Improving Accessibility of Transaction-centric Web Objects

Muhammad Asiful Islam, Faisal Ahmed, Yevgen Borodin, Jalal Mahmud[*] I.V. Ramakrishnan

Department of Computer Science, Stony Brook University
Stony Brook, NY, USA
{maislam, faiahmed, borodin, jmahmud, ram}@cs.sunysb.edu

Abstract

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, e.g., shopping, paying bills, making travel plans, etc. Such transactions span multiple web pages and require that users find clickable objects (e.g., "add-to-cart" button) which are essential for transaction progress. While this is fast and straightforward for sighted users, locating the clickable objects causes considerable strain for blind individuals using screen-reading technology. Screen readers force users to listen to irrelevant information sequentially and provide no interface for identifying relevant clickable objects.

This paper addresses the problem of making clickable objects readily accessible, which can substantially reduce the information overload that is otherwise experienced by blind users. A static knowledge base of keywords constructed from the captions of clickable objects does not provide enough learning capability for identifying clickable objects which do not have any captions (e.g., image buttons without alternative text). In this paper, we present an Information Retrieval based technique that uses the *context* of *transaction-centric* objects (e.g., "add-to-cart" and "checkout" buttons) to identify and classify them even when their captions are missing. In addition, the technique utilizes a reinforcement mechanism based on user feedback to accommodate previously unseen captions of objects as well as new categories of objects. We provide user study and experimental evidence of the effectiveness of our algorithm.

Keywords

Context, Web Transaction, Online, Web Object, Vector Space Model.

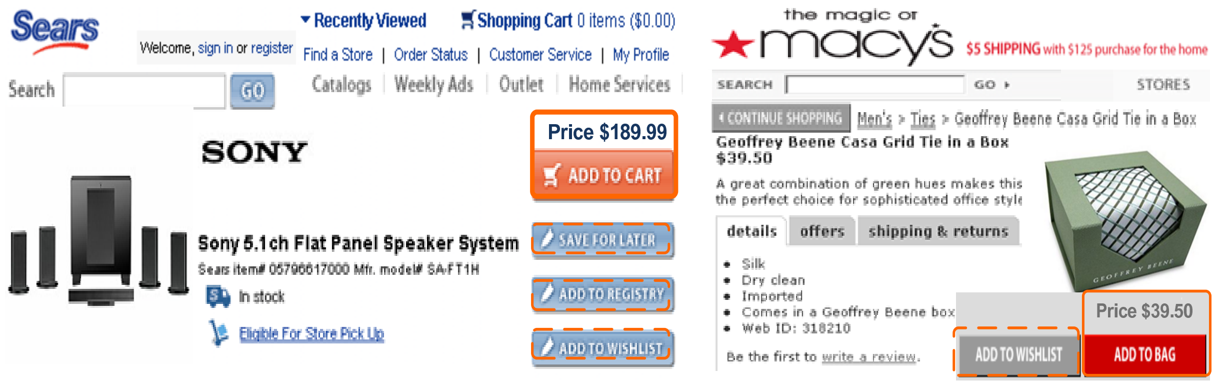
[*]IBM Research - Almaden, San Jose, CA. jumahmud@us.ibm.com

1 Introduction.

Over a relatively short period of time the Web has evolved into an ecosystem where anyone can co-exist, communicate, find information, shop, bank, and pay bills online. On the downside, the advances that have made this possible have also widened the Web accessibility divide between sighted and blind users. People suffering from vision loss have to use screen-readers, such as JAWS [1] or Windows Eyes [2], to access the Web. Screen-readers typically narrate web pages sequentially, while allowing users to move backward and forward within the pages. Unfortunately, screen-readers do not offer content-filtering, making web browsing very time consuming and causing significant information overload.

Web browsing has never been easy for blind people; in addition, it poses more challenges while performing *web transactions*, such as shopping, paying utility bills and booking flight tickets, etc. While people with visual impairments can usually perform online transactions, provided all important buttons are properly labeled, it takes them a considerable amount of time to find those buttons and go through a transaction. The only feasible navigation approach, incorporated in extant screen readers, is tabbing through the links and buttons on every web page. Another possibility is to try remembering in what part of the page a certain action-button occurs. Needless to say, any changes to the layout of web sites can cause significant frustration and more wasted time. Manually searching a page for certain keywords can be sometimes helpful, but a variety of terms used to describe similar entities by different content providers can be overwhelming. For example, a "search" button can be labeled as: "search" in one site, "find" in another, and "go" in the third site.

To avoid having to search manually, one can construct a shallow knowledge base (KB) made up of keywords appearing in the captions of these clickable elements and do simple keyword matching to locate them.


Sears
Recently Viewed
Shopping Cart 0 items ($0.00)
Welcome, sign in or register
Find a Store | Order Status | Customer Service | My Profile
Search
GO
Catalogs | Weekly Ads | Outlet | Home Services
SONY
Price $189.99
ADD TO CART
Sony 5.1ch Flat Panel Speaker System
Sears item# 05796617000 Mfr. model# SA-FT1H
In stock
Eligible For Store Pick Up
SAVE FOR LATER
ADD TO REGISTRY
ADD TO WISHLIST
(a)
the magic of macy's
$5 SHIPPING with $125 purchase for the home
SEARCH
GO
STORES
CONTINUE SHOPPING
Men's > Ties > Geoffrey Beene Casa Grid Tie in a Box
Geoffrey Beene Casa Grid Tie in a Box
$39.50
details
offers
shipping & returns
Silk
Dry clean
Imported
Comes in a Geoffrey Beene box
Web ID: 318210
Be the first to write a review.
Price $39.50
ADD TO WISHLIST
ADD TO BAG
(b)

A great combination of green hues makes this the perfect choice for sophisticated office styl

Figure 1: Web Pages with Clickable Transaction-centric Web Objects

In our earlier work we had used such a simplistic approach in the context of exploring user interfaces for doing non-visual web transactions [6]. There are quite a few drawbacks with this approach. Firstly, capturing many of these captions and their variations a priori in the KB is not only difficult, but is also limiting in terms of scalability, i.e., doing so does not provide for the possibility of expanding the KB as new variations are encountered. Secondly, the matching process itself can suffer from poor recall/precision performance. For example, a match for the keyword "add" (ignoring the case) locates the "add-to-cart" button, as well as several other irrelevant elements (indicated by dotted rectangles in Figure 1(a)). On the other hand the exact match for the keywords "add to cart" eliminates all these irrelevant elements in Figure 1(a) but looking for the same exact match in Figure 1 (b) will not find the "correct" button with the caption "ADD TO BAG". Lastly, either approach will fail to identify action buttons without captions (e.g., image buttons with no alternative text such as the "add-to-cart" button in Figure 2(a)).

Addressing all of the above limitations of keyword matches will go a long way towards substantially reducing the information overload experienced by blind users when doing web transactions. More generally, it will push the state of the art in web accessibility beyond extant screen reading technology.

1.1 Overview of Proposed Approach. In this paper we describe a new algorithmic technique to address the shortcomings of keyword-based searches for locating *transaction-centric clickable web objects*, e.g., "search", "add to cart", "view bill", "pay bill" - action buttons and links that are essential for transaction progress.

