 11th Intl. World Wide Web Conf. (WWW'02)*, pp. 232–241, 2002.
- [40] Y. Zhai and B. Liu. Extracting Web Data Using Instance-Based Learning. *In Proc. Of 6th Intl. Conf. on Web Information Systems Engineering (WISE'07)*, pp. 318–331,2007.
- [41] Zhang, N. R. (2001). Hidden Markov models for information extraction. Final Project, Stanford NLP Group, Stanford University, CA, USA.