Serving Comparative Shopping Links Non-invasively

Renan Cattelan[†], Darko Kirovski[‡], and Deepak Vijaywargi[◇]
[†] Universidade de São Paulo, São Carlos, SP, Brazil
[‡] Microsoft Research, Redmond, WA, USA
[◇] University of Washington, Seattle, WA, USA

Contact: darkok@microsoft.com

Technical Report MSR-TR-2008-014
January 2008

Microsoft Research
One Microsoft Way Redmond, WA 98052, USA
http://research.microsoft.com

Serving Comparative Shopping Links Non-invasively

ABSTRACT

Marketing and commerce are two activities that have primarily funded the economic lifecycle of the Internet. From seller's perspective, the objective is to sway user traffic onto its Web-site which offers merchandise and/or service. Strategies to fulfill this goal typically range from push and pull ads to spam. Ads are presented at publishers' sites which offer content popular among the general public. From buyer's perspective though, shopping has become an overly distractive and deceiving process as search engines paired with recommendation systems often spread their results over many Web-pages so to present as many as possible ads.

We propose a simple, user-friendly tool which aims to offer comparative shopping to the consumer with minimal distraction. The key idea is to detect whether a specific Web-page is commercial, i.e., whether it sells an individual product or service. The detection is performed in real-time at the client with focus on exceptionally low false positives. For each commercial page, we aim to identify the product name P from its hypertext and send P to a knowledge server which would respond with a list $\mathbb{L}$ of URLs at which P is sold in increasing order of pricing. The browser would then present $\mathbb{L}$ in an non-invasive fashion to the user. The user could browse K sites in $\mathbb{L}$ with only K clicks resulting in a simple and effective shopping experience. In this paper, we present certain statistical properties of collected commercial Web-pages, introduce a novel classifier of Boolean spaces used in the detector, and compare its performance to an SVM with quadratic programming.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce; I.2.11 [**Artificial Intelligence**]: Intelligent agents; H.4.m [**Information Systems**]: Miscellaneous; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*learning*.

General Terms

Algorithms, experimentation, performance.

Keywords

Comparison Shopping, Feature Extraction, Text-Mining.

1. INTRODUCTION

On-line shopping should be a simple and enjoyable process for a typical consumer considering the global accessibility of information on the Web. Unfortunately, in most cases it is not. The three actors in the process: merchants, Web intermediaries, and consumers, have crystal clear objectives. Merchants resort to both push and pull [1, 2] ads in order to make the general public aware of their existence and/or their current commercial offers. Ads get published at Web-pages with popular or forced (e.g., pop-ups, ad aggregators) content with hope of drawing traffic to the merchant. Consumers, when decided to purchase a specific product, opt to seek for the best "deal" on the Web, which is typically a minimum of a function over product's price vs. merchant's reputation. For a given product, the very few merchants who fall in this category usually have razor-thin margins. The complication in the economic ecosystem stems from the fact that ad publishers' revenues are directly proportional to the number of served ads via the pay-per-click ad delivery model.[1] Thus, it is in their best interest to obfuscate the shopping process that benefits the consumer. As a result, Web giants such as Google, Yahoo, eBay, Microsoft, and Amazon.com all contribute to the creation of the $9B (in 2006 [3]) on-line advertisement market built mainly with a purpose to disrupt the consumer on the Internet while shopping or not. In this paper, we focus on a technology that is disruptive to the non-branding advertisement business on the Web as it aims to interject the current shopping process and deliver to the consumer the information necessary to make decisions while shopping.

A great survey and analysis of the effect of shopbots (softbots [4] for shopping resources on the Web) to on-line retailers and consumers can be found in [5, 6, 7]. Several years ago the prediction that shopbots would decimate the retailers on the Web as they are forced into a pricing war, did not materialize for numerous reasons: shopbots were specialized, difficult to find on the Web, and finally, they quickly moved to biased recommendation systems due to difficulty maintaining profitable operations. Recent success of peer-to-peer networks such as BitTorrent and community assembled and controlled encyclopedia-style databanks such as Wikipedia, has shown that small collaborative effort and donated computing power and storage from participants can build useful, economically self-sustainable powerful distributed systems.

1.1 Non-invasive Comparative Shopping

We present a user-friendly tool for comparative shopping. It has a simple goal, once the browser has a high-probability signal that the user is shopping for a product P, ask a server to provide a list $\mathbb{L}$ of URLs of Web merchants who offer P in

[1]Pay-per-conversion ad systems are infrequent and not popular among Web intermediaries due to the technical difficulties they pose and relatively inferior revenue potential.

increasing order of price. The browser then displays $\mathbb{L}$ in an non-invasive manner to the user who can browse $\mathbb{L}$ without distractions. For a list of K URLs the consumer could visit all of them in only K mouse-clicks. Figure 1 illustrates the proposed system using an example.

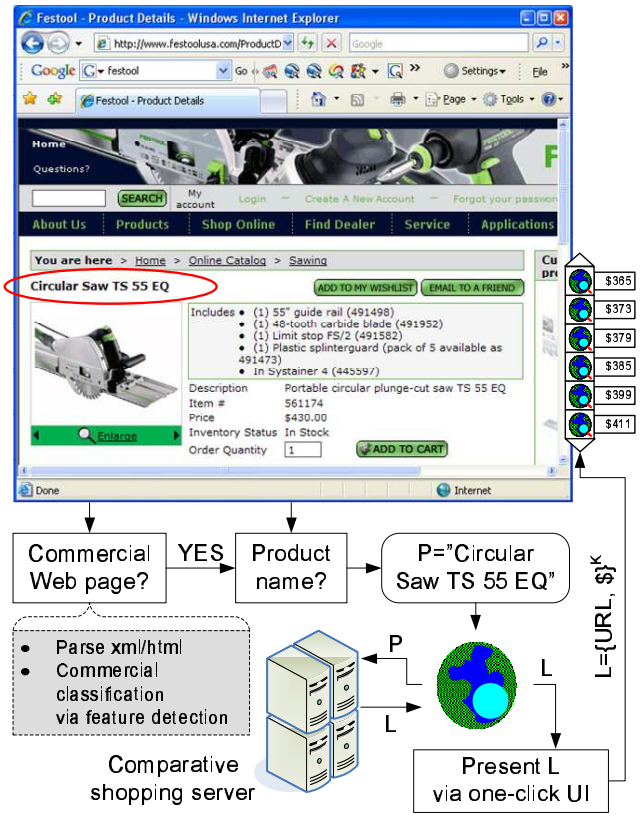

Figure 1: Basic Block Diagram of the proposed comparative shopping tool. The platform consists of a client-based tool that detects whether a Web-page is commercial and a product extraction modeling tool.

The key component of our comparative shopper plug-in is a detector of commercial Web-pages. A page is deemed commercial if its visitor can add a product or service into a shopping cart.[2] The detection is performed transparently and in real-time at the client-side only as the user browses the Web. Client-side detection is an imperative for two reasons: protection of user's privacy and reduction of server's workload. To minimize user distraction, we want to build a detector that offers an exceptionally low rate of false positives. This requirement is unique with respect to related work as it has never been singled out as a primary concern.

