CoAggregator: User-Driven Approach to Web Aggregation

Svitlana Vakulenko
Graduate School of Computer Science
Saarland University
66123 Saarbrücken, Germany
s9svvaku@stud.uni-saarland.de

Sudhir Agarwal
Institute for Applied Informatics and Formal
Description Methods (AIFB)
Karlsruhe Institute of Technology (KIT)
76131 Karlsruhe, Germany
sudhir.agarwal@kit.edu

ABSTRACT

For increasingly sophisticated use cases an end user needs to extract, combine, and aggregate information from various (dynamic) web pages belonging to different websites. Current search engines do not focus on combining the information from various web pages in order to answer the overall information need of the user. Semantic Web and Linked Data usually take a static view on the data and rely on providers' cooperation. Web automation scripts, initially developed for testing websites, allow end users to capture their browsing activities as executable processes and share them with other end users. Web aggregators promise to find best offers on the market in order to save money and time of the customers. However, the web aggregators are opaque and thus hard to trust for end users regarding completeness of the results and fairness of the ranking. Furthermore, when there are more than one aggregator for one domain end users face the same problem of aggregating their results manually. Web automation scripts, initially developed for testing websites, allow end users to capture their browsing activities as executable processes and share them with other end users. In this paper, we present a script-based approach for allowing end users to collaboratively create information aggregators.

Keywords

Semantic Web, end user development, web aggregation, crowdsourcing, the Deep Web.

1. INTRODUCTION

For many practical purposes end users need information that is scattered across multiple websites. Static websites can be reached and their content can be indexed by the crawlers of state of the art search engines. Search engines results often contain links to web pages with similar content even though the information need of the user might require pages with complementary information. In order to obtain the required information an end user needs to pose multiple queries to a search engine, browse through the hits, and aggregate the required information fragments outside of the found web pages. The case of dynamic websites to access the information in the Deep Web [2] is even more complex and still an open challenge for search engines since it is hard for automatic crawlers to sensibly interact with the dynamic websites. Furthermore, indexing such information is not a suitable technique since the information underlying dynamic websites changes so rapidly that the index becomes quickly outdated.

In contrast to search engine crawlers, end users are able to reach the dynamic web pages. Information retrieval has focused on analyzing such click trails of millions of end users mainly for the purpose of improving web search results. Click trails can be used as endorsements to rank search results more effectively [3], trail destination pages can themselves be used as search results [10], and the concept of teleportation can be used to navigate directly to the desired page [9]. Similarly, large-scale studies of web page revisitation patterns [1] focus on how often users revisit the same page, while ignoring how people get there. The statistics based click analysis methods typically do not consider semantics of user queries and pages. As a result, a frequently used and thus recommended path may not necessarily satisfy the information need, and end users still require to figure out themselves which of the recommended web pages are actually relevant for them and which interactions are required with which web pages.

Web aggregators promise to find best offers on the market in order to save money and time of the customers. However, the web aggregators also have some limitations from the point of view of end users. (i) are opaque and thus hard to trust for end users regarding completeness of the results and fairness of the ranking. (ii) when there are more than one aggregator for one domain end users face the same problem of aggregating their results manually .

The automated wrapper generation tools for the Deep Web [8] are restricted for one-step web processes that consist of one web-form and one response page. In many professional scenarios the processes for gathering required information require multiple user interactions such as multiple form submission with web-sites.

OXPath [7] and CoScripter [4] record user actions on the web for further reuse and information extraction. Web automation scripts, initially developed for testing websites, allow end users to capture their browsing activities as executable processes and share them with other end users. Number of scripts in public repositories prove that users are

interested in developing and using web automation scripts (Greasemonkey script repository[1] currently has more than 80,000 scripts, IBM CoScripter repository[2] - almost 6,000 scripts). However, the larger repository grows the more difficult it becomes to locate useful scripts. It is often easier to create a new script than find and reuse existing one. This leads to unnecessary duplication, decreases reuse and collaborative development of scripts. Moreover multiple site-specific scripts may have the same goal and it is useful to integrate them. However, such scripts, created by distributed authors, are heterogeneous and, therefore, are hard to integrate. Our system goes beyond this scope and offers semantic script integration for aggregation of information extracted from different web sites.

Our Contribution: We introduce a platform that equips script authors with common semantics and enables easy interface-based script integration. In this way script repository turns into semantically aware mash-up application. We demonstrate how such a system shall be used for real-time information extraction from the Deep Web [2] sources. Moreover, our approach provides an incentive to the end users to contribute to the shared repository because of the added value of reusing existing scripts alongside the new scripts.

