An approach for the semantic contextualization of the web advertisement process

Ljiljana Stojanovic
FZI Karlsruhe
Haid-und-Neu Strasse 10-14
76131 Karlsruhe
+49 721 9654804
Ljijana.Stojanovic@fzi.de

Roland Stuehmer
FZI Karlsruhe
Haid-und-Neu Strasse 10-14
76131 Karlsruhe
+49 721 9654872
stuehmer@fzi.de

ABSTRACT

The modern advertisement theory is based on the "contextual priming effects": the product attributes primed by the ad context may result in the formation or change of beliefs about the advertised brand, thereby affecting consumers' evaluations of the brand. Therefore, a web ad should be tailored as much as possible to the user's current context (interests) in order to affect the user' attention appropriately.
In this paper we present an approach for the semantic-based personalized advertising on the web.

Keywords

Personalized advertisement, complex event processing, semantics

1. INTRODUCTION

A contextual web advertising system scans the text of a website for keywords and returns advertisements to the webpage based on what the user is viewing. Contextual advertising has made a major impact on earnings of many websites. Because the advertisements are more targeted, they are more likely to be clicked, thus generating revenue for the owner of the website (and the server of the advertisement). However, despite being targeted, current approaches for contextual web advertising are not personalized, i.e. they are not taking into account the user, but only the characteristics of the web site. On the other hand, the modern advertisement theory is based on the "contextual priming effects": the product attributes primed by the ad context may result in the formation or change of beliefs about the advertised brand, thereby affecting consumers' evaluations of the brand. Therefore, an ad should be tailored as much as possible to the user's interests in order to affect the user appropriately. Consequently, this implies a need for real-time tracking a web user's behavior in order to detect her/his current interests, by assuming that her/his current interest will correlate to the visited elements in a web page. Moreover, due to the different contexts that can be found in a web page, such personalized ads should be dynamically changed, according to the changes in the user's interests. However, due to the request/response style of web communication, the user's behavior cannot be captured in the real-time (on the client side) easily and is therefore omitted from the traditional web advertisement process.

Modern web technologies are enabling more client-side control of the user's behavior and there is already work done in developing technologies for gathering a web user's behavior while browsing AJAX-based web pages [1].

In this paper we leverage on that work in developing an approach for the dynamic and personalized web advertisement. In the nutshell of the approach is the real-time and complex processing of the user's behavior in a web page.

In fact, the user's interaction with a web page is interpreted as a set of events, which are combined in order to discover the "very current" interest of the user. Events, simple or complex, are models for things that happen e.g., when a user interacts with a Web page. Events are consumed in some meaningful way e.g., for monitoring reasons or to trigger actions such as responses. Semantics is used for a better interpretation of the user's behavior by taking into account the meta information assigned to parts of the web page, which the user has visited. The user's interest/profile generated in this way is used for the very personalized ad generation.

Additionally we define a model for updating ads once the current user's interest has been changed such that displayed ads are not any more the most relevant one. In that way our approach supports dynamic adaptation of ads ensuring their high relevance for the user.

In this paper we present the whole approach for the personalized and dynamic web advertisement, including the technical architecture for detecting and composing (semantic) events in Web clients, that is, as explained above, the basic mechanism for discovering and updating real-time profile of a web user (i.e. her/his interests). Additionally we demonstrate the validity of the approach in two evaluation studies.

The paper is structured as follows: In Section 2 we describe methods for tracking a user's behavior in a web page (as a set of semantically enriched events), whereas in Section 3 the methods for complex processing of these events are given as an approach for discovering the current interest of the web user. Section 4 presents the architecture for generating personalized and dynamic web ads based on detecting “unusual” user’s behavior, whereas Section 5 contains some implementation details and in Section 6 we present some evaluation results. We will discuss related work and conclude the paper in the last remaining sections.