To address the above problem, we introduce a conceptually novel classifier. To the best of our knowledge, it is the first classifier that seeks to expand Boolean functions in an N-dimensional Boolean space $\mathbb{B} = \{0,1\}^N$ using fast and simple heuristics. The positive and negative training dataset is represented in this formalism as two potentially intersecting Boolean functions f_P and f_N respectively, which are not specified over a vast majority of $\mathbb{B}$. Growing f_P and f_N in $\mathbb{B}$ with respect to (or even without) additional input, is the key classification objective. In this paper, we demonstrate the effectiveness of the novel classifier by comparing its performance to an SVM with quadratic programming.

In case a Web-page is commercial, our tool then aims to identify the product name P from its hypertext. This objective has proven to be prohibitively difficult to solve in the generic case. Thus, we propose a solution that uses a simple longest common substring algorithm to produce a list of candidate product names $\mathbb{P}$ extracted from the hypertext. This list $\mathbb{P}$ is then sent to the server which in turn resolves the product name P from its e-commerce URL database.[3]

The server replies to the user with a list $\mathbb{L} = \{u, s\}^K$ of K URL-price (u, s) tuples sorted in increasing order of price and/or other criteria. Each URL u_i is a pointer to a Web-page that sells P at price s_i. User's browser then presents $\mathbb{L}$ in a non-invasive but effective manner. For example, the browser would use minimal real-estate to display $\mathbb{L}$ by semi-transparently overlaying $\mathbb{L}$ with the current Web-page so that the user could browse all K links with K clicks. The browser could offer to the user a quick way to remove the display of $\mathbb{L}$, e.g., by a quick random motion of the mouse.

The objective of our simple comparative shopping tool is to transparently aid the user while shopping by offering comparative shopping lists only when the user is browsing commercial content. For user convenience, the tool is not designed as a recommendation system or an ad-display platform although it could. We do not want to pose a restriction on the implementation of the server side. The database that contains digested and sorted commercial offerings could be centralized or peer-to-peer; it could be compiled by a single entity or by collaborative effort of users who report their unbiased shopping experiences into the database.

1.2 Paper Organization

The remainder of the article is organized in the following way. Section 2 presents related work in the domain of comparative shopping tools, recommendation systems, and text mining algorithms. Section 3 focuses on formally defining the problem and establishing the statistical model used in the detection. Section 4 describes the detector implementation and outlines several statistical properties of commercial Web-pages that influenced the construction of our models. Finally, Sections 5 and 6 present the experimental results and outline conclusions.

2. RELATED WORK

We review the related work in the domain of shopping agents, recommendation systems, and other assistance tools for e-commerce on the Web.

2.1 Shopbots

First shopping agents were developed with the emergence of the World Wide Web. The first shopbot, BargainFinder, developed in 1995 by Andersen Consulting [8], enabled consumers to compare prices of music CDs sold over the Internet. The engine required merchants to register with the shopbot and present their HTML layouts. Unbiased price comparisons quickly turned retailers away from the shopbot. A solid survey of early shopbots can be found in [9]. Collaborative filtering – a community-organized recommendation

[2]We formally define a commercial page later.

[3]For example, Microsoft's MSN consistently updates a list of over 16 million names of products sold on the Web.

system – for shopping has been deployed in numerous shopping agents such as Ringo [10] (later augmented into FireFly [11]) and Amazon.com.

The ShopBot was an agent developed at University of Washington with an aim to parse software shopping sites provided by a consumer in order to find the most inexpensive merchant [12]. ShopBot used artificial intelligence techniques to parse form-based HTML documents and extract features such as product name and price. While BargainFinder had to be hand-tailored to browse the commercial pages of a merchant, ShopBot was one of the first engines that used automated techniques to achieve the same goal. Maes et al. introduced Tete@Tete, a brokering agent dispatched by a consumer or merchant into the digital bazaar to autonomously represent their best interests [11]. While this technology was focused on pricing it was also capable of parsing and categorizing commercial pages.

The only comparison shopping agents available to consumers that are surviving in the commercial realm are biased, presenting results only from companies with whom they collaborate. Examples include MySimon, DealTime, PriceScan, and others.

2.2 Web-Page Classification

Web-page classification and extraction of information in general terms has been addressed using Bayesian classifiers [13], support vector machines (SVMs) [14], [15], [16], relational learning [17], and mapping-convergence algorithms that do not require negative data for training [18]. Different approaches to this problem have been taken from the perspective of assembling and consulting user interest [19], [20], clustering [21], and relying upon interface agents that monitor user actions such as Letiza [22] and WebWatcher [23]. Detectors that target the general classification case typically do not perform on par compared with specialized detectors – hence, we do not review this body of work in detail.

Web-page classifiers concentrated on commercial pages have been mostly developed using "wrappers," HTML templates that map to commercial pages by a specific vendor, and thus reveal information that is distinct for this page. An example of such a system is MORPHEUS, a comparative shopping agent proposed in [24].

2.3 Recommendation Systems

Finally, recommendation systems within the realm of e-commerce and shopping agents have been proposed in several publications [26], [27], [28], [29], [30], [31], [32]. These systems can be deployed within our proposed system as an additional feature – as our proposal revolves around an efficient interjection of the on-line shopping pattern, it could be a facilitator for recommendation systems.

Our proposal differs from related work as it entails a novel point of deployment for the shopping agent which imposes a strong requirement for low false positives. Regardless of application, the proposed detector is built around a novel classification paradigm for Boolean spaces that shows great promise due to its performance, speed, simplicity and intuitive and potentially interactive nature of training.

3. PROBLEM AND MODEL DEFINITIONS

In this section, we define the problem of commercial page detection and present our novel statistical model used for probabilistic decision making.

3.1 Detection of Commercial Web-Pages

INSTANCE: Web-page H in hypertext format, e.g., HTML or XML.
QUESTION: Does H sell explicitly exactly one product P, i.e., can the user by any direct action defined in H, add P to a shopping cart that belongs to the merchant Web-site presenting H?

From the above definition, the question is answered affirmatively even in the case when H, which sells P, and the shopping cart are not hosted within the same Web-site. In addition, pages that sell more than one product are not considered as a target for our application. Typically, pages where users can purchase more than one product, have distinct hyperlinks for each individual product being sold. By following a specific hyperlink, the user usually reaches a Web-page dedicated to offering only one product.

3.2 Product-name Extraction

INSTANCE: A commercial Web-page page H in hypertext format, e.g., HTML or XML, that sells a single product P. P is unknown to the detector but exists as substring in H.
QUESTION: What product is being sold in H?

Certain commercial Web-pages contain product names exclusively encoded in an image and thus, unavailable as a substring in H. As we did not consider optical character recognition in our system, at this point we do not pertain to provide a semantically correct answer to the posed question in this specific situation. In this case, our tool would return a best-effort candidate list $\mathbb{P}$, which is not likely to be resolved at the server, or in certain cases an empty candidate list $\mathbb{P} = \emptyset$.