In our demonstration we provide an example in a traveling domain where all scripts aim at extracting the cheapest-priced ticket using different web sources given an itinerary. Other use cases for web aggregation vary through all the search domains and may include car rental, shopping, library services. For each of this domain a full-fledged ontology and a stand-alone access point (hub) shall be designed, although they may be also integrated on the semantic level and share common functionality (e.g. input/output formatting methods). The scripts are to be centered around such hubs in order to classify and meaningfully interconnect them.

2. COAGGREGATOR: A TOOL FOR END USER INFORMATION AGGREGATION

We propose a system that supports information extraction and aggregation by automating some of the tedious steps. The system can also be applied to the so-called web processes, which are rather typical for modern e-commerce websites, and may cause state change. Sample use case scenarios to be listed include: car rental, ticket/hotel reservation, best offer search processes. The main aim of the proposed system is a separation into a centralized server-side knowledge base of scripts and a client-side toolkit to allow personalized information extraction from the original sources in real time.

2.1 Formalization of Scripts

An end user browsing process is a sequential process that coordinates the execution of multiple websites. An end user has a local knowledge base, and the browsing activities that an end user carries out can be categorized into input, output, and local (w.r.t. to the end user knowledge base) actions. An input action causes addition of knowledge from a website into the knowledge base, the output activity emits (without deleting) knowledge from the knowledge base to a website, and a local action causes changes in the knowledge base independent of the websites such as deletion or alignment of knowledge. Such browsing processes can be easily modeled by a process algebra such as π-calculus [5] with the syntax $\mathbf{0} \mid c[\mathbf{x}].P \mid c\langle\mathbf{y}\rangle.P \mid \tau.P$, where $\mathbf{0}$ denotes the process that does nothing and used as termination symbol, $c[\mathbf{x}].P$ denotes a process that inputs some values along the channel c, binds them to vecx, and then behaves like the process P, $c\langle\mathbf{y}\rangle.P$ denotes a process that outputs values $\mathbf{y}$ along the channel c, and then behaves like the process P, and finally $\tau.P$ denotes a process that performance a local action, and then behaves like the process P. A local action is an action performed by the end user in his/her local knowledge base in order to structure the knowledge as per user's needs. The set of local actions available to an end user depends on the data model of the knowledge base, e.g. a relational model will allow different operations than a graph based model. In order to allow semantic reasoning about the values of the process we annotate them with a domain ontology O_D expressed in $\mathcal{ALC}$ (attributive concept language with complements) [6]. E.g., input parameters $\mathbf{x}$ and the communication channel $\mathbf{c}$ of an input activity are process resources and further described in O_D. With $\mathcal{ALC}$ we can describe not only the types of process variables but also their relationships with other process variables. For example, if an input activity has two parameters of type 'Person', we can also describe that the first person should be father of the second person. Precisely, the local knowledge modeled as ABox of $\mathcal{ALC}$ can be modified by adding or removing following types of axioms: (i) add sameAs relation between two individuals, (ii) add typeOf relation between an individual and a concept, (iii) domain specific relationships between two individuals (object properties), and (iv) relationships between an individual and a literal (data properties) .

2.2 Script-based Aggregation

Each script fulfills a certain purpose. For sophisticated and complex requirements many scripts need to be combined and their execution need to be coordinated. A coordination process (also referred to as a hub) is a sequential process that invokes its component scripts. Formally, a coordinating process can be seen as a process running in parallel to its component scripts. The composition of a coordinating process C with its component scripts $P_1, \ldots, P_n$ can be described with the help of the composition operator as $C \parallel P_1 \parallel \ldots \parallel P_n$. The variables of the coordinating process and variables of the component scripts are annotated with semantically aligned domain ontologies in such as way that at least automatic type checking (and ideally checking of preconditions of input values is possible). The dataflow between coordinating process and the component scripts takes place by connecting an input (output) activity of the coordinating process with an output (input) activity of a script. The local activities of a coordinating process can be deterministic computation procedures such as MIN, MAX, AVERAGE, COUNT etc. The order of the activities of the coordinating process determines the order of the overall process that an end user executes.

2.3 Implementation

Figure 1 illustrates the conceptual architecture of CoAggregator. An end user interacts with a web page which consists of a web form. For each source chosen by the user the web page (hub) issues a request to the repository and returns specification of a script (name, url, input and output vari-

