# Goal-Oriented Web-site Navigation for On-line Shoppers

Daniel Deutch Tova Milo Tom Yam

Tel Aviv University
{danielde,milo,tomyam}@cs.tau.ac.il

## ABSTRACT

Web-sites for on-line shopping typically offer a vast number of product options and combinations thereof. While this is very useful, it often makes the navigation in the site and the identification of the "ideal" purchase (where the notion of ideal differs among users) a confusing, non-trivial experience. This demonstration presents ShopIT (ShoppIng assitanT), a system that assists on-line shoppers by suggesting the most effective navigation paths for their specified criteria and preferences. The suggestions are continually adapted to choices/decisions taken by the users while navigating. ShopITis based on a set of novel, adaptive, provably optimal algorithms for TOP-K query evaluation.

## 1. INTRODUCTION

On-line shopping is extremely popular nowadays, with millions of users purchasing products in shops that provide a Web interface. It is common for on-line shops to offer a vast number of product options and combinations thereof [14, 8]. This is very useful but, at the same time, makes shopping rather confusing. Indeed, it is often very difficult to find the specific navigation path in the site (i.e. a flow of user clicks / choices / actions) that will lead to an "optimal" result, best suiting the needs and preferences of the given user.

Consider for example an on-line store that allows users to assemble computers from a variety of component parts. The store offers various processors, motherboards, screens etc. Consider a user that is interested in buying a cheap computer with Intel processor inside. Suppose that the user can get a good price by first registering to the store's customers club, then passing through some advertisement page that provides such members with discount coupons, and finally buying a certain set of components (including a certain Intel processor) that, when purchased together with the above coupons, yields the cheapest overall price. Clearly, the user might be interested in knowing this information if she is after the deal with the best price. Alternatively, the user may prefer combinations where the delivery time is minimal, or may want to use the experience of others and view the most popular navigation paths (and purchases thereof), assembling an Intel-based computer. If, for instance, the above proposed cheap deal is not among them, the user would more carefully check the quality of the deal's components.

We present here ShopIT (ShoppIng assitanT), a system that assists on-line shoppers by suggesting the most effective navigation paths for their specified criteria (e.g. "Intel processor inside") and preferences (e.g. low price, delivery time, popularity). When the user starts her navigation in the site, she may specify her constraints and her ranking function of interest, and have the system compute and propose (an initial set of) top-k ranked navigation flows, out of these conforming to the constraints. The user then continues her navigation taking into account the presented recommendations, but may also make choices different than those proposed by the system. For instance, ShopIT might have proposed the purchase of a DDR motherboard and a Pentium 4 processor, but the user might have nevertheless chosen an RDRAM motherboard. In this case, new recommendations, consistent with the actual choices made by the user, are automatically computed. Namely, the system dynamically proposes new top-k *continuations* that are up to date with the user's current navigation. The user may similarly modify her selection criteria or preferred ranking.

Several challenges arise in the development of such a system. First, the number of possible navigation flows in a given web-site is not only large but infinite, as users may navigate back and forth between pages. Hence, enumerating and ranking all relevant flows is clearly not an option. Second, it is critical to maintain a fast response time in order to provide a pleasant user experience. Finally, as explained above, the computation must be flexible and adaptive, to account for run-time user choices. To address these needs, ShopIT uses a novel, dynamic and adaptive top-k query evaluation algorithm. Our algorithm can be formally proved to be *optimal* [6] and our experiments show that it is very efficient in practice.

**Demonstration Scenario.** We demonstrate the operation of ShopIT on a Web shop that simulates the on-line computers store of *Yahoo! Shopping* [14]. The shop is based on real-life data obtained from Yahoo! Shopping and the only reason that we use a simulated version is technical: avoiding to commit actual purchases in the site.

First, we shall present to the audience the original "stripped-down" version of the Yahoo! Shopping-like store (without the ShopIT recommendations component) and ask them to shop for component products of their likings. Then, we shall turn on ShopIT and demonstrate how it simplifies and improves the shopping experience. We will show how users specify their interests and their preferences for ranking, and how corresponding proposals for best navigation flows in the store's web-site are presented. Given these initial recommendations, we will illustrate different possible interactions with the system. Specifically, we will consider users that follow the recommended flows and, in contrast, users that make choices different than those recommended by ShopIT or dynamically change their search criteria or ranking preferences. We will illustrate how the system's recommendations are continually adapted to the concrete choices made by users and to changes in their interests/preferences. We will also compare the overall deal price obtained by ShopITto the initial one obtained by the users without it (and emphasize also the saving in the time invested in searching for deals).

