MICROFORMATS BASED NAVIGATION ASSISTANT
A Non-intrusive Recommender Agent: Design and Implementation

Anca-Paula Luca and Sabin C. Buraga
Department of Computer Science, "A.I. Cuza" University of Iasi – 16, Berthelot, 700483 Iasi, Romania

Keywords: Recommender system, prediction, microformats, semantic markup, web interaction.

Abstract: The multiple ways in which we rely on the information available on the web to solve increasingly more tasks encountered in day-to-day life has led to the question whether machines can help us parse the amounts of data and bring the interesting closer to us. This kind of activity, most often, requires machines to understand human defined semantics which, fortunately, can be easily done in today's web through semantic markup. The purpose of the proposed project is to build a flexible tool that understands the behaviour of a user on the web and filters out the irrelevant data, presenting to the user only the information he/she is most interested in, while being as discreet as possible: the user is required no preference settings, no explicit feedback.

1 PREAMBLE

Navigating the web each day, accessing numerous websites containing information from various domains can be overwhelming sometimes, making the obvious or the interesting hard to get, due to the huge amount of useless data surrounding it. The explosion of the amount of information on the web from the past few years has satisfied our need of information, but also invaded our web lives with considerable amounts of unnecessary data we must surf through in order to get to the things we really want.

In the actual stage of the web, some patterns in published information and users' requests have emerged: there are numerous sites for blogs, social groups, collaborative bookmarks, collaborative knowledge, products/companies presentations, news portals – all aligned to the social web (O'Reilly, 2005; Shadbolt, Hall & Berners-Lee, 2006). There also are a lot of users taking advantage of this information: relying on the web for communicating with friends and family, researching different topics, staying up-to-date with most recent news and events.

In this context, semantics tend to repeat on the web: either it is the semantic of the published information (a publisher's point of view); either it is the semantic of the needed information – a user's point of view. The patterns are reflected in the markup of the information as well: resembling elements, attribute names and values, and imbrications can be noticed (Celik & Marks, 2004).

In the actual circumstances, the microformats initiative (Microformats, 2008) tries to specify a frame for this kind of patterns: standards of publishing information with these most frequent semantics (thus semantically marking the data) and push the web to the new stage where information is equally accessible for humans as well as for machines. With more and more websites adapting their markup to follow these standards to produce semantic markup, the idea of a tool that comprehends and simulates a user's behavior on the web becomes a need, it gets consistent and closer to implementation. Such an instrument can increase the efficiency of a user's sessions on the internet, measured as a proportion of the information gained per time spent on internet.

The existing tools focus either on microformats processing – detection, presentation and storage – without trying to assimilate them from a semantic point of view, either on the detection of semantics using the "classical" semantic web methods (such as RDF description of metadata and/or ontology specifications), or parsing hypertext as ordinary text, using standard text classification methods. Most of the tools require as well configurations, training data or explicit feedback from the user.

The innovations of our approach are in using microformats as semantic sources in the task of

Luca A. and Buraga S. (2008).
MICROFORMATS BASED NAVIGATION ASSISTANT - A Non-intrusive Recommender Agent: Design and Implementation.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - HCI*, pages 54-61
DOI: 10.5220/0001696600540061

"understanding" the web – for the arguments sustaining this decision see also (Celik & Marks, 2004). The purpose is to achieve our goal without human effort: the user is not requested to change his/her navigation behavior to adapt to the new tool or to provide it training data.

This paper is structured as follows: first, in section 2, we will define the microformat term and we will describe it's possible usage in the context. Then, in the next section, we will present the model used for data and the recommending system (Adomavicius & Tuzhilin, 2005) that constitute the foundation of the project. Section 4 focuses on the presentation of the application to the user: the user interface – design and, most important, interaction. After enumerating different related approaches, the paper ends with an outline of the discussed topics and presents the further research ideas.

2 MICROFORMATS

According to the *microformats.org* website, the microformats definition is: *Designed for humans first and machines second,* microformats *are a set of simple, open data formats built upon existing and widely adopted standards.*

A more accessible definition is the following: Microformats *are simple conventions for embedding semantics in HTML to enable decentralized development.*

Even more precise than this, microformats are conventions for XHTML (Extensible HyperText Markup Language) elements names, attribute names and associated values, with precise semantics – see also (Allsopp, 2007), (Haine, 2006) and (Suda, 2006).

