Transaction Models for Web Accessibility

A Dissertation Presented
by
Jalal Uddin Mahmud

to
The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of
Doctor of Philosophy
in
Computer Science
Stony Brook University

August 2008

Stony Brook University
The Graduate School

Jalal Uddin Mahmud

We, the dissertation committee for the above candidate for the degree of Doctor of Philosophy, hereby recommend acceptance of this dissertation.

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Transaction Models for Web Accessibility

by

Jalal Uddin Mahmud

Doctor of Philosophy

in

Computer Science

Stony Brook University

2008

The Web has evolved into a dominant digital medium for conducting many types of online transactions such as shopping, paying bills, making travel plans, etc. Such transactions typically involve a number of steps spanning several web pages. For sighted users these steps are relatively straightforward to do with graphical web browsers. But they pose tremendous challenges for visually impaired individuals. This is because screen readers, the dominant assistive technology used by visually impaired users, function by speaking out the screen's content serially. Consequently, using them for conducting transactions can cause considerable information overload.

But usually one needs to browse only a small fragment of a web page to do a step of a transaction (e.g., choosing an item from a search results list). Based on this observation this dissertation develops a model-directed transaction framework to identify, extract and aurally render only the "relevant" page fragments in each step of a transaction. The framework uses a process model to encode the state of the transaction and a concept model to identify the page fragments relevant for the transaction in that state. The two models are constructed from labeled transaction sequences using traditional classification and automata learning methods.

Next, we relax the requirement of needing fully labeled training data. Specifically, we present a framework to mine transaction models from partially labeled click stream data generated by transactions, where some or all the labels could be missing. Not having to rely exclusively on (manually) labeled click stream data has important benefits: Visually impaired users do not have to depend on third party (e.g., sighted users) for constructing transaction models. This makes it possible to mine personalized models from transaction click streams associated with sites that visually impaired users visit regularly. Since partially labeled data is relatively easier to generate, scaling up the construction of domain-specific transaction models (e.g., shopping, airline reservations, bill payments, etc.) is feasible. Lastly, adjusting the performance of deployed models over time with new training data is also doable.

In terms on techniques used for mining we expand our repertoire to include web content analysis to partition a web page into segments consisting of semantically related content elements, contextual analysis of data surrounding clickable elements in a page and clustering of page segments based on contextual analysis.

We provide qualitative and quantitative experimental evidence of the practical effectiveness of our models in improving user experience when conducting online transactions with non-visual modalities.

To my father.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my wholehearted thanks to my advisor, Professor I.V. Ramakrishnan, for his guidance and generous support throughout my graduate studies. His enthusiasm for research motivated me to solve interesting research problems. He inspired me to be an independent and creative researcher, and I hope to carry this on throughout my career. I would also like to thank the other members of my committee: Professor C.R. Ramakrishnan, Professor David Warren and Professor Amanda Stent and my external thesis examiner Dr. Yan-Wu Huang from IBM T.J. Watson Research.

I am grateful to my mother who stayed back in Bangladesh during the last four years of my PhD studies. Without her support, I could not have come this path after I lost my father when I was only eighteen.

I express my thanks to my wife, Nusrat Sultana Priyanka who was a source of inspiration for me during the last couple of years of my graduate studies.

My sincere thanks to all the systems and administrative staff in the department including Cindy Scalzao, Betty Knitweiss, Brian Tria, and Kathy Germana.

I was fortunate to work with some very brilliant and motivating fellow graduate students during the last four years. Specially, I like to express my thanks to Yevgen Borodin, with whom I coauthored several papers. My special thanks to Zan Sun, Chang Zhao and apologies to all the others whom I could not enumerate in this limited space.

Chapter 1

Introduction

Over a relatively short period of time the Web has evolved into an ecosystem where anyone can communicate, find information, shop, bank, and pay bills online. With the expansion of Web users, Web transaction activities (e.g., buying a CD player from an online store, paying a utility bill) are also growing rapidly.

There are two essential components to a Web transaction: (i) locating the relevant content, such as a search form or the desired item in a Web page, and (ii) performing a sequence of steps, such as filling out a search form, selecting an item from the search result and doing a checkout. For completing a transaction these steps usually span several pages.

The primary mode of interaction with the Web is via graphical browsers, which are designed for visual interaction. Most Web pages contain banners, ads, navigation bars, and other data distracting us from the information. As we browse the Web, we have to filter through a lot of irrelevant data. We quickly scan through the rich engaging content in Web pages scripted for e-commerce and locate the objects of interest easily. Moreover, the spatial organization of content in these pages helps users comprehend the sequence of steps necessary to complete a transaction.

Now consider scenarios where visual interaction is impossible (*e.g.*, when the user is a visually handicapped individual). Speech interfaces offer narrow interaction bandwidths making it cumbersome and tedious to get to the pertinent content in a page. For instance, state-of-the-art screen readers and audio browsers (*e.g.*, JAWS [1], IBM's Home Page Reader [11], Windows-Eyes [37]) provide almost no form of filtering of the content in a Web page, resulting in severe *information overload*. This problem is further exacerbated when such an interaction spans several pages as in an online transaction.

In particular the loss of spatially organized content due to simple reading of the screen contents makes it difficult for users to comprehend the sequence of transactional steps. While content summarization can compensate somewhat for this loss, it alone is inadequate for handling the information overload that the user faces. Thus, there is a need for developing techniques to facilitate Web transactions using non-visual modalities that are far less cumbersome than current approaches.

We capture the two aspects of a transaction, namely its operation sequence and content identification, by a *process model* and an *ontology* respectively. The ontology describes the set of *semantic concepts* occurring in Web pages, which are considered essential for conducting Web transactions in a particular domain, e.g., online shopping.

The circled elements in Figure 1 are examples of such concepts. The sequence of actions is captured by a process model, which is a deterministic finite state automaton. Each state, representing an atomic operation in a transaction, is associated with a set of semantic concepts drawn from the ontology. When the model makes a transition to a state during the course of a transaction, a Web page is provided to the state as an input. If the concepts associated with the state are present in the page, then they alone are identified and presented to the user.

Thus a process model can overcome the information overload problem for non-visual Web transactions. For instance, if the page shown in Figure 1 (a) is given as the input to a state associated with the concepts “Item Taxonomy” and “Search Result”, only the two circled items in the figure will be identified and presented to the user. In this thesis, we establish a formal definition for such a transaction model. In our initial work [81, 82], we demonstrated that by coupling content semantics with model-directed navigation facilitated by the process model, we can overcome the information overload problem by delivering relevant content at every step of the transaction. A supervised machine learning technique was used to learn the transaction model from manually labeled Web transaction sequences. Sighted users labeled these sequences, as well as instances of concepts in a Web page. A predefined ontology describing concepts and associated operations was used.

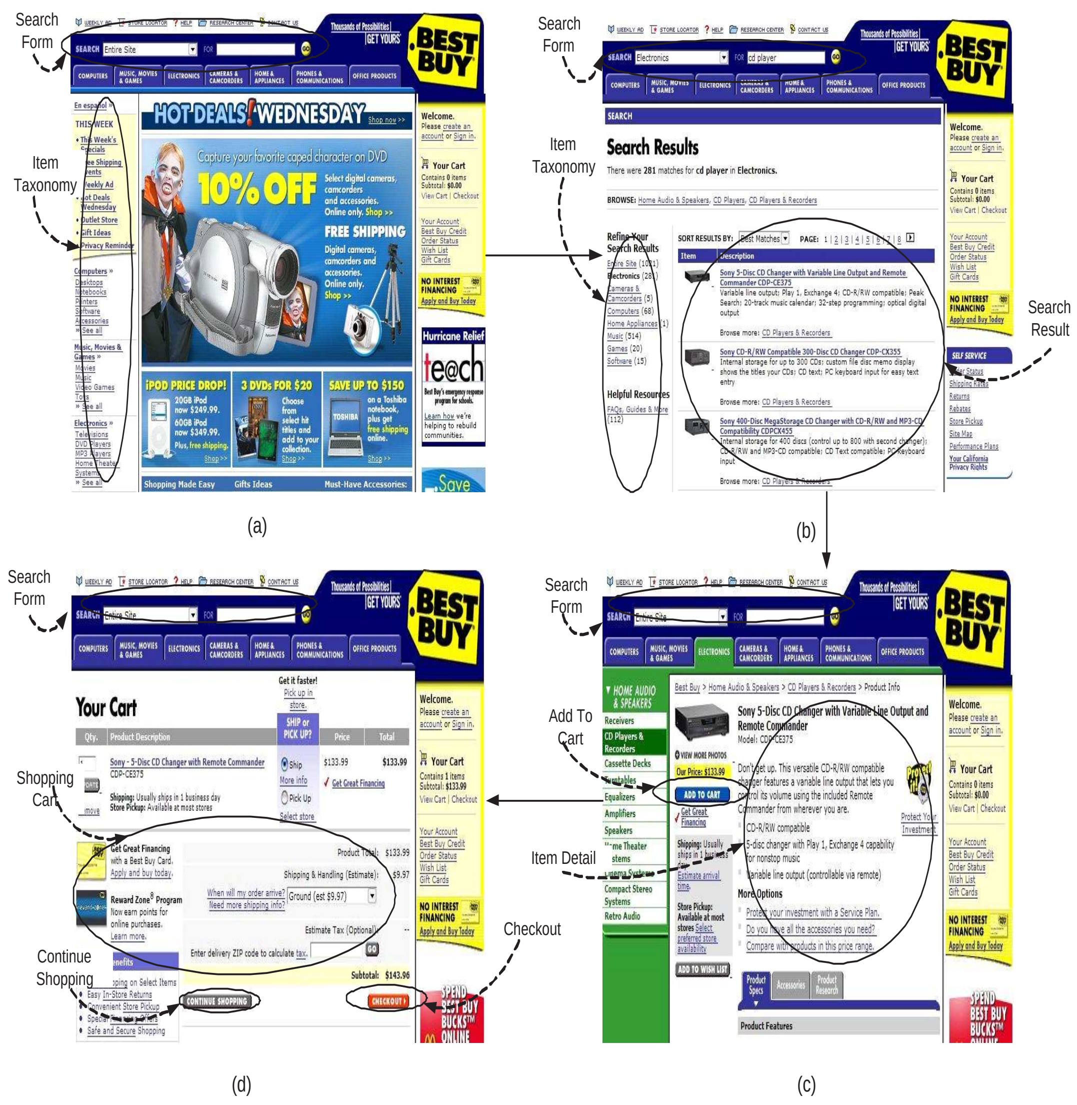
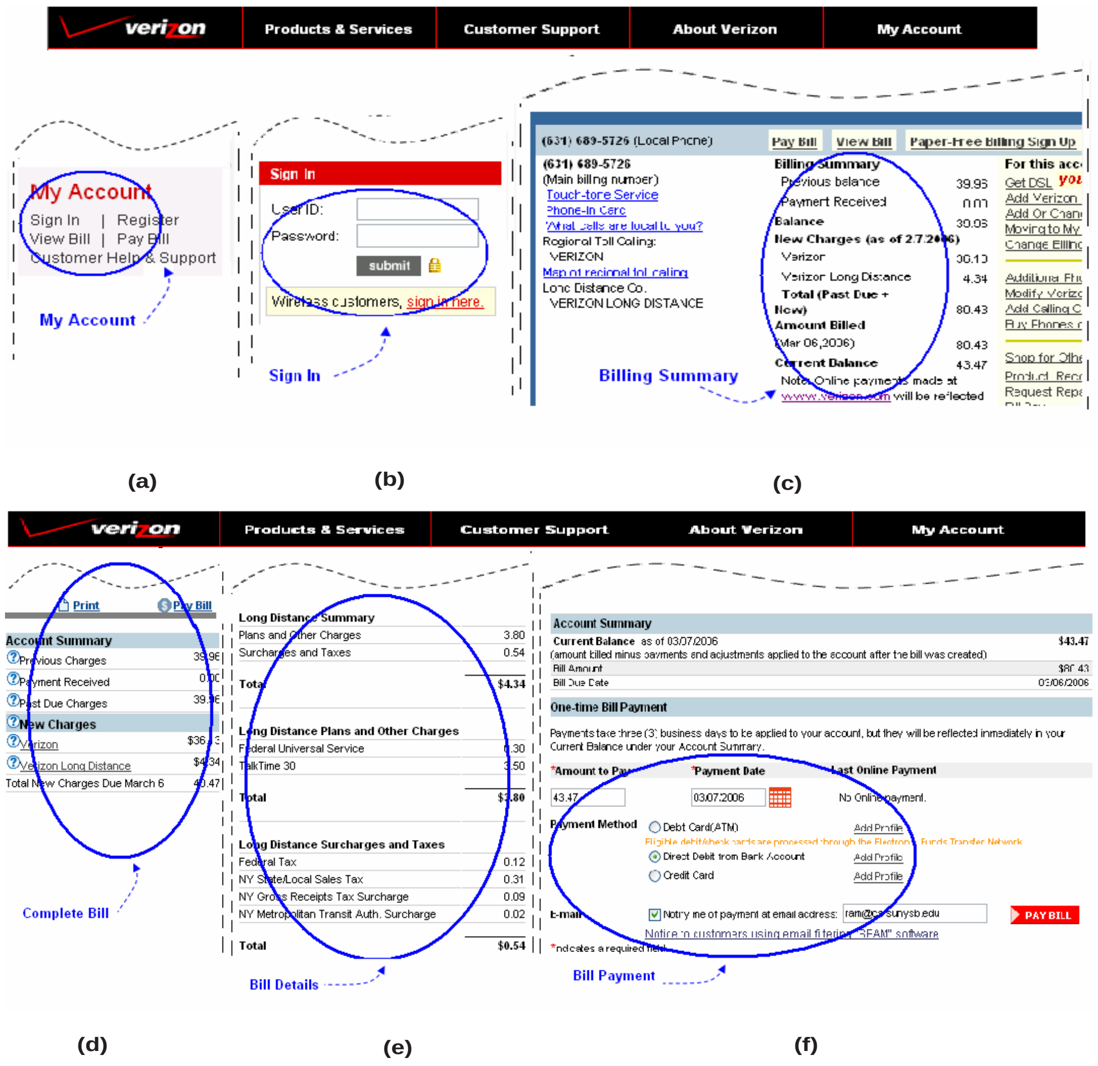


Figure 1: A Web Transaction in Online Shopping Domain



My Account
Sign In | Register
View Bill | Pay Bill
Customer Help & Support

Sign In
User ID:
Password:
submit
Wireless customers, sign in here.

(631) 689-5726 (Local Phone) Pay Bill View Bill Paper-Free Billing Sign Up

(631) 689-5726
(Main billing number)
Touch-tone Service
Phone-In Card
What calls are local to you?
Regional Toll Calling:
VERIZON
Map of regional toll calling
Long Distance Co.
VERIZON LONG DISTANCE

Billing Summary	
Previous balance	39.96
Payment Received	0.00
Balance	39.96
New Charges (as of 2.7.2006)	
Verizon	36.13
Verizon Long Distance	4.34
Total (Past Due + New)	80.43
Amount Billed (Mar 06,2006)	80.43
Current Balance	43.47

Note: Online payments made at www.verizon.com will be reflected

For this acc
Get DSL
Add Verizon
Add Or Chan
Moving to My
Change Billing
Additional Ph
Modify Verizo
Add Calling C
Buy Phones
Shop for Othe
Product Reco
Request Repa

Print Pay Bill

Account Summary	
Previous Charges	39.96
Payment Received	0.00
Past Due Charges	39.96
New Charges	
Verizon	$36.13
Verizon Long Distance	$4.34
Total New Charges Due March 6	40.47

Long Distance Summary	
Plans and Other Charges	3.80
Surcharges and Taxes	0.54
Total	**$4.34**
Long Distance Plans and Other Charges	
Federal Universal Service	0.30
TalkTime 30	3.50
Total	**$3.80**
Long Distance Surcharges and Taxes	
Federal Tax	0.12
NY State/Local Sales Tax	0.31
NY Gross Receipts Tax Surcharge	0.09
NY Metropolitan Transit Auth. Surcharge	0.02
Total	**$0.54**

Account Summary	
Current Balance as of 03/07/2006 (amount billed minus payments and adjustments applied to the account after the bill was created)	**$43.47**
Bill Amount	$80.43
Bill Due Date	03/06/2006

One-time Bill Payment

Payments take three (3) business days to be applied to your account, but they will be reflected immediately in your Current Balance under your Account Summary.

*Amount to Pay	*Payment Date	Last Online Payment
43.47	03.07.2006	No Online payment.

Payment Method
( ) Debit Card(ATM) Add Profile
Eligible debit/check cards are processed through the Electronic Funds Transfer Network
(●) Direct Debit from Bank Account Add Profile
( ) Credit Card Add Profile

E-mail ✓ Notify me of payment at email address: ram@cs.sunysb.edu PAY BILL
Notice to customers using email filtering "SPAM" software

*indicates a required field

Figure 2: A Web Transaction in Utility Bill Payment Domain

However, such a supervised approach is not readily scalable to different content domains. Besides, the supervised approach makes it almost impossible for blind users to build their own personalized transaction models for Web sites that they have to use regularly, e.g. online banking, utility bill payment.

Utilizing unsupervised mining methods that learn models from unlabeled data, we can address the above limitations of supervised methods. Towards that, we develop a technique to learn transaction models from unlabeled or partially labeled transaction sequences. The salient aspects of this technique are highlighted below:

- We develop the notion of context [19, 57–60] for Web objects (e.g. link, button).
- We use the context of the Web object accessed by the user in a Web transaction, coupled with geometric segmentation [39] to retrieve the Web page segment containing that object.
- These segments are clustered based on generic features. The clustered segments represent the semantic concepts in the ontology. A SVM-classifier [89] is automatically learnt for each such concept.
- In our prior work, we used a DFA learning [70] technique to learn the process model. DFA learning requires a sizable number of negative examples which are often difficult to obtain. For unsupervised model mining, we have developed a new process model learning algorithm which only uses complete transaction sequences (positive examples), i.e. completed transactions.

1.1 Organization

The rest of this thesis is organized as follows: Chapter 2 describes the data structures and Web content analysis techniques (geometric analysis and contextual analysis), that have been developed and are used for transaction model mining. In Chapter 3, definitions and formal semantics are established for the rest of the thesis. Chapter 4 describes our initial work to mine transaction models using a supervised approach. Chapter 5 describes the unsupervised approach to mine transaction models. Related research appears in Chapter 6 and conclusion in Chapter 7.

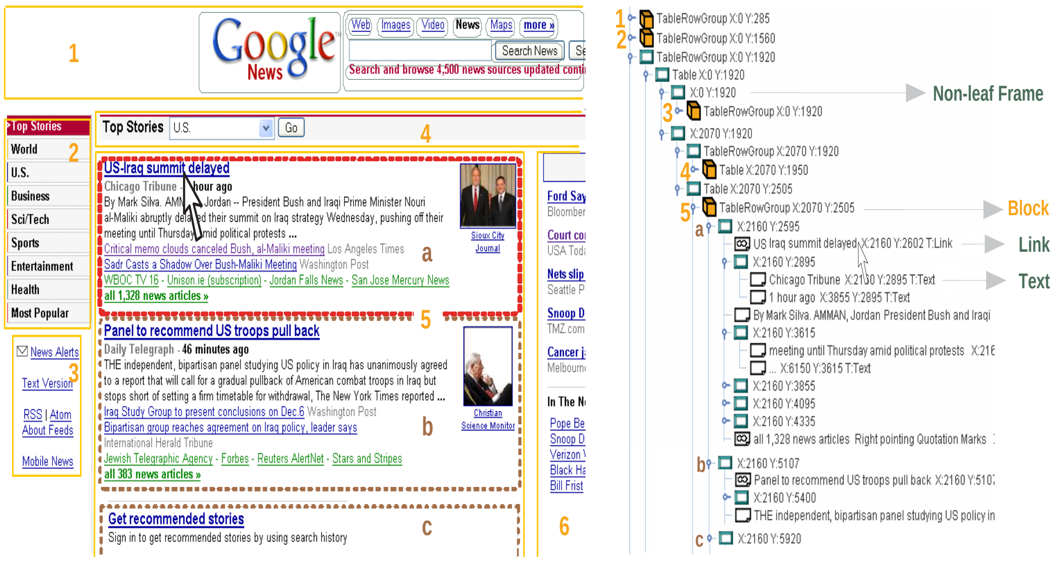
Chapter 2

Preliminaries

In this chapter, we present a brief description of the data structure we use for Web content analysis as well as some of the content analysis techniques, we use for transaction model mining.

2.1 Frame Tree

The *Frame Tree* [59, 60] of a page is Mozilla [62]'s internal representation of a Web page, *after* the Web page has been presented for rendering on the screen. This is a tree-like data structure that contains Web page content, along with its formatting information, which specifies how that Web page has to be rendered on the screen. A frame tree is composed of nested $frames$, so that the entire page is a root frame, containing other nested frames down to the smallest individual objects on the page. For example, Figure 3 shows a snapshot of the Google News front page and the corresponding frame-tree partially expanded to demonstrate the types of frames. We distinguish between the following classes of frames: text, links, images, image-links, and non-leaf frames. Section 1 of Figure 3 (a) shows several nested frames enclosed by rounded boxes. We will continue referring to any node of a frame tree as a $frame$.



US-Iraq summit delayed
Chicago Tribune - 1 hour ago
By Mark Silva. AMMAN, Jordan -- President Bush and Iraqi Prime Minister Nouri al-Maliki abruptly delayed their summit on Iraq strategy Wednesday, pushing off their meeting until Thursday amid political protests ...
Critical memo clouds canceled Bush, al-Maliki meeting Los Angeles Times
Sadr Casts a Shadow Over Bush-Maliki Meeting Washington Post
WBOC TV 16 - Unison.ie (subscription) - Jordan Falls News - San Jose Mercury News
all 1,328 news articles »

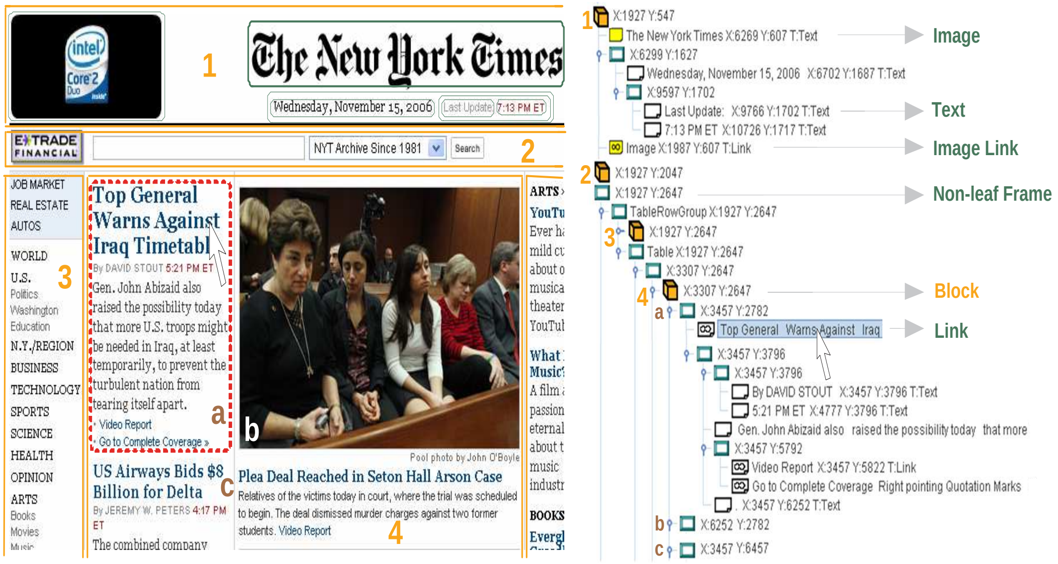
Panel to recommend US troops pull back
Daily Telegraph - 46 minutes ago
THE independent, bipartisan panel studying US policy in Iraq has unanimously agreed to a report that will call for a gradual pullback of American combat troops in Iraq but stops short of setting a firm timetable for withdrawal, The New York Times reported ...
Iraq Study Group to present conclusions on Dec.6 Washington Post
Bipartisan group reaches agreement on Iraq policy, leader says
International Herald Tribune
Jewish Telegraphic Agency - Forbes - Reuters AlertNet - Stars and Stripes
all 383 news articles »

Get recommended stories
Sign in to get recommended stories by using search history

```
1 TableRowGroup X:0 Y:285
2 TableRowGroup X:0 Y:1560
  TableRowGroup X:0 Y:1920
    Table X:0 Y:1920
      X:0 Y:1920
    3 TableRowGroup X:0 Y:1920
      X:2070 Y:1920
        TableRowGroup X:2070 Y:1920
        4 Table X:2070 Y:1950
        Table X:2070 Y:2505
        5 TableRowGroup X:2070 Y:2505
          a X:2160 Y:2595
              US Iraq summit delayed X:2160 Y:2602 T:Link
              X:2160 Y:2895
                Chicago Tribune X:2160 Y:2895 T:Text
                1 hour ago X:3855 Y:2895 T:Text
              By Mark Silva. AMMAN, Jordan President Bush and Iraqi
              X:2160 Y:3615
                meeting until Thursday amid political protests X:216
                ... X:6150 Y:3615 T:Text
              X:2160 Y:3855
              X:2160 Y:4095
              X:2160 Y:4335
              all 1,328 news articles Right pointing Quotation Marks
          b X:2160 Y:5107
              Panel to recommend US troops pull back X:2160 Y:510
              X:2160 Y:5400
              THE independent, bipartisan panel studying US policy in
          c X:2160 Y:5920
```

(a) Source Page (b) Source Frame Tree

Figure 3: Context Identification in Google News



Top General Warns Against Iraq Timetable
By DAVID STOUT 5:21 PM ET
Gen. John Abizaid also raised the possibility today that more U.S. troops might be needed in Iraq, at least temporarily, to prevent the turbulent nation from tearing itself apart.
Video Report
Go to Complete Coverage »

US Airways Bids $8 Billion for Delta
By JEREMY W. PETERS 4:17 PM ET
The combined company

Pool photo by John O'Boyle
Plea Deal Reached in Seton Hall Arson Case
Relatives of the victims today in court, where the trial was scheduled to begin. The deal dismissed murder charges against two former students. Video Report

```
1 X:1927 Y:547
    The New York Times X:6269 Y:607 T:Text
    X:6299 Y:1627
      Wednesday, November 15, 2006 X:6702 Y:1687 T:Text
      X:9597 Y:1702
        Last Update: X:9766 Y:1702 T:Text
        7:13 PM ET X:10726 Y:1717 T:Text
    Image X:1987 Y:607 T:Link
2 X:1927 Y:2047
  X:1927 Y:2647
    TableRowGroup X:1927 Y:2647
    3 X:1927 Y:2647
      Table X:1927 Y:2647
        X:3307 Y:2647
        4 X:3307 Y:2647
          a X:3457 Y:2782
              Top General Warns Against Iraq
              X:3457 Y:3796
                X:3457 Y:3796
                  By DAVID STOUT X:3457 Y:3796 T:Text
                  5:21 PM ET X:4777 Y:3796 T:Text
                Gen. John Abizaid also raised the possibility today that more
                X:3457 Y:5792
                  Video Report X:3457 Y:5822 T:Link
                  Go to Complete Coverage Right pointing Quotation Marks
                . X:3457 Y:6252 T:Text
          b X:6252 Y:2782
          c X:3457 Y:6457
```

(a) Source Page (b) Source Frame Tree

Figure 4: Context Identification in NYTimes

2.2 Geometric Segmentation

We use the observation that semantically related content elements in a web page exhibit spatial locality [64, 65] and often share the same alignment (matching X or Y coordinate) on a web page. Since a frame tree represents the layout of a web page, we infer that geometrical alignment of frames may imply semantic relationship between their respective content. A frame is **consistent** if its descendants are consistently aligned either along X or Y axes.

A *Maximal Semantic Block*, or simply **block**, is the largest of the consistent frames on the path from a leaf to the root of a frame tree. Thus, it is likely to be the largest possible cluster containing semantically related content items. For example, Figure 4(b) shows how the alignment information is used to cluster the New York Times Web page into maximal semantic blocks: banner labeled as 1, search - 2, taxonomy - 3, and news - 4.

The *FindBlocks* algorithm is used to find the blocks in a frame tree. The algorithm runs a depth-first search over the frame tree and recursively determines whether the frames are consistent, ignoring the alignment of leaf-frames. A frame is *consistently X-aligned* if all of its non-leaf descendants are X-aligned. Similarly, a frame is *consistently Y-aligned* if all of its non-leaf descendants are Y-aligned. Otherwise, the frame is not considered to be consistent. In this case, all of its children are marked as *blocks*.

Algorithm *FindBlocks*
Input: *Frame*: node of a frame tree
Output: *Blocks*: set of maximal semantic blocks

```
1.  Identify all children $C_1, C_2, \ldots, C_m$ of Frame
2.  $Frame.IsConsistent \leftarrow$ true
3.  for $j \leftarrow 1$ to $m$
4.      do if $C_j.IsLeaf$ = false
5.          then FindBlocks($C_j$)
6.              if $C_j.Alignment$ = NONE
7.                  then $Frame.IsConsistent \leftarrow$ false
8.  if $Frame.IsConsistent$ = false
9.      then for $j \leftarrow 1$ to $m$
```

```
10.              do if $C_j.Alignment \neq$ NONE
11.                  then $Blocks \leftarrow Blocks \cup \{C_j\}$
12.    else  $Frame.Alignment \leftarrow GetAlignment(Frame)$
13.          if $Frame.Alignment$ = NONE
14.            then for $j \leftarrow 1$ to $m$
15.                   do if $C_j.Alignment \neq$ NONE
16.                        then $Blocks \leftarrow Blocks \cup \{C_j\}$
17. return $Blocks$
```

The *FindBlocks* algorithm uses the *GetAlignment* algorithm to check whether the children of a frame have matching alignment. That is, the *GetAlignment* algorithm determines that a frame is *X-aligned* if all of its children are aligned on the left, right, or center of the X-axis. Y-alignment of a frame is computed in a similar fashion.