As a last part of the demonstration, we allow users to look "under-the-hood" of our system. Specifically, we will show graphical representations of the queries constructed out of the user requests and of possible matching navigation flows, and illustrate main features of our adaptive top-k query evaluation algorithm.

## 2. TECHNICAL BACKGROUND

We provide in this section some background on our model for Web applications and for queries over such applications. The described model serves as the basis for the ShopIT application.

**Web-based Applications.** Our model for Web applications, introduced in [2, 5], abstractly models applications as a set of *(nested) DAGs* - Directed Acyclic Graphs - each intuitively corresponding to a specific function or service [2, 4, 7]. The graphs consist of activities (nodes), and links (edges) between them, that detail the execution order of activities. Each activity is represented by a pair of *nodes*, the first standing as the activity's *activation point* and the second as its *completion point*. Activities may be either atomic, or compound. In the latter case their possible internal structures (called *implementations*) are also detailed as DAGs, leading to the nested structure. A compound activity may have different possible implementations, corresponding to different user choices, link traversals, variable values, etc. These are captured by logical formulas (over the user choices, variable values, etc.) that guard each of the possible implementations. A Web-based application may be *recursive*, where an (indirect) implementation of some activity $a$ contains another occurrence of $a$.

For example, a schematic (partial) description of the model of our on-line computers store is depicted in Fig. 1. The possible implementations of each activity are depicted here as bubbles, attached to the activity node. A customer starts by logging into the system, authenticating and giving her credit card details. Then, she may choose a product (e.g. Motherboard, CPU, etc.) out of a list. In parallel (i.e. by opening a new tab) she may review suggestions for hot deals or buy discount coupons. Upon a choice of product type, she is directed to a page where the different brands that the store offers for this product type are listed, and the corresponding products can be selected. The user can next choose to cancel her selection, to search for more products, or alternatively to exit and pay. Note the recursive nature of this application: users can re-start the search (i.e. call $S1$) an unbounded number of times, each time purchasing another item.

**Navigation Flow.** A navigation flow (in a given Web application) corresponds to concrete choices of implementations for the compound activity in the application. For instance, a possible navigation flow in our computer store is one where the user first reviews the possible deals, then chooses to purchase an Intel Motherboard, subsequently cancels her choice, buys a discount coupon and selects a CPU by HP, etc. Note that the number of possible navigation flows may be extensively large even for relatively small-scaled applications. In fact, for *recursive* applications such as the one depicted in our example, the number of possible flows may be *infinite* (as there is no bound on the number of times that a user may go through $S1$.)

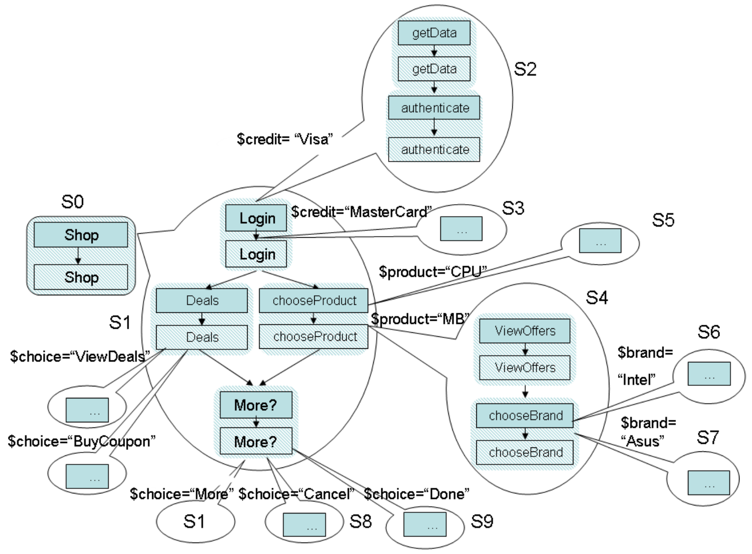


**Figure 1: Web-Based Application (Schematic Representation)**

**Ranking.** The rank of navigation flows is derived using two functions, namely, $cWeight$ and $fWeight$. Function $cWeight$ assigns a weight to each single (implementation) choice within a flow, depending on the course of the flow thus far and the objective that the user wishes to optimize, e.g., monetary cost or likelihood. Function $fWeight$ aggregates the per-choice weights in a single score for the entire flow. As an example, it can sum up the individual monetary costs to compute the total cost of the flow, or it may multiply the per-choice likelihoods to derive an overall likelihood. Notions such as discounts and combined deals are factored within the $cWeight$ function, which is aware of flow history. We do not place any restrictions on $fWeight$ except that it satisfies the standard notion of monotonicity.