[1] http://userscripts.org

[2] http://coscripter.researchlabs.ibm.com

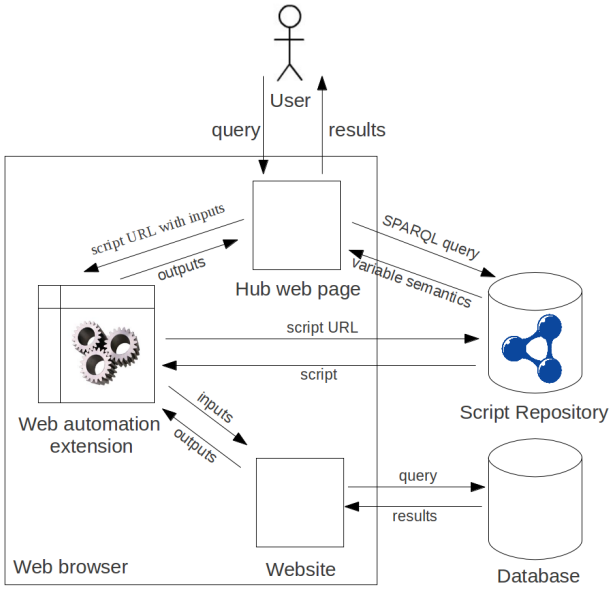

Figure 1: Conceptual Architecture of CoAggregator

ables) which automates access to the particular web source. This information is used in order to start web automation extension provided with script url containing user input from the web form as values for the script variables. On the next step extension fetches script body from the repository and executes the script. If execution was successful it will return extracted information back to the web page where it will be automatically placed in a result table below the form.

2.3.1 *Webform-based Front-end*

CoAggregator interface provides functionality for the real-time search and contribution to the repository of script. The search form consists of the necessary fields for the request to be submitted and multi-choice field for the sources to be queried, which constitute the number of scripts available for the aggregation.

Webform interface reflects content of the underlying ontology in a more usual for the user way, as names for the form fields. The scripts are grouped around the domain-specific hub web page according to their use/semantics (goal or the output they return and input they consume). In-built functions are to be provided in order to allow presentation of the same semantic context with different masks applied. For example, date formats, translation, upper/lower case for the textual inputs.

The following steps are required in order to integrate a new script into the system: (i) Create script with CoScriptor (record/edit); (ii) Name variables according to the naming convention used in the system (look up corresponding field heads in the web-form); (iii) Register script in the system (provide script name and url).

2.3.2 *Technologies used for Implementation*

The knowledge base (meta repository) is implemented as a Virtuoso triple store storing RDF files on a server and accessible through a SPARQL end-point. The system may reuse public IBM CoScripter repository, which enables collaborative script editing in a form of a wiki. Scripts are stored as a plain textual files in JSON format.

Web aggregator is implemented as a JavaScript-enabled web page. The page invokes and exchanges data with a browser automation extension (IBM CoScripter v.2.210 for Mozilla Firefox, extended with a listener to the events coming from our web page), which executes the scripts. GeoNames API is used for the look up of valid city, country names and IATA codes for the nearby airports. Date Format JavaScript function [3] was reused for transforming between different date formats. Such built-in functions help to uniquely identify entities and automatically translate them into other formats used on actual web-sites. The scope of web sources that can be aggregated by our prototype is currently limited to HTML/JavaScript-based web pages.

3. DEMO DESCRIPTION

CoAggregator interface provides functionality for the real-time search and contribution to the repository of script. The search form consists of the necessary fields for the request to be submitted (destination/departure/date in the traveling domain) and multi-choice field for the sources to be queried, which constitute the number of scripts available for the aggregation.

Show case 1: Flight search. A user specifies a search query and chooses the web-sites to search on (the scripts). After pushing the "Search" button no more actions from the user is required. All the possible combinations of the provided search parameters will be used as an input for each of the scripts corresponding to particular web sources. The system will automatically fetch (screen scrape) data from the chosen web sites in real time.

The scripts simulate user actions on the web form interface (filling the input fields and button clicking) as fast as possible and wait for the web site response. Thus, the accumulated waiting time comes from the server responds and depends on the server, which can not be avoided in the case of manual interaction also. The main benefit provided by the tool is that it automates repetitive actions of refilling similar web forms with the same information all over again. The user is free to monitor the search process or use this loading time for any other activity, while script execution is run in the meantime in the background using the input data provided on the previous step. Though, the automated script execution can be interrupted at any time by a user or automatically, for example, when the web process requires a user input, e.g. entering a CAPTCHA to prevent access to the data by automated crawlers. This is another reason in favor of personalized script execution as opposed to bulk information gathering by the 3rd party aggregation servers.

After the system finished execution all the results extracted from the original websites are automatically placed in the result table on the initial web page below the query web form. Single result table makes analysis of the output information easier and more efficient bringing it together in a single format, for example, the same currency or time units. The "Buy!" button brings the user one click away from the original source web-site where (s)he can purchase a ticket of the corresponding itinerary. The script is rerun again with the corresponding ticket itinerary and stopped at the final point where more information, e.g. data of a credit card, is required from the user to proceed.