2.1 Important Features

The key principles in designing microformats are the simplicity – they are designed to solve a specific problem – and the loose connectivity – they represent small pieces loosely joined together to form larger blocks and to express increasingly complex semantics, without decreasing their semantic expressivity through connections.

Microformats achieve their goal either by adding to the (X)HTML markup (elemental microformats), either by specifying a set of attribute values for XHTML existing elements and imbrications of such elements to be the frame for a piece of content (compound microformats). Certain microformats are definitively specified, while others are in work in progress.

Regardless of this, microformats are widely spread – either explicit or through the semantic of content and similar structure of markup, with the possibility of actually being explicited.

2.2 Representative Microformats

The list of the current official microformats is: *hCalendar, hCard, rel-license, rel-nofollow, rel-tag, VoteLinks, XFN, XMDP, XOXO, adr, geo, hAtom, hResume, hReview, rel-directory, rel-enclosure, rel-home, rel-payment, Robots Exclusion, xFolk.*

The microformats useful for a navigation assistant are the ones that encapsulate the content as well as properties of the specific content:

- *rel-tag* specifies that the current page or a portion of is marked with a tag. The tag for a piece of content is, usually, a single word that expresses a keyword for the content, or the topic of the content. It is a frequent practice to use multiple tags for a piece of content.
- *geo* allows the description of a location using geographic coordinates (latitude and longitude). This microformat can be embedded into other microformats such as *hCard* or *hCalendar*, to mark the location of an entity or an event.
- *adr* specifies an address, properly marked with fields for country, city, street and so on. This microformat is also embeddable into other microformats such as *hCard* or *hCalendar*, either joined or not by a *geo* microformat.
- *hCard* denotes a full description of an entity: a person (most often), an organization, a company, etc. It specifies fields for the name of the entity, the nickname, an address, a website and other information.
- *hCalendar* encapsulates a calendar entry (an event): date, description, address, etc.
- *hReview* is defined to be used in publishing reviews for different items. It contains fields for title, description, *hCard* of reviewer, *hCard* of reviewed, date of review, etc.
- *hAtom* mirrors the Atom syndication method, enabling the embedding of an Atom feed in (X)HTML.

Other details are provided by (Allsopp, 2007) and (Suda, 2006).

2.3 Example

The following is an example of using *hCalendar* to mark the ICEIS 2008 conference:

```
<div class="vevent">
  <!-- event description -->
  <a class="url"
   href="http://www.iceis.org/">
    <span class="summary">
 ICEIS '08
    </span>
  </a> &ndash;
  <div class="description">
   10th International Conference on
   Enterprise Information Systems
  </div>
  takes place between
  <!-- event date -->
  <abbr class="dtstart"
      title="20080612">12</abbr> and
  <abbr class="dtend"
      title="20080616">16</abbr>
   June 2008 in
  <!-- event location -->
  <div class="location adr">
   <span class="locality">
        Barcelona
   </span>
   <span class="country-name">
        Spain
   </span>
  </div>
</div>
```

Note that it is easy to extract the information of the event by both a human user (who will actually see the result of the browser processing of the HTML) and an automated tool (that can process markup to extract the URL of the event, a description, a summary and the dates for the event).

3 PROPOSED RECOMMENDING SYSTEM

In this section, we will present the models used for determining the users' preferences and the prediction algorithms.

We start with a presentation of the collected data and the associated data model and then we describe the definition the program associates with the notion of preferred content and the methods used to determine which piece of content is of interest and which is not.

3.1 Data Model

Although there are multiple sources for content information in a web page, our purpose is to build a tool that will only use content semantics provided by microformats markup. Because of this, the following discussion will refer to microformatted web pages – documents that have associated various microformats.

A web page is assumed to be composed by one or more blocks of data: pieces of content that belong each to certain categories of topics (one topic or more) and, most important, which are separable from the rest of the blocks – a program can identify and extract such a block from the web page.