Algorithm *GetAlignment*
Input: *Frame*: node of a frame tree
Output: *Alignment* : alignment of *Frame*'s descendants

```
1.  Identify all children $C_1, C_2, \ldots, C_m$ of $Frame$
2.  $XFirst \leftarrow C_1.X$
3.  $YFirst \leftarrow C_1.Y$
4.  $XAlignedDescendants \leftarrow$ true
5.  $YAlignedDescendants \leftarrow$ true
6.  $Alignment \leftarrow$ NONE
7.  for $j \leftarrow 2$ to $m$
8.      do if $C_j.IsLeaf$ = false
9.           then $XCord \leftarrow C_j.X$
10.               $YCord \leftarrow C_j.Y$
11.               if $XCord \neq XFirst$
12.                 then $XAlignedDescendants \leftarrow$ false
13.               if $YCord \neq YFirst$
14.                 then $YAlignedDescendants \leftarrow$ false
15.               if $C_j$ is not X Aligned
16.                 then $XAlignedDescendants \leftarrow$ false
```

Questions	Possible Responses
Does this block contain important content ?	Yes/No
What amount of the block contents are geometrically aligned ?	All/Some/None
What amount of the block contents are semantically related ?	All/Some/None

Table 1: Questionnaire for Evaluation of Geometric Segmentation Algorithm

```
17.                 if $C_j$ is not Y Aligned
18.                   then $YAlignedDescendants$ ←false
19. if $XAlignedDescendants$ = true
20.   then $Alignment$ ←$XAlign$
21. if $YAlignedDescendants$ = true
22.   then $Alignment$ ←$YAlign$
23. return $Alignment$
```

2.2.1 Evaluation of Geometric Segmentation

We evaluated the algorithm *FindBlocks* using 20 pages from 20 Websites, 5 Web sites in each of the domains: *shopping, news, services, personal.* 10 sighted CS graduate students were used as evaluators. To help them, a visual tool was designed [18]. None of the evaluators were HearSay [20] developers. They were trained on how to use the evaluation tool. Each Webpage was evaluated by 2 evaluators. For each Webpage used in the evaluation, the geometric segmentation algorithm was applied to retrieve the list of blocks. Then, evaluators were presented with a set of questions for each of the segments blocks. The questions are shown in table 1. They were also asked whether there was content on this page that should be grouped into a block but was not identified as a block by the algorithm.

Using the responses collected from the evaluators, we computed recall, precision and F-measure of the segmentation algorithm. Specifically we computed these metrics for each feature of the algorithm, i.e. geometric alignment and semantic relatedness. Let C denote the total number of blocks for which all contents are geometrically aligned, P denote the total number of blocks for which some contents are geometrically aligned, I denote the total number of blocks for which no contents are geometrically aligned, and M denote the total number of missing blocks(i.e. there

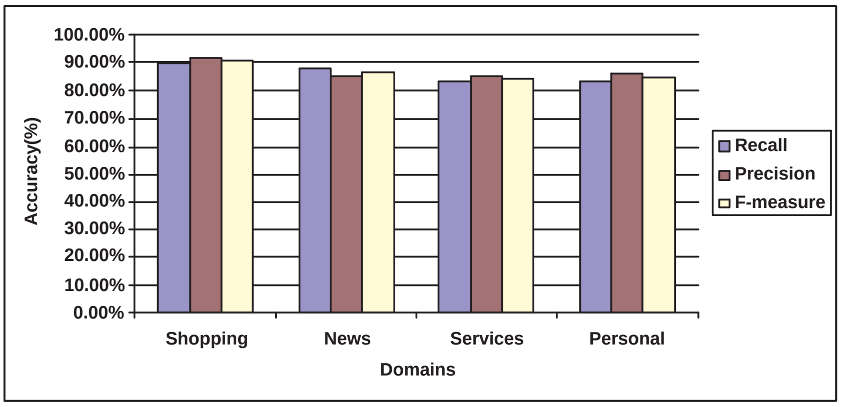


Figure 5: Geometrical Alignment Accuracy for Core Blocks

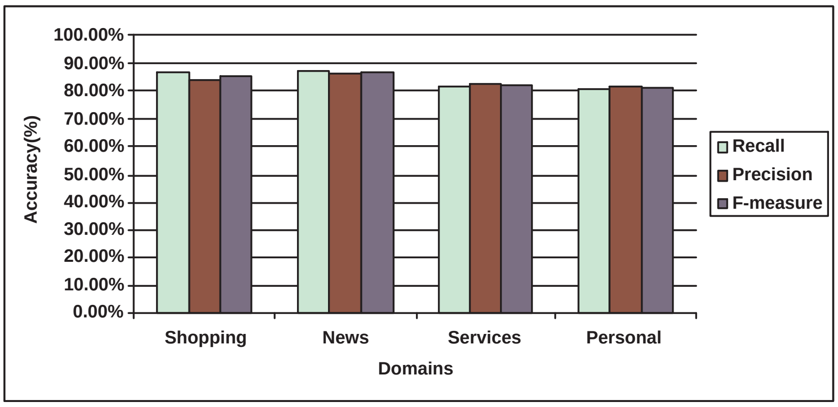


Figure 6: Geometrical Alignment Accuracy for All Blocks

was content on this page that should be grouped into a block but was not identified as a block by the algorithm). Then, recall is defined as (C + P*0.5)/(C+P+M) and precision is defined as (C + P*0.5)/(C+P+I). Note that, we give 0.5 weight to each partially correct response, assuming that it is equally probable to be correct or incorrect. F-measure is the simple harmonic mean of recall and precision. Using a similar formula, we calculated the metrics for semantic relatedness of blocks.

Figures 5, 6, 7, 8 show the experimental results. We observe that geometric segmentation algorithm exhibits reasonably high precision, recall and f-measure for all cases.

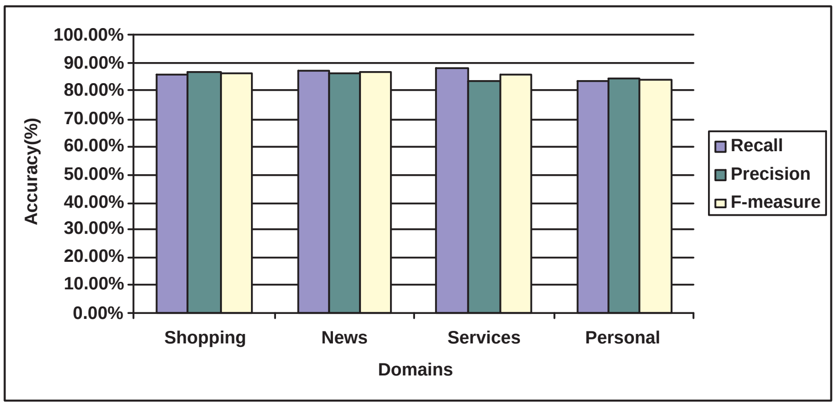


Figure 7: Semantic Relatedness Accuracy for Core Blocks

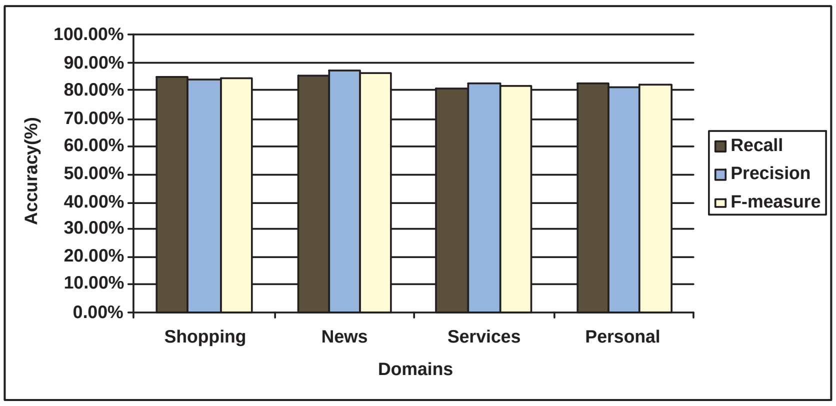


Figure 8: Semantic Relatedness Accuracy for All Blocks

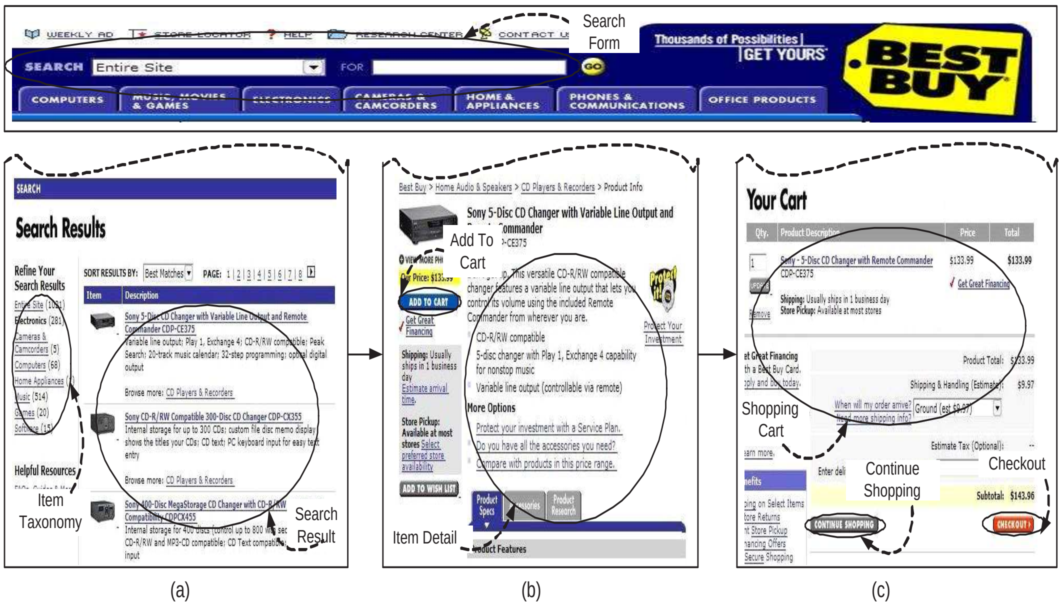


Figure 9: Concept Segments in Web Pages

2.3 Web Objects

A **Web Object** is an atomic element of a Web page. A Web object is associated with an id that uniquely identifies that object. Each Web object also has several attributes, e.g. text, style, location.

For example, the button Checkout is a Web object in figure 9 (c). The text "Checkout", the button's geometric coordinates in the Web page, etc. are various attributes of this Web object. Let Obj denote a Web object and $Attr$ denote the set of attributes of that object. Let L denote the **Attribute Mapping** function to retrieve the set of attributes $Attr$ of a Web object Obj, i.e. $L(Obj) = Attr$.

Let us consider the button "Checkout" in figure 9 (c) which has unique id $F567585$ and the following four attributes.

Text: "Checkout"

X-Cord: 220

Y-Cord: 85

Style: Button

Given the unique id $F567585$ as parameter, the function L returns the set $Attr = \{“Checkout'', 220, 85, Button\}$

2.4 Context Analysis

Here we describe how we collect the context of a Web object. To collect the context, a topic-detection algorithm is applied to the information surrounding the object. We gather the text that shares a common topic [5, 6] with the textual content of the Web object as context.

2.4.1 Context Identification

The Context of a Web object is the content around the Web object that maintains the same *topic* as the text content of the Web object.

Consider Figure 4, showing the front page of The New York Times Web site and the corresponding frame tree. The link (a Web object) “Top General Warns Against Iraq Timetable” is indicated by an arrow. The context of this is the text surrounded by the dotted line. Notice how the topic changes from one headline to another.

A block, produced by the Geometric Segmentation algorithm, ideally represents a segment of text on the same topic, but may have several topics within it. Therefore, we limit topic boundary detection and context collection to the block containing the Web object. Context collection begins from the content of the object and expands around the object until the topic of the text changes. A simple cosine similarity technique is used to detect the boundaries of the topic, see equation (1).

The *FindContext* algorithm initializes the *Context* multiset with the words and word combinations (bigram and trigram), excluding the function words (i.e. stop words), from the object and its non-link siblings; the text in the link siblings is ignored because links tend to be semantically independent of each other, i.e. have different topics (e.g., each news headline is a link and starts a new topic). It then collects all text pertaining to the same topic around the Web object, adding the words to the *Context* multiset.

In the NYTimes example, Figure 4(a), the user follows the link (a Web object) “Top General Warns Against Iraq Timetable”, indicated by the mouse pointer. The multiset is initialized with the words collected from the link node e.g. “general”, “top”, “warns”, etc. of the frame tree, indicated by a mouse pointer in Figure 4(b), as well as from the non-leaf sibling (e.g. the node with X coordinate 3457, Y coordinate 3796) which follows the link node e.g. “david”, “stout”,etc. . The multiset now contains single words (e.g. “general”, “david”, “stout”, “gen” “john”, etc.), their bigrams (e.g., “david stout”, “gen john”), and trigrams (e.g. “gen john abizaid”).

After the initialization stage, the context of the link is collected, starting from the parent frame of the link node, by expanding the context to include the frame’s siblings. The siblings are divided into *PredList* and *SuccList*, containing the predecessor and successor siblings respectively, to expand the context window in both directions. Next, the geometric distances[1] are calculated between the initial frame and its siblings and the siblings are sorted accordingly.

Again, in our example, the parent frame of the link in the frame tree is the node labeled as *“a”* in Figure 4(b). The node does not have any predecessor siblings. Its successor siblings, labeled as *“b”* and *“c”*, are respectively 2795 and 3675 pixels away from frame *“a”*. Hence, we start with the sibling “b”, construct multiset *SText* from the sibling’s text, and compare its content to the content of the *Context* multiset. The comparison is done using cosine similarity of the multisets. More formally, for any two multisets M_1 and M_2, their cosine similarity is defined as:

$$Cos(M_1, M_2) = \frac{|M_1 \cap M_2|}{\sqrt{|M_1|}\sqrt{|M_2|}} \tag{1}$$

In the above formula, each multiset, created from a passage of text, is considered to be a vector. The cosine of the angle between the vectors is equal to 1 if the passages are identical, and 0 if they are dissimilar. Two multisets are said to be similar if their cosine similarity is above a threshold. The threshold that best determines whether a topic changes between the *Context* and the *SText* multisets is computed statistically (described in Section 2.4.3).

[1]Geometric distance between two frames is the Euclidean distance between their upper-left corners on the screen.

If the cosine similarity between the multisets is above the threshold, i.e. a topic boundary is not detected, the multisets are merged. Otherwise, expansion of the context window in that direction is stopped. The process continues until the *Block* boundary is reached or when there is no direction along which to expand. At that point, the algorithm returns the *Context* multiset as the context of the link.

Algorithm *FindContext*
Input: *ObjectNode*: leaf-frame containing the Web Object
Output: *Context*: multiset with collected context

```
1.  Context ← non-function words, their bigrams and trigrams from ObjectNode
    and its non-link siblings
2.  Let ancesBlock be the ancestor Block of ObjectNode
3.  if $ancesBlock \neq ObjectNode.Parent$
4.    then Node ← ObjectNode.Parent
5.         Expand ← true
6.         repeat
7.             ChildList ← Node.Parent.Children
8.             Let PredList and SuccList be the lists of predecessors and succes-
               sors of Node in ChildList, sorted by their geometric distance from
               Node
9.             StopExpand ← false
10.            repeat
11.                Sibling ← PredList.Next
12.                SText ← non-function words, their bigrams, trigrams from
                   Sibling
13.                Similarity ← $Cos(Context, SText)$
14.                if $Similarity > Threshold$
15.                  then Context ← $Context \cup \{SText\}$
16.                  else StopExpand ← true
17.                       Expand ← false
18.            until PredList.IsLast or StopExpand
19.            Repeat line 9 to 17 for SuccList
20.            Node ← Node.Parent
21.        until Node = ancesBlock or Expand = false
22. return Context
```

Continuing with the example in Figure 4, the algorithm collects the text from the closest sibling frame *“b”*, corresponding to the news item “Plea Deal in Seton Hall Arson Case”. The multiset *SText*, constructed for this frame, now contains {“plea”, “deal”, “seton”, ..., “plea deal”, ..., “plea deal seton”, ...}. It computes the cosine similarity of the *Context* and a *SText* multisets, which turn out to be below the threshold. The algorithm detects a topic boundary between the content of the multisets and, therefore, stops expanding the context window and returns the *Context* multiset. The context of the followed link, Figure 4(a), is enclosed by the dotted line.

2.4.2 Context Segment

A **Context Segment** of a Web object is the segment of the Web page, where the content of the section maintains the same topic as the text content of the Web object.

The Context analysis algorithm is used to retrieve the context surrounding a Web object. However, the context collection algorithm is modified to return a **Context Segment** instead of the context multiset containing words, bigrams and trigrams. This modification is straightforward. Start with a single frame tree node, that contains the Web object. This frame tree node represents our initial context segment. Next try to expand context in both directions using the topic boundary detection algorithm. As the context is expanded, update the context segment in both directions.

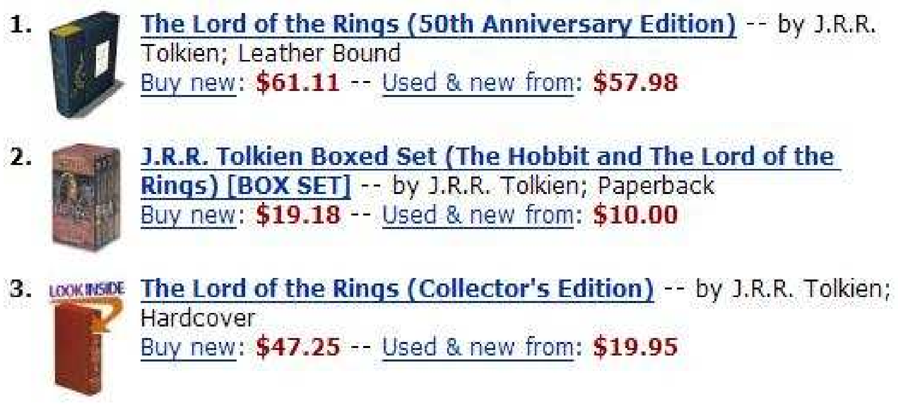
For example, the section of the web page marked as dotted rectangle is the context segment for the link pointed to by the arrow in Figure 4 (a).

A **Context Segment** is repeated if its presentation pattern is repeated within a geometric segment. For example, Figure 10 illustrates a repeated context segment.

For another example, the section of the web page marked as dotted rectangle is the context segment for the link pointed to by the arrow in Figure 3 (a). Note how the presentation pattern of this segment is repeated within the geometric segment (each news items are presented using the same pattern).

2.4.3 Evaluation

We used twenty-five Web sites for evaluation and data collection, 5 Web sites in each of 5 content domains: *news, books, consumer electronics, office supplies,* and *informational*. The informational category included various Web sites, such


1. The Lord of the Rings (50th Anniversary Edition) -- by J.R.R. Tolkien; Leather Bound
Buy new: $61.11 -- Used & new from: $57.98

2. J.R.R. Tolkien Boxed Set (The Hobbit and The Lord of the Rings) [BOX SET] -- by J.R.R. Tolkien; Paperback
Buy new: $19.18 -- Used & new from: $10.00

3. The Lord of the Rings (Collector's Edition) -- by J.R.R. Tolkien; Hardcover
Buy new: $47.25 -- Used & new from: $19.95

Figure 10: An Example of a Repeated Pattern

as MTA Long Island Rail Road [2] and Medicaid[3]. Thirty CS graduate students were used as evaluators. It was impractical to get quantitative measurements of the accuracy of context collection algorithm with blind users. This is because, we did our evaluation for hundreds of pages and from each page we asked our evaluators to select a link and its context. It takes a lot of time for a blind user to reach the link, (compared to sighted users). Sighted users can quickly decide what is the context surrounding a link and can select that. However, it takes a significant amount of time for a blind user to listen to the content of surrounding text and decide what is the context of the link.

Therefore, we used sighted students to obtain the accuracy of the context collection algorithm.

2.4.3.1 Data Collection

To have an efficient infrastructure for experiments, a visual tool for viewing frame trees as well as collecting data was designed [18]. We also embedded a Web browser to aid the data collection. We manually collected around *1000* Web pages to calculate a threshold for topic identification. During the data collection stage, the participants were asked to select any link and the context around it on the

[2]http://www.mta.nyc.ny.us/lirr
[3]www.cms.hhs.gov/home/medicaid.asp

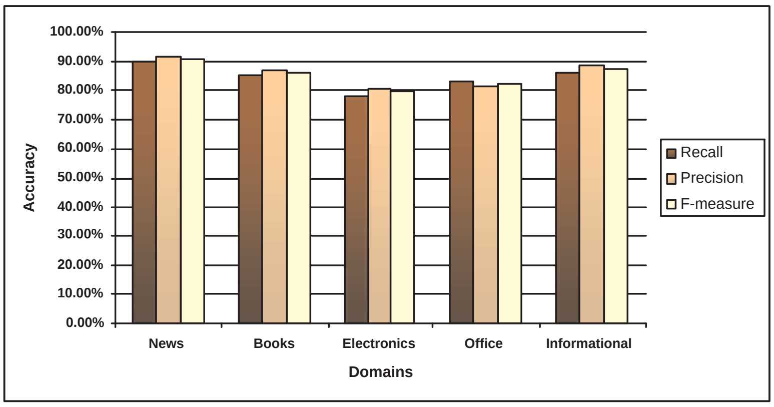


Figure 11: Performance of Context Identification

Web pages. The frame trees, corresponding to the pages, were automatically saved together with the user selections.

2.4.3.2 Evaluation Result

We used the collected source pages to statistically compute the performance of our context identification algorithm and the threshold for our topic boundary detection algorithm, as described in Section 2.4.1. We used 50% of the page samples to estimate the threshold value and the remaining 50% were used to calculate the performance of the context identification algorithm in terms of recall, precision, and F-measure.

Let M_1 be the multiset with the context selected by the user, and let M_2 be the multiset computed by our algorithm, then, the recall value for the context identification algorithm is $|M_1 \cap M_2|/|M_1|$, and the precision value is $|M_1 \cap M_2|/|M_2|$. The F-measure is calculated by taking the harmonic mean of recall and precision.

We used the F-measure to estimate the cosine similarity threshold. We designed a greedy algorithm that started with an unrealistically high threshold, used our context identification algorithm to find the context of the selected links in the 500 sample Web pages, compared the results with the human-selected context, and, then, adjusted the threshold value iteratively until it converged to the F-measure that locally could not be improved any further. Specifically, we set the threshold

$T_1 = 1$, $n = 1$, and $\delta = 0.1$; we compared the F-measures F_n and F_{n+1} while adjusting the threshold $T_{n+1} = T_n - \delta$ iteratively, as long as $F_n < F_{n+1}$. Then, we used a binary-search approach to converge to the optimal threshold T_{opt} between T_n and T_{n+1}, where the F-measure F_{opt} was the local maximum. Once the threshold was determined, we used the remaining 50% of the human-selected context to compute the average recall, precision, and F-measure in each of the 5 domains. Figure 11 summarizes the experimental results. We observe that, context identification achieved higher recall, precision, and F-measure with the “News” and “Informational” Web sites; higher performance of context identification in these domains can be explained by the fact that they are better organized and have more textual content. Context identification performance received the worst score in the “Electronics” domain, and average scores in “Books” and “Office Supplies” Web sites, most likely, because e-commerce Web sites crowd their pages with more diverse information, preferring images over text.

Chapter 3

Transaction Model

In this chapter, we formalize the notion of a Web transaction and transaction models.

3.1 Transaction Concept

A Web page can be partitioned into a logical structure of segments (i.e. sections of a webpage) containing related Web objects. Using an underlying ontology of concepts present in Web pages, we classify these segments to these concepts and assign the corresponding concept names to the classified segments. The ontology describes the set of semantic concepts occurring in Web pages, which are relevant for conducting Web transactions in a particular domain. These semantic concepts present in a transaction ontology are called **Transaction Concepts**.

In this thesis we use the term **Transaction Concept** and **Concept** interchangeably.

Note that, the ontology (i.e. concepts and operations) are given apriori.

3.1.1 Concept Segment

The Web page segment containing a transaction concept is called a **Concept Segment**

A concept segment may contain one or more Web objects. We have observed that Web objects contained within a concept segment are presented using similar

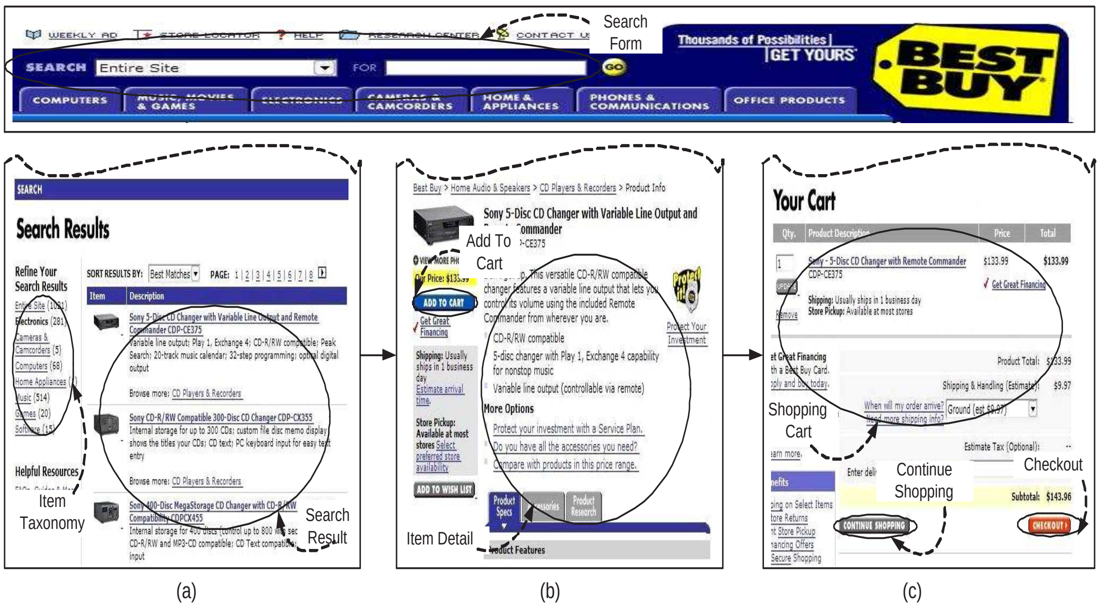


Figure 12: Concept Segments in Web Pages

presentation style.

Figure 12 shows three pages from the Bestbuy Website. Concept segments are encircled in each page. The concept segment labeled as “Item Taxonomy” in figure 12 (a) contains a collection of Web objects presented using the same presentation style (each of the Web objects in this collection is a link).

Table 2 shows concept names in our shallow ontology for the online shopping domain (e.g. books, consumer electronics, office supplies).

3.1.2 Concept Features

Each concept is associated with features. The features extracted from the content of the concept segments are used for classifying the segments to the concept. For example, the segment containing the concept “AddToCart” in figure 12 (b) contains the word “addtocart” as feature.

Concept	Operation Name
Shopping Cart	view_shoppingcart
Add To Cart	add_to_cart
Edit Cart	update_cart
Continue Shopping	continue_shopping
Checkout	check_out
Search Form	submit_searchform
Search Result	item_select
Item List	item_select
Item Taxonomy	select_item_category
Item Detail	show_item_detail

Table 2: Concepts in Ontology for Online Shopping.

3.2 Transaction Operation

A Web transaction is composed of sequence of operations on Web objects. Each transaction operation is an atomic activity on a Web object.

For example, following a link (a Web object) to select an item from a list is a transaction operation. Submitting a search form (a Web object) to search for a product is another transaction operation.

Each transaction operation is associated with a concept present in the ontology. Table 2 shows concept names and associated operation names in our shallow ontology for the online shopping domain. When the user selects a concept when presented with the list of concepts in a Web page, the corresponding **Operation** is invoked.

A transaction operation O performed on a Web object Obj is denoted as $O(Obj)$. However for simplicity, we often do not specify Obj on which the transaction operation O is performed.

We may label O by one of the names in the ontology. Such transaction operations are called **Labeled Transaction Operations**. For example, $submit\_searchform$ is a labeled transaction operation.

A transaction operation which is not labeled by the operation name in the ontology is an **Unlabeled Transaction Operation**.

In this thesis, we use the term **Operation** and **Transaction Operation** interchangeably.

3.3 Transaction Sequence

A Web transaction sequence is a non-empty sequence of transaction operations. Let O_1,O_2, ..., O_n is a sequence of transaction operations. Then, the corresponding transaction sequence is denoted as TS where $TS = O_1, O_2, \ldots, O_n$
For example, the following sequence of operations is a **Transaction Sequence**:
$select\_item\_category$. $select\_item\_category$. $select\_item$. add_to_cart. $check\_out$.
In this transaction sequence, each operation is labeled. This is a **Labeled Transaction Sequence**. The following is another example of a labeled transaction sequence:
$select\_item\_category(Obj\_1)$. $select\_item\_category(Obj\_2)$. $select\_item(Obj\_3)$. $add\_to\_cart(Obj\_4)$. $check\_out(Obj\_5)$.
In this sequence, we include the Web Objects on which each transaction operation is performed. A transaction sequence is called an **Unlabeled Transaction Sequence** if all the transaction operations in that sequence are unlabeled. For example, following sequence of operations is an **Unlabeled Transaction Sequence**: O_1. O_2. O_3. O_2. O_4.
The following is another example of an unlabeled transaction sequence.
$O\_1(Obj\_1)$. $O\_2(Obj\_2)$. $O\_3(Obj\_3)$. $O\_2(Obj\_4)$. $O\_4(Obj\_5)$.
The transaction sequence is called a **Partially Labeled Transaction Sequence** if it includes both labeled and unlabeled operations. For example, the following sequence of operations is a **Partially labeled Transaction Sequence**:
$O\_1(Obj\_1)$. $select\_item(Obj\_2)$. $O\_3(Obj\_3)$. $add\_to\_cart(Obj\_4)$. $check\_out(Obj\_5)$.