2. TRACKING A WEB USER’S BEHAVIOUR

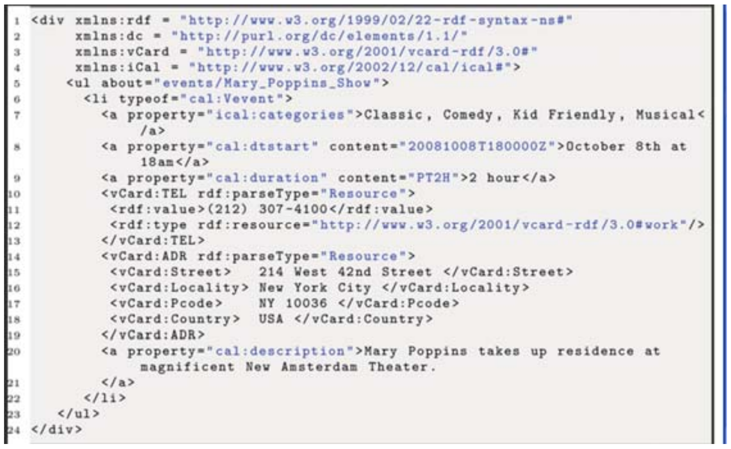
The main issue in enabling personalization of the web usage is to enable capturing of actions or changes in Web documents. These can be treated as events, which an event-driven system will react to. For our use case of advertising we will focus on events created from a user’s interaction with Web documents. After having extracted events from a Web document, they must be processed in order to interpret them semantically, to be able to react on them appropriately. The following two subsections describe our approach for these two issues: generation and processing of Web events.

2.1 Simple Event generation

A simple event in Web clients is characterized by two dimensions; the type of event (e.g. click, mouseover) and the part of the Web page, where the event occurred (e.g. a node of the Document Object Model of the Web document). This node is, however, just a syntactical artifact of the document as it is presented in a Web browser. Adding this node or parts of it to the event body will not significantly add meaning to the event and not ease the understanding of the event for the recipient of the event.

We therefore propose to add semantic information to the event which pertains to the actual domain knowledge that the Web page is about. In order to enable this, the first step is to represent the content of a Web page in a form that can be used for generating meaningful events. To do so without having to manually annotate every Web document, we envision a mechanism, which ensures the relevance of the annotations. This can be done in many (semi-) automatic ways, e.g. by providing Web forms (page templates), which for a given user’s input, automatically adds the proper semantic relationships between the form fields. In this way all user generated content will be annotated. The Web forms are created based on supported vocabularies for a particular Web site. Our particular focus is on widely spread vocabularies such as Dublin Core, Creative Commons, FOAF, GeoRSS and OpenCalais. Regarding the format of structured data, RDFa [2], eRDF and Microformats are all good candidates for this purpose. They support semantics embedded within actual Web page data and allow reusable semantic markup inside of Web pages. In our implementation we use RDFa, since in comparison to eRDF it is a more encouraged candidate by the W3C. Comparing it further to Microformats, RDFa is more flexible in mixing different existing vocabularies.

In the remaining part of this section we give an example demonstrating the generation of events in the context of a Semantic Advertising scenario. The ad space is a part of the Web page which can be dynamically filled by an ad provider as a response to an event the client sends. In our approach ad content is created based on a current user’s attention. In order to accomplish this we need as much (meta-) information as possible about the content of the Web page. Therefore, we assume semantically enriched Web content such that context extraction is easier and more precise. Additionally, every page is split up in a number of Semantic Web Widgets (SWW). We introduce Semantic Web Widgets as self-contained components annotated with semantic data and displayed in a Web page. Semantic Web Widgets give a high-level description of the content, and provide the basic context of data contained in the widgets. For instance on a news portal incorporating semantic advertising one widget could be used for listing all news belonging to one subcategory, e.g., politics, another one for arts, etc. In Figure 1 we show an RDFa example of the semantic description for an arts event listed in a widget related to musicals. The code snippet presents an event named “Mary Poppins Show” described using RDF Schemata for Dublin Core, vCard and iCal vocabularies. Information such as categories, start and duration of the musical are provided together with contact information, location and so on.



```
<div xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:dc = "http://purl.org/dc/elements/1.1/"
     xmlns:vCard = "http://www.w3.org/2001/vcard-rdf/3.0#"
     xmlns:iCal = "http://www.w3.org/2002/12/cal/ical#">
   <ul about="events/Mary_Poppins_Show">
     <li typeof="cal:Vevent">
       <a property="ical:categories">Classic, Comedy, Kid Friendly, Musical<
           /a>
       <a property="cal:dtstart" content="20081008T180000Z">October 8th at
           18am</a>
       <a property="cal:duration" content="PT2H">2 hour</a>
       <vCard:TEL rdf:parseType="Resource">
        <rdf:value>(212) 307-4100</rdf:value>
        <rdf:type rdf:resource="http://www.w3.org/2001/vcard-rdf/3.0#work"/>
       </vCard:TEL>
       <vCard:ADR rdf:parseType="Resource">
        <vCard:Street>   214 West 42nd Street </vCard:Street>
        <vCard:Locality> New York City </vCard:Locality>
        <vCard:Pcode>    NY 10036 </vCard:Pcode>
        <vCard:Country>  USA </vCard:Country>
       </vCard:ADR>
       <a property="cal:description">Mary Poppins takes up residence at
           magnificent New Amsterdam Theater.
       </a>
     </li>
   </ul>
</div>
```