[3](c) 2007-2009 Steven Levithan MIT license

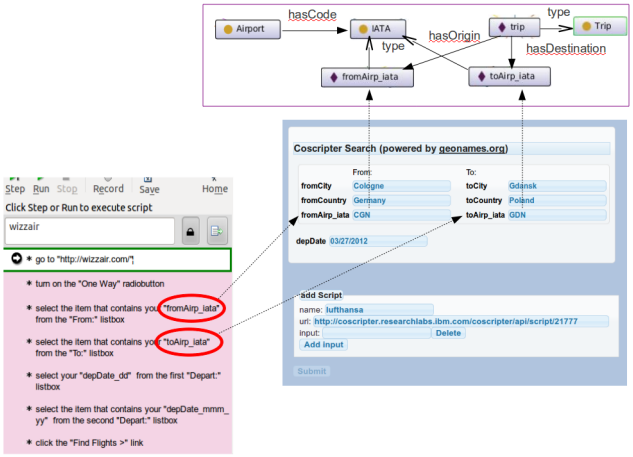

Figure 2: Adding a new script to an existing aggregator

Show case 2: Adding a new script. Any user is free to contribute and integrate new scripts into the system automating information retrieval from other web sources. In order to do this the user creates a script - a sequence of user actions. User toggles the web browser extension into recording mode and shows a sample session of user interaction with a web site for the sake of extracting certain information piece. It concluded with marking the extraction area and saving it into an output script variable (the information goal).

Then, the user substitutes actual values with names of the corresponding input variables using the naming convention imposed by the system. Figure 2 shows the integration of a new script in an existing aggregator for the domain of flight ticket offers. The web form provides the intuition behind the variable use as it initially collects the input data from the user and passes it on to the scripts. Names in front of the form fields suggest user the appropriate names for the script variables. However, difference between the data format consumed by the script has to be explicitly specified using the built-in transformation functions which are connected to the web form. Field name will be automatically updated when user adjusts the appropriate input format. By linking to the hub variables, the script obtains semantic descriptions for its own variables in terms of domain ontology, which provides a single non-ambiguous view on variable semantics.

The user provides script ID that helps system to locate the script in the repository. The system automatically extracts names of the variables from the body of the script, generates the corresponding script profile (semantic description) and adds it to the knowledge base. The web aggregator main page, when refreshed, lists also the newly added script, which now can be used along with the other existing scripts.

4. CONCLUSION AND OUTLOOK

We proposed design of a web aggregator that is shaped by a user community itself. End users get the opportunity to control web aggregation process and extend it to new web sources. Our approach provides an incentive for the end users to contribute to the common repository as they gain value from using their own scripts alongside other scripts.

Semantically enhanced scripts accumulated by CoAggregator can be reused and automatically processed. For example, semantic description of script variables enables semantic search for scripts by type of information they consume and provide, which is more efficient than a keyword-based search in a script repository. CoAggregator automates direct interaction between the web source and the end user avoiding mediation through a 3rd party aggregator web site. Moreover, script integration is easy and information retrieval process is transparent for the end users. Hence, users are able to monitor the execution process in real time to assure the origin and actuality of the retrieved information.

Our solution may be enhanced with automated methods for web process discovery and composition. However, we want to emphasize the role of crowd-sourcing as a way to equip the system with human-friendly interface for executable how-to knowledge exchange and fault tolerance.

5. REFERENCES

[1] Eytan Adar, Jaime Teevan, and Susan T. Dumais. Large scale analysis of web revisitation patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1197–1206. ACM, 2008.

[2] Michael K. Bergman. The deep web: Surfacing hidden value. *The Journal of Electronic Publishing*, 7(1), 2001.

[3] Mikhail Bilenko and Ryen W. White. Mining the search trails of surfing crowds: identifying relevant websites from user activity. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 51–60. ACM, 2008.

[4] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. Coscripter: automating & sharing how-to knowledge in the enterprise. In *Proceedings of the 26th annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1719–1728, New York, NY, USA, 2008. ACM.

[5] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, Parts I and II. *Journal of Information and Computation*, 100:1–77, 1992.

[6] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.

[7] Andrew Jon Sellers, Tim Furche, Georg Gottlob, Giovanni Grasso, and Christian Schallhart. Taking the oxpath down the deep web. In *EDBT*, pages 542–545, 2011.

[8] Pierre Senellart, Avin Mittal, Daniel Muschick, Rémi Gilleron, and Marc Tommasi. Automatic wrapper induction from hidden-web sources with domain knowledge. In *WIDM*, pages 9–16, 2008.

[9] Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI*, pages 415–422, 2004.

[10] Ryen W. White and Jeff Huang. Assessing the scenic route: measuring the value of search trails in web logs. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 587–594. ACM, 2010.