**Top-k query results.** The users' search criteria are modeled by queries. These use *navigation patterns*, an adaptation of the tree-patterns, found in existing query languages for XML, to nested application DAGs [2]. Our top-k query evaluation algorithm gets as input the schematic representation of the application, the user query and the chosen ranking metric, and efficiently retrieves the qualifying navigation flows with the highest rank. The algorithm operates in two steps. First, it generates a refined version of the original application representation, describing only flows that are relevant to the user request. Then it greedily analyzes the refined representation to obtain the best-ranked flows. (The reader is referred to [6] for details.) The choices made by the user throughout the navigation are modeled as additional constraints/relaxations to the original query, and an efficient adaptive evaluation technique is employed to update the query result.

**Related Work.** We end this section with a short review of related work. We have already mentioned popular shopping web-sites such as [14, 8]. Unlike ShopIT, their ranking mechanism ranks, separately, items in each distinct category, based on built-in specific ranking metric. The global effect of a full navigation flow that may include, e.g., registration to customers clubs, collection of coupon discounts, specific user choices, is not accounted for.

A variety of *Recommender Systems* (e.g. [12, 13]) appear in the literature. However, as mentioned in [1], they provide rather low flexibility, with a recommendation method that is hard-wired and not configurable to fit user needs[1]. These works also typically do

[1]An exception are OLAP-based approaches that are still considered an open research problem.

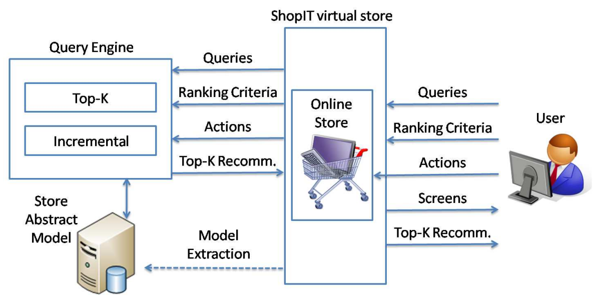


**Figure 2: System Architecture**

not support recommendations on multiple items. Successful commercial tools such as [11, 10] share similar characteristics and often specialize in specific domains, e.g. movies, music, restaurants. In contrast, we propose here a flexible generic approach that addresses the common, multi-item, shopping scenario and identifies navigation flows that best match the users criteria and preferences. The importance of customizable recommendation systems was recently recognized in [9], where such a flexible system was introduced in the context of relational data. The (possibly recursive) semi-structured shape of Web-applications introduces unique challenges for top-k computation, that are not found in a relational environment [6].

We stress that while the demonstration scenario considers a specific example of an on-line computer store, the underlying recommendation system presented here is generic, and an adaptation to other settings will only require the replacement of our Front-End that retrieves application and products data from Yahoo! Shopping, by a different Front-End.

## 3. SYSTEM OVERVIEW

ShopIT is implemented in C++, uses the SQLLite database and php GUI, and runs on Windows XP. Figure 2 depicts the system architecture. We give here a brief overview of the main components and their interaction.

**Store Model.** The abstract model of the on-line store application and its $cWeight$ information are stored in the `ShopIT` database.

The first component, namely the application abstract model, was manually configured following the logical flow structure of Yahoo! Shopping application. We note however that, in general, many Web-based applications are specified in declarative languages such BPEL [3] (the standard for Web-based business processes) and then an automated extraction of their abstract model structure is possible [2]. The products information, *including compatibility relation in-between products*, as well as additional parameters such as products cost, discount deals, shipment time etc. were retrieved via a standard Web interface provided by the Yahoo! web-site. The $cWeight$ function was automatically derived to reflect this data.