3.3.1 Complete Transaction Sequence

A transaction sequence is called **Complete** if it completes a transaction (e.g. buying a MP3 player from Amazon, paying the utility bill in Verizon). The following is an example of complete transaction sequence:

$select\_item\_category.\ select\_item\_category.\ select\_item.\ add\_to\_cart.\ check\_out.$

3.3.2 Incomplete Transaction Sequence

A transaction sequence is called **Incomplete** if it does not represent a complete transaction (e.g. A MP3 player is selected but not added to shopping cart or checked out). The following is an example of an incomplete transaction sequence: $select\_item\_category.\ select\_item\_category.\ select\_item.$

3.4 Transaction Model

A **Web Transaction Model** captures the semantics of a Web transaction. The model uses a shallow ontology of transaction concepts, an underlying concept model (i.e. concept classifiers) to identify instances of a concept from a Web page, and a process model to present concept instances in a Web page associated with the current state, and to make a transition to the next state as the user selects an operation associated with the concept. In the next subsections, we define the process model and the concept model.

3.4.1 Process Model

The process model is a deterministic finite state automaton (DFA) that captures the set of transaction sequences. Each state is associated with a set of concepts drawn from the ontology. When user selects an operation associated with a concept in a state, the model makes a transition to the next state. As a result, a Web page is provided to that state as an input. If the concepts associated with the state are present in the page, then they alone are identified and presented to the user.

3.4.1.1 Formal Definition

Formally, a process model is defined as follows: Let $C = \{c_0, c_1, \ldots\}$ be a set of transaction concepts, and $I(c)$ denote the set c of concept instances. Let

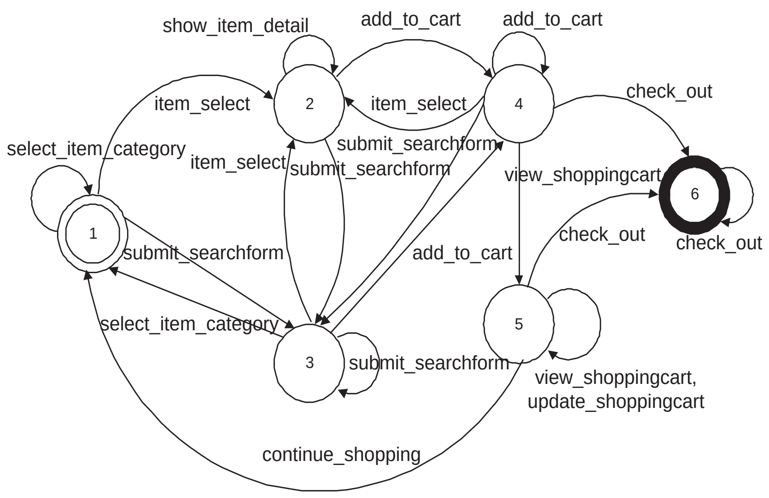


Figure 13: Process Model Example

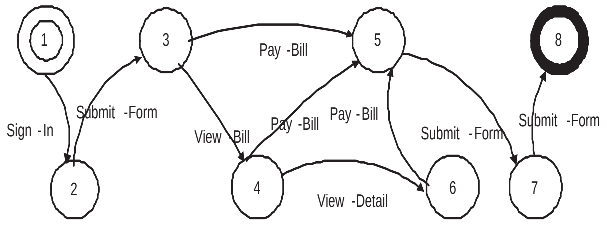


Figure 14: A Process Model for Utility Bill Payment Domain

$Q = \{q_0, q_1, \ldots\}$ be a set of states. With every state q_i we associate a set $S_i \subseteq C$ of concepts. Let $O = \{o_0, o_1, \ldots\}$ be a set of operations. An operation o_i can take parameters. A transition δ is a function $Q \times O \rightarrow Q$, and a concept operation ρ is also a function $C \rightarrow O$. Operations label transitions, *i.e.*, if $\delta(q_i, o) = q_j$ then o is the label on this transition. An operation $o = \rho(c)$ is enabled in state q_i whenever the user selects an instance of $c \in S_i$ and when it is enabled a transition is made from q_i to state $q_j = \delta(q_i, o)$.

Note that, here a process model is defined as a DFA. This is because, selecting an operation from a state always leads to a single state (and hence this is DFA). Associating concepts with every state does not violate any characteristics of a DFA since there is a mapping function from concept to operation and operations can be simply treated as alphabet symbol of an automaton.

Technically a concept instance is the occurrence of a concept in a Web page. For example, the circled items in Figure 1 are all concept instances. But for brevity we choose not to make this distinction explicitly and use concepts and concept instances interchangeably when referring to content segments in Web pages.

3.4.1.2 Process Model Example

Figure 14 illustrates a process model. The concepts associated with state q_1 are “Item Taxonomy”, “Item List”, and “Search Form”. This means that if these concept instances are present in the Web page given to q_1 as its input, they will be extracted and presented to the user. The user can select any of these concepts. We say that the user chooses the “Item Taxonomy” concept whenever he selects a particular category of item in the taxonomy and upon selection the corresponding operation *select_item_category* (see table 2) is invoked. This amounts to fetching a new Web page corresponding to the selected category and a transition is made to state q_1. When the user selects the “Search Form” concept he is required to supply the form input upon which the *submit_searchform* operation is invoked. This amounts to submitting the form with the user-supplied form input. A Web page consisting of the search results is generated and a transition is made to q_3. Lastly a user can select an item from a list of items in the “Item List” concept. This will result in a Web page describing the item selected and the transition labeled *item_select* is made to state q_2. The state transitions of other states can be similarly described.In the figure, state q_6 is deemed as the final state. We have omitted the payment steps following checkout. Hence q_6 which is entered upon a *check_out* operation is deemed as the final state.

3.5 Concept Model

Concept models, i.e. concept classifiers are used to identify semantic concepts present in Web pages. For example, a model (i.e. classifier) for the concept “add_to_cart” is used to identify instances of this concept from Web pages. The concept models are trained from set of example concept instances. For example, a concept classifier for “add_to_cart” concept can be trained by many concept instances (e.g. “add to cart” buttons), given as training example. Using the trained concept model, concept instances are identified and presented to the user. Each concept is associated with an operation, e.g. an “add to cart” concept is associated with the operation “add_to_cart”, i.e. adding an item to shopping cart. When the concept instance is identified and presented to the user, user can does the associated operation (e.g. user can click the “add to cart” button to add the item to shopping cart). As a result of this operation, a transition is made to the next state of the process model, concept instances for the page mapped to that state are identified and presented to the user.

Chapter 4

Mining Transaction Models: Supervised Approach

The components of a transaction model are: (i) a process model and (ii) concept models to identify the concept instances from Web pages. In this chapter, we describe our initial work to mine such transaction models from transaction sequences using a supervised approach. The approach is supervised since we manually label the transaction sequences. Section 4.1 describes process model learning in details. Concept Identification is described in Section 4.2.

4.1 Process Model Learning

We build the process model using DFA learning techniques, a thoroughly researched topic (see Chapter Chapter 6 for related work). In the DFA learning problem the training set consists of two sets of example strings, one labeled positive and the other negative. Only strings in the positive set are in the DFA's language. The objective is to construct a DFA that is consistent with respect to these two sets, *i.e.*, it should accept strings in the positive set while rejecting those in the negative set. We adapted the heuristic in [68] for learning our process model, the choice being mainly dictated by its simplicity and low complexity.

We used **Labeled Transaction Sequences** to learn the process model. The sequence *submit_searchform*. *item_select*. *add_to_cart*. *check_out* is one example

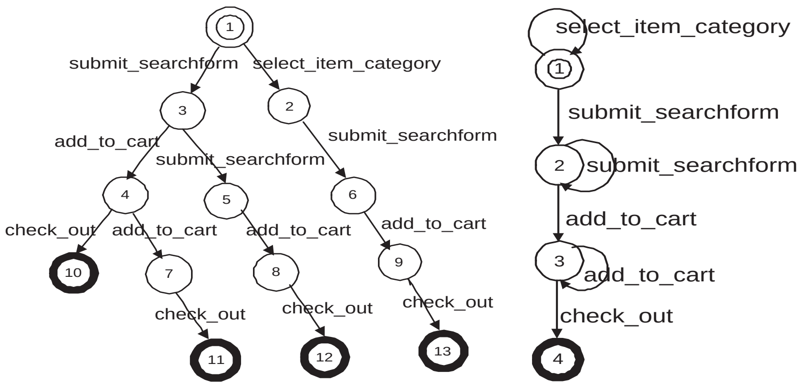


Figure 15: (a) A Prefix Automata. (b) Learned DFA

of a labeled transaction sequence.

These training sequences are (manually)labeled "completed" and "not completed". The positive example set ($S+$) consists of sequences labeled "completed" while the negative example set ($S-$) consists of those labeled "not completed".

We first construct a prefix tree automaton as shown in 15(a) using only the examples in the positive set $S+$.

In 15(a), the sequence of operations along each root-to-leaf path constitutes a string in $S+$. For this example the negative set $S-$ consists of the strings: $\{$*check_out, submit_searchform. add_to_cart, submit_searchform. check_out, select_item _category.add_to_cart. check_out*$\}$.

The prefix of every string in $S+$ is associated with a unique state in the prefix tree. The prefixes are ordered and each state in the prefix tree automaton is numbered by the position of its corresponding prefix string in this lexicographic order.

Next we generalize the prefix tree automaton by state merging. We choose state pairs (i, j), $i < j$ as candidates for merging. The candidate pair (i, j) is merged if it results in a consistent automaton.

For example, merging the pair (1,2) is consistent whereas (3,4) is not merged as the resulting automata will accept the string *submit_searchform. check_out* in $S-$.

The DFA that results upon termination of this merging process on the above example set is shown in Figure 15(b).

4.2 Concept Detection

After a frame tree is generated, our task is to associate an instance of a concept to some nodes in the frame tree. The subtree rooted at such a node represents the smallest segment of the Web page covering the concept instance. Towards this, we have designed a feature space for frame tree nodes. For each node in the frame tree, its features are collected as a vector in the feature space. The features are then used to build a statistical concept model. The model is trained using prelabeled nodes. For a new frame tree, each node is ranked by the concept model and the one with the highest rank is selected as the instance of the concept. Below we give a description of the feature space.

4.2.1 Feature Space

Each node in the frame tree can be represented by a vector of values representing its attributes, namely features. The set of feature vectors forms a multi dimensional feature space.

In our concept identification problem, given a frame tree node p, $n_{f_i,p}$ denotes the frequency of occurrence of feature f_i in p. We use the following categories of features in the analysis:

4.2.1.1 Word features

These are features drawn from the text encapsulated within a frame tree node. An instance of a concept usually contains a set of particular words occurring more often than others. For example, given the Search Form concept, an instance usually contains words like “search” and “find”. Such concept specific words are of great importance in identifying the instance. Word features are the most widely used features in textrelated machine learning tasks such as text categorization.

We collect the following word features: unigram, bigrams, trigrams, and their stemmed[1] counterparts.

[1]Word stemming is done using Porter’s stemmer [73]

For a leaf node in the frame tree, word features are drawn from its own text while for an internal frame tree node, the words present in all the leaves within the subtree rooted at it are aggregated. Stop words are ignored in both cases. $n_{f_i,p}$ is the number of times f_i occurs in the text of p.

Consider a frame tree node p, that contains the text "Nice Looking Video Player". Linguistic feature analysis collects the following features from this node:
Unigram: Nice, Looking, Video, Player
Bigram: Nice Looking, Looking Video, Video Player
Trigram: Nice Looking Video, Looking Video Player
Stemmed: Nice, Look, Video, Play
Stemmed-Bigram: Nice Look, Look Video, Video Play
Stemmed-Trigram: Nice Look Video, Look Video Play

4.2.1.2 Pattern features

These are features representing the visual presentation of content. In content-rich Web pages, it is often the case that the presentation of a semantic concept exhibits similarity across sites. For instance, in Figure 1(b), each item is presented as a link with the item name, followed by a short text description, and ending with miscellaneous text information. Similar visual presentation can also be found on other sites. The pattern features capture these presentation similarities. The basic pattern features are links, text, and image found in leaf frame tree nodes. The basic pattern features are link, text and image. Complex pattern features are any sequence of basic or complex pattern features. For example, the sequence $linklinktext$ is a complex pattern feature. we compute all the basic and complex pattern features occurring in the concept segment. Like word features, $n_{f_i,p}$ is the number of times f_i occurs in the subtree rooted at p.

4.2.2 Concept Model

Our concept identification task is to assign a score to every node p in the frame tree given a concept c. The node covering only the instance of c gets the highest score.

A Bayesian concept model consists of two components: (i) a probability distribution on the frequency of occurrence of features, and (ii) a probability distribution on the number of nodes present in the entire subtree of a frame tree node. A collection of frame trees whose nodes are (manually) labeled as concept instances serve as training set for learning the parameters of these distributions.

A collection of frame trees whose nodes are (manually) labeled as concept instances serves as the training set for learning the parameters of these distributions.

A maximum likelihood approach is used to model the distribution of a feature in a concept. Given a training set of L frame tree nodes identified as instances of concept c_j, the probability of occurrence of a feature f_i in c_j is defined using Laplace smoothing as:

$$P(f_i|c_j) = \frac{\sum_{p \in L} n_{f_i,p} + 1}{\sum_{i=1}^{i=|F|} \sum_{p \in L} n_{f_i,p} + |F|}$$

where $n_{f_i,p}$ denotes the number of occurrences of f_i in frame tree node p and $|F|$ is the total number of unique feature. The number of nodes within the subtree of a frame tree node for a concept c_j is modeled as a Gaussian distribution with parameters mean u_{c_j} and variance σ_{c_j} defined as:

$$\mu_{c_j} = \frac{\sum_{p \in L} |p|}{|L|}, \sigma_{c_j} = \sqrt{\frac{\sum_{p \in L}(|p| - \mu_{c_j})^2}{|L| - 1}}$$

For new frame trees, the probability $P(c_j|p)$ of a node p being an instance of concept c_j is proportional to $P(p|c_j)$ assuming an uniform distribution for $P(c_j)$. We use a modified multinomial distribution to model the likelihood $P(p|c_j)$:

$$P(p|c_j) = (\frac{\overline{N}!}{N_{f_1,p}! \cdots N_{f_{|F|},p}!}) \times \prod_{i=1}^{i=|F|} P(f_i|c_j)^{N_{f_i,p}}$$

where $\overline{N} = K \times e^{(|p| - \mu_{c_j})^2/(2\sigma_{c_j}^2)}$, with K being a normalized total feature frequency count, $|p|$ being the total number of frame tree nodes within the subtree rooted at p, and $N_{f_i,p}$ is a scaled value of $n_{f_i,p}$ such that $\sum_i N_{f_i,p} = \overline{N}$. Note that the above formulation of the likelihood takes into consideration both the *number of nodes* within p as well as the frequencies of the various features in the content

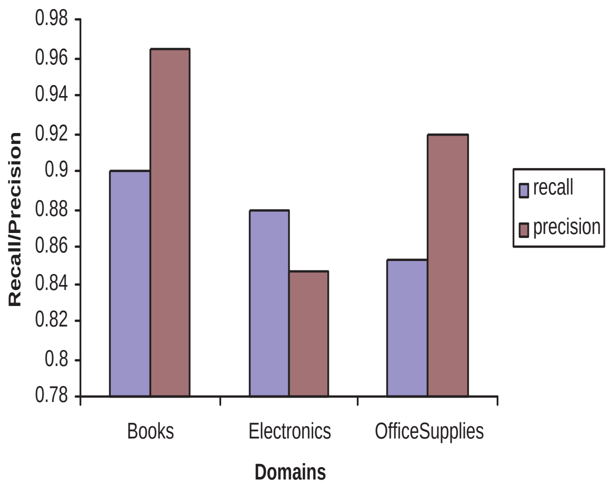


Figure 16: Recall/Precision of the Learned Process Model.

encapsulated within *p*. This results in a tight coupling between content analysis and document structure during concept identification. The frame tree node with the maximum likelihood value is identified as the concept instance.

4.3 Evaluation

Here we describe the experimental performance of our learned models, i.e. process models and concept models.

4.3.1 Performance of Process Model Learning

We collected 200 example transaction sequences from 30 Web sites. These were sequences whose elements are concept operations as illustrated in Figure 15. A number of CS graduate students (all were sighted) were enlisted for this purpose. Specifically each student was told to do around 5 to 6 transactions with a Web browser and the sequences were generated by monitoring their browsing activities. They labeled a sequence as “completed” whenever they were able to complete the

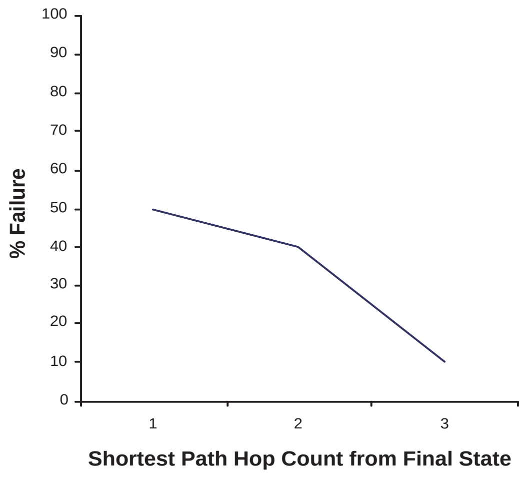


Figure 17: Failure Analysis of the Learned Process Model.

transaction; otherwise they labeled it as “not completed”. We used 120 of these sequences spanning 15 Web sites (averaging 7 to 9 sequences per site) as the training set for learning the process model. The remaining 80 were used for testing its performance. The learned model is shown in Figure 14 (page 26). The first metric that we measured was its recall/precision[2]. They were 90%/96% for the books domain, 86%/88% for the consumer electronics domain and 84%/92% for the office supplies domain. The second metric we measured was the number of transitions that remained to be completed when a true trace (completed transaction) in the test set failed to reach the final state. We observed that more than 50% of such failures ended one hop away from that state. That means even in case of a failure, the user will be only one hop away from the final state. A fast error recovery technique (e.g. a technique that may store the history of the previous states, so that user may go back to the previous state in case of a failure and try other operations from that state) can be designed with such a process model.

[2]Recall for a process model is the ratio of the number of completed transactions accepted by the model over the total number of completed transactions. For Precision, this denominator becomes the total number of accepted transactions (completed and not completed).

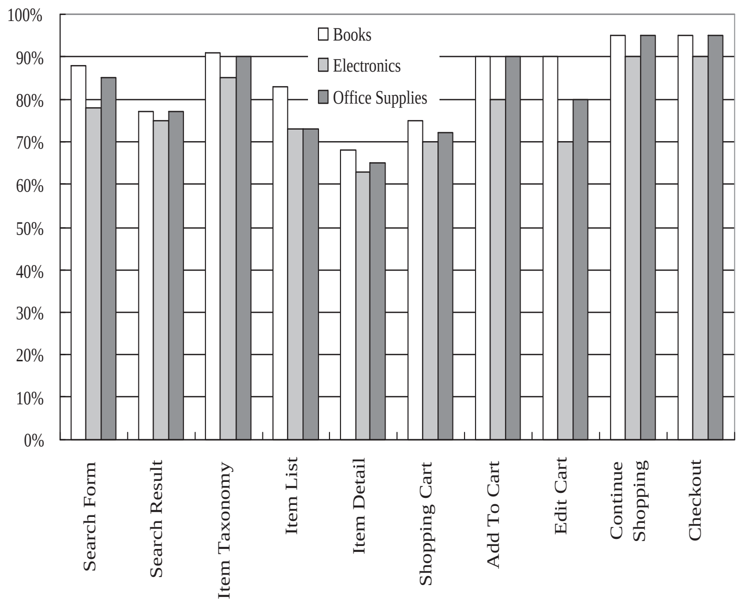


Figure 18: Recall for Concept Extraction.

4.3.2 Concept Extraction Performance

We built a statistical concept model for each of the concepts in Table 2.

Recall that the five concepts in the upper half of the table are generic for all the three domains whereas those in the lower half are domain-specific. For instance the feature set of a list of books differs from that of consumer electronic items. We built one model for each concept in the upper half of the table and three - one per domain – for each concept in the lower half.

The concept models were built using the techniques described previously. To build the model for each of the five generic concepts we collected 90 pages from 15 out of the 30 Web sites. For each of the domain specific concepts we collected 30 Web pages from five Web sites that catered to that domain.

Note that pages containing more than one concept were shared during the building of the respective concept models. These models drive the concept extractor at runtime.

We measured the recall[3] of the concept extractor for each concept in the ontology. Roughly 150 Web pages collected from all of 30 Web sites was used as the test data. Figure 18 shows the recall values for all of the 10 concepts in each of the three domains.

An examination of the Web pages used in the testing revealed that the high recall rates (above 80% for “Item Taxonomy”, “Search Form”, “Add To Cart”, “Edit Cart”, “Continue Shopping” and “Checkout”) are due to the high degree of consistency of the presentation styles of these concepts across all these Web sites. The low recall figures for the “Item Detail” (about 65% averaged over the three domains) and “Shopping Cart” (about 70%) are mainly due to the high degree of variation in their features across different Web sites. A straightforward way to improve the recall of such concepts is to use more training data. However even this may not help for concepts such as “Add To Cart” that rely on keywords as the predominant feature. Quite often these are embedded in a image precluding textual analysis. It appears that in such cases the local context surrounding the concept can be utilized as a feature to improve recall.

Observe that we did not measure the precision value (precision is defined as TP/(TP+FP), TP = true positive, FP = false positive). In some cases concept identifier may detect a non-concept (e.g. a button labeled “Add to Wishlist”) as a concept instance and present it to the user. Such cases are false positive and will drop the precision value. Although a drop in precision value will increase information overload, we felt that recall is more imortant for a transaction and measured only recall value. This is because, it is more important for us to find a true instance of a concept (e.g. an “add to cart” button) so that the user does not miss that and can complete the transaction.

[3]Recall value for a concept is the ratio of the number of correctly labeled concept instances in Web pages over the actual number of concept instances present in them.

4.4 Guide-O-Speech: A Prototype for Conducting Model-directed Transaction using Non-Visual Modality

Transaction models can be applied to reduce information overload in Non-visual Web transactions. We built a prototype, Guide-O-Speech, to conduct Web transactions using non-visual modalities. We describe its use scenario and performance in this section.

4.4.1 Use Scenario

Alice, who is a visually impaired individual, is planning on replacing her broken CD player with a new one from Best Buy. She uses our Web transactional system for this task. To begin, she speaks Best Buy's URL. After retrieving the home page, the system analyzes this page, extracts the two concepts in it, namely "Item Taxonomy" (the circled item on the left in Figure 19(b)) and "Search Form" (the circled item on the top of Figure 19) and asks Alice to choose one of them. Alice says "Search Form" and in response the system reads out the drop-down items in the search form pausing briefly after each item. Alice can pick an item at any time by either saying the item name or its number. Alice says "Electronics" and the system prompts her for the electronic item she wishes to search for.

Alice responds with "CD Player". The search form filled with these two parameters is submitted that results in fetching the page containing the search results shown in Figure 19(b). The system extracts the "Search Result " concept and begins reading out the brief description associated with each CD player in this list. Alice says "item 1" to follow the link associated with the 1st player (CDP-CE375) in the list to the page containing a detailed description of this player (Figure 19(c)). In this page three concepts, namely "Search Form", "Item Detail" and "Add To Cart" are extracted. Alice is asked if she wishes to hear the product details. When Alice responds in the affirmative, the system reads out the detailed description of the CD player she picked earlier. At the end Alice is asked if she wishes to add this to her shopping cart. Alice responds "yes". The system follows the link labeled *Add*

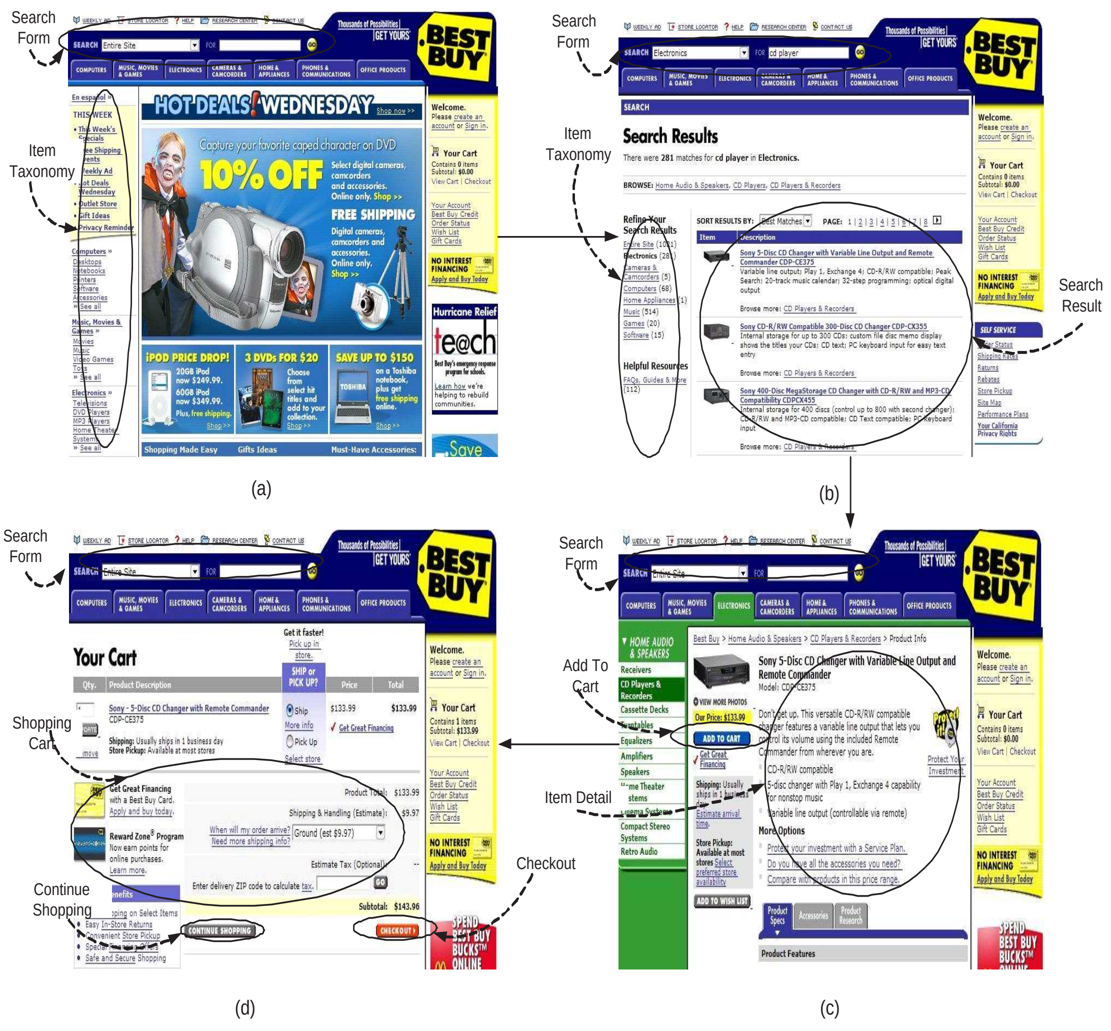


Figure 19: A Web Transaction in Online Shopping Domain

To Cart in Figure 19(c) to the page shown in Figure 19(d). In this page the concepts of “Search Form”, “Shopping Cart”, “Checkout” and “Continue Shopping” are extracted. When presented with these choices Alice chooses “Checkout”. To complete the transaction Alice must provide credit card information upon checkout. Its details have been omitted as they are quite similar to the form filling step described in the first step of the scenario. At any point Alice can also say any one of a set of general-purpose navigation commands, such as “Back to top page”, “Start over”, “Repeat” (last item) or “Stop”. Besides, on a laptop or desktop computer Alice could also use a keyboard in addition to speech to interact with Guide-O.

4.4.2 Evaluation

4.4.2.1 Experimental Setup

We used a 1.2 GHz desktop machine with 256 MB RAM as the computing platform for running the Guide-O-Speech system. To do that we installed our own VoiceXML interpreter along with off-the-shelf speech SDK components. We used 30 CS graduate students as evaluators.

Evaluators were asked to measure the total time taken to complete the transactions with Guide-O-Speech. The screen was disabled and evaluators had to interact with the system using a headphone and a keyboard. For baseline comparison, evaluators were also asked to conduct another experiment with the JAWS screen reader on the same set of transactions. For every page used in a transaction sequence they were asked to record the time it took JAWS to read from the beginning of the page until the end of the selected concept’s content. The sum of these times over all the pages associated with the transaction denotes the time taken to merely listen to the content of the selected concepts with a screen reader.

4.4.2.2 Quantitative Evaluation Result

Here we describe the evaluation results of the usability evaluation of the Guide-O-Speech system. For this evaluation, 9 Websites were used. Evaluators conducted roughly 5 to 6 transactions on each of them. We calculated mean (μ) and standard deviation (σ) for all the measured metrics.