Note that due to the semantic nature of this question, it is hard to answer it exactly without some prior knowledge. For example, humans have no problems answering this question; otherwise the point of a commercial Web-page would not exist. Isolating the brand name, the actual product code, decoding related abbreviations in the product code are tasks relatively simple for most humans – however difficult to achieve for algorithms without proper priors.

To simplify the problem, we demand that the algorithm returns a list of candidate names that could be resolved as P at a server which owns a list of all product names sold on the Internet. Such lists are not hard to distill by crawling Websites of known merchants and using their Web-page layouts to extract the desired data or simply by browsing a database of commerce bar-codes.

Finally, for our decision making we did not use the two-dimensional layout of the input hypertext. This layout is always computed by browsers for display purposes, however in many cases it is hard to reach as browser APIs for this task are not clearly defined. If available, this information could improve the decision making as locations of key commercial actuators in H such as product name and image, add-to-cart button, and price are typically correlated. For simplicity and speed, our detectors operate only on the hypertext without requiring access to any of the files pointed to by the hypertext.

4. A PROBABILISTIC DECISION MODEL OVER A BOOLEAN SPACE

In this section, we present in detail the detectors developed for the problems outlined in Section 3. We first focus

on the commercial page detector. To a large extent, this detector is crucial for the proposed system. We speculate that its rate for false positives would dictate user adoption.

Let's consider a cardinality-L set of heuristic functions $\mathbb{H} = \{h_1, \ldots, h_L\}$ as an input to our detection algorithm $\mathcal{D}$. The algorithm outputs an estimate on a specific Boolean hypothesis $B \in \{0, 1\}$ (i.e., {false, true}), where B depends upon $\mathbb{H}$, $B = \mathcal{D}(\mathbb{H})$. For now, we assume that each heuristic h_i(web page w) $\in \{0, 1\}$ takes as in input a pointer to a Web-page w and outputs a Boolean value. An example of such a function could be a test whether a Web-page contains the string `Price:`. We describe the heuristic functions that we have used in our experiments in Section 5. When computing $\mathcal{D}\left[\mathbb{H}(\text{web page } w)\right]$, we define $\mathcal{D} : \{0, 1\}^L \rightarrow \{0, 1\}$ as a Boolean function over the Boolean outputs of individual functions in $\mathbb{H}$, where each of these functions is fed with the same argument w. Consequently, as illustrated in Figure 2, we use this notation to define a detector $\mathcal{D}$ that for a given Web-page w establishes whether it is commercial or not.

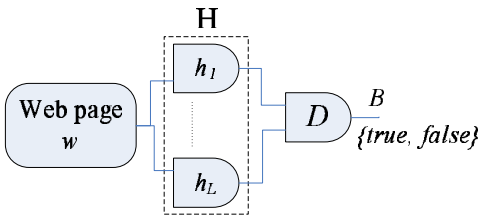

Figure 2: An illustration of the functions involved in the decision making model.

The algorithm $\mathcal{D}$ is trained using a positive and negative dataset, i.e., collection of Web-pages with a priori knowledge on whether they result in $B = 1$ (positive) or $B = 0$ (negative). We denote these input sets as $\mathbb{T}_P$ and $\mathbb{T}_N$ and their cardinalities as L_P and L_N respectively. When $\mathbb{H}$ is applied to all Web-pages from both the positive and negative dataset, we obtain two multi-variable Boolean functions:

$$f_P = \bigcup_{\forall t \in T_P} \{h_1(t), \ldots, h_L(t)\} \quad (1)$$

$$f_N = \bigcup_{\forall t \in T_N} \{h_1(t), \ldots, h_L(t)\}. \quad (2)$$

Just as in traditional Boolean logic, we use "don't cares" to group points in $\{0, 1\}^L$ into terms, e.g., 011 and 001 are grouped as $0*1$ using a "don't care" symbol $*$. Detailed review of Boolean logic can be found in [33].

Functions f_P and f_N represent two sets of distinct points grouped into potentially overlapping terms in $\{0, 1\}^L$. If the heuristics are discriminatory with respect to the desired hypothesis test, the intersection of $f_P \cap f_N$ should be either an empty set or a set with small cardinality. The ratios:

$$\varepsilon(T)_P \equiv \frac{||f_P \cap f_N||}{||f_N||} \quad (3)$$

$$\varepsilon(T)_N \equiv \frac{||f_P \cap f_N||}{||f_P||} \quad (4)$$

define the probability of a false positive, $\varepsilon(T)_P$, and false negative, $\varepsilon(T)_N$, within the training set.

Our objective is to extrapolate our decision making functions over the undecided part of the Boolean space $U = \{0, 1\}^L - (f_P \cup f_N)$ using heuristics, i.e., expand both f_N and $f_P - (f_P \cap f_N)$ so that $U = \emptyset$ while minimizing the probability of false positives on the set of actual experimental data. We denote the resulting bipartitioning of $\{0, 1\}^L$ as f'_P and f'_N. We want to minimize false positives at the cost of increased false negatives. Thus, we assign the intersection $f_P \cap f_N$ to f'_N and partition the remainder of the Boolean space in conservative fashion when expanding upon f'_P. Once the expansion process is completed, we set $\mathcal{D} = f'_P$. This objective is illustrated using Figure 3.

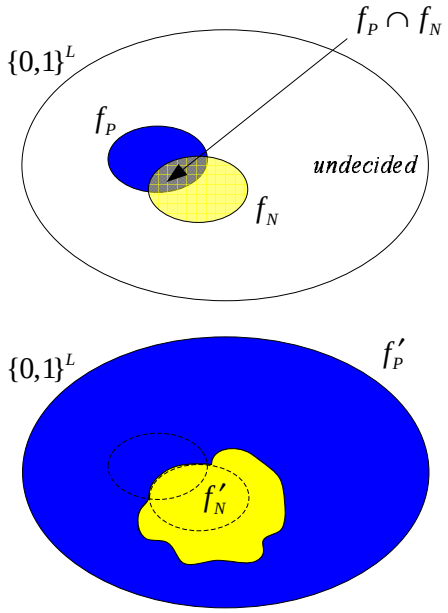

Figure 3: An illustration of bipartitioning a binary space into two complementary functions f'_P and f'_N.

For our application, we would want to accomplish as low as possible false positive rates (hopefully below $\varepsilon_P < 10^{-3}$) while achieving reasonably low false negative rates ($\varepsilon_N < 0.2$). Such results offer great compromise between: *a*) *user convenience* – not interrupting users due to false positives and *b*) *system functionality* – the fact that false negatives do not matter that much in our system. Since consumers are likely to shop while browsing the Web back and forth between the results of a search engine and commercial pages, one can notice that the detector has several chances to display the comparative shopping list. The probability that the detector does not provide the comparative shopping list after M visited commercial pages, is equal to ε_N^M.

4.1 Expansion of Boolean Functions

We first introduce the notion of directional Hamming distance (dHd) $d(a, b)$ between two terms $a = \{a_1, \ldots, a_L\}$ and $b = \{b_1, \ldots, b_L\}$ where $a_i, b_i \in \{0, 1, *\}$ using Table 1.