Figure 1. An example for a musical listed in a Semantic Web Widget.

2.2 Event enrichment

In this subsection we focus on enriching simple events with semantics from the context of the Web page in which the event occurred.

A simple event in Web clients is characterized by two dimensions; the type of event (e.g. click, mouseover) and the part of the Web page, where the event occurred (e.g. a node in the Document Object Model of the Web document). Subscribing to simple events of these types therefore requires the specification of type and the specification of the node or nodes where the events may originate. Both dimensions are retained in an event instance by using the attributes jsEventType and cssSelector (see Figure 2 for more explanations).

In order to better understand these events and make sense of what happened we must enrich the content of events when they are produced. The jsEventType tells us what a user has done and the cssSelector tells us where on the Web page the user did it. However, the latter is a purely presentation-dependent measure. There is no semantics which has any meaning beyond the context of a specific Web page structure. We propose to extract presentation-independent semantic information from the Web page if present. Instead of creating events from interaction with purely syntactic items of a Web document, we create events about interaction with semantic concepts which the document stands for. As an example, an event should not represent e.g., a click on a certain headline element of a Web document but rather a user's interaction with an article talking about politics and certain persons mentioned within.

To annotate a Web page with semantic data such as the topics of an article, we use RDFa. Defined in [2] RDFa is a means of adding RDF data to existing Web pages by using inline XHTML attributes.

After detecting an event which happened in the context of a certain DOM node of a Web document, we collect all semantic information in the Web page about the thing that is reported in that given DOM node. We currently achieve this by employing the client-side RDFa library ubiquity (http://ubiquity-rdfa.googlecode.com/). The lifting of context is achieved in a two-phase process. In the first phase we collect the list of RDF subjects of possible triples. This is done close to where the event happened in the document to provide accurate context. In the second phase we collect every triple with these subjects from the overall document in order to provide a very rich context. To find valid subjects the first phase traverses the node where the event happened and its complete subtree. If the given main node does not contain a subject, the immediate dominator node containing a subject is added to the list. This serves two purposes, guaranteeing a single root subject for orphan properties and objects in the subtree and guaranteeing a non-empty result set.

In the second phase all triples with the given subjects are collected from the entire document tree. The gathered triples are then reified and appended as a bag to the event payload. Even if the event itself becomes part of more complex events during the process of correlating and aggregating events, this basic data is retained as part of the simple event.

3. UNDERSTANDING THE USER'S INTEREST – COMPLEX EVENT PROCESSING

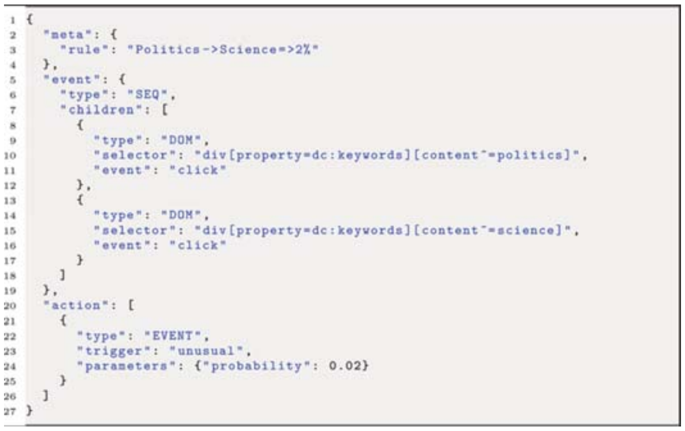
Simple events extracted from Web documents must be combined in order to detect complex situations which might be interpreted as a user's interest. This is the task of Complex Event Processing. Detecting the behavior of Web users according to our proposal is divided into design time and run time. The design time consists of (i) semantically enhancing the Web page and then (ii) recording average viewing statistics of the annotated elements, e.g. from log files. From the statistical data we generate client-side rules. Once these rules are created they are pulled by the next client request and loaded into the rule engine for the run time.