Web Sites	Voice Interactions		Pages Explored		Time Taken (using) Guide-O-Speech		JAWS Screen Reader	
	μ	σ	μ	σ	μ	σ	μ	σ
Amazon	8	0	5	0	306.1	13.73	1300	120.2
BN	9	0.82	5	0	351.5	40.78	1130	176.78
AbeBooks	10.8	0.96	6	0.82	384.9	31.35	700	176.78
Amazon	9.8	3.1	5.2	1.26	386.6	60.15	1413	130.11
CompUSA	9	1.15	6	0.82	360.5	13.34	931.5	211.42
tigerdirect	9.4	1.29	6	0.82	357.7	30.07	1293	212.13
OfficeMAX	10	0.82	5.8	0.96	341.6	23.69	686	61.52
OfficeDepot	10	2.16	6	1.41	309.9	45.87	604	55.23
QuillCorp	9.6	1.73	5.2	0.96	382.6	49.61	625	51.84

*All times are in seconds .

Table 3: Guide-O-Speech Performance.

Table 3 lists the metrics measured for each Web site. Observe that Guide-O-Speech compares quite favorably with the best-of-breed screen readers and hence can serve as a practical assistive device for doing online transactions.

4.4.2.3 Qualitative Evaluation

To gauge user experience we prepared a questionnaire for the evaluators (see Table 4). They were required to answer them upon completing the quantitative evaluation. The questions were organized into two broad categories – system (S1 to S4) and concepts (C1 to C3) – the former to assess the overall functionality and usability of the system and the latter to determine the effectiveness of the semantic concepts in doing Web transactions. All of the concept questions except S1 and S2 required a yes/no response. From the responses we computed the mean percentages shown in the table.

A large percentage of evaluators felt that the concepts presented were self explanatory and contained enough information based on which they could take the right steps to make progress on their transactions (response to question C1). Some evaluators felt that notification of promotional offers, coupons, *etc.* was important and that such concepts ought to be presented (response to question C2).

Most were able to find the items they were looking for (response to question S1). However at times they were unable to complete the transaction (the “no” response to questions C3 and the unfinished transactions in S2). Analysis of such

C1	Did you find the concepts used in doing the transaction informative?	93.33%
C2	Do they capture all the useful information in the Web pages?	76.67%
C3	Did they help in accomplishing the transaction?	86.67%
S1	How often were you able to find the desired item?	96.67%
S2	How often were you able to complete the transaction for the item found?	93.33%
S3	Do you feel that the system restricted your navigation?	80%
S4	Did you find the system useful for conducting transactions?	96.67%

Table 4: Questionnaire with Response

transactions revealed that in many cases the problem arose because: (a) the expected concepts in a state were not extracted; (b) the extracted concepts were mislabeled; or (c) the model could not make the correct transition. The last two problems could be addressed by training the concept extractor and the process model with more examples.

A number of evaluators felt that they expected more flexibility on how they can complete the transactions (response to question S3). Observe that the number of possible paths to complete a transaction is limited by the training data and hence this criticism can be addressed with more training. Overall they all felt that the system was adequate to do their tasks (response to question S4).

Evaluators also had general comments. In particular they all felt that the system requires help utilities to assist users to become familiar with the use and effects of each concept.

Chapter 5

Mining Transaction Models: Unsupervised Approach

In this chapter, the techniques of mining transaction models from unlabeled or partially labeled transaction sequences are described. Algorithm *LearnTransactionModel* illustrates a high level overview of unsupervised transaction model learning.

The main steps to mine transaction models from transaction sequences are as follows.

- Label unlabeled operations in each transaction sequence
- Learn process model and concept models from labeled sequences.

Consider Table 5. The table shows 4 Web transaction sequences. The first transaction sequence (illustrated in figure 20) is from “Amazon.com” and is a completely unlabeled sequence. The second transaction sequence (illustrated in figure 21) is from “AbeBooks.com” and is a partially labeled sequence. The third transaction sequence (illustrated in figure 22) is from “CircuitCity.com” and is a partially labeled sequence. The fourth transaction sequence (illustrated in figure 23) is from “Bestbuy.com” and is a partially labeled sequence. In Figures 20, 21, 22, 23, the transaction operations are shown using arrows. The segments containing the concepts associated with each transaction operations are also shown using rectangles.

The concept segments, containing labeled Web objects, are labeled with the concept name. Other concept segments are unlabeled.

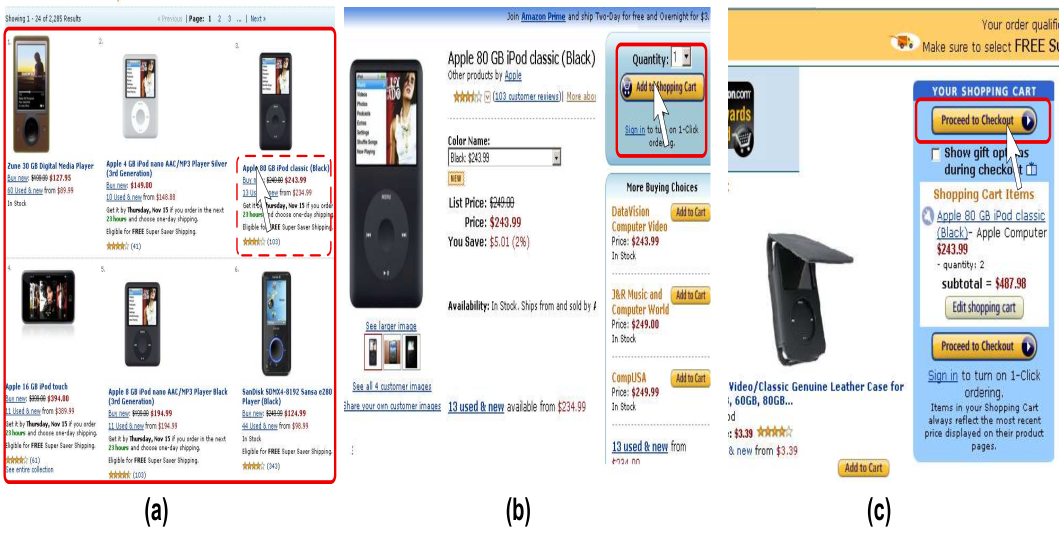


Figure 20: A Web Transaction in Amazon.com

Section 5.1 describes how we can assign labels to such concept segments. Next, in Section 5.2, we describe how concept models are learned using labeled sequences. Learning process models from labeled sequences is described in Section 5.3.

```
Algorithm LearnTransactionModel
Input: Sequences: A Set of Transaction Sequences
Output: Model (ProcessModel, ConceptModels): A Transaction Model
1.  if Sequences.Unlabeled
2.     then $labeledSequences \leftarrow GenerateLabel(\text{Sequences})$
3.  if Sequences.Partiallabeled
4.     then $labeledSequences \leftarrow GenerateLabel(\text{Sequences})$
5.  if Sequences.Labeled
6.     then $labeledSequences \leftarrow Sequences$
7.  $ConceptModels \leftarrow LearnConceptModel(\text{labeledSequences})$
8.  $ProcessModel \leftarrow LearnProcessModel(\text{labeledSequences})$
9.  return (ProcessModel, ConceptModels)
```

No.	Training Sequence
1	$< 073A133F, 0A3B135E, 0A7324F3 >$
2	$< select\_item(041A122F), 031AB23F, 0A14B25F, 04F354A2 >$
3	$< select\_item(145AB2D1), 05F354A1, 731DA231, 873A11F2 >,$
4	$< select\_category(01A561AF), 81121F2A, 02141F4B, 02345A21 >$

Table 5: Example Training Sequences

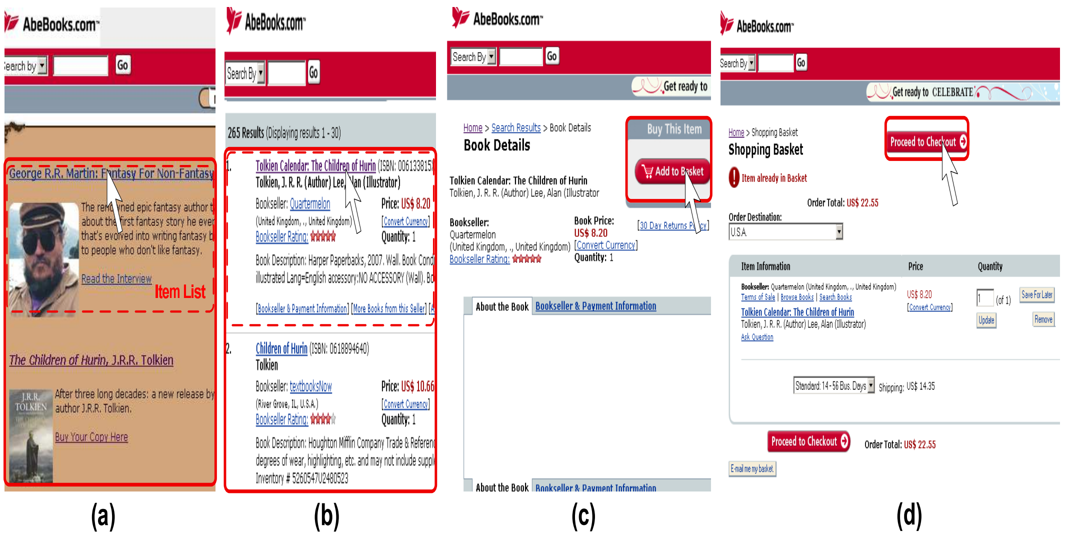


Figure 21: A Web Transaction in AbeBooks.com

5.1 Label Generation

The label generation step associates a label with each unlabeled operation. The algorithm *GenerateLabel* illustrates the label generation steps. Briefly, the algorithm does the following.

Line 1 of the algorithm *GenerateLabel* (**ExtractConceptSegment**) uses context analysis and geometric segmentation (as described in Section Chapter 2) to get the **Concept Segment** (see Section 3.1 for definition of concept segments) containing the web object. Section 5.1.1 gives more details about concept segments retrieval. Next, we cluster similar concept segments (line 2 of the algorithm *GenerateLabel*). Section 5.1.2 describes this process. We assign a concept label to each cluster, and associate each of the segments with a label. The concept labels are also used to label the transaction operations in an unlabeled or partially labeled sequence (line 3 of *GenerateLabel*). Section 5.1.3 describes this process in more details.

```
Algorithm GenerateLabel
Input: Sequences: A Set of Unlabeled or Partially Labeled Transaction Sequences
Output: labeledSequences: A Set of Labeled Transaction Sequences
1.  $ConceptSegments \leftarrow ExtractConceptSegments(\text{Sequences})$
2.  $labeledSegmentSets \leftarrow ClusterSegments(\text{ConceptSegments})$
3.  $labeledSequences \leftarrow LabelSequences(\text{labeledSegmentSets,Sequences})$
4.  return labeledSequences
```

5.1.1 Concept Segment Retrieval

Web objects are contained within concept segment. Given a Web object Obj, we can retrieve its concept segment using our Web content analysis technique, which we will describe now.

We observe that, some concept instances contain objects with repeating presentation pattern, e.g. a list of items. Concept segment for such a concept instance is the geometric segment that contains those objects presented in a repeating pattern and sharing similar alignment in the web page. In figure 22(a), geometric segment marked using solid rectangle contains list of objects with similar presentation style, which is the semantic concept “Item List”.

Other concept instances does not contain such a collection of objects with repeating pattern. A single clickable object (i.e. button, link) is the instance of a concept in those cases. Concept segment for such a concept instance contains the object and its surrounding text that shares common topic with the caption of the object. This is same as context segment of the object. In figure 22 (b), concept segment is shown using solid rectangle.

Thus the algorithm to retrieve the concept segment from a given Web object is as follows:

- We apply repeated pattern analysis algorithm [39] to find repeated patterns in geometric segment containing the Web object.
- If the Web object is presented as a repeated pattern, then we return the geometric segment as concept segment.
- Otherwise, we return the context segment as concept segment.

The detailed algorithm is described in Algorithm *ExtractConceptSegments* and *GetConceptSegment*.

Consider the second sequence in Table 5. The first Web Object in the sequence is labeled with the operation name "select_item".
Hence the concept segment containing this Web object is shown with the corresponding concept name "ItemList" in figure 21 (a).

Algorithm *ExtractConceptSegments*
Input: *Sequences*: A Set of Unlabeled or Partially labeled Transaction Sequences
Output: *ConceptSegments*: A Set of Concept Segments

```
1.  $ConceptSegments \leftarrow \emptyset$
2.  $n \leftarrow Sequences.Size$
3.  $i \leftarrow 1$
4.  repeat
5.      $Sequence \leftarrow Sequences(\mathrm{i})$
6.      $WebObjectsSeq \leftarrow Sequence.WebObjects$
7.      $m \leftarrow WebObjectsSeq.Size$
8.      $j \leftarrow 1$
9.      repeat
10.         $WebObject \leftarrow WebObjectsSeq(\mathrm{j})$
```

```
11.           $ConceptSegment \leftarrow GetConceptSegment(\text{WebObject})$
12.           if $WebObject.Labeled$
13.             then $ConceptSegment.Label \leftarrow WebObject.Label$
14.           $ConceptSegments \leftarrow ConceptSegments \cup ConceptSegment$
15.           $j \leftarrow j + 1$
16.       until $j$ =m+1
17.       $i \leftarrow i + 1$
18. until $i$ =n+1
19. return $ConceptSegments$
```

Algorithm *GetConceptSegment*
Input: *Object*: a WebObject in a Web Page
Output: *ConceptSegment*: A Concept Segment

```
1.  $ContextSegment \leftarrow$ Context Segment Containing the WebObject
2.  $GeometricSegment \leftarrow$ Geometric Segment Containing the WebObject
3.  Apply Pattern Analysis in $GeometricSegment$
4.  if $ContextSegment.Repeated$
5.    then $ConceptSegment \leftarrow GeometricSegment$
6.         $ConceptSegment.isRepeated \leftarrow$ true
7.    else $ConceptSegment \leftarrow ContextSegment$
8.         $ConceptSegment.isRepeated \leftarrow$ false
9.  return $ConceptSegment$
```

5.1.2 Concept Segment Clustering

Once the concept segments are retrieved, we cluster them to put similar segments into the same cluster.
First, we describe the concept segment features, cluster features and similarity metrics used for clustering. We then describe the clustering algorithm.

5.1.2.1 Concept Segment Features

We extract **Word Features** and **Pattern Features** from non-repeating concept segments (e.g., a concept segment containing the concept instance of “AddToCart”) and **Pattern Features** from repeating concept segments (e.g., a concept segment containing the concept instance of “ItemList”). Here we describe the features.

Word Features: These are features drawn from the text encapsulated within the concept segment.

We collect the following word features: unigram, bigrams, trigrams, and their stemmed[1] counterparts.

For example, consider the concept segment illustrated in figure 21 (d). The word features extracted from the concept segments are “Proceed”, “Checkout”, “Proceed to”, “to Checkout”, “Proceed to Checkout”, etc.

Special Fields (e.g., Number, Date, Money) are tagged as Features. So \$249.99 in figure 22 (b) is tagged as MONEY .

Pattern Features: These are features representing the visual presentation of content. The basic pattern features are link, text and image. Complex pattern features are any sequence of basic or complex pattern features. For example, the sequence $linklinktext$ is a complex pattern feature. we compute all the basic and complex pattern features occurring in the concept segment.

In concept segment illustrated in figure 20 (a), some of the pattern features are “linklinktext”, “linktext”, ”text”.

5.1.2.2 Concept Segment Similarity Computation

Jaccard Similarity [69] is used to measure the similarity between two concept segments. For concept segment S_i and S_j, the Jaccard Similarity is defined as follows.

$$J(S_i,S_j) = \frac{|F(S_i) \cap F(S_j)|}{|F(S_i) \cup F(S_j)|} \tag{2}$$

Here $F(S_i)$ and $F(S_j)$ are set of features for concept segment S_i and S_j.

The Jaccard similarity between the concept segments is equal to 1 if they are identical, and 0 if they are completely different.

[1]Word stemming is done using Porter’s stemmer [73]

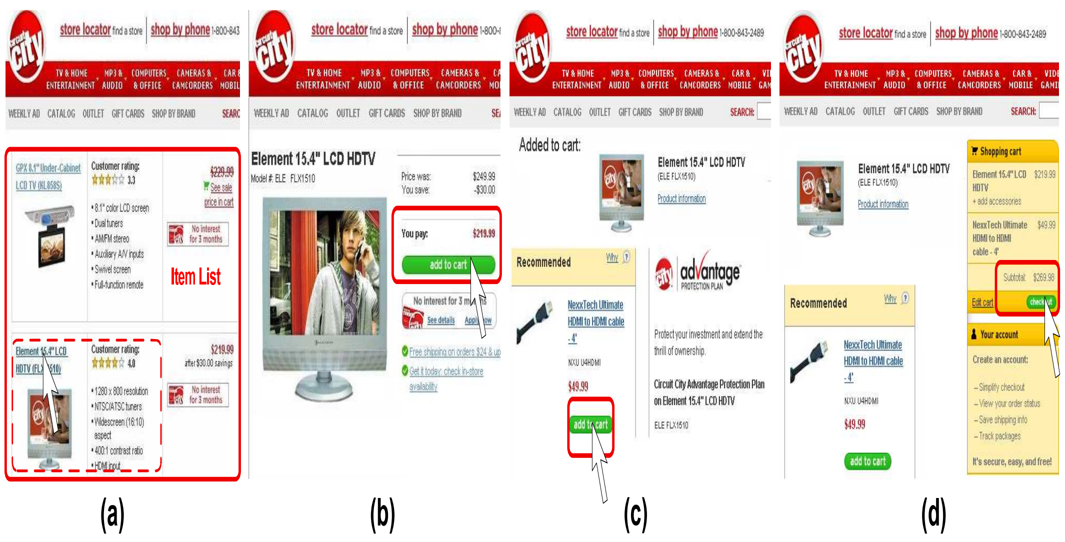


Figure 22: A Web Transaction in CircuitCity.com

5.1.2.3 Intra-Cluster Similarity

Intra-Cluster Similarity measures the similarity of the concept segments in a cluster [80]. Let us, C is a cluster with n concept segments. S_1, S_2, ..., S_n are concept segments in cluster C. Then, intra-cluster similarity is defined as:

$$Intra(C) = \frac{1}{(n-1)\cdot n}\sum_i \sum_j J(S_i, S_j) \tag{3}$$

A high value for this similarity score indicates that concept segments in the cluster are very similar, i.e. homogeneous. On the other hand, a low intra-cluster similarity indicates that concept segments are quite dissimilar, i.e. heterogeneous. Therefore, a high value of intra-cluster similarity is desired. Note that, intra-cluster similarity is undefined (0/0) for singleton clusters, i.e. when n = 1.

For example, consider the cluster in figure 25 (a). It has 4 concept segments. Jaccard Similarity values are as follows.

$\mathrm{J}(S_1, S_2) = 0.8, J(S_1, S_3) = 0.75, J(S_1, S_4) = 0.6,$
$J(S_2, S_3) = 0.65, J(S_2, S_4) = 0.55, J(S_3, S_4) = 0.75.$
$Then, Intra - ClusterSimilarityis:$

$Intra(C) = 2 * (0.8 + 0.75 + 0.6 + 0.65 + 0.55 + 0.75)/12$
$Intra(C) = 0.6833$

5.1.2.4 Inter-Cluster Similarity

Inter-Cluster Similarity measures similarity between two clusters [80]. If C_i, C_j are two clusters and S_{i_1}, S_{i_2}, ..., $S_{i_{ni}}$ are concept segments in cluster C_i and S_{j_1}, S_{j_2}, ..., $S_{j_{nj}}$ are concept segments in cluster C_j. Then,

$$Inter(C_i, C_j) = \frac{1}{n_i \cdot n_j} \sum_m \sum_n J(S_{i_m}, S_{j_n}) \tag{4}$$

Here S_{i_m} denote the m th concept segment in cluster i and S_{j_n} denote the n th concept segment in cluster j.

A high value for this similarity score indicates that clusters are very similar. On the other hand, a low inter-cluster similarity indicates that concept segments in different clusters are quite dissimilar. Therefore, a low value of inter-cluster similarity is desired.

Let us consider the clusters in figure 25 (a) and 25 (b).

The Jaccard Similarity scores between the concept segments in first cluster (25 (a)) and the concept segments in second cluster (25 (b)) are as follows:

J(S_{1_1}, S_{2_1}) = 0.05, J(S_{1_1}, S_{2_2}) = 0.052, J(S_{1_1}, S_{2_3}) = 0.06, J(S_{1_1}, S_{2_4}) = 0.05, J(S_{1_1}, S_{2_5}) = 0.09.

J(S_{1_2}, S_{2_1}) = 0.08, J(S_{1_2}, S_{2_2}) = 0.012, J(S_{1_2}, S_{2_3}) = 0.09, J(S_{1_2}, S_{2_4}) = 0.015, J(S_{1_2}, S_{2_5}) = 0.019.

J(S_{1_3}, S_{2_1}) = 0.08, J(S_{1_3}, S_{2_2}) = 0.032, J(S_{1_3}, S_{2_3}) = 0.07, J(S_{1_3}, S_{2_4}) = 0.04, J(S_{1_3}, S_{2_5}) = 0.09.

J(S_{1_4}, S_{2_1}) = 0.021, J(S_{1_4}, S_{2_2}) = 0.032, J(S_{1_4}, S_{2_3}) = 0.016, J(S_{1_4}, S_{2_4}) = 0.02, J(S_{1_4}, S_{2_5}) = 0.012.

Here S_{1_1} is the 1st concept segment in cluster C_1, S_{2_1} is the 1st concept segment in cluster C_2, and so on.

Using equation 4, we compute Inter(C_1, C_2) as follows:

Inter(C_1, C_2) = (0.05 + 0.052 + ... + 0.012)/20 = 0.0465

5.1.2.5 Quality of Clustering

Quality of clustering measures goodness of clustering [80]. In other words, it measures how good the clusters are. It is desired that concept segments in a cluster be similar (i.e. intra-cluster similarity should be high) and concept segments in different clusters be dissimilar (i.e. inter-cluster similarity should be low).

Say we have n concept segments and C_1, C_2, ..., C_k are k non-singleton clusters (i.e. clusters with more than one concept segment) of those segments. Intra-Cluster Similarities are Intra(C_1), Intra(C_2), ..., Intra(C_k).
Inter-Cluster Similarities are Inter(C_1, C_2), Inter(C_1, C_3),.........,Inter(C_{k-1}, C_k). Then quality of clustering is defined based on the ratio of weighted average inter-cluster to weighted average intra-cluster similarity:

$$\phi^{\mathrm{Q}} = 1 - \frac{\sum_{i=1}^{k} \frac{n_i}{n-n_i} \sum_{j \in \{1,\ldots,i-1,i+1,\ldots,k\}} n_j.\mathrm{Inter}(C_i, C_j)}{\sum_{i=1}^{k} \mathrm{Intra}(C_i)} \tag{5}$$

The quality value, $\phi^{\mathrm{Q}} \in$ [0, 1] (ϕ^{Q} is negative in case of inverse/pathological clustering, i.e. when intra-cluster similarities are low and inter-cluster similarities are high).

Note that we compute quality of clustering only from non-singleton clusters. This is because intra-cluster similaritiy is undefined (0/0) for singleton clusters [80].

Let us consider the clusters in figure 25. The clusters are denoted as C_1 and C_2.

Intra(C_1) = 0.68, Intra(C_2) = 0.51, Inter(C_1, C_2) = 0.0465.

Then, quality of clustering is,

1 - (4/5 * (5 * 0.0465) + 5/4 * (4 * 0.0465))/(4*0.68 + 5*0.51) = 0.92

5.1.2.6 Clustering Algorithm

Here we describe the clustering algorithm which is illustrated in *ClusterConceptSegments*. Line 1 initializes the set of clusters to be the empty set. Lines 2 to 11 iterate over each concept segment and does the following: If the concept segment is labeled, then the concept segment is added to the cluster with the same label. Otherwise a new cluster is created and the concept segment is added to the new cluster.

For the example training sequences in table 5, 15 concept segments are shown in figure 20, 21, 22, 23. 2 of the concept segments are labeled with concept name "ItemList" and another concept segment is labeled with concept name "ItemTaxonomy". Therefore, we have 14 clusters initially. One of the clusters is labeled "ItemList" and contains 2 concept segments labeled "ItemList" (these concept segments are shown in figure 21 (a) and figure 22 (a)). Another cluster is labeled "ItemTaxonomy" (shown in figure 23 (a)). Thus we have 12 unlabeled and 2 labeled clusters.

Lines 12 to 20 construct pair of clusters from list of clusters such that at least one cluster in a pair is unlabeled. The inter-cluster similarity values are also computed.

For our example, we construct 132 unlabeled cluster pairs (both the clusters in the pair are unlabeled) and 24 labeled cluster pairs (one of the clusters in the pair is labeled). We compute the inter-cluster similarity value for each of the pairs.

Line 21 invokes the algorithm *GetQuality* which returns the quality value of the current clusters. This algorithm computes quality value (as defined in section 5.1.2.5) from non-singleton clusters, and returns that value. However, it returns "Undefined" if none of the clusters are non-singleton. The quality value is saved as maximum quality value. Current clusters are saved as best clusters (line 22). For our example, the current quality value is 0.13231.

Lines 23 to 34 is the main clustering loop. At each step of the iteration, the most similar clusters are retrieved by invoking the algorithm *GetMostSimilarClusters*(line 24). The algorithm sorts the cluster pairs based on their similarity value (i.e. inter-cluster similarity) and the pair with the highest similarity value is returned. The clusters with highest similarity value are merged (figure 24).

The algorithm continues until we are left with only 1 cluster or the number of unlabeled clusters becomes 0 or no similar clusters are found to merge (line 34). For our example, the algorithm continues until we have only 2 labeled clusters.

Line 35 returns the clusters (*BestClusters*) with the highest quality value. For our example, *BestClusters* contains 4 clusters. 2 of the clusters are labeled and the others are unlabeled. Figure 25 shows the unlabeled clusters.