**Query Engine.** The query engine is composed of two components. The first is the Top-k queries evaluator that receives, as input, from the user, her search criteria and chosen ranking metric, and computes the initial suggestion of top-k qualifying navigation paths. `shopIT` supplies a Graphical User Interface that allows users to specify their criteria for search (we will see an example of such query GUI in Section 4). The specified criteria are compiled into a *navigation pattern*, which in turn is evaluated over the Web Application model.

The second component is the *adaptive* recommendation engine, that is continuously informed about the user actual navigation choices

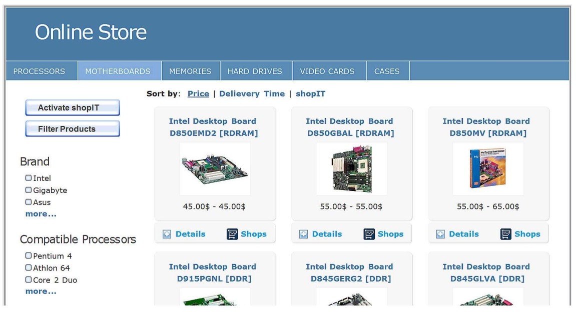


**Figure 3: The original Web-store application**

(or changes to her search criteria and ranking choice) and adapts the offered top-k suggestions accordingly.

We have designed the query engine so that it is accessible through an API that allows the placement of user queries and preferences, and the retrieval of the corresponding recommendations. This general API can be used to incorporate ShopIT within a given website.

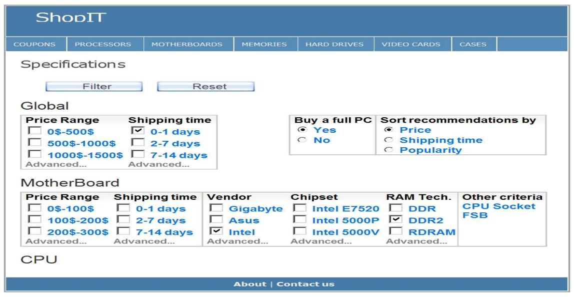
**The ShopIT virtual store.** Users interact with a virtual store that wraps the original store. User actions are passed, through the API, to the ShopIT engine and to the (original) store application. The obtained recommendations are then presented to the user alongside with the resulting store screens (see Fig. 5). Each recommendation consists of a sequence of proposed actions, such as "click on button X", "choose option Y at box Z", etc., and is accompanied by its corresponding weight (e.g. total price, likelihood, etc.). For "in house" applications that allow interference with their graphical interface (like our simulated Yahoo! Shopping store), ShopIT further reorders the items on the screen to reflect their relative "precedence" (i.e. the rank of the best navigation flow in which they participate).

## 4. DEMONSTRATION

As explained above, we illustrate the system operation by considering a specific on-line store that simulates the computers store of *Yahoo! Shopping* [14]. The shop is based on real-life products data obtained from Yahoo! Shopping through their public Web interface. The store allows users to assemble computers based on a variety of components, divided into categories (e.g. motherboards, CPUs, memories). In each category, products can be filtered based on some selection criteria (e.g. vendor name) and may be sorted based on certain attributes (e.g. price). Figure 3 shows a sample screen of the original store describing the available motherboards (MBs), sorted by price. The first three are fairly old, cheap, MBs while the following are newer, hence more expensive, MBs.

The assembly of a reasonably priced, functional computer is a non-trivial task: not all components are compatible, and some selections of cheap components force the selection of other highly priced or error-prone compatible components. For instance, the first three old MBs in the above example support only RDRAM memory modules, which are very expensive, while the newer MBs also support cheap DDR memory. Furthermore, even for a specific combination of components, there may be several ways of purchasing them (with or without a customers club membership, a certain discount package, etc.). Finding the best deal is not easy.

In this demonstration we will show how ShopITsimplifies this challenging task. We demonstrate the user's interaction with the system, as well as what happens "under the hood". The demonstration is structured as follows.