For the run time we have developed a client-side event-condition-action (ECA) rule engine. It uses a lightweight rule language which supports ECA rules described in more detail in [3].

Very briefly, we JSON-Rules, our client-side rule language that resembles a lightweight reaction rule language tailored to the needs of Rich Internet Applications, specifically applications that profit from or require Complex Event Processing, condition evaluation on a working memory, and running rule actions written in JavaScript. As a representation for our rules we use JSON, because it is natively usable within JavaScript. JSON can specify objects, arrays and primitives. Rule objects in our JSON-Rules language contain the three attributes event, condition and action. The event part consists of patterns in the event pattern language Snoop [4]. Snoop contains a fairly comprehensive list of Boolean and temporal operators. They are modeled in our ontology. What is missing in Snoop are operators which inspect the contents of input events such as attributes other than timestamps and type. Therefore, we added a FilterEvent as an example of what is needed to filter events by their content.

The condition part consists of conjunctive predicates over variables from a working memory. The action part in turn contains one or more JavaScript code blocks to gain a maximum degree of versatility for the rule author. Alternatively for rule actions we offer to trigger certain desired events as well as manipulations of the working memory. The latter types of action offer greater declarativity while formulating rules. This increase is, however, bought at the price of some flexibility. Thus, we still offer all three kinds of rule actions which can be freely mixed.

Figure 2. Example of a single Rule



```
{
  "meta": {
    "rule": "Politics->Science=>2%"
  },
  "event": {
    "type": "SEQ",
    "children": [
      {
        "type": "DOM",
        "selector": "div[property=dc:keywords][content~=politics]",
        "event": "click"
      },
      {
        "type": "DOM",
        "selector": "div[property=dc:keywords][content~=science]",
        "event": "click"
      }
    ]
  },
  "action": [
    {
      "type": "EVENT",
      "trigger": "unusual",
      "parameters": {"probability": 0.02}
    }
  ]
}
```

The rules on the client serve to detect users exhibiting interesting behavior as learned from the average usage patterns. The user causes events to occur by interacting with the Web page, detected by the event processor and rule engine. Rules are triggered which create intermediate events in a hierarchy of event abstraction. These events are subsequently accumulated until sufficient interest according to the ad provider is recorded (threshold achieved) and actions can be taken by further rules.

The distinction between run time and design time in this section is not a strict temporal distinction as the names would suggest. Rather, because new users will inevitably alter our knowledge of what is interesting there is a loop in the process, feeding back from the run time into the design time to evolve new rules for future users.

Figure 2 shows an example rule. It can be automatically created from analyzing histories of interesting behavior. The only requirement is knowledge, that e.g. states that only two percent of users look at a politics item followed by a science item. The actual rule consists of an event part starting at line 5 and an action part starting at line 20. The rule resembles an event-condition-action rule where the condition is left blank, i.e. is always true.

4. GENERATING THE PERSONALIZED ADS DYNAMICALLY

Figure 3 shows a rough architecture of our approach: Part b) on the right hand side of the figure depicts the components of our client-side rule engine. Multiple event sources provide input for the event detection, creating complex events. Also, a working memory submits its changes to a Rete network, evaluating rule conditions. The logic for both the event detection and condition evaluation is supplied by rules from a repository, generated from past user activities. Part a) on the left hand side places the client-side components above the protocol boundary dividing client and server. Below on the server or several distributed servers hold the Web content as well as the advertising content. The Web content is annotated, providing semantic relations to the advertisements. Short-term user models provide a temporal model of how a user interacts with the Web content. The ad provider analyses user models to provide up-to-date and personalized advertisements.