Algorithm *ClusterConceptSegments*
Input: *ConceptSegments*: A Set of Concept Segments
Output: *Concepts*: A Set of ConceptSegment Sets where Segments Representing Same Concepts are Placed in the Same Set

```
1.  $Clusters \leftarrow \emptyset$
2.  for $i = 1$ to $ConceptSegments.Size$
3.      do if $ConceptSegments(\mathrm{i}).Labeled$
4.          then $Cluster \leftarrow GetCluster$(Clusters,ConceptSegments($i$).Label)
5.              if $Cluster$ = $NULL$
6.                then $Cluster \leftarrow$new Cluster
7.                    $Cluster.Label \leftarrow ConceptSegments(\mathrm{i}).Label$
8.          else
9.              $Cluster \leftarrow$new Cluster
10.         $Cluster.Segments \leftarrow Cluster.Segments \cup ConceptSegments(\mathrm{i})$
11.         $Clusters \leftarrow Clusters \cup Cluster$
12. $ClusterPairs \leftarrow \emptyset$
13. for $i = 1$ to $Clusters.Size - 1$
14.     do for $j = i + 1$ to $Clusters.Size$
15.         do if $Clusters(\mathrm{i}).Label$ = $NULL$ or $Clusters(\mathrm{j}).Label$ = $NULL$
16.             then $ClusterPair \leftarrow$ new Pair ($Cluster(\mathrm{i})$, $Cluster(\mathrm{j})$)
17.                 $Features(\mathrm{i}) \leftarrow Cluster(\mathrm{i}).Features$
18.                 $Features(\mathrm{j}) \leftarrow Cluster(\mathrm{j}).Features.$
19.                 $ClusterPair.Similarity \leftarrow$GetSimilarity($Features(\mathrm{i})$,
                    $Features(\mathrm{j})$)
20.                 $ClusterPairs \leftarrow ClusterPairs \cup ClusterPair$
21. $MaxQuality \leftarrow$GetQuality($Clusters$)
22. $BestClusters \leftarrow Clusters$
23. repeat
```

```
24.       $(first, second)$ ←GetMostSimilarClusters($ClusterPairs$)
25.       **if** $first \neq NULL$ and $second \neq NULL$
26.         **then if** $first.Label = NULL$
27.               **then** ($Clusters$,$ClusterPairs$) ←Merge($Cluster$, $ClusterPairs$,
                    $second$, $first$)
28.               **else**
29.                    ($Clusters$, $ClusterPairs$) ←Merge($Cluster$, $ClusterPairs$,
                    $first$, $second$)
30.             $CurrentQuality$ ←GetQuality($Clusters$)
31.             **if** ($MaxQuality$ = $UNDEFINED$ and $CurrentQuality$
                $\neq$UNDEFINED)**or** (CurrentQuality>MaxQuality)
32.               **then** $MaxQuality$ ←$CurrentQuality$
33.                    $BestClusters$ ←$Clusters$
34. **until** $Clusters.Size = 1$ **or** $Clusters.NumUnlabeled = 0$ **or** $first = NULL$ **or**
    $second = NULL$
35. **return** $BestClusters$
```

Algorithm *GetMostSimilarClusters*
Input: *ClusterPairs*: A Set of Cluster Pairs
Output: $(firstCluster, secondCluster)$: a Cluster Pair

```
1. Sort the pair of clusters based on their similarity value
2. Arrange the cluster pairs from highest to lowest similarity value
3. $(first, second)$ ←next cluster pairs from sorted cluster pairs.
4. **if** $Similarity$(first,second) = 0
5.   **then** $first$ ←$NULL$
6.        $second$ ←$NULL$
7. **return** $(first, second)$
```

Algorithm *Merge*
Input: *Clusters*: A Set of Clusters Containing Concept Segments
Input: *ClusterPairs*: A Set of Cluster Pairs
Input: *firstCluster*: a Cluster
Input: *secondCluster*: a Cluster
Output: *(Clusters, ClusterPairs)*: The updated set of Clusters

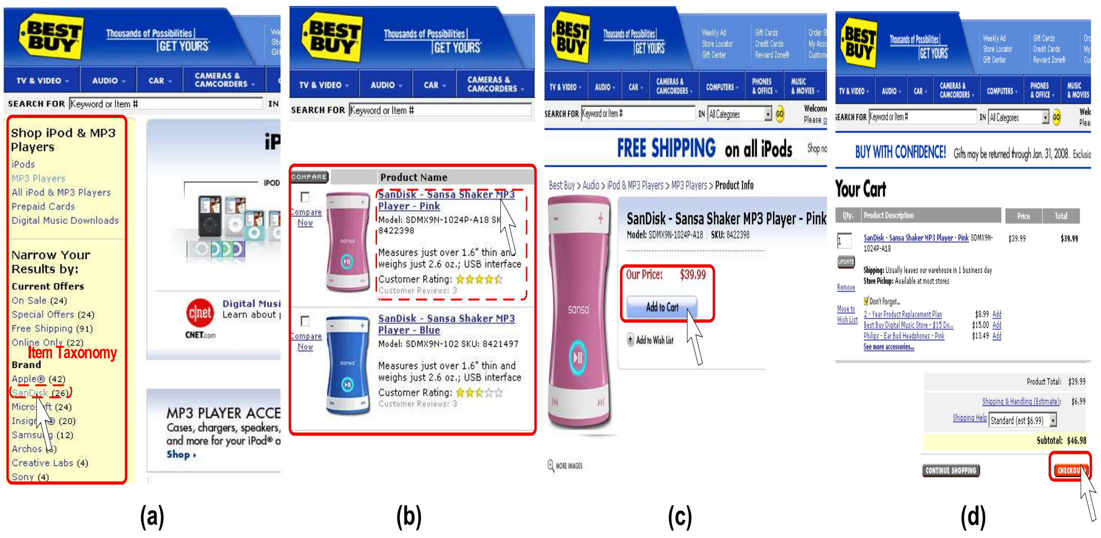


Figure 23: A Web Transaction in BestBuy.com

```
1.  for $i \leftarrow 1$ to $secondCluster.Segments.Size$
2.      do insert $secondCluster.Segments(i)$ to $firstCluster.Segments$
3.  Remove $secondCluster$ from $Clusters$
4.  Compute similarity of $firstCluster$ from other Clusters
5.  Update $ClusterPairs$
6.  return $(Clusters, ClusterPairs)$
```

Algorithm *GetQuality*
Input: *Clusters*: A Set of Clusters Containing Concept Segments
Output: *qualityVal*: The quality of clusters

```
1.  $ClustersNonSingleton \leftarrow Clusters$.GetNonSigletonClusters
2.  if $ClustersNonSingleton.Size = 0$
3.     then $qualityVal \leftarrow UNDEFINED$
4.     else
5.          $qualityVal \leftarrow$ compute quality using equation 5
6.  return $qualityVal$
```

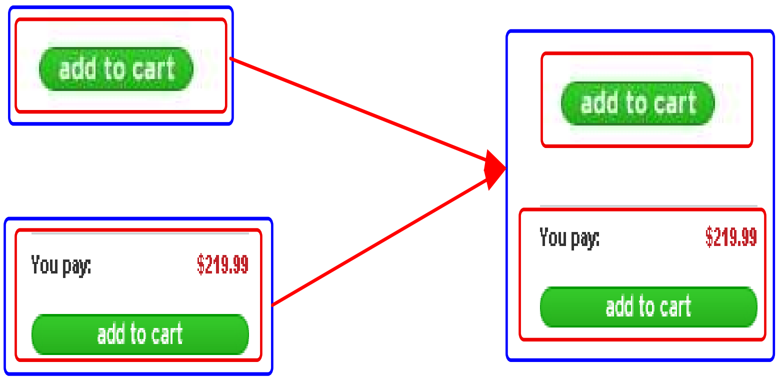


Figure 24: Merging Similar Clusters

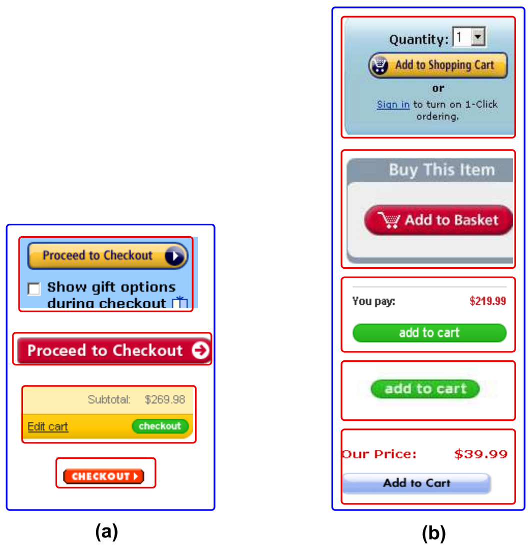


Figure 25: Clusters Containing Unlabeled Concept Segments

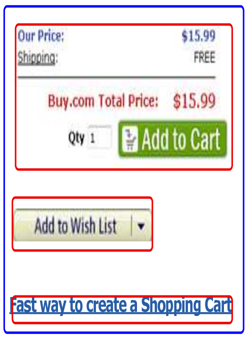


Figure 26: An Example Noisy Cluster

5.1.2.7 Elimination of Noisy Clusters

After the clustering algorithm returns a set of clusters, we eliminate noisy clusters which are:

- singleton unlabeled clusters, i.e. a cluster with a single concept segment. We assume that a concept should have multiple occurences in a collection of transaction sequences.

- non-singleton unlabeled clusters which have Intra-Cluster Similarity below a threshold. (The threshold value 0.01 worked well in practice).

Figure 26 shows an example of a noisy cluster.

5.1.3 Labeling

Once the clustering algorithm clusters the concept segments into a set of clusters, unlabeled clusters are labeled (line 27 to 31 in algorithm *ClusterConceptSegments*). Next the label of each of the clusters is used to label the concept segments and the Web objects in each concept segments. The following two approaches are used for cluster labeling.

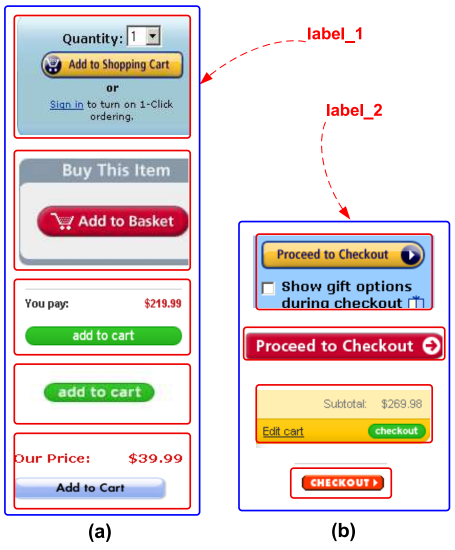


Figure 27: Ad-hoc Labeling

Ad-hoc Labeling

This approach is used when we do not have labeled sequences. In this approach, we assign ad-hoc labels sequentially to the unlabeled clusters. For *m* unlabeled clusters, *m* labels are assigned starting from $label_1$ to $label_m$.

Continuing with our example, the first unlabeled cluster (as shown in Figure 25(a)) is labeled as “label_1” and the second unlabeled cluster (as shown in Figure 25 (b)) is labeled as “label_2”. The clusters in Figure 25(a) and 25(b) after ad-hoc labeling are shown in Figure 27.

Labeling using Co-training

We described how we label the unlabeled clusters in ad-hoc way. Another simple idea is to generate the label of each unlabeled cluster from the concept segments present in that cluster. However, that would not give us meaning label used in that domain (e.g. a label “search form” may not be available in this approach). Observe that when any concept segment in a cluster is labeled, the cluster is also labeled with that label. For example, a concept segment labeled “ItemList” also labels the cluster containing that segment as “ItemList”. Therefore, an unlabeled cluster means none of the concept segments in that cluster are labeled. We can label such an unlabeled cluster using co-training approach. In this approach, labeled transaction sequences are used to label the unlabeled sequences. The underlying principle is described in [13], where unlabeled data is labeled with the help of labeled data.

Specifically, we will label the unlabeled clusters constructed by our clustering algorithm with the help of labeled clusters constructed from labeled sequences.

We apply the concept segment retrieval algorithm on the labeled sequences to get a set of concept segments for each sequence. Since the sequences are labeled, we associate the concept label with each concept segment (e.g. a concept segment labeled with “SearchForm”). We construct clusters containing concept segments with the same label.

Let $A = A_1,A_2,...,A_M$ denote the set of clusters (either labeled or unlabeled) constructed by our algorithm and F(A_1),F(A_2),...,F(A_M) denote the feature vectors computed from each such cluster. Let $H = H_1,H_2,...,H_N$ denote the set of clusters constructed from labeled data and F(H_1),F(H_2),...,F(H_N) denote the feature vectors computed from each such cluster.

Next, we take an unlabeled cluster A_i from the set of clusters constructed by our algorithm and compute the inter-cluster similarity between A_i and H_j, for all clusters H_j in the set of user-labeled clusters. The cluster A_i which is most similar to H_j is labeled with the label of H_j. Then, the cluster H_j is removed from the set H. This procedure is applied repeatedly until all the clusters in the set A are labeled, or the set H is empty. In the second case, the unlabeled clusters in set A are assigned ad-hoc labels.

Let us consider the labeled sequences in Table 6. The first transaction sequence is shown in Figure 28 and the other one is shown in Figure 29. We apply the concept segment retrieval algorithm for each sequence to retrieve the concept segments as shown in Figure 28, 29. Next we construct four labeled clusters containing those concept segments, (Figure 30).

The first cluster contains the concept segment from Figure 28 (a) and labeled “ItemTaxonomy”. The second cluster contains concept segments from Figure 28 (b) and 29 (b) and labeled “AddToCart”. The third cluster contains the concept segment from Figure 29 (a) and labeled “ItemList”. The final cluster contains the concept segments from Figure 28 (c) and Figure 29 (c)and labeled “CheckOut”.

No.	Training Sequence
1	$< select\_category(093A233E), add\_to\_cart(0B7424A3), check\_out(632A213F) >$
2	$< select\_item(03EA162B), add\_to\_cart(0A942522), check\_out(7A1B513B) >$

Table 6: Example Labeled Transaction Sequences:

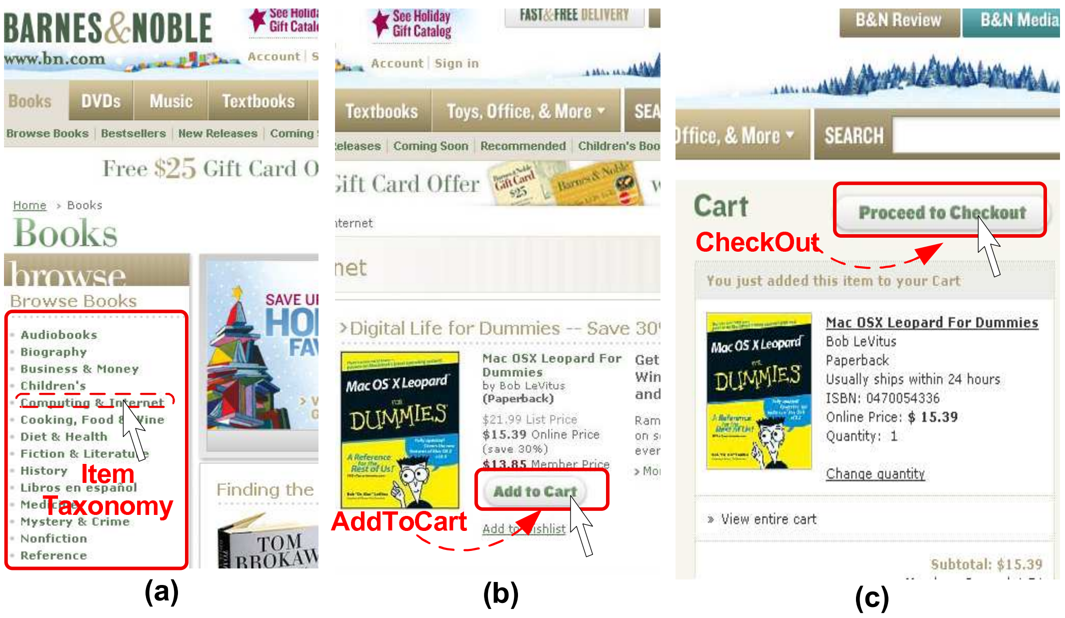


Figure 28: A Web Transaction in BN.com

Now consider the unlabeled cluster (returned by our algorithm) shown in Figure 25 (a). We compute similarity of this cluster with all the labeled clusters. Similarity values are shown as labels of the arrows in figure 30.

This cluster is found to be most similar with the user labeled cluster “AddToCart”. Hence we label this cluster as “AddToCart”, (Figure 30).

Then, we remove the user labeled cluster “AddToCart” from the set of user labeled clusters. Next, we take the unlabeled cluster shown in Figure 25 (b) and compute its inter-cluster similarity with the user labeled clusters, “ItemTaxonomy”, “ItemList” and “Checkout”. This cluster is most similar to the user labeled cluster “CheckOut”. Hence the unlabeled cluster is labeled “CheckOut”, (Figure 30).

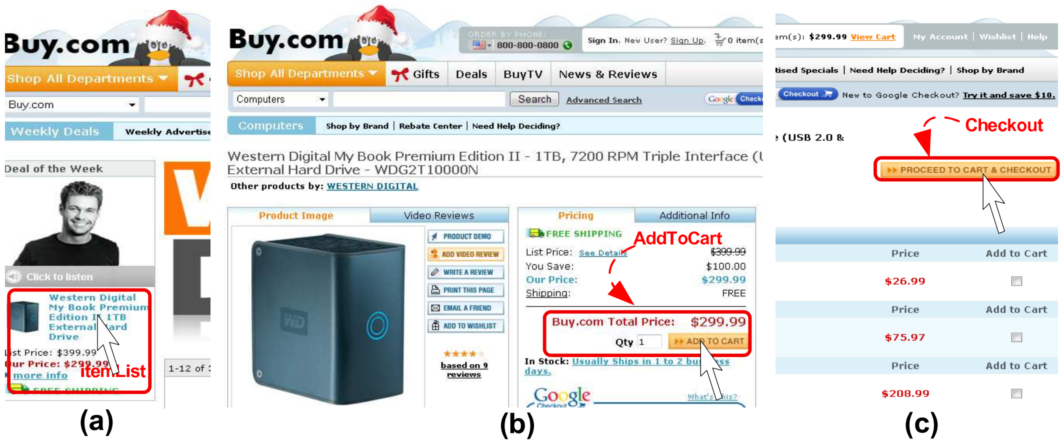


Figure 29: A Web Transaction in Buy.com

5.1.3.1 Web Object Labeling

Once the clusters are labeled, the concept segments in each cluster are also labeled(line 31 of algorithm *ClusterConceptSegments*). For example, the concept segments of the clusters in Figure 31 are labeled with the label of the clusters. Next, we associate the *WebObject* (i.e. URL) which is embedded within the concept segment with the operation name corresponding to the label of the segment.

Continuing with our example, the *WebObject*'s in Table 5 are labeled with the operation name for each concept label. Table 8 shows the corresponding labeled transaction sequences. For simplicity, operations are written as *operationName* instead of *operationName*(WebObject).

5.2 Unsupervised Learning of Concept Models

The label generation method described in the previous section associates concept labels with segments containing the *WebObject*. It also clusters similar segments. Each cluster represents semantic concept. We may use each such cluster as a classifier to classify a concept segment as instance of that concept. For example, given a concept segment and a set of clusters, we can compute the inter-cluster similarity between the single cluster containing that segment and each of the clusters.

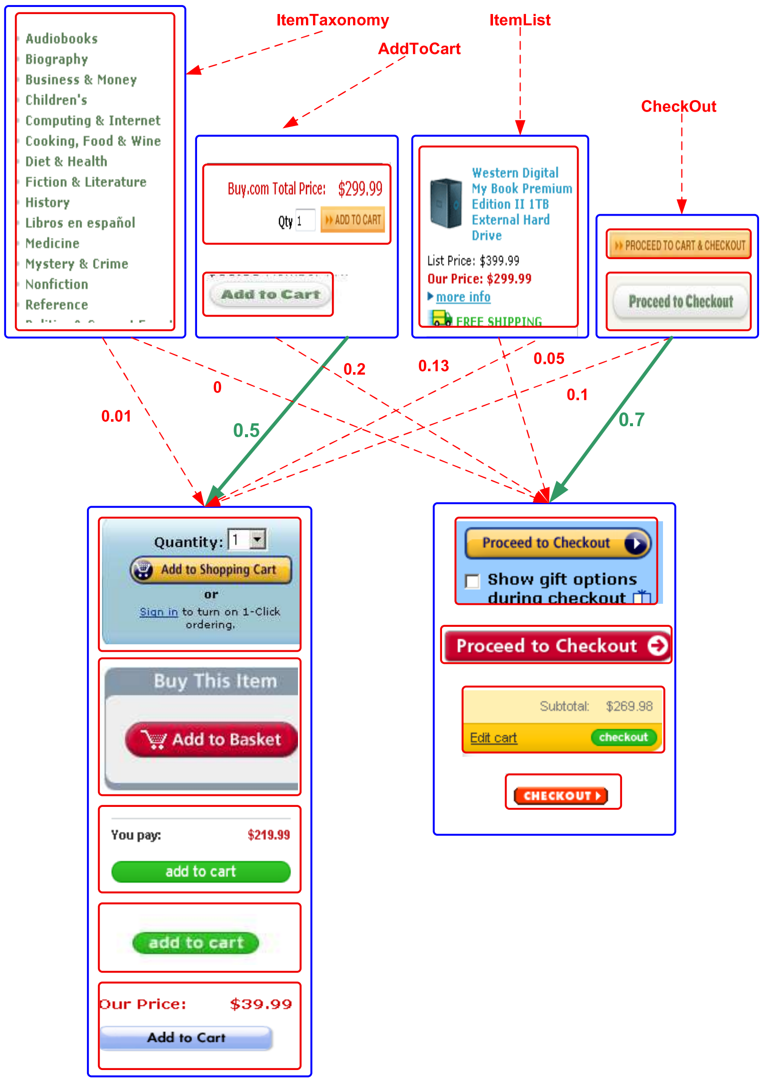


Figure 30: Labeling Unlabeled Clusters using Labeled Clusters

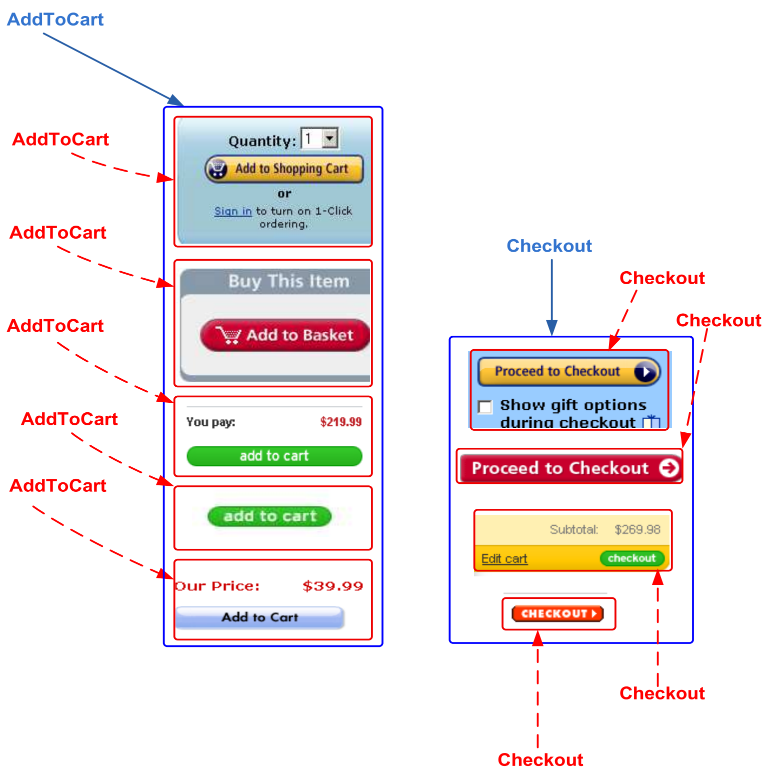


Figure 31: Concept Segments Labeling using Cluster Label

Then we can label the concept segment (i.e. categorize the segment) with the label of its most similar cluster. This approach is simple and will work for many cases, but it has some shortcomings:

- Inter-cluster similarity does not weight the features. Observe that jaccard similarty metric does not consider importance of a particular feature. Using uniform weight for each feature may result poor performance, especially when there are common features across different concepts.
- The entire process of clustering is done in a batch mode and hence this is offline. Once a set of transaction sequences are collected, we apply our algorithm. If clusters have to be used for classification, then they have to stored. That means, all the concept segments extracted from the transaction sequences have to be stored. This is not scalable when the transaction models are learned from a large number of sequences. Moreover, it is very calculation intensive. Very large cluster, i.e. a cluster with many concept segments can make computation of inter-cluster similarity difficult. From a computational point of view, it may require a long time and is not suitable for online processing.
- The simple approach described above assumes that any segment is an instance of a concept since it classifies a given segment as an instance of a concept class when the segment gets the highest similarity with the cluster corresponding to that concept. However, it can be a noisy segment. To eliminate such noisy segments, we have to use a threshold for inter-cluster similarity value for each clusters. Manually fixing such threhsolds is not justifiable. Hence, these thresholds have to be determined experimentally.

However, a statistical model, e.g. Support Vector Machine (SVM) [89] has the following advantages:

- An SVM model learned from features vectors of the training examples computes weight (i.e. contribution) of each individual feature to classify an instance as a member of the class. This is more justifiable and accurate than using uniform weights for each feature which is the case when we compute inter-cluster similarity.

- An SVM models are usually learned as a binary classifier to classify an instance as either member or not member of the class. Therefore, there is no need to use thereshold for each cluster to eliminate noisy segments.
- A learned SVM model does not store each data point, it stores only the support vectors which are needed to classify a given instance as a member of that class. SVM computation is generally faster than computing similarity value from large number of data points.
- SVM is a well-known statistical model used in classification and regression. It has been successfully applied to build classifier from both linearly seperable and non-seperable data points.

Therefore, we automatically learn a support vector machine (SVM) [23, 89] for each such concept. Each such concept models are used as a binary classifier to classify a given concept segment as either instance or not instance of the concept.

Algorithm *LearnConceptModel*
Input: *Sequences*: A Set of Labeled Transaction Sequences
Output: *Models*: A Set of Statistical Models

```
1.  $ConceptSets \leftarrow \emptyset$
2.  $n \leftarrow Sequences.Size$
3.  $i \leftarrow 1$
4.  repeat
5.      $Sequence \leftarrow Sequences$(i)
6.      $WebObjectsSeq \leftarrow Sequence.WebObjects$
7.      $m \leftarrow WebObjectsSeq.Size$
8.      $j \leftarrow 1$
9.      repeat
10.         $WebObject \leftarrow WebObjectsSeq$(j)
11.         $ConceptSegment \leftarrow GetConceptSegment$(WebObject)
12.         $ConceptSegment.Label \leftarrow WebObject.Label$
13.         Place $ConceptSegment$ in $ConceptSets$(Label)
14.         $j \leftarrow j + 1$
15.     until $j$ =m+1
16.     $i \leftarrow i + 1$
```

Table 7: Description of Concept Segment Features

Feature	Description
$f_{unigram}$	words present in the segment
f_{bigram}	bigram (pairs of words)
$f_{trigram}$	trigram (triples of words)
$f_{stemUnigram}$	word stems*
$f_{stemBigram}$	stemmed bigrams
$f_{stemTrigram}$	stemmed trigrams
$f_{pattern}$	patterns

```
17. until $i$ =n+1
18. $numConcept \leftarrow$ Total Number of Concepts
19. $Models \leftarrow \emptyset$
20. $i \leftarrow 1$
21. repeat
22.     $NegativeExamples \leftarrow ConstructNegExamples(ConceptSets, i)$
23.     $ConceptModel$(i) $\leftarrow LearnSVM$(ConceptSets($i$),NegativeExamples)
24.     $Models \leftarrow Models \cup ConceptModel$(i)
25.     $i \leftarrow i + 1$
26. until $i$ =numConcept+1
27. return $Models$
```

As with many other machine learning tools, an SVM [23, 89] takes a feature vector as input and produces its classification. Here, we define two classes for each of our concept model: **Concept** and **Not Concept**, and describe each concept segment with a set of feature values.

For example, classes for the concept model "CheckOut" are **CheckOut** and **Not CheckOut**.

We represented each concept segment by a tuple $(\vec{f}, l)$, where $\vec{f}$ is a feature vector for that segment and l is its concept label. A segment can be either instance of a concept: $l = 1$, or not instance: $l = 0$. Segments labeled with concept C_i are used as positive examples to train the SVM model for concept C_i. Segments labeled with concept C_j, where i $\neq$ j are used as negative examples to train the SVM model for concept C_i.

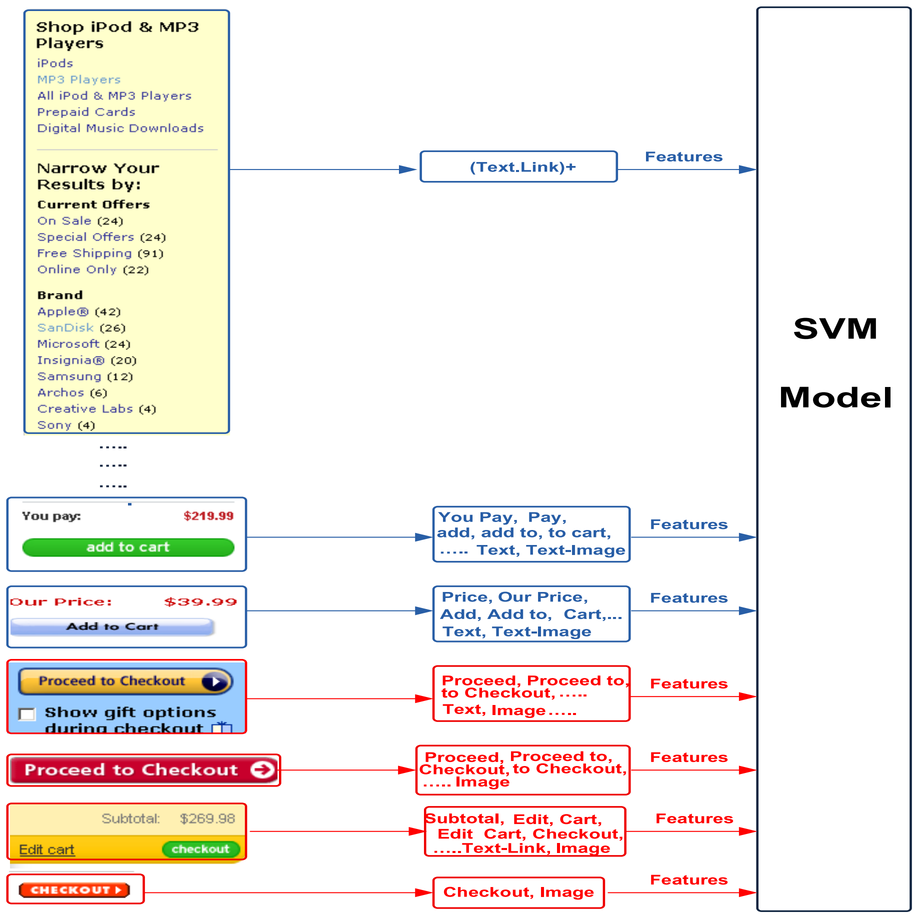


Figure 32: Positive and Negative Examples for "CheckOut" SVM Model

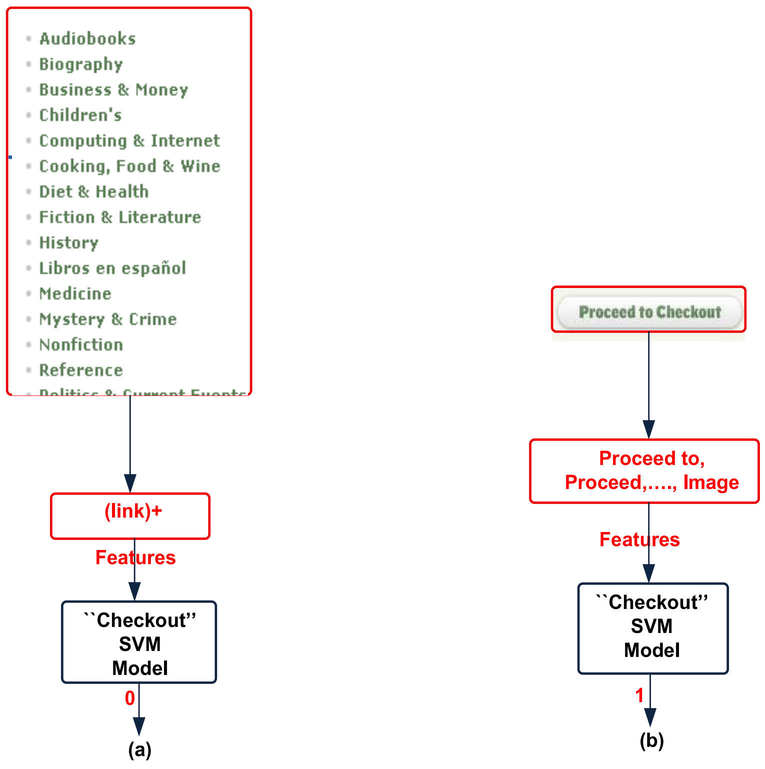


Figure 33: Classification of Segments using "CheckOut" SVM Model

Note that, we did not manually label the concept segments. The labels are generated as a result of clustering them and hence this is automatic. Thus in contrast to traditional SVM learning which requires manually labeled training data and hence is supervised, we used an unsupervised approach to learn the SVM models.