From Table 1 we see that the distance metric is such that it measures the probability of a_i not being included in b_i assuming 0s and 1s occur equiprobably – which results in its directionality. The dHd of L-dimensional terms is computed as $d(a, b) = \sum_{i=1}^{L} d(a_i, b_i)$.

a_i	b_i	$d(a_i, b_i)$	a_i	b_i	$d(a_i, b_i)$
0	0	0	0	$*$	0
0	1	1	1	$*$	0
1	0	1	$*$	0	0.5
1	1	0	$*$	1	0.5
			$*$	$*$	0

Table 1: The definition of directional Hamming distance for two Boolean scalars.

4.1.1 Term Merger

Now we present a simple heuristic for expansion of Boolean functions. It is predominantly based upon the fact that two terms from f_X that have small dHd could be encompassed in a container term if it does not have a significant intersection and/or is at minimal dHd with the opposite function $f_{\neg X}$.[4] By merging terms with a small dHd d, we inherently render several, say K, heuristics in the functional locality of these terms as "don't cares," where $K \sim d$. Here directionality is important as it allows for better granularity of decision making while merging terms.

Term Merger
Input: terms $a, b \in f_X \subset \{0,1\}^L$
if $\min[d(a,b), d(b,a)] < \vartheta_0$
for $i = 1, L$
$c_i = a_i \cup b_i$
(✠) **if** $\min_{\forall t \in f_{\neg X}} d(c,t) > \vartheta_1$
Replace $\{a, b\}$ in f_X with c

Table 2: Function that merges two terms a and b that belong to a common expanding function f_X.

Table 2 defines a simple algorithm that merges two terms a and b that belong to a common function f_X under expansion. When growing f_X, we consider all possible term pairings $\{a, b\} \in f_X$. Constants ϑ_0 and ϑ_1 are used as thresholds in the merger process; the first as a limit on the maximum dHd between two terms and the second as a limit on how closely we would allow c to grow with respect to the opposite function $f_{\neg X}$. Thus, ϑ_1 directs how conservative we want to be while growing f_X – in our application we are particularly concerned of growing f_P, thus we use a higher ϑ_1 when expanding f_P. Alternatively, if we allow aggressive growth of f_X, the test marked with ✠ could be replaced with another one: **if** $||c \cap f_{\neg X}|| < \vartheta_2$, where ϑ_2 is another constant that limits the cardinality of the intersection between c and $f_{\neg X}$. In our application, we could allow for such expansion of f_N on the account of reducing f_P for the overlap $f_P = f_P - c \cap f_P$.

4.1.2 Q out of R

The aforementioned technique is effective when dependencies among heuristics can be efficiently described using common formats for presenting Boolean functions such as the algebraic normal form (ANF), e.g., Boolean sum of products. This is not always the case. One particularly useful expansion function – "Q out of R" – where any Q out R variables that are true, yield a consolidation of a specific signal (to true or false) used further in the detection process. Representing such a binary function is cumbersome using ANF. Thus, we opt to create additional functions in $\mathbb{H}$ which aggregate such signals via integer summation of binary outputs from existing heuristic functions:

$$g_R^Q(\mathbb{H}_R, w) \equiv \begin{cases} 1, & \sum_{\forall h \in \mathbb{H}_R} h(w) \geq Q \\ 0, & \text{else} \end{cases} \quad (5)$$

where w is an input Web-page and $\mathbb{H}_R$ is a subset of R heuristic functions from $\mathbb{H}$. The new heuristic g_R^Q is then applied to both datasets $\mathbb{T}_P$ and $\mathbb{T}_N$. At that point we are interested in the separation effect (via Eqn. 3 and 4) that g_R^Q alone has on the two sets. If it yields satisfactory separation, g_R^Q is included in $\mathbb{H}$ which then increments the dimensionality of the considered Boolean space.

4.1.3 Putting It Altogether

We apply the techniques presented in the previous two subsections in concert to heuristically find a solution to the overall problem. The overall training process can be done in numerous ways, even manually in an interactive fashion as each step gets validated with quantifying the overlap/minimum distance between the expanding f'_P and f'_N. Still, we propose a simple two-phase automated greedy algorithm for extrapolation of these functions.

In the first phase, we extrapolate f'_N aggressively with a small ϑ_1. We may even allow certain overlap with f'_P which immediately updates $f'_P = f'_P - (f'_P \cap f'_N)$. In this phase we try expanding upon each pair of terms in f'_N.

In the next phase, we extrapolate f'_P conservatively with a lower/higher ϑ_0/ϑ_1 parameter set for any term coupling in f'_P. Then, we identify candidate heuristic subsets for combining heuristics using the "Q out of R" method. One greedy strategy is:

$$\varrho = \frac{1}{||\mathbb{T}_P||} \sum_{\forall h \in \mathbb{H}_R} \sum_{\forall t \in \mathbb{T}_P} h(t) \gg \frac{1}{||\mathbb{T}_N||} \sum_{\forall h \in \mathbb{H}_R} \sum_{\forall u \in \mathbb{T}_N} h(u), \quad (6)$$

where $\mathbb{H}_R$ includes a random subset of R functions from $\mathbb{H}$ which yield only several 1s across $\mathbb{T}_P$ and nearly no 1s across $\mathbb{T}_N$.

If the requirement from Eqn. 6 is met, we estimate $Q_0 \approx \varrho$ and find the best aggregation function with the "Q out of R" method among several values for Q; all of them close to Q_0. We repeat this process for several random subsets $\mathbb{H}_R$, say until there are no inclusions of additional heuristics into $\mathbb{H}$ for M consecutive iterations. Finally, we include the new heuristics into the decision making process by repeating the first phase over the expanded heuristic set and setting $\mathcal{D} = f'_P$ at the end.

4.2 A Probabilistic Version

In this subsection we introduce a set of modifications to the introduced deterministic decision model whose objective is to introduce probabilistic methods as a qualitative improvement. We start by introducing a supplemental function $h' : \{\text{web page } w\} \rightarrow \mathbb{R}$ designed to return a probability that the heuristic function $h()$ returns a Boolean true over the same argument: $h'(\text{web page } w) \equiv \Pr[h(w) = 1] \in [0, 1]$. Using $h'()$, we introduce a measure of certainty about the input to the decision model. Let's denote the set of such functions as: $\mathbb{H}' = \{h'_1, \ldots, h'_L\}$.

[4] $X \in \{P, N\}$. If $X = P$, then $\neg X = N$.