On the other hand we anticipate annotations to be mostly used on elements at, or not far below the level of single widgets or paragraphs. Reasons for this are of practical nature, in keeping the number of events manageable. Handling too much detail might have further adverse effects at this point, creating a large number of event types which are almost never used (created or consumed). There might, for example, be no measurable interaction of the user with a certain word in a Web page, whereas the surrounding paragraph might encounter detectable mouse clicks or mouse hovering/movement.

As mentioned in the introduction, web ads should be continuously updated to the web user's interests, which implies a need for the automatic triggering of a new ad, once that user's interests has been dramatically changed. In this work we use the notion of unusual behavior as the criteria for generating a new ad. In the following we describe that principle shortly.

In order to form complex event expressions, the RFDa annotations are combined with a temporal model. Such expressions group the user's atomic actions into temporal contexts like e.g. sequences of clicks. Determining sequences of interest is based on analyzing historical (log) data statistically. By using data mining algorithms for click streams such as [5], historical data is transformed into knowledge about unusual sequences of interaction such as clicks. Subsequently, the corresponding complex event expressions can be created.

This process can be done automatically. A simple sequence along with its confidence might be "politics" followed by "flowers" with a low confidence of 2%. This means that from previous users only a fraction of 2% have looked at a politics widget followed by looking at a flowers widget.

This pattern in the users behavior can be treated as unusual, i.e. his/her interests for "politics" and "flowers" are distinguished from the interest of others, so that this can be used for developing a very personalized ad. In fact, we argue that more information content (for generating ads) is stored in the exceptional behavior, than in the usual/expected one. A simple explanation is that expected behavior is too general to detect what is specific in the behavior of the customers (cf. example from the brick and mortar environment from Introduction). Once when enough "unusual behavior" is accounted for a user a new ad should be issued.

Such an ad will very likely attract the attention of the user, since it directly corresponds to his short-term profile. Further processing of e.g. the time interval within the two participating events could be envisioned.

Each complex event expression is embedded in an event-condition-action rule with the probability as the consequence. The consequence forms another event which is processed further by higher-level rules.

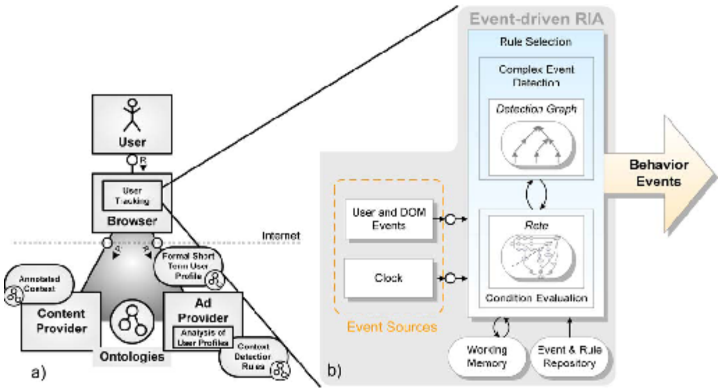


Figure 3: Architecture: a) Logical Architecture b) Client-side User Behavior Analysis.

In order to enable such a processing, we extended the set of traditional event processing operators with two additional ones *Filter*(E1; condition) and Thres(E1; threshold) as follows:

Filter is modeled like event masks. The Filter enforces a condition on each occurrence of event E1. This allows e.g. for fine-grained content-based filtering/masking of events.

Thres is another content-based operator which we need to extend the Snoop algebra with. Thres(E1; threshold) accumulates the events of type E1 until the boolean function "threshold" returns true, releasing all accumulated events as a complex event and starting accumulation anew.

5. IMPLEMENTATION: CLIENT-SIDE EVENT-ENABLED RULE ENGINE

For our implementation we chose JavaScript from the available Web programming languages, for reasons of widespread availability. The data structures and program logic we implemented are roughly divided into the following areas: adapters for the rule language and remote event sources, the working memory, condition representation and evaluation as well as complex event detection.

For Complex Event Processing we are using a graph based approach as proposed in [4]. Initially the graph is a tree with nested complex events being parents of their less deeply nested sub-events, down to the leaves being simple events. However, common subtrees may be shared by more than one parent. This saves space and time compared to detecting the same sub-events multiple times, and renders the former tree a directed acyclic graph.