Continuing with our example in the previous section, Figure 32 shows concept segments used as positive examples and some of the concept segments used as negative examples for training the SVM model for "Checkout" concept. Positive examples are concept segments taken from the "Checkout" cluster. Negative examples are concept segments taken from "AddToCart", "ItemTaxonomy" and "ItemList" (not shown in Figure 32 because of limited space) clusters. Some of the features extracted from the concept segments are also shown in Figure 32.

The learned SVM concept model is used to classify a given segment as a concept. Given a segment B_i, we compute the feature values for it. Next, we use the learned SVM models to label the segment as as either instance or not-instance of the concept and get the associated probability values. Note that, the segment can be identified as an instance of a concept by more than one SVM model (and which is an ambiguity). In that case, we label the segment as an instance of the concept for which corresponding SVM model returns the highest probability value.

For example, Figure 33 shows two concept segments. One of them is classified as a "Checkout" concept (33(b))and the other one is not classified as a "Checkout" (33 (a)) concept by "Checkout" SVM model.

5.3 Learning Process Model

In the previous chapter, we have described process model learning using a DFA-based technique. There are some drawbacks of using DFA learning to learn a process model from transaction traces.

5.3.1 Limitation of using DFA learning

- The goal of DFA learning is to learn the smallest size DFA that is consistent with respect to a set of positive and negative training examples. This is an NP-hard [9, 10, 38] problem. As a result, there are efficient heuristics for DFA learning(*e.g.,* [33, 66, 68]) in polynomial time. However, training examples must certain properties to learn a target DFA using these heuristics. For example, Parekh and Honavar [70] assume a structurally complete set [2]. Oncina [68] assumes a characteristics set [3] of samples. Web transaction traces are collected by maintaining logs of transaction activities done by users. It is quite diffcult to collect traces that satisfies such properties, which are necessary for DFA learning in polynomial time.

- DFA learning requires a sizable number of negative examples which are often difficult to obtain from logs of transaction activities done by users. Moreover, collecting negative traces may introduce spurious concepts in the model as a result of clustering.

  One solution to get negative examples is to generate them from positive examples. Specifically, each subsequence of a positive example may be used as a negative example. However this has the following drawback.

  - This assumes that a subsequence of a completed transaction can not complete another transaction.
    For example, consider the positive example "select_item_category"."select_item". "add_to_cart"."check_out". If we generate a negative example "select_item". "add_to_cart"."check_out" from the above mentioned positive example, then the resultant model won't accept "select_item"."add_to_cart"."check_out". However, this transaction sequence can also complete a transaction. Thus, generating negative examples from subsequences of positive examples may result in rejecting transaction sequences by the model which should be accepted.

5.3.2 Inference of the Process Model from Positive Examples

For unsupervised transaction model mining, we have developed a learning algorithm that can learn a process model from only positive examples, i.e. complete

[2]A set S+ is said to be structurally complete with respect to a DFA A if S+ covers each transition of A and uses every element of the set of final states of A as an accepting state [34, 70]

[3]A characteristics set S= S+ $\cup$ S- is such that S+ is structurally complete with respect to the target and S- prevents merging of any two states that are not equivalent

transaction sequences. We observe that:

- If a transaction sequence with a consecutive repeating operation subsequence completes a valid transaction, then discarding the repeat (i.e. having the operation subsequence only once in the whole sequence) or inserting any number of such repeating operation subsequence consecutively in the sequence should also complete a valid transaction.

  For example, consider the following sequence $select\_item.select\_item.$ $add\_to\_cartcheck\_out$. This completes a transaction (selecting an item, selecting another item, adding that item to the shopping cart and then checking out). The sequence contains exactly one consecutive repeated subsequence, $select\_item$. Removing that operation gives another transaction sequence $select\_item.$
  $add\_to\_cart.check\_out$. This is also a valid transaction (selecting an item, adding that item to the shopping cart and then checking out). Similarly, the following sequence $select\_item.select\_item.select\_item.add\_to\_cart.$
  $check\_out$ also completes a transaction.

So given a set of completed transaction sequences the aforementioned insert and delete operations allow a limited degree of generalization. We will learn a process automaton (i.e. process model) to accept these kinds of generalized sequences from a training set T of completed transaction sequences. The details are as follows:

Definition 1 (Language of transaction sequences) *Given a training set T, the language of transaction sequences* [4]*, denoted by $\mathcal{A}(T)$ is the* smallest *set such that:*

- *$T \subseteq \mathcal{A}(T)$, and*
- *for all $x \in \mathcal{A}(T)$ such that $x = pmms$ (p is the prefix, s the suffix and m the repeated middle, all possibly empty), $pm^ks \in \mathcal{A}(T)$ for every $k > 0$.*

Note that $\mathcal{A}(T)$ generalizes T, and is the language we seek to learn from T. In particular, we generalize T such that any consecutively repeating substring in T is now permitted to repeat an arbitrary number of times. The language $\mathcal{A}(T)$ has

[4]*The language definition and the algorithm to construct process automaton were developed with Prof. C.R. Ramakrishnan*

an important property: it is closed with respect to training sets in the sense that adding any string in the language to the training set does not change the language. Formally,

Theorem 1 (Closure) *Let T be a given training set and $\mathcal{A}(T)$ be the corresponding language of transaction sequences. Then, for all S such that $T \subseteq S \subseteq \mathcal{A}(T)$, $\mathcal{A}(S) = \mathcal{A}(T)$.*

The proof of this property follows from the monotonicity of $\mathcal{A}$: i.e. if $T \subseteq S$ then $\mathcal{A}(T) \subseteq \mathcal{A}(S)$. The property of closure indicates "stability" of the learned language since no string in the language could have been added to the original training set to construct a different (more general) language.

The definition of $\mathcal{A}$ is does not directly give a procedure to construct an automaton that accepts $\mathcal{A}(T)$. We now outline such a procedure.

Definition 2 *Let $\mathcal{R}(T)$ be the set of regular expressions over the alphabet of T, defined as follows:*

- $T \subseteq \mathcal{R}(T)$
- $\forall x \in T$ *such that* $x = pmms$, $pm^+s \in \mathcal{R}(T)$.

Note that $\mathcal{R}(T)$ is a finite set of regular expressions (REs); in particular, if the largest sequence in T is of length k, then $|\mathcal{R}(T)| = O(k^2|T|)$, and the largest regular expression in $\mathcal{R}(T)$ is of length k^2 or less.

Let $\mathcal{L}(r)$ denote the language of a regular expression r. The language of a set of regular expressions is the union of the languages of each of its elements: i.e. if R is a set of regular expressions, then $\mathcal{L}(R) = \cup_{r \in R} \mathcal{L}(r)$.

The language of regular expressions $\mathcal{R}(T)$ constructed from the training set is identical to $\mathcal{A}(T)$, the language of transaction sequences learned from T, as formally stated below.

Theorem 2 *For all sets of training sequences T, $\mathcal{A}(T) = \mathcal{L}(\mathcal{R}(T))$.*

The above theorem can be proved by considering the usual least fixed point (iterative) construction of $\mathcal{A}(T)$, and showing that the least fixed point computation will converge in two steps to $\mathcal{L}(\mathcal{R}(T))$.

Note that $\mathcal{R}(T)$ gives us an effective procedure for constructing the transaction automaton. For each regular expression in $\mathcal{R}(T)$ we construct the corresponding nondeterministic finite automaton (NDFA) using Thomson's construction [46]. The automaton for $\mathcal{L}(\mathcal{R}(T))$ is simply the union of all the individual automata. Based on the argument above on the size of $\mathcal{R}(T)$ and Thomson's construction, it follows that if k is the length of the longest sequence in T, then the automaton for $\mathcal{A}(T)$ thus constructed is of size $O(k^4|T|)$, and can be constructed in time $O(k^4|T|)$. Finding more efficient construction algorithms and building smaller automata are topics of future research.

Next, we describe such a learning algorithm in detail.

5.3.2.1 Unsupervised Learning of the Process Model

The steps for learning the process model are as follows.

- Start with training transaction Sequences and a set containing regular expressions, which is initially empty.
- For a sequence TS_i, find all possible non-empty subsequences m such that $TS_i = pmms$
  - for each such $pmms$, generate a regular expression pm^+s and insert it to the set containing regular expressions
  - if no non-empty m is found, then insert TS_i to the set containing regular expressions
- Get the union of the regular expressions from the set containing regular expressions
- Build a non-deterministic automaton from the regular expression constructed above using a standard algorithm [4, 46, 84] and return the automaton as the learned process model from training sequences.

The detailed learning algorithm is illustrated in *LearnProcessModel*. Line 1 of the algorithm initializes the set *ExpSet* to be empty. Line 2 to 10 is the main iteration loop. Line 3 takes each training example, and line 4 invokes the algorithm *FindConsecutiveRepeatingSubsequences*. This algorithm generates all possible subsequences in a sequence given as parameter (line 4 and line 5 of algorithm

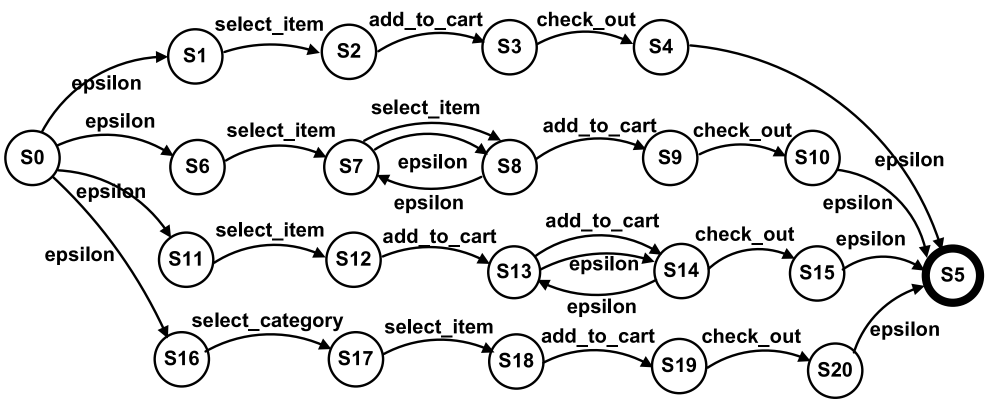


Figure 34: Non Deterministic Process Model (with epsilon transitions) Constructed from Regular Expressions in Table 9

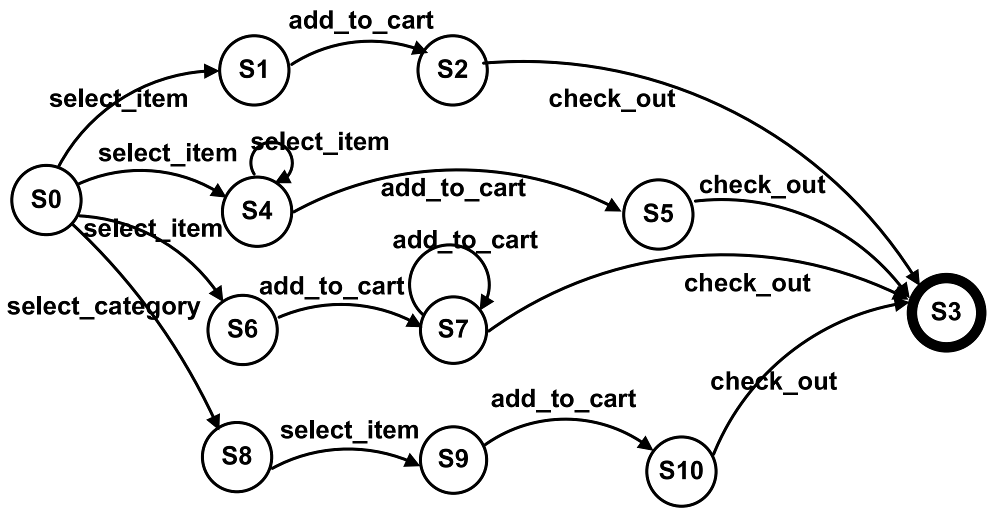


Figure 35: Non Deterministic Process Model after removing epsilon moves from the model in Figure 34

FindConsecutiveRepeatingSubsequences). Next it checks for consecutive repeating subsequences (line 6 of *FindConsecutiveRepeatingSubsequences*). If such a repeating subsequence is found, then the algorithm breaks the entire sequence in the form *pmms* (line 7, 8, 9 of *FindConsecutiveRepeatingSubsequences*) where,

- *p* is the prefix which can be empty.
- *m* is the repeating middle part.
- *s* is the suffix which can be empty.

Algorithm *LearnProcessModel*
Input: *Sequences*: A Set of Training Sequences Given as Positive Examples
Output: *(Model)*: The learned process model

```
1.  $RegExpSet \leftarrow \emptyset$
2.  for $i \leftarrow 1$ to $Sequences.Size$
3.      do $Sequence \leftarrow Sequences(i)$
4.         $PreMidSuffSet \leftarrow FindConsecutiveRepeatingSubseq(Sequence)$
5.         if $PreMidSuffSet.Empty$
6.            then $ExpSet \leftarrow ExpSet \cup Sequence$
7.            else
8.                 for $j \leftarrow 1$ to $PreMidSuffSet.Size$
9.                     do $(Prefix, Middle, Suffix) \leftarrow PreMidSuffSet(j)$
10.                       $ExpSet \leftarrow ExpSet \cup Prefix.Middle^{+}.Suffix$
11. $NonDeterministicProcessModel \leftarrow$ BuildNFAFromRegExp($ExpSet$)
12. return $(NonDeterministicProcessModel)$
```

Algorithm *FindConsecutiveRepeatingSubsequences*
Input: *Sequence*: A Training Sequence
Output: *(PrefixMidMidSuffixSet)*: A Set where Each Entry is of the form (Prefix, Middle, Suffix)

```
1.  $PrefixMidMidSuffixSet \leftarrow \emptyset$
2.  for $i \leftarrow 1$ to $Sequence.Size$
3.      do for $j \leftarrow 1$ to $Sequence.Size$
4.            do $SubStr \leftarrow Sequence.SubString(i, j)$
5.               $SubStrNext \leftarrow Sequence.SubString(j+1, j+ SubStr.Length)$
```

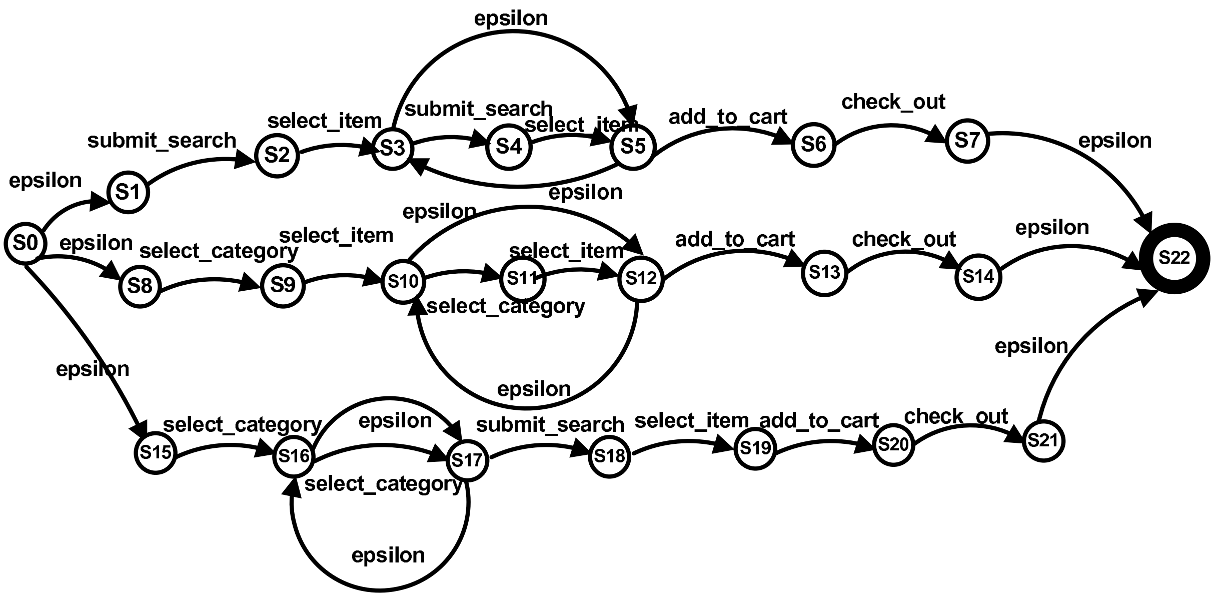


Figure 36: Non Deterministic Process Model Constructed from Regular Expressions in Table 11

```
6.              **if** $SubStr$.Equals($SubStrNext$)
7.                **then** $Prefix \leftarrow Sequence.SubString(1, i\text{ -}1)$
8.                     $Middle \leftarrow SubStr$
9.                     $Suffix \leftarrow Sequence.SubString(j + SubStr.Length + 1,$
                       $Sequence.Size)$
10.                    $PrefixMidMidSuffixSet \leftarrow PrefixMidMidSuffixSet$
                       $\cup$(Prefix, $Middle$, $Suffix$)
11. **return** *($PrefixMidMidSuffixSet$)*
```

The algorithm inserts (line 10 of *FindConsecutiveRepeatingSubsequences*) each such prefix, middle and suffix into a set (*PrefixMidSuffixSet*) which is returned as output in line 4 of algorithm *LearnProcessModel*.

For example, consider the training sequences in Table 8. The first sequence is *select_item.add_to_cart.check_out* which does not contain any repeated subsequence. Therefore, line 4 of the algorithm *LearnProcessModel* returns the empty set. For the next sequence (*select_item.select_item.add_to_cart.check_out*), the

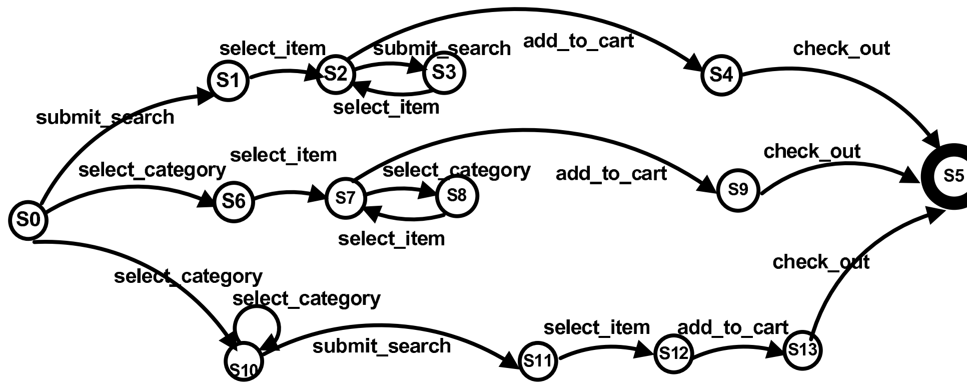


Figure 37: Non Deterministic Process Model after removing epsilon moves from the model in Figure 36

subsequence
select_item is repeated consecutively. This sequence has a unique *pmms* representation, where *p* is *empty*, *m* is *select_item* and *s* is *add_to_cart.check_out*. The next sequence (*select_item.add_to_cart.add_to_cart.check_out*) also has a unique *pmms* representation where *p* is *select_item*, *m* is *add_to_cart* and *s* is *check_out*. The final sequence in the table 8 *select_item_category.select_item.add_to_cart.check_out* does not contain a repeated subsequence and hence line 4 of the algorithm (*LearnProcessModel*) returns the empty set. Line 5 of the algorithm *LearnProcessModel* checks whether the set of (Prefix, Middle, Suffix) is empty. If this set is empty (i.e. no consecutive repeating subsequence in the sequence), the sequence is added to the set containing Regular Expressions (line 6). However, for each non-empty such set, a Regular Expression is added to the set *ExpSet*. The form of each such expression is $Prefix.Middle^{+}.Suffix$.

Continuing with our example, the first and fourth training sequences are added to the set *ExpSet* (since the set of ((Prefix, Middle, Suffix) become empty for these sequences). For the second sequence, the Regular Expression $select\_item^{+}.add\_to\_cart.check\_out$ is added to the set. For the third sequence, $select\_item.add\_to\_cart^{+}.check\_out$ is added to the set. The set of Regular Expressions is shown in Table 9.

No.	Training Sequence
1	$< select\_item, add\_to\_cart, check\_out >$
2	$< select\_item, select\_item, add\_to\_cart, check\_out >$
3	$< select\_item, add\_to\_cart, add\_to\_cart, check\_out >$,
4	$< select\_item\_category, select\_item, add\_to\_cart, check\_out >$

Table 8: Labeled Training Sequences for Process Model Learning

No.	Regular Expressions
1	$< select\_item, add\_to\_cart, check\_out >$
2	$< select\_item^+, add\_to\_cart, check\_out >$
3	$< select\_item, add\_to\_cart^+, check\_out >$,
4	$< select\_item\_category, select\_item, add\_to\_cart, check\_out >$

Table 9: Regular Expressions for Sequences given in 8

Once we have a set of Regular Expressions, we construct an NFA from the expression using a standard algorithm [46, 84] (line 11). The resultant Non Deterministic Process Model is shown in Figure 34.

Now we illustrate the process model learning algorithm with the labeled examples in Table 10. The set of regular expressions for these examples are shown in Table 11. $weconstructNFAfromtheexpressions$using standard algorithm. The resulting Non-Deterministic Process Model is shown in Figure 36.

Note that it is possible to convert the non deterministic process model returned by our algorithm to a deterministic process model. However, the existing subset construction algorithm that converts a NFA to DFA is computationally expensive (exponential complexity). So we will use the non-deterministic process model returned by our algorithm to avoid such computation.

No.	Training Sequence
1	$< submit\_search, select\_item, submit\_search, select\_item, add\_to\_cart, check\_out >$
2	$< select\_category, select\_item, select\_category, select\_item, add\_to\_cart, check\_out >$
3	$< select\_category, select\_category, submit\_search, select\_item, add\_to\_cart, check\_out >$

Table 10: Labeled Training Sequences for Process Model Learning

No.	Regular Expressions
1	$< (submit\_search, select\_item)^+, add\_to\_cart, check\_out >$
2	$< (select\_category, select\_item)^+, add\_to\_cart, check\_out >$
3	$< select\_category^+, submit\_search, select\_item, add\_to\_cart, check\_out >$

Table 11: Regular Expressions Computed for the Sequences in 10

In Section Chapter 3, we defined a process model as a deterministic automaton. According to that definition: when the model makes a transition to a state during the course of a transaction, a Web page is provided to the state as an input. If the concepts associated with the state are present in the page, then they alone are identified and presented to the user.

However, in a nondeterministic process model, we have to consider a set of states instead of a single state at any given step of a transaction. When the model makes a transition, it makes a transition to a set of states (possibly a set with a single state). As a result the Web page is provided to the set of states as an input. If the concepts associated with the states (from that set of states) are present in the page, then they alone are identified and presented to the user. This modification is very straightforward and hence a non-deterministic process model can also be used to conduct transactions.

5.4 Personalized Transaction Models

Concept instances shows variability in texts and presentation patterns. However, this variability is less common in a single Website. Therefore, clustering concept segments from different Websites often places concept segments in the wrong cluster. For example, an “AddToCart” concept instance in “Amazon.com” contains the text ”add to cart” but the same concept instance in “Buy.com” contains the text “buy now”. The concept segments collected from these instances may be placed in different clusters if other textual and pattern features do not have enough match.

However, a single user most often visits some common Web sites. A Web transaction model for each such site can capture the user’s personalized transactions in that Website. Therefore, we can also mine personalized transaction models using the algorithm described in the previous sections.

5.5 Experiments

We conducted a series of experiments to evaluate the performance of the transaction model learning algorithms.

5.5.1 Data Collection

To have an efficient infrastructure for experiments, a visual tool [18] was designed for viewing frame trees, as well as collecting data. A Web browser was also embedded to aid the data collection. During the data collection stage, participants were asked to select any link, or submit a search form to navigate from one page to another page. As they navigate from one page to another page, the frame trees, corresponding to the source and the destination pages, were automatically saved together with user selections. This is how we collected unlabeled transaction sequences. To aid the collection of labeled sequences, participants chose the name of the transaction operation for each navigation. They also selected the nodes of the frame tree as instances of the concept associated with the operation.

5.5.2 Datasets

Around *500* transaction sequences from 36 Websites in the online shopping domain (books, electronics, office supplies) were manually collected using the data collection method described above. 15 CS graduate students (all sighted) were used to collect transaction sequences. Around *300* transaction sequences were unlabeled and the remaining *200* were labeled. We denote the set of unlabeled sequences as $UnlabeledSeq$, and the set of labeled sequences as $UserLabeledSeq$. We applied our concept segment retrieval algorithm on sequences in $UnlabeledSeq$ and $UserLabeledSeq$ to get set of concept segments, $C_{Unlabeled}$ and $C_{UserLabeled}$. The set $C_{UserLabeled}$ contained 807 labeled concept segments (total eight concept labels were used). and the set $C_{Unlabeled}$ contained 1092 unlabeled segments.

We split the set $C_{UserLabeled}$ into $C_{TestSegments}$ and $C_{LabeledSegments}$. $C_{TestSegments}$

Cluster Label	Number of User Labeled Segments
$SearhForm$	91 (67 in $C_{TestSegments}$, 24 in $C_{LabeledSegments}$)
$AddToCart$	118 (91 in $C_{TestSegments}$, 27 in $C_{LabeledSegments}$)
$ContinueShopping$	100 (72 in $C_{TestSegments}$, 28 in $C_{LabeledSegments}$)
$CheckOut$	114 (83 in $C_{TestSegments}$, 31 in $C_{LabeledSegments}$)
$ShoppingCart$	98 (74 in $C_{TestSegments}$, 24 in $C_{LabeledSegments}$)
$EditCart$	96 (70 in $C_{TestSegments}$, 26 in $C_{LabeledSegments}$)
$ItemList$	93 (68 in $C_{TestSegments}$, 25 in $C_{LabeledSegments}$)
$ItemTaxonomy$	97 (75 in $C_{TestSegments}$, 22 in $C_{LabeledSegments}$)

Table 12: Segments in User Labeled Clusters

Dataset	No of Sequences	No of Concept Segments	Labeling
$UnlabeledSeq$	300	1092	Unlabeled
$UserLabeledSeq$	200	807	User Labeled

Table 13: Description of Datasets.

contained 600 segments and was used as a validation set to determine the performance of the clustering algorithm, and also to construct the test data sets (The test sets are described in Table 14). $C_{LabeledSegments}$ contained 207 segments and was used to label unlabeled segments (described in Section 5.1.3).

Distribution of user labeled concept segments to each concept label is presented in table **??**. Note that, since the concept segments in both $C_{TestSegments}$ and $C_{LabeledSegments}$ are labeled, 8 labeled clusters are contructed from them. This is done by constructing a cluster for each concept label. Let $Cluster_{user}$ (constructed from $C_{TestSegments}$ which contains 600 concept segments) and $Cluster_{labeling}$ (constructed from $C_{TestSegments}$ which contains 207 concept segments) denote the set of labeled clusters.

5.5.3 Performance of Clustering

Note that, our test set may contain only unlabeled concept segments (T_U) or a fraction of the concept segments can be labeled (e.g. 5% labeled). We apply the clustering algorithm on segments in testset (testsets are described in 17), and eliminate the noisy clusters. Let $Cluster_{algo}$ denotes the set of clusters constructed

Test Set.	Description
T_U	Concept Segments are Unlabeled
T_5L	5% Concept Segments are Labeled
T_10L	10% Concept Segments are Labeled
T_15L	15% Concept Segments are Labeled
T_20L	20% Concept Segments are Labeled
T_25L	25% Concept Segments are Labeled
T_30L	30% Concept Segments are Labeled
T_35L	35% Concept Segments are Labeled
T_40L	40% Concept Segments are Labeled
T_45L	45% Concept Segments are Labeled
T_50L	50% Concept Segments are Labeled

Table 14: Test Datasets for Clustering Performance Evaluation

Test Set.	Number of Clusters
T_U	12
T_5L	11
T_10L	9
T_15L	12
T_20L	12
T_25L	11
T_30L	10
T_35L	10
T_40L	9
T_45L	8
T_50L	8

Table 15: Number of Clusters Constructed from each Test Dataset

from the testset. We label the clusters in $Cluster_{algo}$ using the labeled clusters, $Cluster_{labeling}$ (the labeling procedure is described in section 5.1.3). Number of clusters constructed (after eliminating noisy clusters) for each testset is presented in 15.

We will determine the performance of clustering in terms of recall/precision/f-measure [5] for each concept present in $Cluster_{user}$.

[5]recall value of clustering instances (i.e. segments) of a particular concept is the ratio of number of concept instances (i.e. segments) clustered correctly (i.e. labeled as instance of that concept) over

Let $Cluster_{user_i}$ denotes the cluster containing concept segments labeled as concept *i* by user and $Cluster_{algo_i}$ denotes the cluster containing concept segments labeled as concept *i* by our algorithm.

We take each concept segment from $Cluster_{user_i}$ and compare its label (i.e. user label) with the label assigned by our algorithm. Let N_{total_i} be the total number of concept segments in $Cluster_{user_i}$. $N_{correct_i}$ be the number of concept segments which are present in both $Cluster_{user_i}$ and $Cluster_{algo_i}$. $N_{incorrect_i}$ be the number of concept segments which are present in $Cluster_{algo_i}$ but not present in $Cluster_{user_i}$. Then, the recall value for the clustering algorithm for concept *i* is $N_{correct_i}$ / N_{total_i}, and the precision value is $N_{correct_i}$/ ($N_{correct_i}$ + $N_{incorrect_i}$). The F-measure is calculated by taking the harmonic mean of recall and precision.

For each concept, we compute recall/precision/f-measure of clustering using the above formula. Figure 38 shows performance of clustering (in terms of f-measure). Note that, clustering performance is higher (around 80% f-measure) for some concepts (e.g. SearchForm, AddToCart, ContinueShopping, CheckOut) than the other concepts. This is because of the consistent textual and pattern features present in the segments of that cluster.