Next, we alter the definition of the decision circuit $\mathcal{D}$. Its objective is redefined as to produce the probability $b \in \mathbb{R}$ that the resulting Boolean decision is true based upon the probabilistic input $b = \mathcal{D}'(\mathbb{H}') \in [0, 1]$. Similarity to the deterministic case is preserved by the introduction of probabilistic `NOT`, `AND`, and `OR` gates. Their output is equal to $r = \texttt{NOT}(p) = 1 - p$, $r = \texttt{AND}(p, q) = pq$, and $r = \texttt{OR}(p, q) = 1 - (1 - p)(1 - q)$ respectively, where p and q are probabilities/gate[5] inputs and r is the output probability. If $h'_i(w) = 0.5$, we set $h_i(w)$ to a random Boolean value. By using exclusively such gates to construct $\mathcal{D}'$, we enable propagation of probabilities throughout the logic network and compute the probability that a certain decision is true based upon a collection of input probabilities.

Since the design of the decision network $\mathcal{D}'$ depends only upon the expected Boolean values for each variable (i.e., output of a heuristic function in $\mathbb{H}'$) in the system, the training set is represented using Boolean functions f_P and f_N. The growth of f_P and f_N is performed in deterministic fashion as presented in Subsection 4.1. The constructed decision network now outputs a probabilistic support quantifier, $\mathcal{D}'(\mathbb{H}'(w)) \lesseqgtr \alpha\theta$, which is compared to a threshold:

$$\theta = \mathcal{D}'(h'_1, \ldots, h'_L), (\forall h'_i \in \mathbb{H}')h'_i = \frac{1}{2}. \quad (7)$$

where α is a real scalar. As opposed to the actual Boolean function f'_P which controlled the error rates in the deterministic version, here, it is predominantly $\alpha\theta$ that controls the error rates.

For brevity and simplicity, in this paper we do not formally present the characteristics of the developed model – we focus on the engineering details as well as the empirically observed detection performance.

Finally, note that although the model is relatively simple it is, to the best of our knowledge, novel. The fact that the training datasets are modeled as Boolean functions and first expanded using heuristics to bipartition the multi-dimensional Boolean space of interest, is distinct to existing models that operate in such spaces such as deterministic and probabilistic Boolean networks [36], [37], deterministic classifiers with Bayesian extensions [38], support vector machines [39], and Bayesian networks [40]. A survey of text classifiers is presented in [41].

5. IMPLEMENTATION

In this section, we present a prototype implementation of the client-side of the proposed platform. We did not focus on the "server"-side technologies as they could be implemented in several ways. For example, it could be a fully automated system that ties a Web-crawler with a commercial page detector[6] and a commercial data (e.g., product name, price) extractor. Or, it could be a Wikipedia-style community-generated system where consumers maintain a global list of greatest offers on the Web. The requirements for both implementations are substantially different and require distinct solutions. In this paper, we focus on the enabling technology for the system, the client side algorithms.

[5]With a fan-in equal to two.

[6]The requirements imposed upon this detector are different than the set of requirements imposed on the detector presented in this paper. The detector augmented into the Web-crawler would be focused on as low as possible false negatives.

Our client-side prototype implementation consists of three main modules: an HTML Parser, a Commercial Page Detector (CPD) and a Product Name Extractor (PNE). In the remainder of this article, we review each component in more detail.

5.1 Parser

The main objective of parsing HTML files in our application is data extraction. We built a stack-based parser based upon well documented prior work [42] which focuses on extracting specific features that we can use for detection of commercial pages. It is a two-pass parser which in the first stage creates a n-ary tree that structurally represents the HTML tags and actual content, and in the second pass filters out information that does not contribute to the extraction process such as HTML comments, certain multimedia specifications and proprietary formats (e.g., flash clips). Each node in the tree is generated by a tag container (e.g., `<DIV>` *arbitrary content* `</DIV>`), where the subnodes branch and depend upon the encompassed content. Each tag is denoted with its attributes, attribute values, and associated text. For instance, for an HTML context:

`<TD><B CLASS="price"> $99.95 </B></TD>`

we create a root node `<TD></TD>`, with a subnode `<B></B>` with attribute `CLASS`, its value `price`, and associated text `$99.95`.

Our parser is also capable of fetching metadata for specific multimedia content displayed on the HTML header. We specifically search for clues like product description or identification available in the well-known `<META>` tag. The metadata is embedded into the n-ary tree as tag's attribute-value pairs. The structured content is now well positioned for feature extraction and analysis and is used as an input to our commercial page detector.

Finally, note that in this work we do not use HTML layout information, i.e., data about the positioning of HTML constructs once a Web-page is displayed. For example, the relative positioning of a paragraph of text, or relative distance between an image and table, for instance. While this is certainly possible and would likely improve detection performance, we chose not to use it for brevity, simplicity, and dependance upon browser's API. However, including these parameters and observing detection performance is an interesting future work.

5.2 Commercial Page Detector

We explored various statistical properties of the commercial pages and the parameters which distinguishes a commercial page from a non-commercial one.

A commercial page (such as the one presented in Figure 1) which is selling a single product P has a typical layout:

- a product photo which typically dominates the real-estate of the Web-page layout (excluding a potential header image),
- a product description, typically represented as a container with text that includes words such as specifications, dimensions, etc.,
- pricing details, typically an isolated string that follows common pricing formats with an associated string `Price:` in the same or neighboring layout container (i.e., `<DIV>`, `<TD>`),

- shipping information, commonly represented with synonyms for delivery options as well as a form used to accurately compute shipping costs, and, most importantly, and
- an "Add-to-Shopping-Cart" (ATC) button. This button is the key component in identifying a commercial page as it is closely tied to its definition presented in Section 3.

In most cases, an ATC button is either an image or a button, e.g.:

```
<IMG SRC="addtocart.gif" ALT="Add To Cart">
```

Clearly an ATC button can be displayed in an HTML page using numerous variations such as "Add-to-bag" or "Add-to-shopping-basket." To address this issue for this and other heuristics, we built extensive dictionaries for keywords critical for our detector, and their possible combinations. These dictionaries were manually verified due to the relatively small volume of information as well as the challenge of semantic validation of all synonyms.

5.2.1 Set of Heuristics – $\mathbb{H}$

As opposed to using a generic text classifier [41], due to a manageable volume of data, we decided to identify the set of heuristics $\mathbb{H}$ through observation of statistics and general characteristics of commercial and non-commercial Web-pages in our training dataset. For brevity, in this paper we review only ten groups of these heuristics:

$h_1()$ Presence of an "Add-to-Shopping-Cart" button;

$h_2()$ Pricing details – a label describing price and related information about the product;

$h_3()$ Quantity input – a text-form input field where a customer can specify the number of products she wants to purchase;

$h_4()$ Product description – information presenting the product specs, description, highlights or other specific details;

$h_5()$ Presence of a "View-Cart" button;

$h_6()$ Presence of a "Checkout" button;

$h_7()$ Availability details – information about product availability (e.g., "in stock," "store pickup," etc.);

$h_8()$ Shipping information – details about shipping/delivery of the product;

$h_9()$ Recommendation – suggestion of related products (e.g., "People who bought this product also bought");

$h_{10}()$ SKU code – existence of a Stock Keeping Unit identifier in the page's body.

Button-related heuristics evaluate the following HTML constructs: (i) within <FORM> tags, the focus is on <INPUT> subtags or targets of ACTION attributes; or (ii) within an embedded image, the focus is on its filename and/or directory name, on the associated attribute ALT text[7], or on the target URL when the image is assigned a hyperlink.