When using the term event, the distinction must be drawn between event occurrences (i.e. instances) and event types, usually done implicitly. In the detection graph the nodes are event types, they exist before there are any instances. Event instances exist after simple instances arrive and are fed into the graph at the leaves. Complex instances are then formed at the parent nodes, which in turn propagate their results upwards. Every complex event occurrence carries pointers to the set of its constituent event occurrences, so that the events and their parameters can be accessed later. Once an occurrence is computed at a node which is attached to a rule, the state of the associated Rete node is started and actions are triggered.

6. EVALUATION

To evaluate the return of targeted advertisements we created a demo Web page with some news articles. Each news article is contained in a separate part of the page, termed Semantic Web Widget (cf. Section 2.1). Each widget is annotated using RDFa using basic keywords and concepts pertaining to the article. For a user entering our demo, each widget is at first partially concealed. This is done to solicit an action from the user when "unfolding" the widget. Thereby the user expresses interest. This creates explicit events which can then be processed by our engine. Our initial evaluation of the ad quality was performed as follows:

1. We selected three different news domains (politics, culture, sports) in order to prove the domain-independence of the approach and pull into the demo Web page, as separate evaluation sessions.

2. We selected five users (PhD students from the Institute) with different cultural backgrounds.

3. The users should browse the demo Web page and judge about the relevance of generated ad-keywords in the case of a) the keywords generated statistically from the Web page (Google approach) and b) keywords generated by using the event-driven approach described in this paper. In order to ensure a fair comparison, the users did not know which list of ad-keywords was produced by which method.

We ask the users to rate the gathered keywords in terms of relevance to what they had been doing in the news portal and to compare this with a static list of keywords extracted from the overall page. The results are very encouraging: in the average 85% of keywords generated in our approach were described as "very relevant" and 98% as "relevant" (very similar results across all three domains).

The traditional approach achieved 65% success for "very relevant" and 85% success for "relevant" ad-keywords. This result demonstrates the advantages of our approach for generating very relevant ads.

In comparison, Web Usage Mining (e.g., [5]) is used on log files which are analyzed on the server side at certain intervals or possibly in a continuous fashion. It is important, however, to stress that our approach detected all events on the client. Events occurred purely by folding and unfolding widgets as parts of the page. No communication with the server took place and hence no artifacts are visible in server log files. Thus, our approach extends clickstream analysis to regions which were previously invisible to server-based mining techniques.

Moreover, our approach is a truly event-driven application, meaning that we detect events in real-time, as soon as they happen. In contrast, traditional mining techniques function in a query-driven manner where results are only created at intervals, such as daily analyses of the log files.

7. RELATED WORK

In Web advertising there are essentially two main approaches, contextual advertising and behavioral advertising. Contextual advertising [6] is driven by the user's context, represented usually in the form of keywords that are extracted from the Web page content, are related to the user's geographical location, time and other contextual factors. An ad provider (ad serving service) utilizes these meta data to deliver relevant ads. Similarly, a users' search words can also be used to deliver related advertisement in search engine results page, Google's second pillar in online advertising. However, contextual advertising, although exploited today by major advertising players (e.g., GoogleAdsense, Yahoo! Publisher Network, Microsoft adCenter, Ad-in-Motion etc.), shows serious weaknesses. Very often the automatically detected context is wrong, and hence ads delivered within that context are irrelevant. For instance, a banner ad offering a travel deal to Florida can possibly be seen side-by-side to a story of a tornado tearing through Florida. This is happening because the context was determined using purely keywords such as "Florida, "shore" etc (i.e., without taking keyword semantics into account). While there are improvements in contextual advertising (e.g., language-independent proximity pattern matching algorithm [7]), this approach still often leads companies to investments that are wasting their advertising budgets, brand promotion and sentiment. In contrast, our approach utilizes semantics to cure major drawbacks of today's contextual advertising. Semantic Web technologies can be used to improve analysis of the meaning of a Web page, and accordingly to ensure that the Web page contains the most appropriate advertising.