We average the clustering performance (i.e. recall/precision/f-measure) of each concept to get the overall clustering performance. Figure 39 shows performance (f-measure) variation of clustering for different test datasets. As the amount of labeled data was increased, better clustering accuracy was achieved.

5.5.4 Performance of Transaction Model Learning

We computed the performance of transaction models (process model and concept models) for both the sets *UnlabeledSeq* (i.e. set of unlabeled transaction sequences) and *UserLabeledSeq* (i.e. set of labeled transaction sequences). The transaction sequences in *UnlabeledSeq* are labeled using the algorithm described in *GenerateLabel*.

Next, each of the sets was divided into training and testing. We used 90% sequences for training, 10% for testing and performed a standard 10-fold cross

the total number of that concept instance (i.e. segment). For precision, the denominator becomes the total number of instance (segment) labeled as that concept (either correctly or incorrectly). F-Measure is the simple harmonic mean of recall and precision

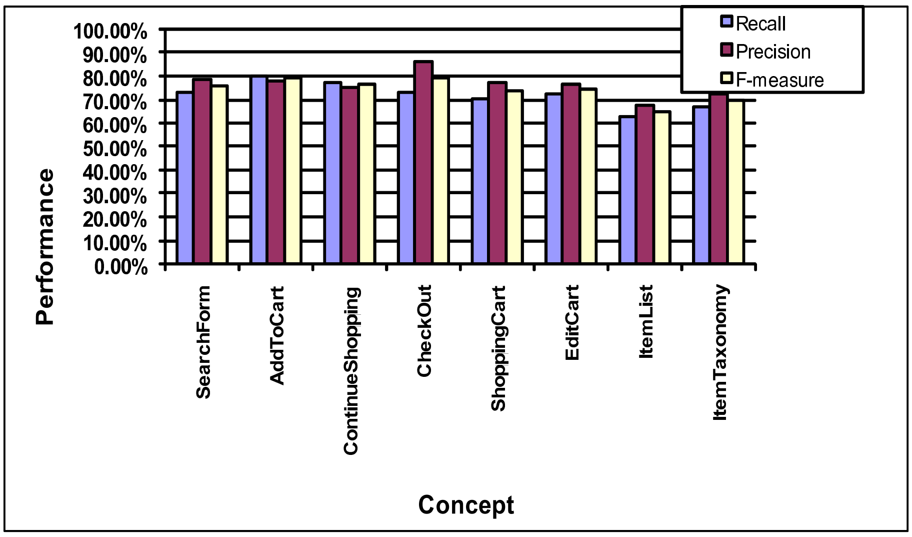


Figure 38: Performance of Clustering for Each Concept

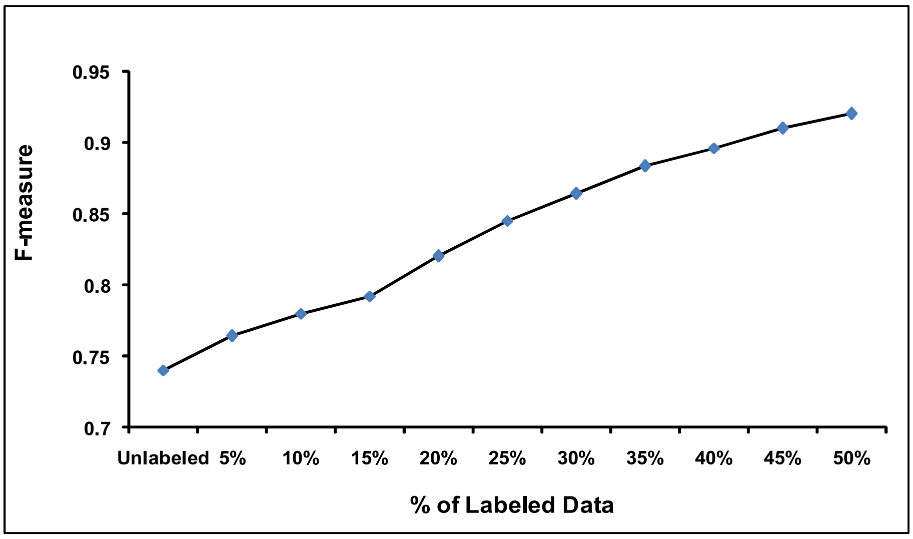


Figure 39: Performance of Clustering with Increasing Amount of Labeled Sequences

Dataset	No. of States in the Process Model
$Dataset_1$	45
$Dataset_2$	35
$Dataset_3$	32
$Dataset_4$	29
$Dataset_5$	33
$Dataset_6$	36
$Dataset_7$	35
$Dataset_8$	42
$Dataset_9$	34
$Dataset_{10}$	28

Table 16: Process Model Learning Statistics for each Training Set Generated from Unlabeled Sequences ($UnlabeledSeq$)

Dataset	No. of States in the Model
$Dataset_1$	34
$Dataset_2$	32
$Dataset_3$	29
$Dataset_4$	22
$Dataset_5$	28
$Dataset_6$	34
$Dataset_7$	27
$Dataset_8$	37
$Dataset_9$	31
$Dataset_{10}$	25

Table 17: Process Model Learning Statistics for each Training Set Generated from Labeled Sequences ($UserLabeledSeq$)

validation.

5.5.4.1 Performance of Process Model Learning

Note that the transaction sequences used for training were complete sequences. The test set contained both complete and incomplete transaction sequences.

We computed the recall/precision/f-measure [6] of the process model learning.

[6]Recall for a process model is the ratio of the number of completed transactions accepted by the

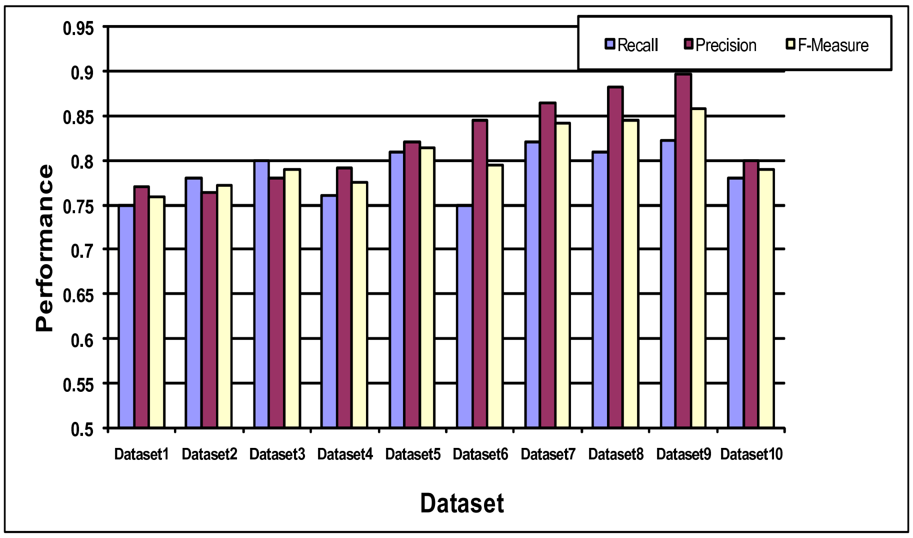


Figure 40: Performance of Process Model Learning from Unlabeled Sequences

Figure 40 shows the recall/precision/f-measure of the learned process model from unlabeled sequences (i.e. dataset *UnlabeledSeq*). Figure 41 shows the recall/precision/f-measure of the learned process model from labeled sequences (i.e. dataset *UserLabeledSeq*)

Note that performance of process model learning from labeled sequences (i.e. dataset *UserLabeledSeq*) is higher than the performance of process model learning from unlabeled sequences (i.e. dataset *UnlabeledSeq*). This is because concept segments extracted from unlabeled sequences are clustered and then concept labels are generated for them. Clustering of unlabeled concept segments introduces inaccuracy in the dataset *UnlabeledSeq*. On the other hand, concept segments in the dataset *UserLabeledSeq* are user-labeled and do not contain such inaccuracies. Hence, performance of process model learning for this dataset becomes higher than the other one.

model over the total number of completed transactions. For Precision, this denominator becomes the total number of accepted transactions (complete and incomplete). F-measure is the harmonic mean of recall and precision

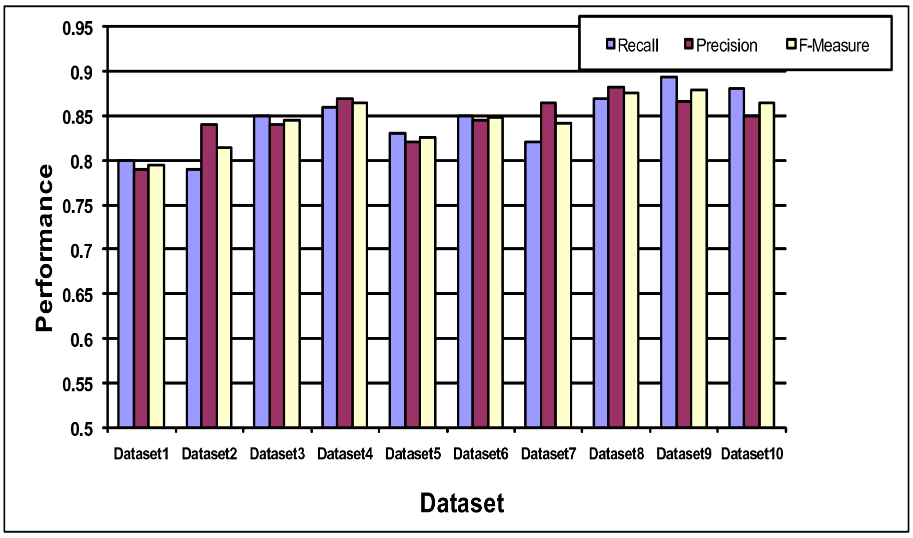


Figure 41: Performance of Process Model Learning from Labeled Sequences

5.5.4.2 Concept Identification

We measured the recall[7], precision and f-measure of concept identification.

Figure 42 shows the recall/precision/f-measure of the concept identification from unlabeled sequences (i.e. dataset *UnlabeledSeq*). Figure 43 shows the recall/precision/f-measure of the concept identification from labeled sequences (i.e. dataset *UserLabeledSeq*)

We average the concept identification performance of each concept to get the overall concept identification performance. We used 90% sequences for training, 10% for testing and performed a standard 10-fold cross validation.

Figure 44 shows the cross validation performance of the concept identification when concept models are learned from unlabeled sequences (i.e. dataset *UnlabeledSeq*). Figure 45 shows the cross validation performance of the concept identification when concept models are learned from labeled sequences (i.e. dataset *UserLabeledSeq*)

[7]Recall value for a concept is the ratio of the number of correctly labeled concept segments over the actual number of segments which are instances of that concept. For Precision, this denominator becomes the total number of segments labeled as instance of that concepts. F-measure is the harmonic mean of recall and precision

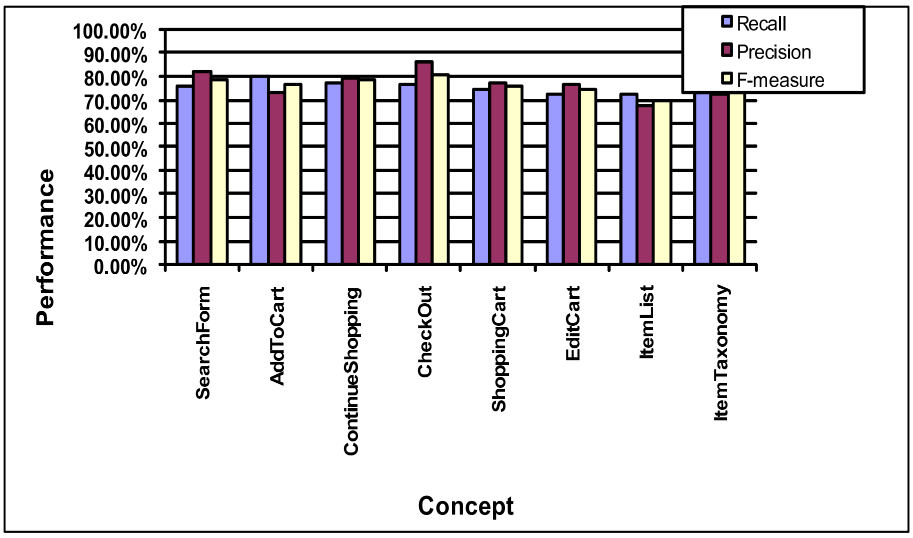


Figure 42: Performance of Concept Identification from Unlabeled Sequences

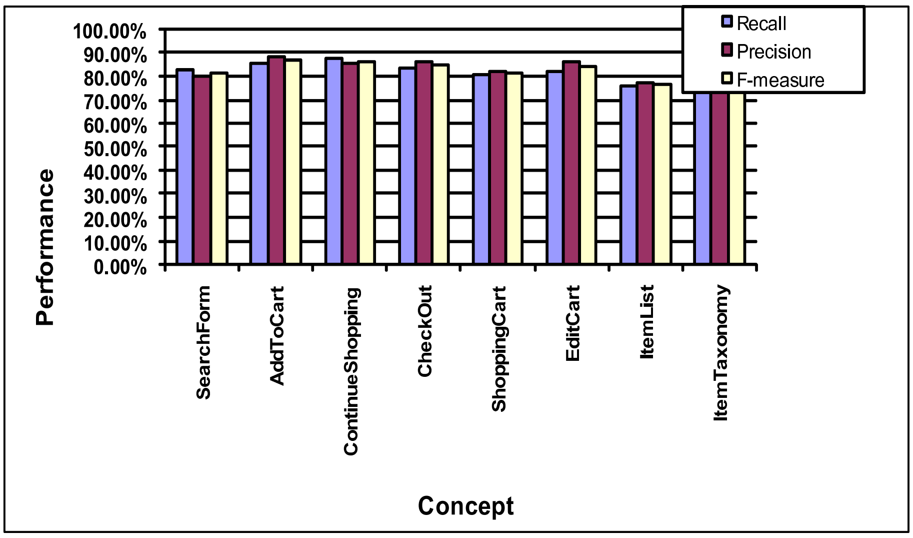


Figure 43: Performance of Concept Identification from Labeled Sequences

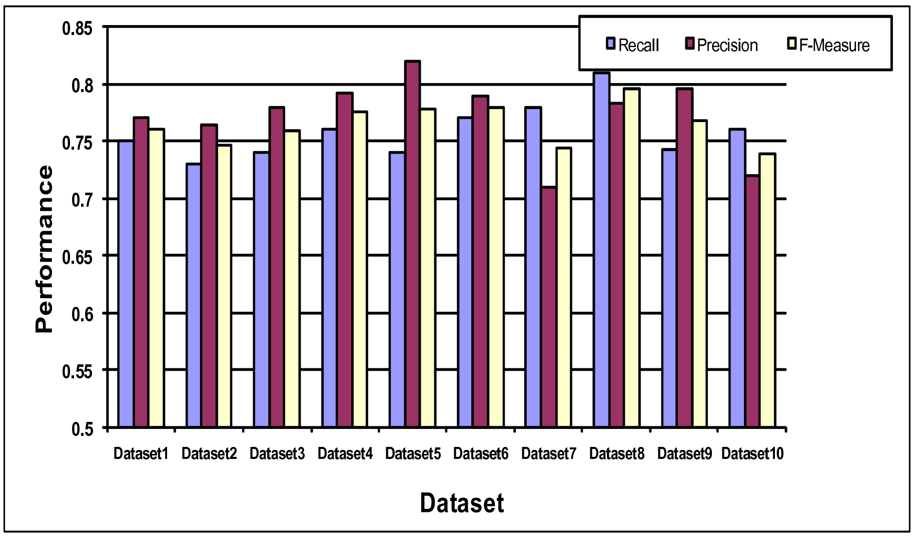


Figure 44: Cross Validation Performance of Concept Identification when Concept Models are Learned from Unlabeled Sequences

Note that, performance of concept identification from labeled sequences (i.e. dataset *UserLabeledSeq*) is higher than the performance of concept identification from unlabeled sequences (i.e. dataset *UnlabeledSeq*). This is because concept segments extracted from unlabeled sequences are clustered, and then concept labels are generated for them. Clustering of unlabeled concept segments introduces inaccuracy in the dataset *UnlabeledSeq*. On the other hand, concept segments in dataset *UserLabeledSeq* are user labeled and do not contain such inaccuracy. Hence, concept identification performance for this dataset becomes higher than the other one.

5.5.5 Single Site Transaction Model Evaluation

We constructed separate transaction models for 3 Web sites (“Amazon.com”, “OfficeMax.com”, “BN.com”) from our dataset (described in section 5.5.2). To construct a transaction model for a specific site, we separated the transaction sequences (both labeled and unlabeled) collected from that site. We had 15 unlabeled and 10 labeled sequences for “Amazon”, 10 unlabeled and 8 labeled sequences for “OfficeMax” and 13 unlabeled and 12 labeled sequences for “BN”. We used our algorithm to assign labels to unlabeled sequences, learned process model and concept

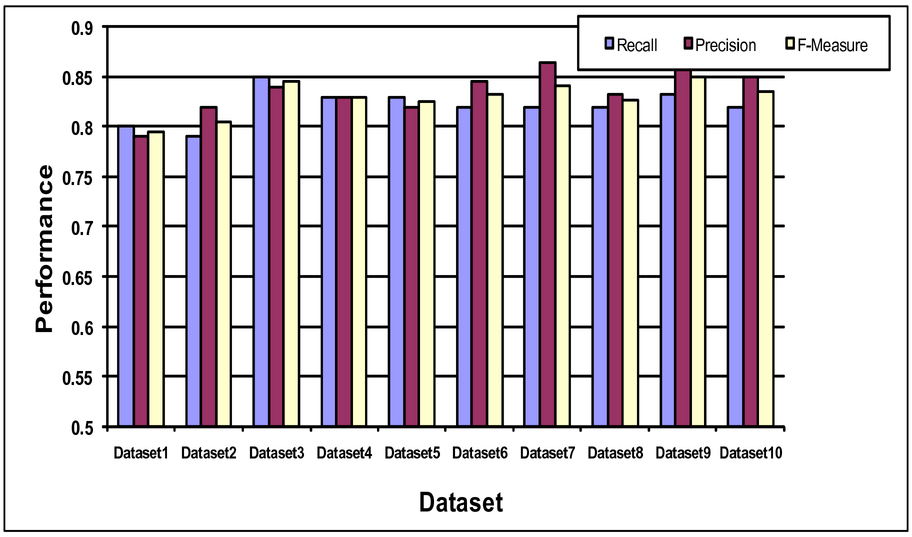


Figure 45: Cross Validation Performance of Concept Identification when Concept Models are Learned from Labeled Sequences

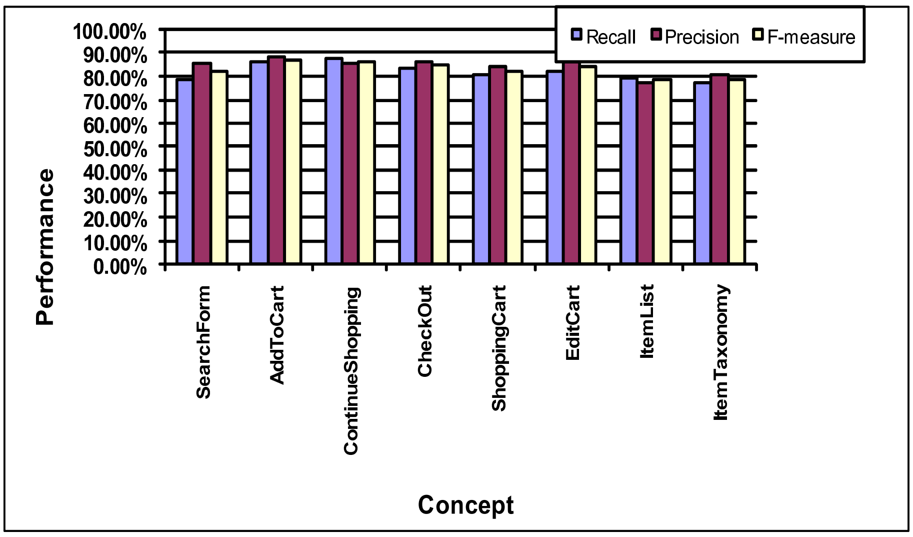


Figure 46: Performance of Clustering Unlabeled Sequences Collected from “Amazon”

models from labeled sequences.

We determined the performance of clustering, process model learning and concept identification from the transaction sequences collected from each site.

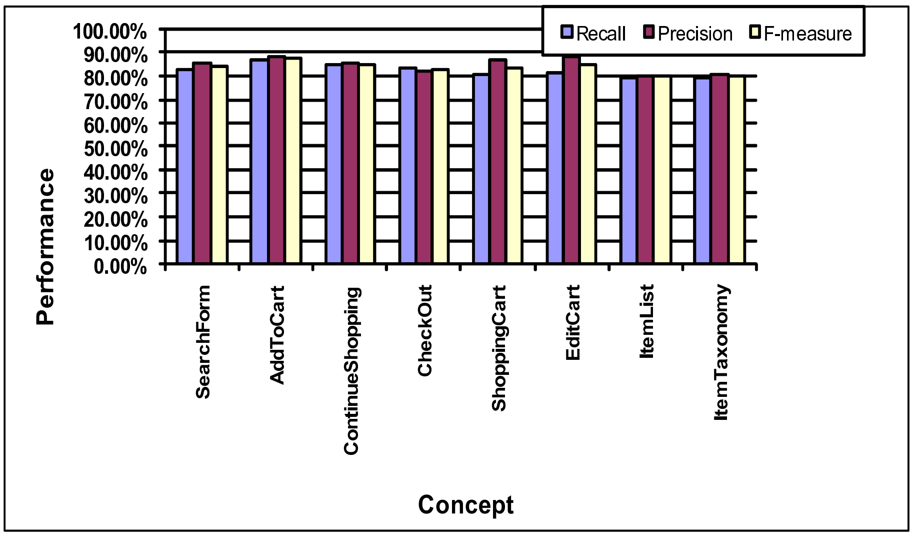


Figure 47: Performance of Clustering Unlabeled Sequences Collected from "OfficeMax"

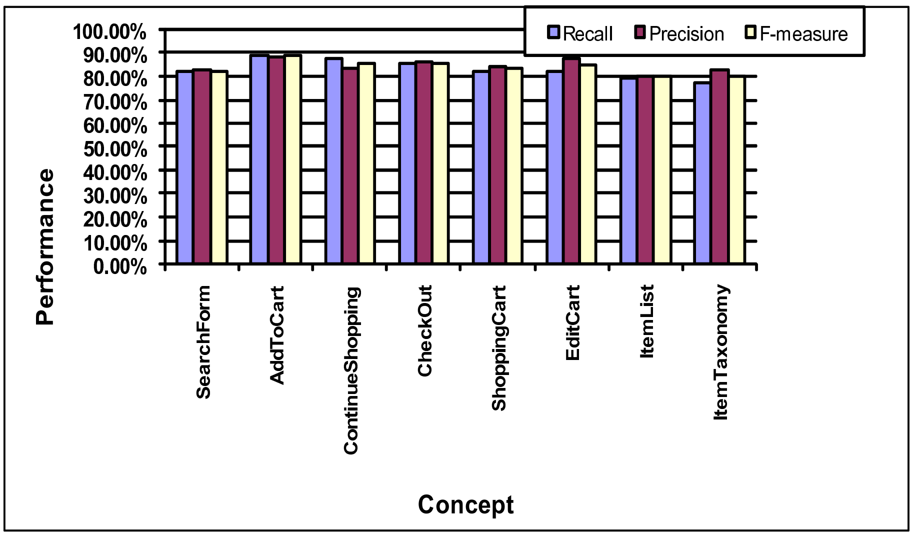


Figure 48: Performance of Clustering Unlabeled Sequences Collected from "BN"

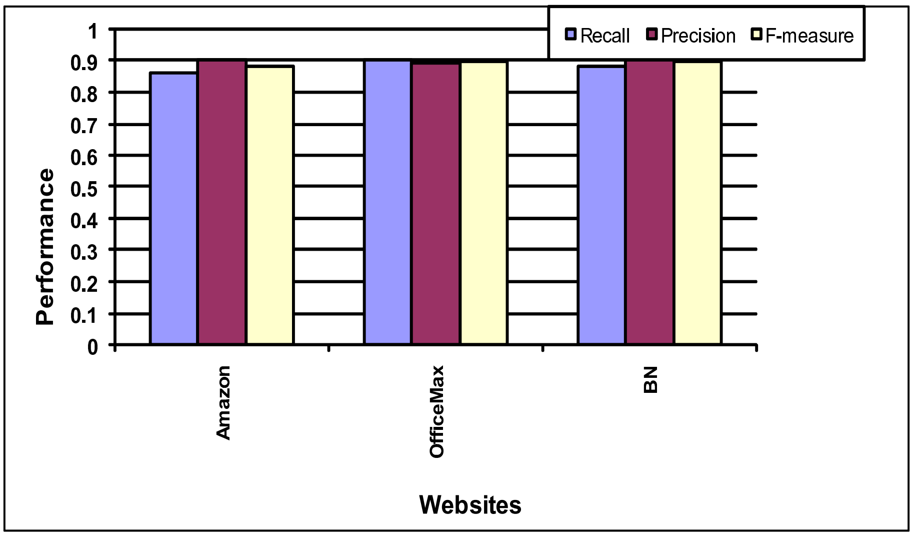


Figure 49: Performance of Learning Process Model from Unlabeled Sequences

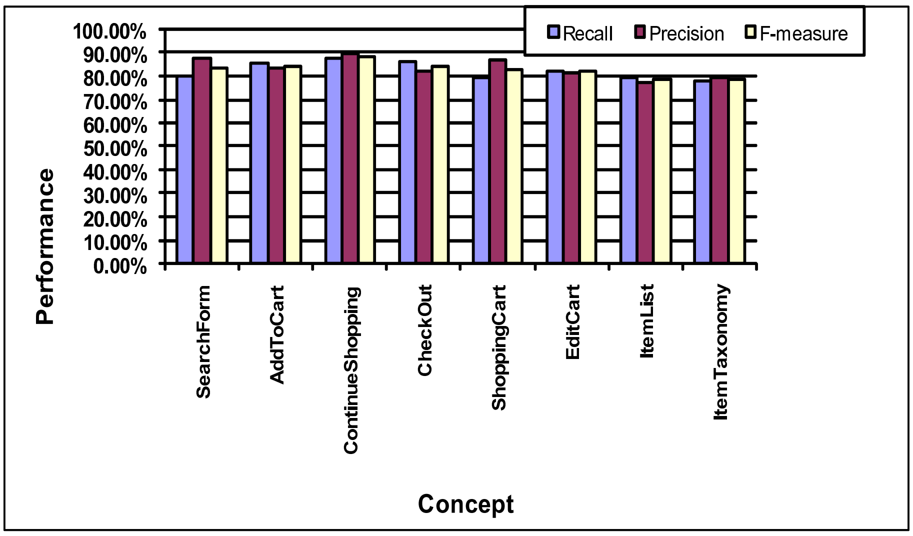


Figure 50: Performance of Concept Identification for "Amazon"

Figures 46, 47, 48, 49, 50, 51, 52, illustrate the experimental results.

Observe that, we got higher performance of clustering (an improvement of approximately 10%), process model learning (an improvement of approximately 12.5%) and concept identification (an improvement of approximately 9%) when the

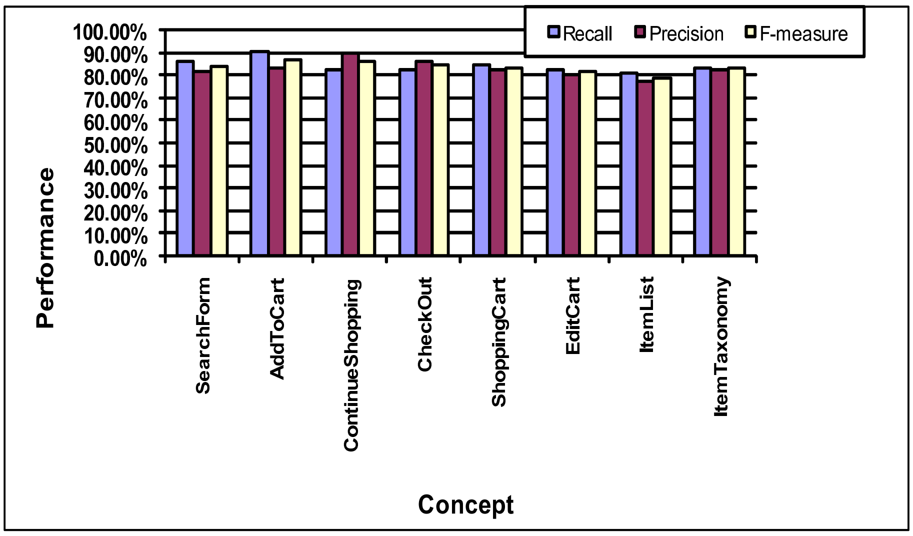


Figure 51: Performance of Concept Identification for “OfficeMax”

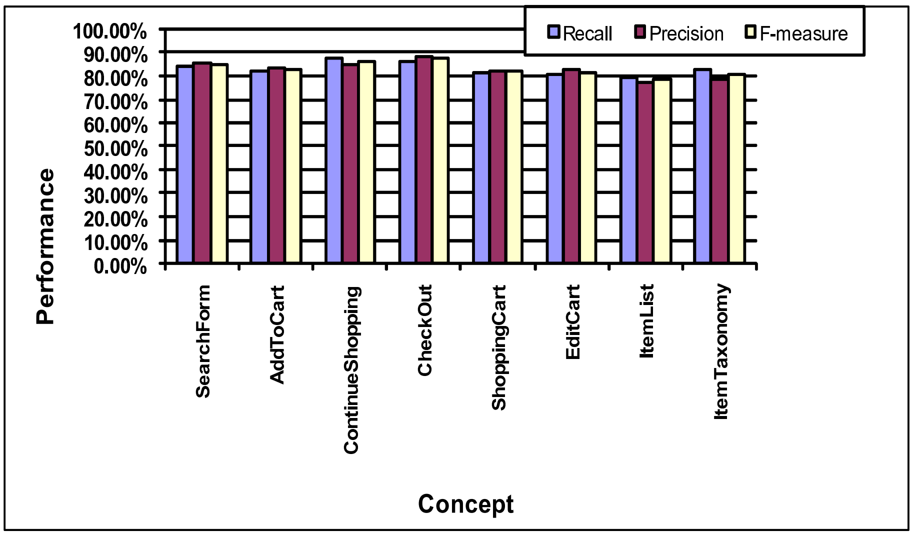


Figure 52: Performance of Concept Identification for “BN”

models are learned from a single Website. This is because, concept instances show less variability in texts and presentation patterns in a single Web site. Therefore, the performance of clustering concept segments from a single site is higher than the performance of clustering concept segments from different sites. A higher performance of clustering also results a higher performance of process model learning and concept identification.