Expressions = { "add to cart", "add to shopping cart", "add to basket", "add to bag", "add to order", "buy it", "add to shopping bag", "store pickup", "buy it now", "order now", "add to shopping list", "add item to cart", "in stock", "also bought", "buy now", "who bought", "buy together", "store locator", "ships within" }
Keywords = { "add", "show", "view", "cart", "basket", "bag", "buy", "order", "now", "item", "product", "description", "details", "features", "specifications", "highlights", "information", "info", "characteristics", "sku", "mgf", "mfr", "part", "shipping", "delivery", "price", "quantity", "qty", "checkout" }
Filename = { "a-t-c", "a_t_c", "/atc.", "buynow", "addtocart", "-atc.", "_atc.", "add_to_cart", "add2cart", "add-to-cart", "button_buy", "addtobag", "add2bag" }

Table 3: Part of the dictionary of words used to facilitate the detection of commercial Web-pages.

The majority of other heuristics can be simply detected by searching for dictionary keywords in the associated text of specific tags such as <B>, <TD> or <A> in a process similar to the just described.

Table 3 displays a list of expressions, keywords, and filename substrings that represents a part of the dictionary of words we use in the deployed heuristics $\mathbb{H}$. These heuristics either search for exact *expressions* and/or arbitrary combinations of *keywords* in the text presented to the user or HTML attributes, or search for *filename substrings* in encompassed files, e.g., images, and URLs.

5.3 Product Name Extraction

As discussed in Section 3, once a Web-page is detected as a commercial one, we report a set of candidate strings that could be used to resolve the product name. The candidate strings are extracted from the Web-page while browsing the n-ary tree for input that triggers the heuristics from $\mathbb{H}$. Thus, both solvers use a single pass over the tree to achieve their objectives. Our PNE detector relies on the observation that the product name is a short isolated string which is both repeatedly specified in the Web-document as well as presented in isolated layout containers. We present a list of locations in a generic HTML file where we focus our attention in the quest to identify the product name:

1. the product name is typically a substring of the page's title.
2. the product name is frequently specified in header tags across the HTML file (e.g., <H1> or <H2>) or in highlight tags (e.g., <B> or <STRONG>).
3. the product name is sometimes included in the descriptive text associated with the image which illustrates the product, e.g.:
   <IMG SRC ="nikon-d300.jpg" ALT="Nikon D300">.

[7]Text displayed by default in case image is not available or prevented from being displayed.

4. The product name is included in the Web-page's meta-data, e.g.:
<META NAME="PRODUCTNAME" CONTENT="NIKON D50">.

We specifically did not look into text font-size as one may anticipate that product names are typically presented in large and/or bold font. We chose not to pursue this heuristic because font information is commonly defined in CSS files that are frequently difficult to parse. Thus, we did not deploy this possibly helpful heuristic.

We apply a variation of the Longest Common Substring (LCS) algorithm [35], to the strings extracted using the aforementioned heuristics. The algorithm typically outputs a set of several substrings, $\mathbb{P}$, that are reported to the knowledge server. Note that the extractor may also extract strings with information which is non-related to the product name – in all experiments we have run and verified semantically, the algorithm never returned information that could be interpreted as pointing to two distinct products. Finally, the knowledge server, due to the fact that it stores a list of all products currently sold on the Web, resolves from $\mathbb{P}$ the actual product name P using a simple matching procedure.

5.3.1 Adversarial Formatting of Commercial Pages

One of the consequences of an unbiased shopping engine is dissatisfaction of online retailers. In order to avoid being parsed and catalogued at the knowledge server and/or detected as commercial pages at the client, retailers may resort to adversarial formatting of their sales Web-pages. Another potential target of adversarial formatting is to fool the Web-crawler into detecting a certain price, then offering another price to the end-consumer. While scripting languages today offer endless possibilities to achieve these tasks – the existence of a popular unbiased shopping engine such as the one we are proposing, may repel consumers from retailers that do not cause an expected reaction by their shopping plug-in. We arguably speculate that such a reaction would produce similar end-consequences to high-margin on-line retailers at the benefit of consumers.

6. EXPERIMENTAL RESULTS

We have implemented a prototype of the technology described in this paper in order to evaluate its error rates. In this section, we overview the experimental benchmark – a dataset that encompasses large number of commercial and non-commercial Web-pages, then review the detection performance of our algorithm on this dataset.

6.1 The Benchmark

We collected a significant corpus of Web-pages to train and test our CPD. The data was collected from real conventional shopping Web-sites. For training, we used a dataset of 10^3 commercial and 10^3 non-commercial pages obtained using random product search from MSN Shopping and Live Search, respectively. We report our detection results on a **different** set of 9871 commercial and 9224 non-commercial Web-pages collected via the same process. We automatically filtered Web-pages containing frames, hypertext embedded images, flash, or abstruse Web-pages (e.g., spam).

The positive datasets (i.e., the commercial datasets) were generated from a list of arbitrary product brands such as "Sony" or "Xbox," queried into the MSN Shopping product engine. The training dataset f_P was constructed by querying MSN Shopping with a list of 6 brands. The resulting 10^3 product Web-pages were downloaded from 116 distinct merchant Web-sites. On the other hand, the positive benchmark dataset was obtained using the same procedure and a list of 48 brands resulting in 9871 product listings from additional 27 distinct on-line retailers (total of 143).

The negative datasets (i.e., the non-commercial datasets) were generated by querying an existing search engine (Live Search) with a list of keywords unrelated to shopping (e.g., "politics," "wiki," "blog," etc.). The negative dataset used for training, i.e., f_N, was generated using a list of 32 distinct keywords that resulted in 10^3 non-commercial Web-pages from 597 different Web-sites. The negative benchmark dataset was obtained using a list of 272 keywords that pointed to 9224 non-commercial Web-pages from 6409 different Web-sites. The negative datasets included pages which described (e.g., review pages, discussion groups and forums, etc.) but did not sell any products.

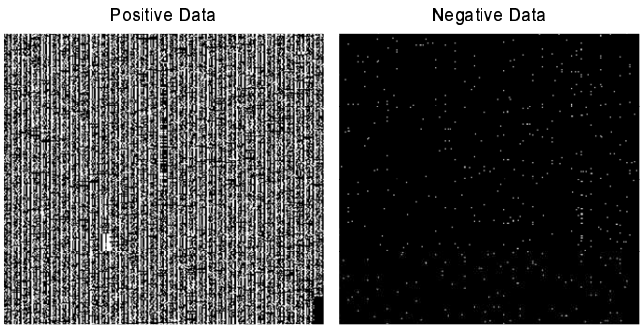

Figure 4: Illustration of the 10-dimensional Boolean input $\mathbb{H}$ for positive and negative data in our benchmark dataset. A black pixel in the image symbolizes a Boolean zero. The original matrices of size 9871×10 for positive data and 9224×10 for negative data were folded into images so that the vertical dimension is 300 pixels.