The second approach to Web advertising is based on the user's behavior, collected through the user's Web browsing history (i.e., behavioral targeted advertising). The behavior model for each user is established by a persistent cookie. For example, Web sites for online shopping utilize cookies to record the user's past activities and thereby gain knowledge about the user or a cluster of users. There are several reasons why behavioral targeted advertisement via cookies is not a definitive answer to all advertisement problems. First, if a user, after browsing the information about an item purchases that item, he or she will not be interested in that particular good afterwards. Therefore, all ads and "special deals" offered to the user later while browsing that Web site are useless. Also, the short-term user interest should be detected more quickly (i.e., during the current user session). Displayed ads need to reflect current moods or transient user interest. For example, a user looking hastily to buy a gift of flowers is not interested in ads related to his/her long-term profile, created during previous purchases unrelated good or services. Further on, there are problems with cookies. Computers are sometimes shared and users get to see ads governed by other user's cookies. Finally, given the European Union's Directive and US legislation concerned with restricted use of cookies, behavioral targeted advertisement based on cookies is not a promising direction for Web advertising.

We believe that short-term profiling (in contrast to long-term profiles created by cookies) is a valid and possibly augmenting approach in terms of personalization and identification of the user's interest. We realize a short-term profiling using client-side Complex Event Processing techniques (cf. Section 2.2), and background semantics (cf. Section 2). Such profiles are automatically detected, are always up-to-date and fully personalized.

The work from [8] describes event processing for Web clients. Events are observed on the client; however, complex events are not detected in the client. All simple events are propagated to the server for detection of patterns. This incurs latency and reduced locality for the processing of events, so the advantages of client-side event processing are lost.

8. CONCLUSION

In this paper we present a novel approach for the personalized and dynamic ad delivery on the web. The approach is based on the complex processing of the semantically enriched events generated out of the user's interaction with web content. Additionally, the approach introduce the notion of "unusual behavior" as a criteria for determining the dynamics of the new ads delivery for a particular user. This work goes beyond the web advertisement use case – in fact it opens possibilities to build event-driven applications for the (Semantic) Web. We envision the future of the (Semantic) Web as a huge, decentralized event repository (the so-called Event Cloud in Event processing terminology), which will contain information about the real-time activities of different Web users. Such an event repository will enable different kinds of processing of the real-time information, making the Semantic Web really active, i.e. the environment can react and adapt itself on the signals sensed from the environment, connecting the Internet of Things with the Internet of Services, two basic elements of the Future Internet.

9. REFERENCES

[1] Schmidt, K.U., Stojanovic, L., Stojanovic, N., Thomas, S.: On enriching ajax with semantics: The web personalization use case. In: Proceedings of the EuropeanSemanticWeb Conference, ESWC2007. Volume 4519 of Lecture Notes in Computer Science., Springer-Verlag (July 2007)

[2] Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: Rdfa in xhtml: Syntax and processing. Online Resource. http://www.w3.org/TR/rdfa-syntax/ (October 2008)

[3] Schmidt, K.U., Stühmer, R., Stojanovic, L.: From business rules to application rules in rich internet applications. Scalable Computing: Practice and Experience 9(4) (2008) 329–340

[4] Chakravarthy, S., Krishnaprasad, V., Anwar, E., Kim, S.K.: Composite events for active databases: Semantics, contexts and detection. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: 20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings, Los Altos, CA 94022, USA, Morgan Kaufmann Publishers (1994) 606–617

[5] Liu, B.: Web Data Mining. Data-Centric Systems and Applications. Springer Berlin Heidelberg (2007)

[6] Kenny, D., Marshall, J.: Contextual marketing–the real business of the Internet. Harvard Business Review 78(6) (2000) 119–25

[7] Schonfeld, E.: Proximic signs deals with yahoo and ebay to turn product listings into contextual ads; taking on adsense. Online Article. http://www.techcrunch.com/2008/01/15/proximic-signs-deals-with-yahoo-andebay-to-turn-product-listings-into-contextual-ads-taking-on-adsense/ (January 2008) Last visited: August 2009.

[8] Carughi, G.T., Comai, S., Bozzon, A., Fraternali, P.: Modeling distributed events in data-intensive rich internet applications. In Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C., eds.: WISE. Volume 4831 of Lecture Notes in Computer Science., Springer (2007) 593–602