ShopIT

COUPONS | PROCESSORS | MOTHERBOARDS | MEMORIES | HARD DRIVES | VIDEO CARDS | CASES

Specifications

Filter | Reset

Global

| Price Range | Shipping time |
|---|---|
| ☐ 0$-500$ | ✓ 0-1 days |
| ☐ 500$-1000$ | ☐ 2-7 days |
| ☐ 1000$-1500$ | ☐ 7-14 days |
| Advanced... | Advanced... |

| Buy a full PC | Sort recommendations by |
|---|---|
| ● Yes | ● Price |
| ○ No | ○ Shipping time |
| | ○ Popularity |

MotherBoard

| Price Range | Shipping time | Vendor | Chipset | RAM Tech. | Other criteria |
|---|---|---|---|---|---|
| ☐ 0$-100$ | ☐ 0-1 days | ☐ Gigabyte | ☐ Intel E7520 | ☐ DDR | CPU Socket |
| ☐ 100$-200$ | ☐ 2-7 days | ☐ Asus | ☐ Intel 5000P | ✓ DDR2 | FSB |
| ☐ 200$-300$ | ☐ 7-14 days | ✓ Intel | ☐ Intel 5000V | ☐ RDRAM | |
| Advanced... | Advanced... | Advanced... | Advanced... | Advanced... | |

CPU

About | Contact us

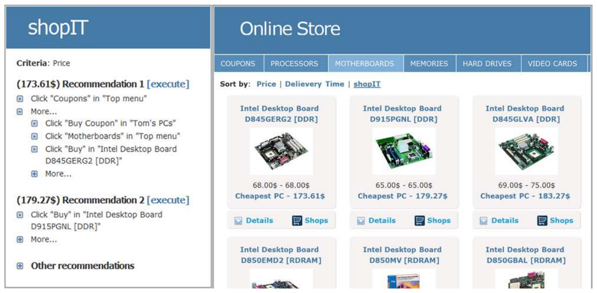
**Figure 4: User Query**

**The Store.** First we will show the original computer store, the many possible navigation paths in its Web-site, and the variety of purchase options, in order to demonstrate the information overload experienced by typical users when faced with such an enormous number of possibilities. Specifically, we shall allow users to navigate freely through the web store, using its standard search mechanism, and to assemble a computer of their likings. Then we will turn to ShopIT and demonstrate how it simplifies and improves the shopping experience by analyzing the various options, in view of users' search criteria and preferences, and identifying those that are most appropriate.

**ShopIT.** We will first illustrate how the store is represented in our abstract model. For that we will show a graphical view of the (weighted) nested DAG representation. We will proceed to demonstrate the system's operation. First we show how the user specifies her search criteria (e.g. a specific items set, or a request for a full compatible computer, possibly with certain components inside), and how she selects her preferred ranking (e.g. price, popularity, delivery time, or some combination). The search criteria may place a requirement over a specific product (e.g. Intel CPU), or over the entire flow (e.g. show only deals with total cost cheaper than 100$). Figure 4 depicts an example for such search criteria, with the user seeking for a deal that comprises of a full computer, containing an intel motherboard with DDR2 RAM technology, with all components supplied overnight. The user further requests the deals to be ordered by their overall price.

Given the user requirements, the system computes the top-k navigation recommendations and presents them to the users alongside the (refined) original shop screen, with the items now ordered according to their current precedence. Two such results are depicted in Figs. 5 and 6 respectively. In Fig. 5 the user asks for ranking based on the overall price. Fig. 6 shows a ranking of the same products, based on popularity of overall deals. We can see that the old, cheap, MBs from Fig. 3 no longer appear here since they both entail a high overall price and are unpopular.

**Flexibility and adaptivity.** Next we will demonstrate the flexibility of the system and the adaptivity to the actual decisions taken by the user. The ShopIT recommendations consist of a sequence of navigation steps that would lead to the "ideal" outcome. The user can either accept the full recommendation by one click (we will first demonstrate how this is done), or start following the proposed navigation sequence step by step (we will demonstrate this next). While navigating, the user may either follow the recommendations or take choices different than those proposed. As an example, we will illustrate a scenario where ShopIT recommended the assembly of a computer with a DDR motherboard, but the user nevertheless chooses an RDRAM motherboard. In this case, new recommendations, consistent with the actual choices made by the user, are automatically computed and presented. We will also illustrate how the


shopIT

Criteria: Price

(173.61$) Recommendation 1 [execute]
- Click "Coupons" in "Top menu"
- More...
  - Click "Buy Coupon" in "Tom's PCs"
  - Click "Motherboards" in "Top menu"
  - Click "Buy" in "Intel Desktop Board D845GERG2 [DDR]"
  - More...

(179.27$) Recommendation 2 [execute]
- Click "Buy" in "Intel Desktop Board D915PGNL [DDR]"
- More...