Consider, for example, a web page that contains news and each piece of news is properly marked (by using *hAtom* or *hCalendar*) – see also Figure 1. All the pieces of news represent blocks of data in the model described above.

Such a block of content is considered to be the unit of content: from a web page, the algorithm will recommend one or more such blocks of content (in the presented example it seems natural to recommend a piece of news, as the unit of information). Naturally, a block will be considered to be the piece of content encapsulated by a microformat (e.g., *hCalendar*, *hAtom*, *hReview*, and *hCard*).

Given a web page, the flow of the recommending program is the following: first, the blocks are identified in the web page. Then, for each block, a preference score is computed (the algorithms are described in section 3.2) and then the recommendation algorithm is updated to take into account the new blocks as well.

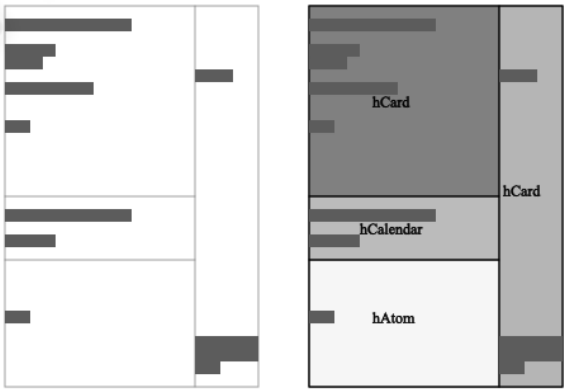

Figure 1: Detecting blocks of data.

3.2 Algorithms

Certain assumptions have been used in order to build the preference model – partially elaborated from (Chakrabarti, 2003):

- The entire semantic markup is semantically correct: any value assigned, through markup, to

an attribute of the content is the real value of that attribute, for the specific content. This assumption might not always hold (it depends on the quality of the markup) but there is no general way to detect or correct such errors in an automated manner.

- In the context of liking or disliking content, a user operates with blocks, as defined in the subsection above. The validity of this assumption is highly correlated with the way the blocks are built, which in turn depends as well on the quality of the markup.
- The user's preferences are expressed in terms of attributes values rather than items and the user tends to associate a greater importance to some attributes.
- A user accesses more often items he/she likes and ignores items he/she does not like.

In consequence, we propose a solution inspired from machine learning methods (Mitchell, 1997) that estimates the degree of interest for a given item by combining two components: the preference for an attribute and the preference for the particular value of that attribute, for the current item:

1. For each property of each type of block, an associated weight is computed as follows: initially, all weights are set equal for all attributes and then they are adjusted using the following assumption: the more the user "trusts" an attribute, the more liked items specify a value for that attribute (the user looks for the items that specify values for the properties he/she trusts more).
2. For each attribute and a given value, we can compute the user preference for that value as the probability that the value is among the values of that attribute for all preferred items.
3. The final estimation of the interest for an item is the weighted sum of values described at 2, by using the weights explained at 1.

We have opted for a memory-based approach: items are stored and then the needed values are computed when a new estimation is required – instead of transforming data into hypotheses of preference and then testing the items against them – for flexibility and data reusability reasons.

This algorithm is able to handle a set of issues that might appear in the recommendation process:

- It can efficiently learn the preferences incrementally, without the necessity of providing "training data". Of course, the results in the first stages will be poor but, as data is collected, they will improve;
- The assumption made about the user behavior regarding liked and disliked items allows the algorithm to work, even if no explicit feedback is provided by the user. As we have mentioned before, this is a key feature of the designed tool;
- By simply setting an expiring date for the stored items (after a period of time they no longer influence the recommending process), the algorithm can deal with dynamic preferences: a user's preferences can change and the agent must change its recommendations accordingly.

3.3 User Interaction

One of the first requirements regarding this aspect is the fact that the application must present its suggestions in real time (when a new web page is accessed) and this must not be done in a *master* manner: the agent only *suggests* the content to be accessed, not *dictates*.

Related to these aspects we will discuss two options for the application: a standalone application and a browser extension.