The crux of our technique is based on the notion of the "*context*" of an object. Informally, by context we mean the bag of words and word combinations collected from the caption of an object, as well as from other content elements in its immediate "vicinity" (indicated by solid rectangles in Figure 2). For example, in Figure 2(a), the bag of words "Our", "Price", "Our Price", "Shipping", "Free","Shipping Free", "Buy.com Total", "Total Price", "Price", "Total", "Qty" is the context for the "Add to Cart" button. Given two contexts, encoded as term vectors in a Vector Space Model (VSM) [7], one can use the Information Retrieval (IR) methods to compute their "similarity" [8]. Thus it will be possible to say that the contexts of the two "add-to-cart" buttons in Figure 2 are similar, whereas they are different for the "add-to-cart" button and "Sign Up" link in Figure 2 (a). (All these buttons and links are pointed to by arrows in the Figure.)

Observe that we can infer that two objects are similar despite variations in the actual captions associated with these objects (e.g., the "add-to-cart" and "add-to-bag" buttons). More interestingly, observe that even though the caption in the "add-to-cart" button in Figure 2 (a) is missing (i.e. has no alternative text) we can still conclude that its context is similar to that of the "add-to-cart" button in Figure 2 (b). Thus encoding captions and contexts, which are unstructured content, as vectors in a VSM and using IR methods based on "similarity" computation exposes their semantics quite naturally.

Based on the idea sketched above, we develop an algorithm for improving the accessibility of such objects. The salient aspects of our algorithm are highlighted below:

- We develop the notion of context for transaction-centric clickable web objects. Based on this notion we build a VSM encoding categories of such ob-

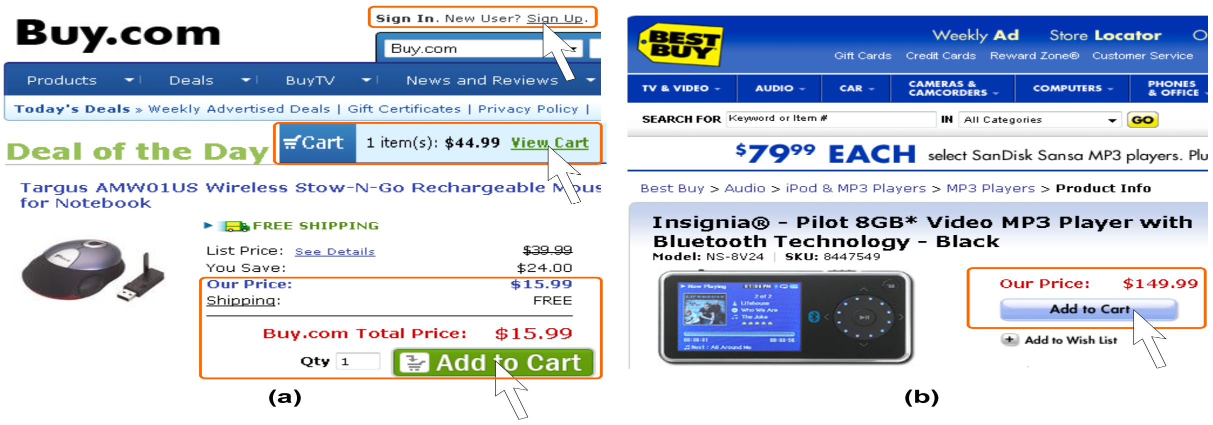

Buy.com
Sign In. New User? Sign Up.
Buy.com
Products
Deals
BuyTV
News and Reviews
Today's Deals » Weekly Advertised Deals | Gift Certificates | Privacy Policy |
Deal of the Day
Cart
1 item(s): $44.99 View Cart
Targus AMW01US Wireless Stow-N-Go Rechargeable Mous
for Notebook
FREE SHIPPING
List Price: See Details $39.99
You Save: $24.00
Our Price: $15.99
Shipping: FREE
Buy.com Total Price: $15.99
Qty 1
Add to Cart
(a)
BEST BUY
Weekly Ad
Store Locator
Gift Cards
Credit Cards
Reward Zone®
Customer Service
TV & VIDEO
AUDIO
CAR
CAMERAS & CAMCORDERS
COMPUTERS
PHONES & OFFICE
SEARCH FOR Keyword or Item #
IN All Categories
GO
$79.99 EACH select SanDisk Sansa MP3 players. Plu
Best Buy > Audio > iPod & MP3 Players > MP3 Players > Product Info
Insignia® - Pilot 8GB* Video MP3 Player with Bluetooth Technology - Black
Model: NS-8V24 SKU: 8447549
Our Price: $149.99
Add to Cart
Add to Wish List
(b)

Figure 2: Clickable Web Objects and their Context

jects. The model provides the computing framework for exposing the semantics latent in the unstructured content that characterizes context.

- The algorithm uses vector similarity to locate and classify such objects in a page even if their captions are new variations of previously seen captions. Consequently there is no need to know all the variations of captions beforehand. More importantly, an object can still be classified to a category even if its caption is missing (e.g. Figure 2(a)). Such an object is assigned the generic name associated with the category.

- The computation of similarity used for classification requires setting thresholds. These thresholds are learnt automatically by cross validation.

- The number of object categories and captions are not fixed a priori; instead, the integration of reinforcement learning based on user feedback with automatic threshold adjustments enables the algorithm to accommodate previously unseen captions of objects and new categories of objects.

- Experimental evidence and preliminary user study suggest that the algorithm is quite effective in practice.

2 Technical Preliminaries.

We use the term *web object* to refer to any element in a web page that has a unique address, e.g., link, button and text elements. We say a web object is *clickable* whenever clicking on the object results in an action, e.g., clicking on a link results in following the link to another web page. We will use the terms web objects and objects interchangeably.

A web page may have several clickable objects. However, only a few of them are needed in transactions. We denote such objects as *transaction-centric* clickable objects, e.g., radio buttons, “add-to-cart”, “checkout” and “login” buttons.

We use the notion of *concept class* to refer to a category of objects. For example, the two “add-to-cart” buttons in Figure 2 belong to the same concept class. We assume that each concept class is assigned a unique, generic concept name such as *ADD_TO_CART* for the concept class consisting of all button elements whose function is to add items to a shopping cart. Objects belonging to a concept class are called *concept instances*. By labeled training data we mean the set of pairs <object, name of the object's concept class>. We will use object class, concept class, and concept interchangeably.

Abstractly, we can associate text content with every object. For example, the caption on a button element is its text; the label on a link element is its text. In Figure 2, the text string *add to cart* is the “add-to-cart” button's text content. The associated text content can be null, as in images with no alternative text.

By *locality* of an Object we mean the set of objects that are “geometrically adjacent” to that Object and are siblings of the Object in the DOM tree [9]. Figure 3 is a fragment of the DOM tree for the page in Figure 2 (a) The rightmost image node in the DOM tree corresponds to the “Add to Cart” button on the page. Its sibling nodes are the text elements “Qty”, “Buy.com Total Price $15.99”, “Shipping Free”, etc. on this page. They constitute the locality of the button.