6.2 Results

Our detector processed both datasets as HTML files. It started from a $L_0 > 100$ number of heuristics that were used to derive the Boolean output to the resulting $L = 10$ heuristics presented in Subsection 5.2.1. The Boolean output of these heuristics for both positive and negative data is presented in Figure 4. In the images, a black pixel symbolizes a Boolean zero. The original matrices of size 9871×10 for positive data and 9224×10 for negative data were folded into bitmaps so that the vertical dimension is 300 pixels. In Table 4, we quantify the number of occurrences of true Boolean output from heuristics in $\mathbb{H}$ for our benchmark of 9871 commercial and 9224 non-commercial Web-pages. From Figure 4 and Table 4, one can observe strong differences between the two datasets – a result that points to the fact that low error rates are achievable in our system.

To that extent, we report our detector's performance using the receiver operating characteristics (roc) diagram in Figure 5. Although numerous configurations of our detection algorithm $\mathcal{D}$ are possible, we identified only a few datapoints on the roc-curve. Each point is specified with a specific Boolean function $\mathcal{D}$. To obtain points T1 and T2 during algorithm training, we used the following "Q out of R" term

merger heuristics:

$$\{Q, R\} = \{\{1, 5\}, \{2, 6\}\}, \quad (8)$$

respectively. We applied the detection engines obtained at training data-points T1 and T2 to our benchmark dataset to obtain data-points B1 and B2 respectively. We obtained B3 by applying the same parameter setup from T2 to $\{Q, R\} = \{4, 8\}$. We also present the performance obtained from two optimized designs of $\mathcal{D}$ for both the training and benchmark datasets marked with $\Diamond$.

Table 4: Number of occurrences of true Boolean output from heuristics in $\mathbb{H}$ for our benchmark of 9871 commercial and 9224 non-commercial Web-pages.

$\mathbb{H}$	Commercial	Non-commercial
$h_1()$	9373	266
$h_2()$	7709	40
$h_3()$	1966	3
$h_4()$	4273	9
$h_5()$	2205	33
$h_6()$	1518	53
$h_7()$	7430	11
$h_8()$	5410	71
$h_9()$	333	3
$h_{10}()$	1188	0

We present the error rates for both the training set ($f_P \cap f_N$) as well as the collected benchmark dataset. From Figure 5 we observe that the performance achieved over our training data (marked with $\square$) was expectedly slightly better then the performance over the benchmark data (marked with $\bigcirc$). We point to a specific data-point (B3) on the benchmark performance curve where our detector achieved a probability of a false positive of 0.0002 (non-commercial pages detected as commercial ones) and a probability of a false negative of 0.089 (commercial pages detected as non-commercial ones). False positives occurred due to surprising content/formatting of Web-pages such as the following example from our negative benchmark dataset:
`http://www.topplebush.com/bushstore.shtml`.

Our product name extractor performed at 100% accuracy as we were able to extract a list of candidate product names for all commercial pages in the positive dataset.

We report that the execution run-times for the developed CPD&PNE depended upon Web-page size. For instance, one of the largest Web-pages in our dataset, a Circuit City shopping Web-page, was parsed[8] in 125.5ms and our CPD&PNE computed $\mathcal{D}$ and P in less than 1 millisecond.

From the obtained results, we speculate that the proposed CPD and PNE detectors would perform well both in the setting of the proposed comparative shopping application (point B3) as well as embedded within a Web-crawler whose task is to browse commercial pages on the Web (point BP).

6.3 Comparison to SVM

In this subsection we elaborate on the performance comparison of our learning algorithm with respect to a support vector machine (SVM) [43]. SVMs are a set of related supervised learning methods used for classification. A special property of SVMs is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers. SVMs map input vectors to a higher dimensional space where a maximal separating hyperplane is constructed. Two parallel hyperplanes are constructed on each side of the hyperplane that separates the data. A threshold hyperplane maximizes the distance between the two parallel hyperplanes. One of the advantages of comparing a learning algorithm to an SVM is the fact that there exist studies that relate SVM performance to other existing classifiers [44].

[8]Includes computing $\mathbb{H}'(w)$.

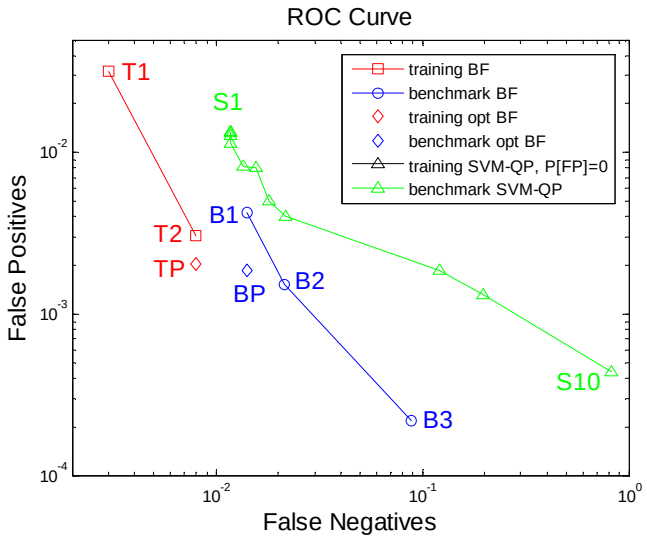

Figure 5: Receiver operating characteristic curve for the novel Boolean classifier (BF) and an SVM-QP classifier (SVM) when applied to our training and benchmark datasets.

We compare our approach to SVM with quadratic programming (QP). We used the MatLab SVM toolbox publicly available from: `http://asi.insa-rouen.fr/enseignants/~arakotom/toolbox/index.html` [45]. We fed the SVM engine with the following input parameters and options: ∞ as the bound on the lagrangian multipliers, 10^{-6} as the conditioning parameter for the QP method, and we used a gaussian kernel. The input data, illustrated in Figure 4, was fed into both classifiers. The SVM-QP roc curve is illustrated in Figure 5. While its training resulted in zero false positives and 0.004 false negatives, SVM demonstrated relatively sub-par performance when applied on the benchmark data. Points S1 through S5 in Figure 5 illustrate its performance on the benchmark data as we moved the normalized threshold hyperplane from -0.5 (S1) to 0.99 (S10). We conclude from our experiments that our classifier performed better than on-par on this task with respect to a well-researched SVM classifier. It is beyond the scope of this paper to further formally analyze the two classifiers.

SVM is known to be a fast classification algorithm. Our detector performed about two orders of magnitude faster on MatLab: an average of 0.03 seconds vs. 1.75 seconds for the 10^3 pages in the training dataset, and an average of 0.25 seconds vs. 15.6 seconds for the 10^4 benchmark dataset. We anticipate that similar performance gap remains in fast

implementations of both technologies. In scenarios where detection is performed on the server, clearly, this substantial advantage in performance is one of the contributions of the proposed learning paradigm.