Chapter 6

Related Work

The work described in this thesis has broad connections to research in Web services, semantic understanding of Web content, process model learning, Web accessibility research, contextual processing, end user programming and automatic information extraction.

6.1 Web Services

Web Services research is an emerging paradigm that focuses on technologies that let service providers to export their functionalities on the Web so as to facilitate automated e-commerce. It has given rise to standardization efforts resulting in languages such as WSDL for describing services, SOAP for accessing services, UDDI for service discovery and integration, BPEL4WS for business process specification, and OWL-S as an ontology for semantic description of service metadata.

Service providers are beginning to utilize these languages for exposing their services (see for example `http://www.amazon.com/webservices`).

The complementary task of annotating service descriptions with semantic metadata has been addressed in [45, 71, 78].

In contrast to these works we address a different kind of annotation problem, namely automatic annotation of different kinds of concepts that can occur in a Web page.

Web services expose very basic functionalities which by themselves are not sufficient to conduct complex transactions. For instance, Amazon's Web service exposes basic tasks such as searching for a product, adding a product into the shopping cart, *etc.* One has to compose these primitive services in order to perform complex transactions.

This problem has been receiving attention lately [3, 24, 76, 85, 90]. All these works typically use process definitions and an ontology to create the composite service with varying degrees of automation. Note that our technique is based on composing operations over Web pages instead of services.

A vast majority of transactions on the Web are still conducted over HTML pages. This focus on Web pages is what sets our work apart from those in Web services. Also note that our approach to Web transactions is quite flexible, in the sense that the users can define their own "personalized" transactional service instead of being confined to whatever service is exposed by the provider.

6.2 Semantic Analysis of Web content

The essence of the technique underlying our concept identification module is to partition a page into segments containing "semantically" related items and classify them against concepts in the ontology.

Substantial research has been done on segmenting Web documents [36,83,91]. These techniques are either domain [36] or site [83] specific or depend on fixed sets of HTML markup [91]. Semantic partitioning of Web pages has been described in [63–65]. These systems require semantic information (e.g. ontologies) to partition a Web page.

In contrast to these research works, our geometric segmentation algorithm does not require any ontology or domain information.

Web page partitioning techniques have been proposed for adapting content on small screen devices [21, 22, 25, 93], content caching [75], data cleaning [79, 92], and search [94]. The VIPS [94] algorithm uses visual cues to partition a Web page into geometric segments. The algorithm extracts nodes from the DOM tree, finds vertical and horizontal separator lines between the nodes, and segments the Web page into regions based on a number of handcrafted rules. This algorithm is used in [79], where the segments are described by a set of features (e.g. spatial features, number of images, sizes, links, etc.). The feature values are then fed into an SVM, which labels the segments according to their importance. The idea of using content similarities for semantic analysis was also recently explored in [96] in the context of Web forms. The fundamental difference between our technique and all the above works is the tight coupling of the logical structure of a page with concept learning. This is because we learn concept models from segments representing an instance of a concept.

Concept identification in Web pages is related to the body of research on semantic understanding of Web content. Powerful ontology management systems and knowledge bases have been used for interactive annotation of Web pages [44, 50]. More automated approaches combine them with linguistic analysis [72], segmentation heuristics [31, 35], and machine learning techniques [27, 40].

Our semantic analysis technique does not depend on rich domain information. Instead, our approach relies on light-weight features in a machine learning setting for concept identification. This allows users to define *personalized* semantic concepts thereby lending more flexibility to modeling Web transactions. Moreover, in contrast to all of these works, our geometric segmentation (partitioning) method does not depend on any domain knowledge or semantic information.

It should also be noted that the extensive work on wrapper learning [51] is related to concept identification. However, wrappers are syntax-based solutions and are neither scalable nor robust when compared to content-based techniques.

6.3 Process Model Learning

Our work on learning process models from user activity logs is related to research in mining workflow process models (see [88] for a survey). However, our current definition of a process is simpler than traditional notions of workflows. For instance, we do not use sophisticated synchronization primitives. Hence we are able to model our processes as DFAs instead of workflows and learn them from example sequences. Learning DFAs is a thoroughly researched topic (see [66] for a comprehensive survey). A classical result is that learning the smallest size DFA that is consistent with respect to a set of positive and negative training examples is NP-hard [9, 10, 38]. This spurred a number of papers describing efficient heuristics for DFA learning(*e.g.*, [66, 68]).

We used the simple yet effective heuristic with low time complexity described in [68] for process model learning using a supervised approach. There are some drawbacks with learning DFAs.

Web transaction traces are collected by maintaining logs of transaction activities done by users. It is quite diffcult to collect traces that satisfies the properties (e.g. characteristics samples) which are necessary for DFA learning in polynomial time. In addition, DFA learning requires a sizable number of negative examples which are often difficult to obtain, specially from logs of user activities.

In this research, we have developed a new process model learning algorithm for unsupervised model mining. In contrast to traditional DFA learning, our algorithm does not require negative examples. We defined the class of Web transaction languages and using such a definition, we proposed a new learning algorithm for process models.

Navigating to relevant pages in a site using the notion of "information scent" has been explored in [26]. This notion is modeled using keywords extracted from pages specific to that site. This is site specific, so keyword based models have to be built for each site. This is labor intensive and is not scalable. In contrast, our process model is domain specific and using it a user can do online transactions on sites that share similar content semantics.

6.4 Web Accessibility Research

Several research projects aiming to facilitate Web accessibility include work on browser-level support [1, 11, 83], content adaptation and summarization [42, 77, 95], organization and annotation of Web pages for effective audio rendition [47,48], ontology-directed exploratory browsing as in our HearSay audiobrowser [20, 74], WebInSight [12], etc.

Some of the most popular screen-readers are JAWS [1] and IBM's Home Page Reader [11, 83]. An example of a VoiceXML browsing system (which presents information sequentially) is described in [67]. All of these applications do not perform content analysis of Web pages. BrookesTalk [95] facilitates non-visual Web access by providing summaries of Web pages to give its users an audio overview of Web page content. The work described in [42] generates a “gist” summary of a Web page to alleviate information overload for blind users.

In our work we need to first filter the content based on the transactional context represented by the states of the process model. Specifically we need a more global view of the content in a set of pages in order to determine what should be filtered in a state.

In [61], we described a simple and scalable approach for conducting Web transactions using a shallow knowledge base that stores features occuring in concepts. Since this approach does not use a process model, it is not possible to represent transactional context (i.e. state) and present concepts depending on the transactional context. In contrast, we can determine what should be filtered in a state using a process model. Process model can also be used to rank different concepts occuring in a page. Currently the model only stores concepts in a state. It can however stores the ranks of the concepts in a state. Such ranks can be computed from the transaction traces. In addition, use of a model-directed approach also helps personalization.

The works describing organization and annotation of Web pages for better audio rendition typically rely on rules or logical structures [47]. In [48], authors propose the idea of extracting content using semantics [48]. They describe a framework for manual annotation of the content w.r.t. a schema, representing the task a user wishes to accomplish. These annotation rules are also site specific, and, hence, not scalable over content domains.

The essential difference between our work and all of the above-mentioned research is that we do not require any domain knowledge in terms of rules.

6.5 Contextual Analysis

The notion of context has been used in different areas of Computer Science research. For example, [49] defines context of a Web page as a collection of text, gathered around the links in other pages that are pointing to that Web page. The context is then used to obtain a summary of the page. Summarization using context is also explored by the InCommonSense system [8], where search engine results are summarized to generate text snippets. Context analysis for non-visual Web access is described in [41, 43], where the context information of a link is used to get the preview of the next Web page, so that visually disabled individuals can choose whether or not they should follow the link. This idea is used in the AcceSS system [2], to get the preview of the entire page. All of these works define the context of the link as an ad-hoc collection of words surrounding it. In contrast, our notion of context is based on the topic similarity of words around the link. We use a principled approach for context analysis using a simple topic boundary detection method [5], confined to geometric segments that have semantically related content.

Contextual browsing is a joint effort by me, Prof. I.V. Ramakrishnan and my colleague, Yevgen Borodin. In his thesis, Yevgen has proposed a unifying interface for aural web access that can substantially improve the user experience and make non-visual web browsing more usable [17]. Yevgen's and my thesis share context collection algorithm. In my thesis, I have used context of Web objects to identify concept segments. On the other hand, Yevgen describes the context-based browsing technique as one of the algorithms to improve Web accessibility. Apart from that, we do not share any overlapping content in our thesis. In contrast to his thesis proposal which aims to develop the architecture, interface, and algorithms for Bridging the Web Accessibility Divide, I have focused on formalizing transaction models for Web Accessibility and developing algorithms for mining them from transaction click streams.

6.6 End User Programming

Several research works on end user programming that relate to our work include programming by demonstration [30, 52, 53], agent learning [7], query from demonstrations [86], learning from instructions [14, 15], building meshups [1] from examples [87], etc.

Programming by demonstration (PBD) [30, 52, 53] allows users to construct a program by simply performing actions in the user interface with which they are already familiar. CoScripter [56] uses this approach to build a collaborative scripting environment for recording, automating, and sharing web-based processes. Other browser recording and playback tools, e.g. iMacros

[1]a web application that integrates data from multiple web sources to provide a unique service

(http://www.iopus.com/imacros/), also use this approach. Using such tools, Web based tasks can be scripted and automated. However, such approaces are supervised and are not scalable across Websites.

PLOW [7] is a collaborative task learning system that learns task models from demonstration, explanation and dialogue. The system couples machine learning, natural language processing and AI techniques to learn such task models. Learning from instructions is described in [14, 15]. For example, Tailor [14] allows users to modify task information through instruction.

Knoblock at. el [86] describes their system Karma, which allows users (without programming experience) to easily build services that integrate information from multiple data sources. In a recent work [87], they described how users with no programming background can easily create Mashups using Karma.

All of these works enable the user to complete a task without programming knowledge. However, they are supervised and hence a lot of user interaction is required throughout the learning process. Some of the works (e.g. [7]) also use domain knowledge to learn a task model.

In contrast, our algorithm of learning transaction models from click streams is completely unsupervised and does not need user interaction in the learning process. Moreover, we do not use any domain knowledge to learn transaction models.

However, the fundamental difference between our work and all the above mentioned work is that we learn transaction models from sequences of operations over multiple Web pages. That allows us to capture transaction context in states of the learned process models and present concepts in a state to the user. Moreover, it aims to help visually disabled users build their transaction models over sites they frequently visit and quickly conduct transactions.

6.7 Automatic Information Extraction

Our algorithm to automatically assigning labels to unlabeled transaction sequences is related to a number of research works [16, 28, 54, 55]. Knoblock at. el describes automatic labeling of data used by a Web service where a classifier is built to label unseen data [55]. Borker at. el [16] uses heuristic features and domain-specific vocabularies, to learn a probabilistic model from a set of training examples for automatic text segmentation. These approaches are supervised and use domain knowledge to build the classifier. In contrast, our label assignment algorithm is completely unsupervised. Automatic information extraction from HTML tables is described in [54]. This work is unsupervised, domain independent and uses Web page structures, templates, etc. to extract information from tables. We also use the geometric structure of the concept instances as well as their content similarities to cluster them.

Dong et al [32] describes Woogle, a Web service search engine that finds similarity of Web services using an unsupervised clustering. They match operations, input outputs, text descriptions to cluster the Web services. In contrast, we cluster segments containing concept segments in a Web page.

The RoadRunner system [28,29] automatically extracts data from Web sites by exploiting similarities in page layouts. It learns the underlying template of Web sites from examples pages using unsupervised approach, and uses it to automatically extract data from Web sites. In contrast, we learn process models from transaction sequences and concept models from Web page segments containing instances of the concepts.

Chapter 7

Discussion

Advances in web technology have considerably widened the web accessibility divide between sighted and blind users. This divide is especially acute when conducting online transactions. Model-directed Web transaction, that uses transaction models (i.e. process model and concept models) to deliver relevant page fragment at each transactional step can improve Web accessibility and substantially reduce the digital divide between sighted and blind users. Our preliminary experimentation [81] seems to suggest that it is possible to achieve such reductions in practice.

In this thesis, we describe such transaction models and techniques to mine them using supervised and unsupervised approaches. First, we discussed some of our Web content analysis techniques. We also defined the formal notions of a Web transaction model and our initial work to mine transaction models using a supervised approach [81, 82]. Next, we described automatic mining of transaction models from unlabeled and partially labeled transaction sequences using unsupervised clustering, classification, and our prior research on web page partitioning and context analysis [60].

There are several avenues for future research. We performed experiments on Websites from online shopping domains. But it is possible to apply our algorithm to mine transaction models for other domains (e.g. flight ticket booking). In our current framework, we need to build process models for each type of Web transactions. It is interesting to investigate the possibility of developing a generic process model to support any type of Web transaction. The transaction system emerged from this research will be used to push the state-of-the-art in assistive web browsing technology. In particular, the transaction system will be used for non-visual Web transactions. Towards that the transaction system will be deployed at the Helen Keller Services for the Blind (http://www.helenkeller.org) to get feedback from the visually handicapped community. Using this transaction system, blind users will be able to personalize the system for Web sites that they need to use on a regular basis and conduct transactions on these sites with the same ease as their sighted counterparts. Finally, integration of our framework with Web services standards is an interesting problem. Success here will let us specify the process model in BPEL4WS which in turn will enable interoperation with sites exposing Web pages as well as those exposing Web services.

Bibliography

[1] JAWS Screen Reader. http://www.freedomscientific.com.

[2] NetFront browser. http://www.access-us-inc.com/Products/client-side/Prod_NetFront.html.

[3] S. Agarwal, S. Handschuh, and S. Staab. Surfing the service web. In *Proc. of Intl. Semantic Web Conf. (ISWC)*, pages 211–216, 2003.

[4] V. A. Alfred, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley.

[5] J. Allan, editor. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers, 2002.

[6] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proc. of ACM Conf. on Informaion Retrieval (SIGIR)*, 1998.

[7] J. F. Allen, N. Chambers, G. Ferguson, L. Galescu, H. Jung, M. D. Swift, and W. Taysom. Plow: A collaborative task learning agent. In *AAAI*, pages 1514–1519. AAAI Press, 2007.

[8] E. Amitay and C. Paris. Automatically summarising web sites - is there a way around it? In *Proc. of the ACM 9th Intl. Conf. on Information and Knowledge Management CIKM*, pages 173–179, 2000.

[9] D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39(3):337–350, 1978.

[10] D. Angulin. Inductive inference of formal languages from positive data. *Information and control,45(2),* pages 117–135, 1980.

[11] C. Asakawa and T. Itoh. User interface of a home page reader. In *Proc. of ACM Intl. Conf. on Assistive Technologies (ASSETS),* pages 149–156. ACM Press, 1998.

[12] J. P. Bigham, R. S. Kaminsky, R. E. Ladner, O. M. Danielsson, and G. L. Hempton. Webinsight:: making web images accessible. In *Proc. of the 8th Intl. ACM SIGACCESS Conf. on Computers and Accessibility,* pages 181–188, 2006.

[13] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of Annual Conf. on Learning Theory (COLT),* 1998.

[14] J. Blythe. Task learning by instruction in tailor. In *Proc. of the 10th Intl. Conf. on Intelligent User Interfaces,* pages 191–198, 2005.

[15] J. Blythe and T. Russ. Case-based reasoning for procedure learning by instruction. In J. M. Bradshaw, H. Lieberman, and S. Staab, editors, *Proc. of Intl. Conf. on Intelligent User Interfaces,* pages 301–304, 2008.

[16] V. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *Proc. of ACM Conf. on Management of Data (SIGMOD),* 2001.

[17] Y. Borodin. Architecture, interface, and algorithms for bridging the web accessibility divide. *A PhD Thesis Proposal Presented to the Department of Computer Science, Stony Brook University*, 2008.

[18] Y. Borodin, J. Mahmud, A. Ahmed, and I. V. Ramakrishnan. Webvat: Web page visualization and analysis tool. *Lecture Notes in Computer Science*.

[19] Y. Borodin, J. Mahmud, and I. V. Ramakrishnan. Context browsing with mobiles - when less is more. In *Proc. of ACM/USENIX MobiSys*, 2007.

[20] Y. Borodin, J. Mahmud, I. V. Ramakrishnan, and A. Stent. The hearsay non-visual web browser. In *Proc. of the 2007 Intl. Cross-disciplinary Conf. on Web Accessibility (W4A)*, pages 128–129, 2007.

[21] O. Buyukkoten, H. Garcia-Molina, and A. Paepcke. Seeing the whole in parts: Text summarization for web browsing on handheld devices. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 652–662. ACM Press, 2001.

[22] O. Buyukkoten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power browser: Efficient web browsing for pdas. In *Proc. of ACM Conf. on Human Factors in Computing Systems (CHI)*, pages 430–437. ACM Press, 2000.

[23] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001. http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[24] L. Chen, N. Shadbolt, C. A. Goble, F. Tao, S. J. Cox, C. Puleston, and P. R. Smart. Towards a knowledge-based approach to semantic service composition. In *Proc. of Intl. Semantic Web Conf. (ISWC)*, pages 319–334, 2003.

[25] Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 225–233. ACM Press, 2003.

[26] E. Chi, P. Pirolli, K. Chen, and J. Pitkow. Using information scent to model user information needs and actions on the web. In *Proc. of ACM Conf. on Human Factors in Computing Systems (CHI)*, pages 490–497. ACM Press, 2001.

[27] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 462–471. ACM Press, 2004.

[28] V. Crescenzi and G. Mecca. Automatic information extraction from large websites. *Journal of ACM*, 51(5):731–779, 2004.

[29] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proc. of Intl. Conf. on Very Large Data Bases (VLDB)*, 2001.

[30] A. Cypher. Watch what i do: Programming by demonstration. *MIT Press*, 1993.

[31] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. Tomlin, and J. Yien. SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 831–840, 2003.

[32] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for web services. In *Proc. of the 30th Intl. Conf. on Very large data bases*, pages 372–383, 2004.

[33] P. Dupont. Incremental regular inference. In *Proc. of the 3rd ICGI-96, Lecture Notes in Artificial Intelligence 1147*, pages 238–250, 1996.

[34] P. Dupont. Incremental regular inference. In *Proc. of Intl. Colloquium on Grammatical Inference (ICGI)*, 1996.

[35] M. Dzbor, J. Domingue, and E. Motta. Magpie - towards a semantic web browser. In *Proc. of Intl. Semantic Web Conf. (ISWC)*, 2003.

[36] D. Embley and L. Xu. Record location and reconfiguration in unstructured multiple-record web documents. In *Proc. of ACM SIGMOD Workshop on the Web and Databases (WebDB)*, 2000.

[37] D. Geoffray. The internet through the eyes of windows-eyes. In *Proc. of Tech. and Persons with Disabilities Conf.*, 1999.

[38] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[39] H. Guo, J. Mahmud, Y. Borodin, A. Stent, and I. Ramakrishnan. A general approach for partitioning web page content based on geometric and style information. In *Proc. of Intl. Conf. of Document Analysis and Recognition (ICDAR)*, 2007.

[40] B. Hammond, A. Sheth, and K. Kochut. Semantic enhancement engine: A modular document enhancement platform for semantic applications over heterogenous content. In V. Kashyap and L. Shklar, editors, *Real World Semantic Applications*. IOS Press, 2002.

[41] S. Harper, C. Goble, R. Stevens, and Y. Yesilada. Middleware to expand context and preview in hypertext. In *Proc. of the 6th Intl. ACM SIGACCESS Conf. on Computers and accessibility (ASSETS)*, 2004.

[42] S. Harper and N. Patel. Gist summaries for visually impaired surfers. In *Proc. of ACM Intl. Conf. on Assistive Technologies (ASSETS)*, pages 90–97. ACM Press, 2005.

[43] S. Harper, Y. Yesilada, C. Goble, and R. Stevens. How much is too much in a hypertext link?: investigating context and preview – a formative evaluation. In *Proc. of the 15th ACM Conf. on Hypertext and hypermedia*, pages 116–125, 2004.

[44] J. Heflin, J. A. Hendler, and S. Luke. SHOE: A blueprint for the semantic web. In D. Fensel, J. A. Hendler, H. Lieberman, and W. Wahlster, editors, *Spinning the Semantic Web*, pages 29–63. MIT Press, 2003.

[45] A. Hess, E. Johnston, and N. Kushmerick. Assam: A tool for semi-automatically annotating semantic web services. In *Proc. of Intl. Semantic Web Conf. (ISWC)*, pages 320–334, 2004.

[46] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[47] M. Hori, G. Kondoh, K. Ono, S. ichi Hirose, and S. Singhal. Annotation-based web content transcoding. In *Proc. of Intl. World Wide Web Conf. (WWW)*, 2000.

[48] A. Huang and N. Sundaresan. A semantic transcoding system to adapt web services for users with disabilities. In *Proc. of ACM Intl. Conf. on Assistive Technologies (ASSETS)*, pages 156–163. ACM Press, 2000.

[49] B. B.-M. J.-Y. Delort and M. R. Enhanced. Enhanced web document summarization using hyperlinks. In *Proc. of the 14th ACM Conf. on Hypertext and hypermedia*, pages 208–215, 2003.

[50] J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick. Annotea: An open RDF infrastructure for shared web annotations. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 623–632. ACM Press, 2001.

[51] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2), 2002.

[52] T. Lau. *Programming by Demonstration: a Machine Learning Approach*. PhD thesis, University of Washington, 2001.

[53] T. Lau, L. Bergman, V. Castelli, and D. Oblinger. Sheepdog: learning procedures for technical support. In *Proc. of the 9th Intl. Conf. on Intelligent user interfaces*, pages 109–116, 2004.

[54] K. Lerman, L. Getoor, S. Minton, and C. Knoblock. Using the structure of web sites for automatic segmentation of tables. In *Proc. of the 2004 ACM SIGMOD Intl. Conf. on Management of data*, pages 119–130, 2004.

[55] K. Lerman, A. Plangprasopchok, and C. A. Knoblock. Automatically labeling the inputs and outputs of web services. In *AAAI*. AAAI Press, 2006.

[56] G. Leshed, E. M. Haber, T. Matthews, and T. Lau. Coscripter: automating & sharing how-to knowledge in the enterprise. In *Proc. of the 26th annual SIGCHI conference on Human factors in computing systems*, pages 1719–1728, 2008.

[57] J. Mahmud. Information overload in non-visual web access: Context analysis spells relief. In *Proc. of ACM Intl. Conf. on Assistive Technology (ASSETS)*, 2007.

[58] J. Mahmud, Y. Borodin, D. Das, and I. Ramakrishnan. Improving non-visual web access using context. In *Proc. of ACM Intl. Conf. on Assistive Technology (ASSETS)*, 2006.

[59] J. Mahmud, Y. Borodin, D. Das, and I. Ramakrishnan. Combating information overload in non-visual web access using context. In *Proc. of Intelligent User Interface (IUI)*, 2007.

[60] J. Mahmud, Y. Borodin, and I. V. Ramakrishnan. Csurf: A context-directed non-visual web-browser. In *Proc. of the 16th Intl. Conf. on the World Wide Web (WWW)*, 2007.

[61] J. Mahmud, Y. Borodin, and I. V. Ramakrishnan. Assistive browser for conducting web transactions. In *Proc. of the Intl. Conf. on Intelligent User Interface (IUI)*, 2008.

[62] `http://www.mozilla.com/firefox/`.

[63] S. Mukherjee, I. Ramakrishnan, and A. Singh. Bootstrapping semantic annotation for content-rich HTML documents. In *Proc. of Intl. Conf. on Data Engineering (ICDE)*. ACM Press, 2005.

[64] S. Mukherjee, G. Yang, and I. Ramakrishnan. Automatic annotation of content-rich HTML documents: Structural and semantic analysis. In *Proc. of Intl. Semantic Web Conf. (ISWC)*, 2003.

[65] S. Mukherjee, G. Yang, W. Tan, and I. Ramakrishnan. Automatic discovery of semantic structures in HTML documents. In *Proc. of Intl. Conf. on Document Analysis and Recognition*, 2003.

[66] K. Murphy. *Passively learning finite automata*, 1996.

[67] `http://www.internetspeech.com`.

[68] J. Oncina and P. Garc. Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis*, 1991.

[69] M. S. Pang-Ning Tan and V. Kumar. *Introduction to Data Mining, Addison-Wesley*.

[70] R. G. Parekh and V. G. Honavar. Learning DFA from simple examples. In *Proc. of the Intl. Workshop on Algorithmic Learning Theory*, 1997.

[71] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. Meteor-s web service annotation framework. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 553–562, 2004.

[72] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. Kim - semantic annotation platform. In *Proc. of Intl. Semantic Web Conf. (ISWC)*, 2003.

[73] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.

[74] I. Ramakrishnan, A. Stent, and G. Yang. Hearsay: Enabling audio browsing on hypertext content. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 80–89. ACM Press, 2004.

[75] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis. Automatic detection of fragments in dynamically generated web pages. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 443–454. ACM Press, 2004.

[76] J. Rao, P. Küngas, and M. Matskin. Logic-based web services composition: From service description to process model. In *Proc. of Intl. Conf. on Web Services (ICWS)*, 2004.

[77] J. T. Richards and V. L. Hanson. Web accessibility: a broader view. In *Proc. of the 13th Intl. Conf. on World Wide Web (WWW)*, pages 72–79, 2004.

[78] M. Sabou, C. Wroe, C. Goble, and G. Mishne. Learning domain ontologies for web service descriptions: An experiment in bioinformatics. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 190–198, 2005.

[79] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for web pages. In *Proc. of the 13th Intl. Conf. on World Wide Web (WWW)*, pages 203–211, 2004.

[80] A. Strehl. *Relationship-based Clustering and Cluster Ensembles for High-dimensional Data Mining*. PhD thesis, The University of Texas at Austin, May 2002.

[81] Z. Sun, J. Mahmud, S. Mukherjee, and I. V. Ramakrishnan. Model-directed web transactions under constrained modalities. In *Proc. of the 15th Intl. Conf. on World Wide Web (WWW)*, pages 447–456, 2006.

[82] Z. Sun, J. Mahmud, I. V. Ramakrishnan, and S. Mukherjee. Model-directed web transactions under constrained modalities. *ACM Transactions on the Web (TWEB)*, 1(3), 2007.

[83] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda. Site-wide annotation: Reconstructing existing pages to be accessible. In *Proc. of ACM Intl. Conf. on Assistive Technologies (ASSETS)*. ACM Press, 2002.

[84] K. Thomson. Regular expression search algorithm. *Communications of ACM*, pages 419–422, 1968.

[85] P. Traverso and M. Pistore. Automated composition of semantic web services into executable processes. In *Proc. of Intl. Semantic Web Conf. (ISWC)*, 2004.

[86] R. Tuchinda, P. Szekely, and C. A. Knoblock. Building data integration queries by demonstration. In *Proc. of the 12th Intl. Conf. on Intelligent user interfaces (IUI)*, pages 170–179, New York, NY, USA, 2007. ACM.

[87] R. Tuchinda, P. A. Szekely, and C. A. Knoblock. Building mashups by example. In J. M. Bradshaw, H. Lieberman, and S. Staab, editors, *Proc. of Intl. Conf. on Intelligent User Interfaces*, pages 139–148. ACM, 2008.

[88] W. van der Aalst and A. Weijters. Process mining: A research agenda. *Computers and Industry*, 53:231–244, 2004.

[89] V. Vapnik. Principles of risk minimization for learning theory. In *D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, Advances in Neural Information Processing Systems 3*, pages 831–838. Morgan Kaufmann, 1992.

[90] A. Wombacher, P. Fankhauser, and E. J. Neuhold. Transforming BPEL into annotated deterministic finite state automata for service discovery. In *Proc. of Intl. Conf. on Web Services (ICWS)*, 2004.

[91] Y. Yang and H. Zhang. HTML page analysis based on visual cues. In *Proc. of Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 2001.

[92] L. Yi and B. Liu. Eliminating noisy information in web pages for data mining. In *Proc. of ACM Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pages 117–118. ACM Press, 2003.

[93] X. Yin and W. S. Lee. Using link analysis to improve layout on mobile devices. In *Proc. of Intl. World Wide Web Conf. (WWW)*, pages 338–344. ACM Press, 2004.

[94] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proc. of Intl. World Wide Web Conf. (WWW)*, 2003.

[95] M. Zajicek, C. Powell, and C. Reeves. Web search and orientation with Brookestalk. In *Proc. of Tech. and Persons with Disabilities Conf.*, 1999.

[96] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *Proc. of ACM Conf. on Management of Data (SIGMOD)*, pages 107–118. ACM Press, 2004.