We define the *context* of a web object to be the text in it, such as a caption of a button, as well as the text around it, delimited by the neighboring objects. In Figure 2 the contexts of the two “Add to Cart” buttons,

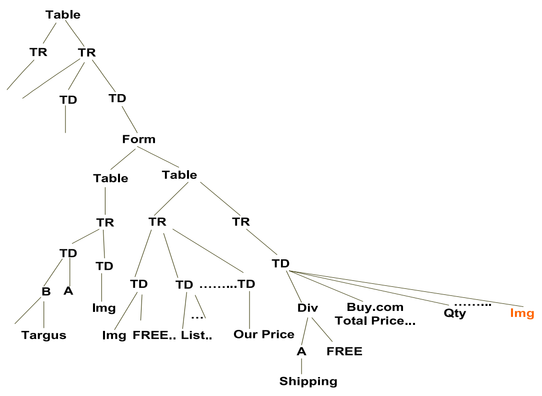


Figure 3: DOM Tree for the page in 2(a)

the "View Cart" button and the "Sign Up" link are constructed from the words appearing in solid rectangles enclosing them.

Since context is essentially unstructured text content, IR offers a natural computing framework for extracting the semantics from such content. IR concepts are reviewed next so as to make the paper self-contained.

The *Vector Space Model* (VSM) is widely used in IR systems for document retrieval [7, 8]. In this model, a document is represented as a vector, where each dimension corresponds to a separate term. Typically terms are single words, bigrams, trigrams or even longer text strings. A term appearing in the document is assigned a non-negative weight. One popular weighting scheme is TF*IDF [7]. It uses the following expression to assign weights:

$$w_{t,d} = \mathrm{tf}_t \cdot \log \frac{|D|}{|\{t \in d\}|} \tag{2.1}$$

In the expression $w_{t,d}$ is the weight of term t in document d; tf_t is the term frequency of term t in document d; $|D|$ is the total number of documents; $\log \frac{|D|}{|\{t\in d\}|}$ is the inverse document frequency; $\{t \in d\}$ is the total number of documents containing the term t.

Suppose $1, 2, .., N$ denote the terms of a document d. Then the weighted document vector v_d for d is:

$$\mathbf{v}_d = [w_{1,d}, w_{2,d}, \ldots, w_{N,d}]^T \tag{2.2}$$

Given a query vector v_q, the cosine of the angle between this vector and a document vector d is the expression:

$$\cos\theta = \frac{\mathbf{v_q} \cdot \mathbf{v_d}}{\|\mathbf{v_q}\| \, \|\mathbf{v_d}\|} \tag{2.3}$$

The cosine value measures the degree of "semantic closeness" between the two vectors. A value of 1 means the vectors are identical, and it is 0 if they are orthogonal. Two vectors are considered to be *similar* if their cosine similarity is above some set threshold.

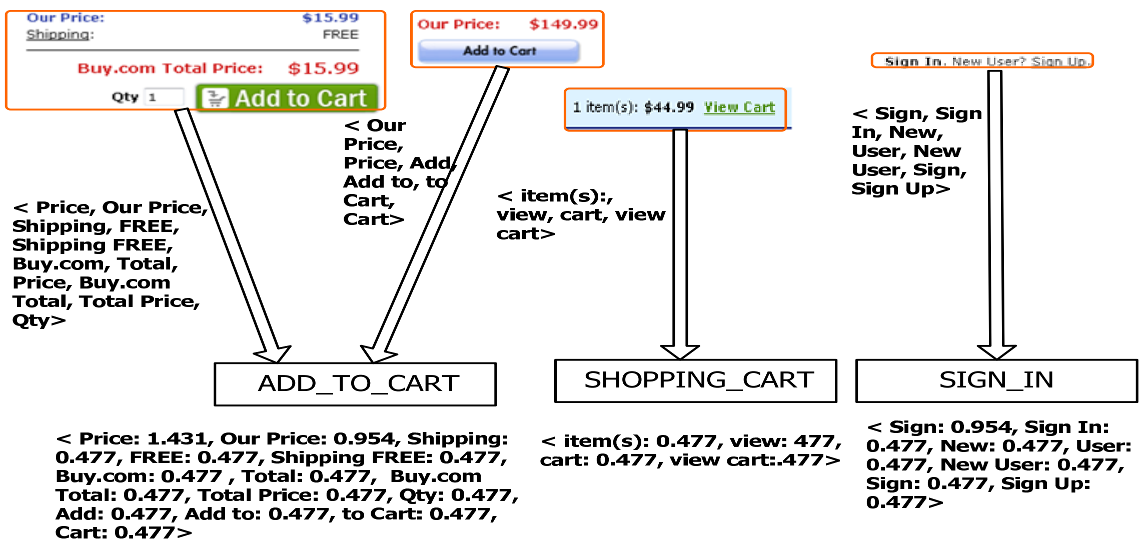
3 Algorithm for Improving Accessibility.

3.1 Overview. Even though most web sites converge on a certain set of descriptive labels (i.e. captions and descriptions) for objects in concept classes, there is still a lot of diversity in the labels used for the same concept. For example, "sign in", "log in", and "my account" - all have the same meaning and essentially indicate the same concept. Evolution of web services continually creates new domains of web transactions, introducing new concepts or new labels for objects in existing concepts. Therefore, it is not practical to collect an exhaustive set of concept classes and a set of their features. Instead, our approach is to let the end-users decide which concepts are relevant and important in their online activities. This means the positive and negative examples of concept instances would be continually supplied by the users. Such a dynamic situation calls for online learning and the ability to "unlearn" both features and classes, as newer concept classes may override older ones.

Although traditional classification technologies, such as Neural Networks [11] and Support Vector Machines [12], can be used for online learning of concepts, the variation in the number of features and concept classes may require changing the structure of NNs and SVMs and retraining over the entire training data.

We propose an efficient online algorithm for concept classification based on IR and standard Machine Learning (ML) principles, such as target function, online, and reinforcement learning. A standard Vector Space Model lies at the core of our IR-based classifier. To store the information necessary for concept classification, we construct a Shallow KB where each concept is represented by a document vector, storing the terms collected from context of the already-seen concept instances and their weights.

3.2 Knowledge Base Construction. Initially the KB is constructed from bootstrapped data consisting of labeled concept instances, i.e., data of the form <object, name of the object's concept class>. Context of each such object is collected. The result is a multiset containing words (excluding stop words, e.g., prepositions, determiners, etc.) and bigrams for each object. These terms are added to the concept vector, and weighted using TF*IDF weighting scheme. Figure 4 shows the KB fragment constructed from labeled concept instances, specifically: the two "add-to-cart" buttons, the "View Cart" button and "Sign Up" link in Figure 2. The KB fragment has 3 concept vectors



Our Price: $15.99
Shipping: FREE
Buy.com Total Price: $15.99
Qty 1
Add to Cart
Our Price: $149.99
Add to Cart
1 item(s): $44.99 View Cart
Sign In. New User? Sign Up.
< Price, Our Price, Shipping, FREE, Shipping FREE, Buy.com, Total, Price, Buy.com Total, Total Price, Qty>
< Our Price, Price, Add, Add to, to Cart, Cart>
< item(s):, view, cart, view cart>
< Sign, Sign In, New, User, New User, Sign, Sign Up>
ADD_TO_CART
SHOPPING_CART
SIGN_IN
< Price: 1.431, Our Price: 0.954, Shipping: 0.477, FREE: 0.477, Shipping FREE: 0.477, Buy.com: 0.477 , Total: 0.477, Buy.com Total: 0.477, Total Price: 0.477, Qty: 0.477, Add: 0.477, Add to: 0.477, to Cart: 0.477, Cart: 0.477>
< item(s): 0.477, view: 477, cart: 0.477, view cart:.477>
< Sign: 0.954, Sign In: 0.477, New: 0.477, User: 0.477, New User: 0.477, Sign: 0.477, Sign Up: 0.477>

Figure 4: A Knowledge-base constructed from labeled concept instances of Figure 2

named ADD_TO_CART, SHOPPING_CART and SIGN_IN. To illustrate the TF*IDF weight computation: observe that the term "Price" has a TF value of 3 (since its frequency is 3) in the concept vector ADD_TO_CART. It occurs only in this concept. Thus, IDF = log(3/1) = 0.477 and so TF*IDF is 1.431 (3 * 0.477).