7. CONCLUSIONS

In this paper, we propose a simple novel classifier that operates over a multi-dimensional Boolean space. Its target application in this work is identification of commercial Web-pages with an emphasis on a low rate of false positives. We enforced such an application from several perspectives. First, existing Web-crawlers that target e-commerce can use it to identify the subspace of the Internet of interest. Second, we point to a novel application where the detector is present at the client and enables an non-intrusive method for reviewing comparative shopping lists. Although here we advocate for unbiased systems, the tool could be used for recommendation and/or advertisement systems (e.g., [46]).

We have shown that on the specific task of commercial page detection, our classifier outperformed a well-researched SVM classifier with quadratic programming. We found our classifier to exhibit two important performance features that make it appealing for both client-side commercial page detection due to low false positives and server-side detection as we found that it was two orders of magnitude faster than the reference SVM.

8. REFERENCES

[1] A.A. Achenbaum and F.K. Mitchel. Pulling Away from Push Marketing. Harvard Business Review, pp.38–40, 1987.

[2] D. Bollier. When push comes to pull: The New Economy and Culture of Networking Technology. Report of the Fourteenth Annual Aspen Institute Roundtable on Information Technology, 2006.

[3] Television Bureau of Advertising. Trends in Advertising Volume. Available on-line at: `http://tvb.org/rcentral/mediatrendstrack/trends/trends.asp?c=2004_2006`.

[4] O. Etzioni and D. Weld. A softbot-based interface to the Internet. Communications of the ACM, Vol.37, no.7, pp.72–76, 1994.

[5] M.D. Smith. The Impact of Shopbots on Electronic Markets. Journal of the Academy of Marketing Science, Vol.30, no.4, pp.442–450, 2002.

[6] P. Markopoulos and J. Kephart. How valuable are shopbots? International Joint Conference on Autonomous Agents and Multiagent Systems, 2002.

[7] J. Bakos. Reducing buyer search costs: implications for electronic marketplaces. Management Science, Vol.43, no.12, pp.1676–1692, 1997.

[8] B. Krulwich. The BargainFinder agent. Comparison Price Shopping on the Internet. In Williams, J., ed., Bots and Other Internet Beasties. SAMS.NET 1996.

[9] P. Jacso. Shopbots: shopping robots for electronic commerce. Online, Vol.22, no.4, pp.14–20, 1998.

[10] U. Doo and P. Maes. Social information filtering: Algorithms for automating "word of mouth." ACM Conference on Human Factors in Computing Systems, pp.210–217, 1995.

[11] P. Maes, et al. Agents that buy and sell. Communications of the ACM, Vol.42, no.3, pp.81–91, 1999.

[12] R. Doorenbos, et al. A scalable comparison-shopping agent for the World Wide Web. International Conference on Autonomous Agents, pp.39–48, 1997.

[13] D. Shen, et al. Web-page classification through summarization. ACM SIGIR Conference on Research and Development in Information Retrieval, pp.242–249, 2004.

[14] S. Dumais and H. Chen. Hierarchical classification of Web content. ACM SIGIR, pp.256–263, 2000.

[15] E.J. Glover, et al. Using web structure for classifying and describing web pages. International Conference on World Wide Web pp.562–569, 2002.

[16] A. Sun, et al. Web classification using support vector machine. International Workshop on Web Information and Data Management, 2002.

[17] D. Freitag. Information extraction from HTML: application of a general machine learning approach. Conference on Artificial Intelligence, pp.517–523, 1998.

[18] H. Yu, et al. PEBL: Web Page Classification without negative examples. IEEE Transactions on Knowledge and Data Engineering, Vol.16, no.1, pp.70–81, 2004.

[19] C. Knoblock and A. Levy. Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments. Stanford University: AAAI Press.

[20] P. Maes and R. Kozierok. Learning interface agents. Proceedings of AAAI, 1993.

[21] L. Ungar and D. Foster. Clustering methods for collaborative filtering. Recommender Systems, 1998.

[22] H. Lieberman. Autonomous interface agents. ACM Conference on Computers and Human Interface, 1997.

[23] T. Joachims, et al. WebWatcher: A tour guide for the world wide web. In Proc. Intl. Joint Conf. on Artificial Intelligence, 1997.

[24] J. Yang, et al. MORPHEUS: a more scalable comparison-shopping agent. International Conference on Autonomous Agents, pp.63–64, 2001.

[25] Q. Shen, et al. Next generation agent enabled comparison shopping. Expert Systems with Applications, Vol.18, no.4, pp.283–297, 2000.

[26] P. Resnick and H. Varian. Recommender systems. Communications of the ACM, Vol.40, no.3, pp.56–58, 1997.

[27] J. Schafer, et al. Recommender systems in e-commerce. ACM Conference on Electronic Commerce, pp.158–166, 1999.

[28] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. Communications of the ACM, Vol.40, no.3, pp.66–72, 1997.

[29] S. Bohte, et al. Market-based recommendation: Agents that compete for consumer attention. ACM Transactions on Internet Technology, Vol.4, no.4, pp.420–448, 2004.

[30] R. Kumar, et al. Recommendation systems: a probabilistic analysis. Journal of Computer and System Sciences, Vol.63, no.1, pp.42–61, 2001.

[31] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE

Transactions on Knowledge and Data Engineering, Vol.17, no.6, pp.734–749, 2005.

[32] Z. Huang, et al. A graph model for E-commerce recommender systems. Journal of the American Society for Information Science and Technology, Vol.55, no.3, p.259–274, 2004.

[33] G. DeMicheli. Synthesis and Optimization of Digital Circuits. McGraw Hill, 1994.

[34] Hyper Text Markup Language Specification. On-line at: `http://www.w3.org`.

[35] D. Gusfield. Algorithms on Stings, Trees, and Sequences: Computer Science and Computational Biology. SIGACT News, Vol.28, no.4, p.41–60, 1997.

[36] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. Journal of Theoretical Biology, Vol.22, pp.437-467, 1969. Boolean Networks.

[37] E.R. Dougherty and I. Shmulevich. Mappings Between Probabilistic Boolean Networks. Signal Processing, Vol.83, no.4, pp.799–809, 2003.

[38] I. Leenen, et al. Bayesian probabilistic extensions of a deterministic classification model. Computational Statistics, Vol.15, pp.355–371, 2000.

[39] K. Sadohara. Learning of Boolean Functions Using Support Vector Machines. International Conference on Algorithmic Learning Theory, pp.106–118, 2001.

[40] J. Pearl. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Francisco, 1988.

[41] A. Hotho, et al. A Brief Survey of Text Mining. LDV FORUM, Vol.20, pp.19–62. 2005.

[42] J. Hammer, et al. Extracting Semistructured Information from the Web. Workshop on Management of Semistructured Data, pp.18–25, 1997.

[43] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, Vol.2, pp.121–167, 1998.

[44] C.M. van der Walt and E. Barnard. Data characteristics that determine classifier performance. Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa, pp.160–165, 2006.

[45] S. Canu, et al. SVM and Kernel Methods Matlab Toolbox. Perception Systèmes et Information, INSA de Rouen, Rouen, France, 2005.

[46] Dealio Inc. Available on-line at: `http://www.dealio.com/about/index.html`.