Other recommendations

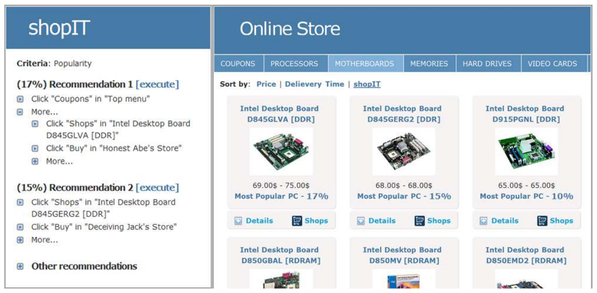
Online Store

COUPONS | PROCESSORS | MOTHERBOARDS | MEMORIES | HARD DRIVES | VIDEO CARDS

Sort by: Price | Delievery Time | shopIT

| | | |
|---|---|---|
| Intel Desktop Board D845GERG2 [DDR] | Intel Desktop Board D915PGNL [DDR] | Intel Desktop Board D845GLVA [DDR] |
| 68.00$ - 68.00$ | 65.00$ - 65.00$ | 69.00$ - 75.00$ |
| Cheapest PC - 173.61$ | Cheapest PC - 179.27$ | Cheapest PC - 183.27$ |
| Details Shops | Details Shops | Details Shops |
| Intel Desktop Board D850EMD2 [RDRAM] | Intel Desktop Board D850MV [RDRAM] | Intel Desktop Board D850GBAL [RDRAM] |

**Figure 5: ShopIT - Ranking by price**


shopIT

Criteria: Popularity

(17%) Recommendation 1 [execute]
- Click "Coupons" in "Top menu"
- More...
  - Click "Shops" in "Intel Desktop Board D845GLVA [DDR]"
  - Click "Buy" in "Honest Abe's Store"
  - More...

(15%) Recommendation 2 [execute]
- Click "Shops" in "Intel Desktop Board D845GERG2 [DDR]"
- Click "Buy" in "Deceiving Jack's Store"
- More...

Other recommendations

Online Store

COUPONS | PROCESSORS | MOTHERBOARDS | MEMORIES | HARD DRIVES | VIDEO CARDS

Sort by: Price | Delievery Time | shopIT

| | | |
|---|---|---|
| Intel Desktop Board D845GLVA [DDR] | Intel Desktop Board D845GERG2 [DDR] | Intel Desktop Board D915PGNL [DDR] |
| 69.00$ - 75.00$ | 68.00$ - 68.00$ | 65.00$ - 65.00$ |
| Most Popular PC - 17% | Most Popular PC - 15% | Most Popular PC - 10% |
| Details Shops | Details Shops | Details Shops |
| Intel Desktop Board D850GBAL [RDRAM] | Intel Desktop Board D850MV [RDRAM] | Intel Desktop Board D850EMD2 [RDRAM] |

**Figure 6: ShopIT - Ranking by popularity**

user may dynamically change her search criteria/ preferred ranking and how the recommendations are instantly adjusted.

At the end of the process, we will present to the users a comparison of their navigation with and without the assistance of ShopIT, in terms of the overall price deal and time of search, to demonstrate the system effectiveness.

To conclude the demonstration and explain the novel top-k ranking mechanism underlying ShopIT, we will pick one of these recommendations and, based on the previously mentioned abstract model, explain its computation.

## 5. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE.*, 17(6), 2005.

[2] C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes. In *Proc. of VLDB*, 2006.

[3] Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel/.

[4] D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella, and F. Patrizi. Automatic service composition and synthesis: the roman model. *IEEE Data Eng. Bull.*, 31(3), 2008.

[5] D. Deutch and T. Milo. Type checking and type inference for queries on business processes. In *Proc. of VLDB*, 2008.

[6] D. Deutch, T. Milo, N. Polyzotis, and T. Yam. Optimal top-k query evaluation for weighted BPs (submitted). 2009. `http://www.cs.tau.ac.il/~danielde/OptimalFull.pdf`.

[7] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of communicating data-driven web services. In *PODS*, 2006.

[8] Ebay. http://www.ebay.com/.

[9] G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Flexrecs: Expressing and combining flexible recommendations (to appear). In *SIGMOD*, 2009.

[10] launch.com. http://www.launch.com/.

[11] netflix. http://www.netflix.com/.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[13] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems*, 1998.

[14] Yahoo! shopping. http://shopping.yahoo.com/.