3.3 Classifying Objects. At a high level the steps to classify an object to a class are: (i) identify its context; (ii) construct its term vector from the words and bigrams in its context; (iii) compute the similarity between the term vector and each of the TF*IDF-normalized concept vectors in the KB; (iv) assign the object to that concept class whose concept vector is most similar to the object's term vector.

The similarity measure is used to compute similarity of term vectors. Specifically, suppose W_1 is the set of words and bigrams in a concept vector C, W_2 is the set of words and bigrams in the term vector built from the object's context, and V_1 and V_2 are TF*IDF-normalized vectors for W_1 and W_2 respectively. Then equation 2.3 in section 2 is used to compute cosine similarity between V_1 and V_2. The cosine similarity formula normalizes the relevance score based on the lengths of the term vector and the concept vector. The TF*IDF measure normalizes the frequency of each term based on the number of concepts sharing that term and the number of terms in a given concept vector. Thus, the longer the text - the lower the score it receives. The term vector is matched against each of the concept vectors. If the cosine similarity score between these two vectors is above the threshold for that concept vector, the object is identified as an instance of that concept. If the object is identified as an instance of multiple concepts (i.e. cosine similarity is above the thresholds for multiple concept vectors), then the highest scoring vector is picked as the final classification. However, it is possible that an object may not be identified as any concept instance (i.e. the cosine similarity is below the threshold for every concept vector). This can happen whenever:

- The object is an instance of a new concept. In such a situation, the user may choose to create a new concept, and give it any name. As a result, the algorithm will create a new concept vector for this class in the KB.
- The object is an instance of an existing concept in the KB. However, the cosine similarity value is below the threshold for the concept since the terms occurring in the object's context have low TF* IDF weights.
- The object is irrelevant (e.g., the link with the caption "Fast way to create a Shopping Cart").

A (cosine) similarity threshold is associated with each concept vector. Lower cosine similarity threshold will result in higher recall and lower precision, resulting in more false positives. Higher threshold value will lower the recall and increase the precision. To start with, we set the threshold to a small value for all concept vectors. As the classifier learns from user feedback (described in the next subsection), the threshold values get automatically adjusted using cross validation.

Algorithm *ClassifyObject* is the high-level abstract pseudo-code for classifying objects:

```
Algorithm ClassifyObject
Input: ClickObj: Object; KB: Current Knowledge Base.
Output: ConceptName: Concept Class
1.  Terms ←Context of the ClickObj
2.  TermVector ←Term vector constructed from Terms
3.  MaxSim ←0
4.  ConceptName ←Null
5.  for j ← 1 to KB.NumberOfConceptVectors
6.      do CurrSim ←Cos(TermVector,
7.      ConceptVector(j))
8.          if CurrSim > Max(Concept(j).Threshold,
9.                  MaxSim)
10.             then ConceptName ←ConceptVector(j).Name
11.                 MaxSim ←CurrSim
12. return ConceptName
```

Figure 5 illustrates object classification. The left part of this Figure shows four objects with the corresponding term vectors constructed from their context. The right part of the Figure shows two concept vectors in the KB and their associated weighted terms. The thresholds for `ADD_TO_CART` and `SHOPPING_CART` concepts are 0.5 and 0.6 respectively. The arrow between an $< object, concept >$ pair is annotated with the value of the cosine similarity between the term vectors of object and the concept. Based on the similarity values the "add-to-cart" and "View Cart" object get classified to `ADD_TO_CART` and `SHOPPING_CART` concepts respectively.

3.4 Updating the Knowledge Base using Reinforcement Learning Technique.

As was mentioned earlier, it is not practical to a priori define all concept classes and captions on objects. Instead, we let the end-user decide which concepts and objects are relevant. This means the concept vectors in the KB need to be updated dynamically; so, we update them incrementally.

Towards that, our algorithm uses a reinforcement technique based on user feedback (learning with a critic) [13]. The idea is as follows: When the IR based classifier misclassifies an object in a web page or fails to identify the object that the user is interested in, as any concept instance, then the user can take one of the following actions which will result in the update of the KB:

Classification Result $\rightarrow$ *User Action*

1) Object not identified as any concept instance $\rightarrow$ Create a new concept class and add the object as its concept instance;
2) Object misclassified $\rightarrow$ Assign it the correct concept label;
3) Object is a false-positive concept instance $\rightarrow$ Remove it from the concept class.

3.4.1 Update operations.

Internally user actions are treated as positive and negative examples. Case (1) is positive and case (3) is negative. Case (2) creates two examples: positive example for the correct classification and a negative example for misclassification. Algorithm *Update* is the pseudocode describing how the KB is updated. Positive examples reward the classifier by increasing the length of the corresponding concept-vector components on lines 2 and 6. Algorithm *Reward* illustrates how we increase the length of the concept vector with the terms occurring in a positive example. First, we increase the frequency of each of these terms in the concept vector (line 3 of the algorithm *Reward*). Next, we update TF*IDF score for these terms in each of the concept vectors where they occur (line 5 of the algorithm *Reward*). To penalize a concept vector with the terms occurring in a negative example, we decrease the frequency of each of these terms (if they already occur) in the concept vector. Next, we update TF*IDF score for these affected terms in each of the concept vectors where they occur.

```
Algorithm Update
Input: C: Terms collected from context of concept instance; L:
    User Label for Concept Instance; T: Original Classification
    Label; KB: Current Knowledge Base.
1.  if $T = Null$ and $L \neq Null$
2.    then KB.Reward(L, C)
3.  if $T \neq Null$ and $L = DELETE$
4.    then KB.Penalize(T, C)
5.         KB.InreaseThreshold(T)
6.  if $T \neq Null$ and $L \neq Null$ and $L \neq T$
7.    then KB.Reward(L, C)
8.         KB.DecreaseThreshold(L)
9.         KB.Penalize(T, C)
10.        KB.InreaseThreshold(T)
```

```
Algorithm Reward
Input: C: Terms Collected from Context of the Concept In-
    stance; L: User Label for Concept Instance;
1.  Vector ←GetConceptVector(L)
2.  for i←1 to C.Size
3.      do Vector.IncreaseFrequency(C(i))
4.         Concepts ←C(i).ConceptVectors
5.         for j←1 to Concepts.Size
6.             do Concepts(j).updateTFIDF(C(i))
```

Observe that the system cannot control the diversity of or errors in the training examples, implying that error rate can fluctuate over time. This is acceptable if the error rate is low and its fluctuations have a low standard deviation, corresponding to a "balanced state." This state of the system can be informally defined as low frequency of user labeling updates due to tolerable error rate. In Section 4, we describe a simulation of repeated updates that demonstrates that the system converges to a "balanced state" in a reasonable time, reaching 90+% accuracy of concept detection and classification.

3.4.2 Threshold Adjustment.

Once an object is misclassified as an instance of concept `T`, we increase the threshold value for the concept vector `T` (line 5, 10 of the

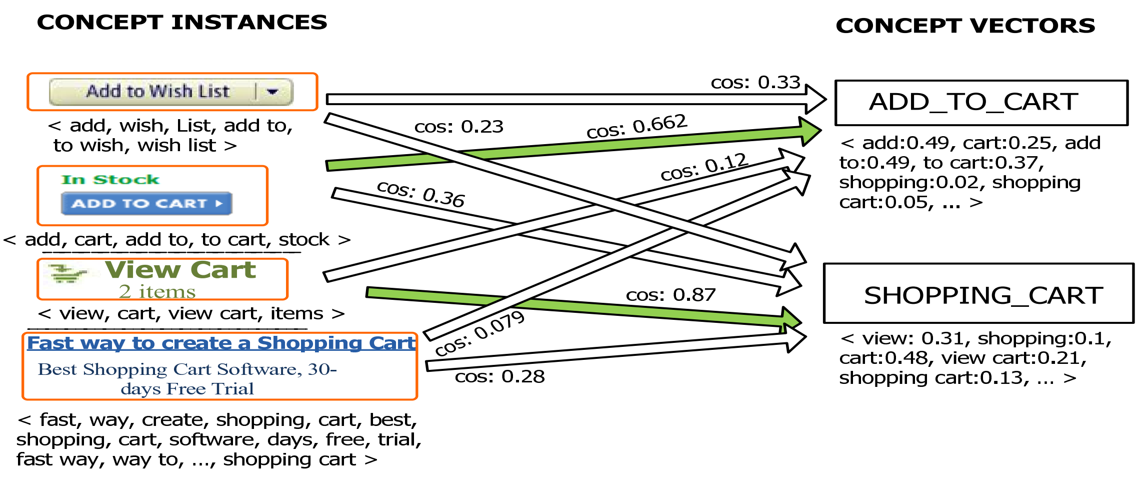

CONCEPT INSTANCES
CONCEPT VECTORS
Add to Wish List
< add, wish, List, add to, to wish, wish list >
In Stock
ADD TO CART
< add, cart, add to, to cart, stock >
View Cart
2 items
< view, cart, view cart, items >
Fast way to create a Shopping Cart
Best Shopping Cart Software, 30-days Free Trial
< fast, way, create, shopping, cart, best, shopping, cart, software, days, free, trial, fast way, way to, ..., shopping cart >
cos: 0.33
cos: 0.23
cos: 0.662
cos: 0.36
cos: 0.12
cos: 0.87
cos: 0.079
cos: 0.28
ADD_TO_CART
< add:0.49, cart:0.25, add to:0.49, to cart:0.37, shopping:0.02, shopping cart:0.05, ... >
SHOPPING_CART
< view: 0.31, shopping:0.1, cart:0.48, view cart:0.21, shopping cart:0.13, ... >

Figure 5: Concept Identification

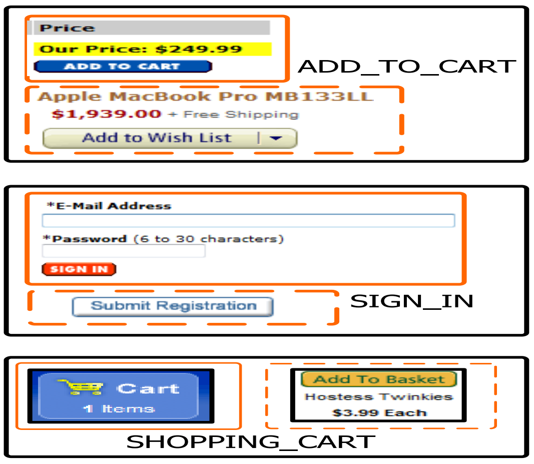


Figure 6: An Example Cross Validation Set Containing Positive and Negative Examples of Concepts

algorithm *Update*). A higher threshold value decreases recall for that concept. If the user assigns the concept label L to the object, then we decrease the threshold value for the concept L (line 8). A lower threshold value increases the recall value for concept L. For adjusting the threshold (i.e., increase/decrease) we use cross validation. We maintain a small cross validation set (roughly 1% of the total data set) containing positive and negative examples of each concepts. Figure 6 is an example of a cross validation set, containing positive examples (marked using solid rectangles) and negative examples (marked using dotted rectangles) for the three concept vectors “ADD_TO_CART”, “SIGN_IN” and “SHOPPING_CART”. Below we describe our cross validation algorithm to adjust threshold values.

3.4.3 Cross Validation Algorithm. Suppose the threshold for a concept vector needs adjustment. Starting with a small value initially the threshold is iteratively increased (decreased) at a fixed rate δ ($-\delta$) for *IncreaseThreshold* (line 5, 10 of the algorithm *Update*) (*DecreaseThreshold* in line 8 of the algorithm *Update*). In an iteration step, the current threshold value in that step is used to run the algorithm on the cross validation set and compute the F-measure (defined in footnote 1). The iteration continues as long as the F-measure monotonically increases. Note that as threshold value increases (decreases), recall (precision) decreases, so that eventually it offsets F-measure gains. So F-measure will eventually decrease. Suppose F-measure increases monotonically until the n^{th} iteration. Let F_n denotes the F-measure in iteration n. Since in the $(n+1)^{th}$ iteration F-measure decreases for the first time, $F_n > F_{n+1}$. Now there exits an F_{max} between F_n and F_{n+1}. We want to find that threshold value Γ_{opt} such that running the algorithm with that threshold value on the cross validation set will result in the F-measure value of F_{max}. Let Γ_n and Γ_{n+1} be the thresholds in iteration n and $n+1$ respectively. We do a binary search over the range $\Gamma_{n+1} - \Gamma_n$. Suppose Γ_i is the threshold value at the split point in this interval. We compute F_i, the F-measure value for the cross validation set with Γ_i as the threshold. If $F_i > F_n$ then we recursively search in the interval $\Gamma_{n+1} - \Gamma_i$ else in the interval $\Gamma_i - \Gamma_n$. The cross validation set used for adjusting the thresholds is also updated incrementally as previously unseen captions of objects are encountered in a page and new concept classes get created.

4 Evaluation.

We collected a dataset of clickable objects and carried out a series of experiments to evaluate various aspects of the algorithm's behavior. In this section we report the experiments and the performances.

4.1 Data collection. We used the Heretrix crawler[16] to collect 3000+ web pages in the shopping domain (books, electronics, office supplies, and others) from over 150 web sites that were used to seed the crawler. The collected web pages were used to extract clickable web objects (an average of 70-100 links and buttons per page) into a repository of nearly 200,000 unique records containing <concept class, caption, context>. We enforced uniqueness in our datasets so that no record could appear in both training and validation datasets and skew the results.

A total of 10 concept classes were identified in the dataset: "CONTINUE_SHOPPING", "REGISTER", "UPDATE QUANTITY", "SHOPPING_CART", "ADD_TO_CART", "SEARCH FORM", "CHECK_OUT", "SIGN_OUT", "DELETE", "SIGN_IN". Of the 200,000 unique records, a total of 5,621 records were identified as concept instances and the rest of the records were labeled as "NON-CONCEPTS". The dataset can now be used to bootstrap the classifier when it is integrated into a screen-reader.

4.2 Experimental Setup. We used the dataset described in Section 4.1 to run a number of automated experiments and test the effectiveness of our algorithm in terms of recall (R), precision (P), and F-measure (F) [1].

We performed a standard 10-fold cross validation over the entire data set. We used 90% of the labeled data as training examples and the remaining 10% for testing. We then rotated the training data 10 times to cover the entire dataset.

We then let the simulation iterate over the entire training dataset 100 times, while we measured the accuracy of the algorithm on the testing dataset, i.e., the testing dataset was reclassified after every 1,000 clickable objects were presented to the hypothetical user at every iteration.

It is noteworthy that in the envisioned use scenario, the user will be more interested in concept identification. While it is important to classify the concepts correctly, it is not critical, since the actual caption on the object (where available) will be read to the user. Therefore, we make a distinction between concept identification and concept classification. The former is concerned with simply recognizing that a clickable object is a concept, whereas the latter is concerned with assigning the object to the correct concept class.

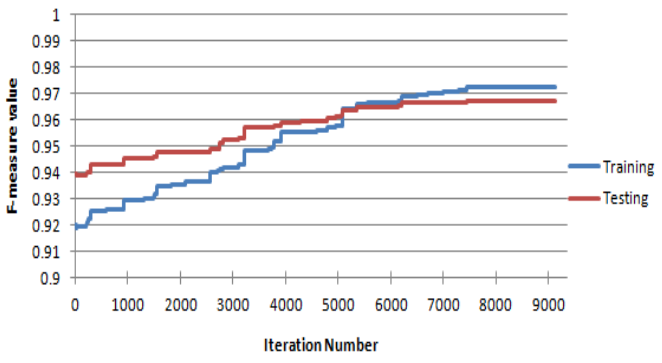


Figure 7: Learning curves of the classifier

4.3 Learning Rate Experiment. To demonstrate that the algorithm did not over-fit for training data, we plotted learning curves (F-measures). Figure 7 shows two different learning curves on one of the 10 pairs of datasets reaching the F-measure of over 92% on both training and testing datasets. For the duration of the experiment, the accuracy showed steady growth over the 9000 iterations on both both training and testing datasets,

4.4 10-Fold Cross-Validation. We ran a 10-fold cross-validation experiment that only checked the accuracy at the end of training; the averages of recall, precision, and F-measure for identification and classification on both training and testing datasets are summarized in Table 1 and Figure 8, and discussed next.

- Only Captions

We first trained the VSM using only captions, thus creating a Caption/Label VSM. The results over training and testing dataset are summarized in Figure 8(a).

It is notable that when the results were measured on the testing dataset, the classifier showed approximately 93.97% recall, and 94.03% precision on the testing dataset. Considering that the entire dataset predominantly contained false positives (only about 2.5% concept instances), this means that 4% of the recalled concepts were false positives. The classification accu-

[1]Precision is the fraction of the documents retrieved that are relevant to the user's information needs.
Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.
F-Measure is the simple harmonic mean of Recall and Precision.

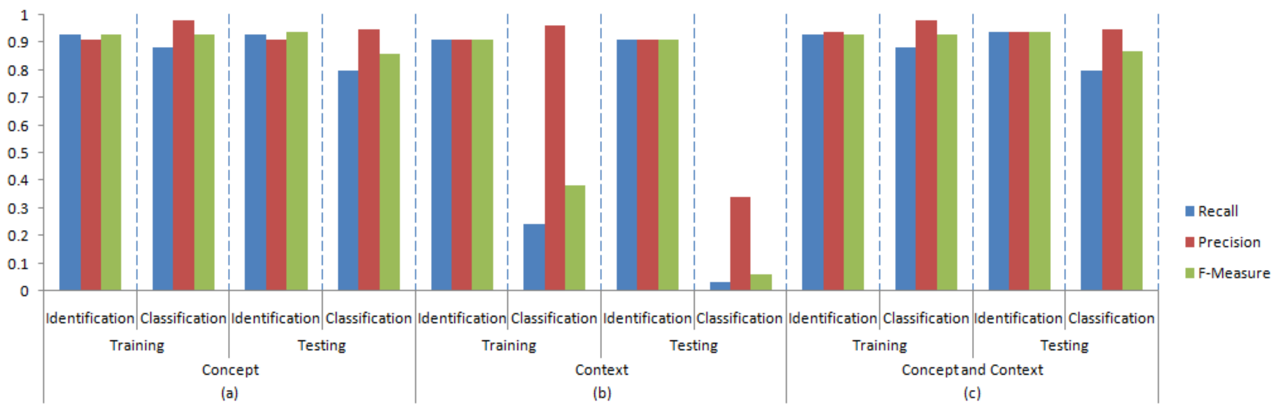


Figure 8: Classification with Concept, Context, and both Concept and Context

racy on the training dataset had an F-measure of 93%, while it was 86% over the testing dataset. However, as we noted before, users are mostly interested in the identification than the classification.

Table 1: Classification with Concept, Context, and both Concept and Context

		Only Concept		Only Context		Concept and Context	
		Training	Testing	Training	Testing	Training	Testing
Identification	R	0.9386	0.9397	0.9116	0.9145	0.9386	0.9402
	P	0.9116	0.9145	0.9143	0.9157	0.9410	0.9403
	F	0.9388	0.9400	0.9129	0.9151	0.9398	0.9402
Classification	R	0.8816	0.8010	0.2454	0.0347	0.8875	0.8087
	P	0.9876	0.9559	0.9659	0.3475	0.9876	0.9559
	F	0.9309	0.8696	0.3899	0.0620	0.9349	0.8762

- Missing captions

We then trained the VSM with only context, thus creating a Context VSM. This experiment was conducted to assess the effectiveness of the algorithm in identifying unlabeled clickable objects, i.e., image-buttons and image-links with no alternative text. It is notable that, of the 5,621 collected concept records, only 2,145 concept instances had any context, so we only used the concept instances that had context. The results of the experiment are summarized in Figure 8(b).

Although the overall classification results seemed rather low, the algorithm was able to correctly recall 91% of the unlabeled concepts, which had been previously completely inaccessible. One possible reason for low classification accuracy was the ambiguities often introduced by context. For example, same word was taken as context for multiple concepts (e.g., the word "shopping" may appear as context for both "shopping cart" and "continue shopping" concepts). The results may have been also affected by a very small number of training examples, because one third of the original concepts instances were used in the experiment. In future, we will conduct further research to determine the best context that improves concept classification accuracy.

- Both Captions and Contexts

Finally, we used both the Label and Context VSMs together to verify whether the use of context had any improvement over the Label VSM accuracy. The two trained VSMs were used in the following manner. If an actionable object had captions then we used only the Label VSM. Context VSM was used only when the actionable object had no captions, i.e., image-buttons and image-links with no alternative text. Figure 8(c) summarizes the results of this experiment.

The same experiment showed overall better results when we combined the two VSMs. Specifically, the algorithm achieved 94.02% recall, and 94.03% precision over the testing dataset. Finally, the F-measure of classification was 93% over the training dataset and 87% over the testing dataset.

4.5 Error Tolerance. A separate experiment was conducted to simulate user labeling errors by introducing random errors into the simulated classification feedback. The experiment showed graceful degradation of the classifier, maintaining high accuracy of classification.

Another scenario that has to be considered in a real-life application is the introduction of alias names for the same concept class. For example, the "SEARCH FORM" concept can be labeled as "find" or "search", resulting in two overlapping concept classes. However, since users are read the original object captions, where available, aliased concept classes will not have any adverse effect on concept identification. The only possible side-effect is that the web objects without alternative text may periodically change their labels, e.g., from "find" to "search", depending on which of the two classes is preferred by the majority of the users. At the same time, since the classifier can learn and unlearn concepts on the fly, the preferred class label may eventually win over, making the other aliased class disappear. In addition, a separate technique may be developed to identify possible aliased classes as candidates for semi- or fully-automated merging. Finally, for the users labeling the objects, their personalized labels will remain consistent, as they are retrieved from label database. We leave the experiments related to alias resolution to future work.

5 Interface and User Study.

5.1 Interface. We used our Hearsay nonvisual web browser [33] to test the effectiveness of our approach. To achieve that, we implemented shortcuts for navigating back and forward between clickable objects. The shortcuts enabled users to quickly iterate over the identified clickable objects. We also added the functionalities for labeling actionable objects and deleting false positives. By labeling an actionable object, users could either add or change its label and, hence, its class assignment. By pressing delete, the users could remove labels from the identified actionable objects. To persist the labels provided by the users, we attached a database that stored user labels. With the database, when the same user revisited the web page, s/he could access the labels s/he had previously created. We believe that the use of the database provides instant gratification for labeling and will encourage users to provide labels when the system is deployed to the real end-users.

5.2 Preliminary User Study. In this section, we present a preliminary user study with a visually impaired person. The objective of this study was to see how the integrated system improves user experience in online transactions.

5.2.1 Experimental setup and evaluations. We have asked a blind subject to visit ten popular shopping web sites (amazon, bestbuy, buy, barnesandnobles, officedepot, macys, target, bullards, flowersonfirst, and abebooks) and go through the process of buying different items such as electronics, computers, clothes, groceries, books, office supplies). The process included searching for a specific item, adding it to cart, updating quantity, and checking out. We conducted experiments with three different conditions, namely: 1) the basic Hearsay system[33], where only the basic screen-reading functionalities were available (e.g., navigating among links, headings, buttons, etc.); 2) Hearsay with concept detection, where a shortcut was provided to navigate between the identified concepts; and 3) Hearsay with concept detection and label support interface, where the labeling interface (discussed in section 5.1) was added to the concept detection. The evaluator was trained on how to use those systems, and was instructed to conduct three transactions on each site. The conditions were balanced (rotated) to eliminate the effect of learning. During the study, we measured the time required by the blind user to buy an item from the specific web site. Figure 9 and Table 2 show the average and the standard deviation of the transaction times with different conditions.

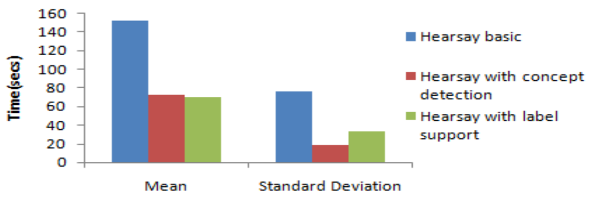


Figure 9: Mean and Standard Deviation(SD) of the transaction time with different conditions

Table 2: Mean and Standard Deviation(SD) of Transaction Time(in seconds)

	Hearsay basic	Hearsay with concept detection	Hearsay with label support
Mean	151.3	71.5	69.8
SD	75.45	17.83	32.37

5.2.2 Analysis of the results. The results of the preliminary user study have shown that using the concept detection and/or label support feature, the subject was able to complete the ten tasks on average twice as fast as with the basic system. The labeling support only ensured that all concepts were correctly identified, which only marginally improved the time it took to go through the transactions. The ability to quickly navigate between the concepts helped the subject avoid doing unnecessary steps and, in this way, helped reduce the information overload. The improved interface received positive feedback from the evaluator. The only criticism was the difficulty of figuring out the association between the add-to-card buttons and the corresponding item descriptions. One avenue for future research would be to associate item description with the add-to-cart concept.

6 Related Work.

This work has broad connections to research in web content analysis, contextual analysis, information retrieval, machine learning, document classification, and accessibility.

6.1 Content Analysis of Web Documents. There exits prior research in content caching [14], data cleaning [15], etc. The idea of understanding semantics of web forms using grammars was explored in [17]. Concept identification in web pages is related to the field of research on semantic understanding of web content. Powerful ontology management systems and knowledge bases have been used for interactive annotation of web pages [18]. More recently, [19] describes an elegant method for annotating form elements using machine learning techniques.

In our work, objects' captions do not always provide sufficient information for identifying and categorizing them. So we explore the idea of using the context of an object. Since context is made up of unstructured data IR methods offer an attractive solution for extracting their semantics. We developed an IR-based learner that utilizes unstructured contextual information of objects to locate and categorize them even if their captions are missing.

6.2 Contextual Analysis. The notion of context has been used in different areas of Computer Science research. For example, [20] defines context of a web page as a collection of text, gathered around the links in other pages pointing to that web page. The context is then used to obtain the summary of the page. Summarization using context is also explored by the InCommonSense system [21], where search engine results are summarized to generate text snippets.

All these approaches define the context of a link as an ad hoc collection of words around it. In [10] we used the context of a link to identify relevant information on the following page. In this paper, we use the context of clickable web objects to identify and classify these objects more accurately (with and without alternative text captions).

6.3 Information Retrieval (IR) and Machine Learning (ML). IR and ML techniques are often used for data extraction, clustering, searching, and classification tasks. In this paper we describe an efficient online reinforcement learning algorithm that can be used to train a classifier for clickable web objects. Our approach combines the vector space IR model described in [7] and the standard techniques generally used in machine learning, specifically online reinforcement learning.

Online learning is a growing subset of machine learning that learn on a per-example basis. There exist a variety of online algorithms, such as online clustering [22], online SVM [23], online Bayesian learning [24], etc. Our algorithm shares some of the features characteristic of online learning algorithms, because it is capable of incrementally learning and unlearning new features and classes as it is provided with new training examples.

Reinforcement learning [13], also called learning with a critic, involves using some performance feedback to improve the accuracy of a machine learning algorithm. [25] describes how bringing a human into the loop can significantly improve the accuracy of classification. Our algorithm also utilizes user feedback to learn the model.

6.4 Document Classification. Identification and classification of clickable web objects can be generalized to document classification with filtering, where the objects and their context can be thought of as documents that need to be classified. A number of methods were proposed for document classification. Examples of document classifiers include: Naive Bayesian [26], k-nearest-neighbors [5], SVM [12], etc. However, most of these methods are used for batch mode document classification. [28] describes an online document classification algorithm, which requires prior knowledge about data distribution. Moreover, the method described in [28] requires an initial training phase. In contrast, our object classifier does not use any prior knowledge or initial training. Although our classifier *can* be bootstrapped with initial data, it is also able to learn from scratch, as users provide feedback on its classification accuracy for every new example.

6.5 Accessibility Research. Several research projects aimed at improving web accessibility include work on browser-level support [1, 29], content adaptation and summarization [30], organization and annotation of web pages for effective audio rendition [31], our work on contextual browsing [10], etc. In contrast to all of the above, our work locates and categorizes clickable transaction-centric objects in web pages. The problem addressed in this paper emerged from our previous work where we explored speech-enabled assistive browser interfaces for conducting transactions [6]. In that work we used a simplistic static KB with keywords as features. In contrast, this paper proposes a robust and scalable solution for the same problem.

7 Conclusions.

In this paper we proposed a new IR-based algorithm for improving accessibility of objects used in web transactions. One of the novel aspects of our algorithm is the use of context of transaction-centric objects to identify and classify them even when their captions are missing. Another interesting aspect of the algorithm is its adaptability to emerging concepts resulting from evolution of web services that create new domains of web transactions.

Web surfing/browsing is becoming pretty common in mobile phones or hand-held devices and people are using those for online transactions. There are over several millions of such smart phones today and the numbers are leaping up everyday. So, the problem of locating transaction-centric actionable objects from web pages would arise in those cases too. Therefore, our solution for improving accessibility of those objects has a broader impact.

We identify some avenues for future research. Screen reading technology enjoys a fairly loyal user base. By incorporating this algorithm as a plug-in into extant screen readers we can readily tap into this pool of users. This is an interesting engineering challenge. Developing a technique facilitating page and site-specific anchoring of personalized labels to give our end-users immediate gratification for their labeling efforts is another intriguing problem. Yet another interesting problem is the extension of our online-learning methodology to the collaborative labeling paradigm [32].

References

[1] *http://www.freedomscientific.com/*

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

[3] Jeffrey P. Bigham, Ryan S. Kaminsky, Richard E. Ladner, Oscar M. Danielsson, and Gordon L. Hempton, *WebInSight:: making web images accessible*, Assets '06, Portland, Oregon, USA.

[4] Yevgen Borodin, Jalal Mahmud, Asad Ahmed, and I. V. Ramakrishnan, *WEBVAT: Web Page Analysis and Visualization Tool*, ICWE, 2007.

[5] Dasarathy, B.V. *Nearest Neighbor (NN) Norms*, 1991.

[6] Jalal Mahmud, Yevgen Borodin and I.V. Ramakrishnan, *An Assistive Browser for Conducting Web Transaction*, IUI '08.

[7] G. Salton and A. Wong and C. S. Yang, *A vector space model for automatic indexing*, Commun. ACM '75, pages 613–620, New York, NY, USA.

[8] Christopher D. Manning and Prabhakar Raghavan and Hinrich Schtze, *Introduction to Information Retrieval*, Cambridge University Press '08.

[9] *Document Object Model (DOM) Technical Reports*, http://www.w3.org/DOM/DOMTR.

[10] Jalal Mahmud and Yevgen Borodin and I. V. Ramakrishnan, *CSurf: A Context-Directed Non-Visual Web-Browser*, WWW '07.

[11] Christopher M. Bishop, *Pattern recognition and Machine Learning*, Springer '06.

[12] V. Vapnik, *Principles of risk minimization for learning theory*, Advances in Neural Information Processing Systems 3, pages 831-838, Morgan Kaufmann '92

[13] Dhillon, I.S. and D.S. Modha, *Concept Decompositions for Large Sparse Text Data Using Clustering*, Machine Learning, vol 42(1/2), pages 143-175, '01.

[14] Lakshmish Ramaswamy, Arun Iyengar, Ling Liu and Fred Douglis, *Automatic Detection of Fragments in Dynamically Generated Web Pages*, WWW '04.

[15] Lan Yi and Bing Liu, *Eliminating Noisy Information in Web Pages for Data Mining*, SIGKDD '03.

[16] *http://crawler.archive.org/*

[17] Zhen Zhang, Bin He and Kevin Chen-Chuan Chang, *Understanding Web Query Interfaces: Best-Effort Parsing with Hidden Syntax*, SIGMOD '04.

[18] J. Kahan and M. Koivunen, E. Prud'Hommeaux and R. Swick, *Annotea: An Open RDF Infrastructure for Shared Web Annotations*, WWW '01.

[19] Hoa Nguyen, Eun Yong Kang and Juliana Freire, *Automatically Extracting Form Labels*, ICDE '08.

[20] J.-Y. Delort, B. Bouchon-Meunier and M. Rifqi. Enhanced, *Enhanced web document summarization using hyperlinks.*, HYPERTEXT '03, pages 208-215.

[21] E. Amitay and C. Paris, *Automatically Summarising Web Sites - Is There A Way Around It?*, CIKM '00.

[22] Zhili Zhang, Changgeng Guo, Shu Yu, De Yu Qi and Songqian Long, *Web Prediction Using Online Support Vector Machine*, ICTAI '05.

[23] Antoine Bordes and Lon Bottou, *The Huller: A Simple and Efficient Online SVM.*, ECML '05.

[24] Kian Ming Adam Chai, Hai Leong Chieu and Hwee Tou Ng, *Bayesian online classifiers for text classification and filtering*, SIGIR '02.

[25] Shantanu Godbole, Abhay Harpale, Sunita Sarawagi and Soumen Chakrabarti, *Document classification through interactive supervision of document and term labels*, PKDD '04, pages 185-196, Pisa, Italy.

[26] Rish, I, *An empirical study of the naive Bayes classifier*,'01.

[27] Thorsten Joachims, *Text categorization with support vector machines: learning with many relevant features*, ECML '98, pages 137-142, Heidelberg, DE.

[28] Zhong, S, *Efficient online spherical k-means clustering*, IJCNN '05, pages 415-422, Montreal, Canada.

[29] Hironobu Takagi, Chieko Asakawa, Kentarou Fukuda and Junji Maeda, *Site-Wide Annotation: Reconstructing Existing Pages to be Accessible*, ASSETS' 02.

[30] Mary Zajicek, Chris Powell and Chris Reeves, *A Web Navigation Tool for the Blind*, ASSETS '98.

[31] Masahiro Hori, Goh Kondoh, Kouichi Ono, Shin-ichi Hirose and Sandeep Singhal, *Annotation-based Web Content Transcoding*, WWW '00.

[32] S. Kawanaka, Y.Borodin and J. Bigham, et al., *Accessibility Commons: A Metadata Infrastructure for Web Accessibility*, ASSETS '08.

[33] Y. Borodin, J. Mahmud, I.V. Ramakrishnan, and A. Stent, *The HearSay Non-Visual Web Browser*, W4A 2007.