The interaction model for this application is the *observable-observer* one: first the agent (the observer) must be notified that the user (the observable) has accessed a new web page and then the user (observer this time) has to be notified about the agent's recommendations (the observable). Two types of observing can be used: *push* (the observable notifies the observer when the state changes) and *pull* (the observer reads the observable's state from time to time). We can assume that, in both cases, a push paradigm can be used for obtaining the current web page (the user must not notify the agent that a new page was accessed) and this is the method to be used in any of the cases since the agent would not require pointless user assistance.

As a standalone application, the agent exists outside the navigation process, in a browser external application. If the agent is built as a push observable then it will have to notify the user, from time to time, that its state has changed (using a dialog window or any other method), which can disturb the user from his/her navigation activity. If the agent is designed using a pull paradigm, thus eliminating the disturbing factor, then the user would be required to switch from one application to another to check the state of the agent, which would causes an overhead big enough to eliminate this option.

The most important note to be made in the case of a browser extension is that the agent can act like *a component of the navigation process itself*, transparent to the user, whose behavior is not

required to change to manage the new application he/she is using.

Clearly, the best option is the implementation as a browser extension, but there are a few other choices to be made, regarding the way the application presents its results to the user. We have chosen to highlight the interesting blocks (using background colours, tool-tips and combinations of the two), in the web page, in the exact place where they appear, for a sum of reasons:

- The model of this interaction corresponds to a push-pull combination: the agent is a push observable that allows a pull behavior from the user: it will alter the web page highlighting the interesting items, but will still provide the whole page to the user so he/she can access any piece of content he/she feels to;
- It does not use page space as a side panel would (frequently used by many applications);
- It does not require a change of focus from the page itself to another area of the screen;
- The recommended items can be observed as the page is scrolled down, it does not require permanently referring a recommendations list;
- Though somehow discrete, the results returned by the application still stand out from the rest of the web page, thus allowing one to easily notice them if this is the desired goal.

Optionally, the recommendations list can be also displayed in a side bar list, for faster access of the user to the recommendations and a greater freedom of choosing the manner he/she browses the recommendations: either as a *pull* or as a *push* observer.

4 ASPECTS REGARDING THE IMPLEMENTATION

We will describe our approach to design and implement a “smart” navigation assistant, using the new web technologies and aiming the largest possible public. The goals are the platform independence and the wide spread standards.

Two main modules of the application can be distinguished: a module that interacts with the navigation process (data collecting and results display) and the module that encapsulates the recommendation engine (storage and prediction).

The implementation of the two as loosely connected components is a key feature, thus enabling independent testing and improvement, and also facilitating reusability in similar contexts. Loose connectivity has been achieved in this situation through a moderator component (a “data manager”) that enables bidirectional communication between the two modules.

An illustration of the architecture presented above is depicted in Figure 2.

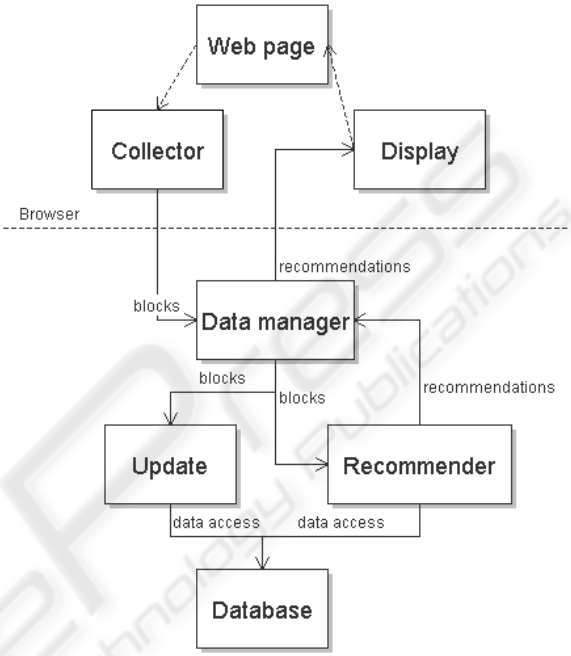

Figure 2: Modules and data flow.

4.1 Data Collecting and Display Module

This component must be analyzed in the context of the extension development interface available from the browser.

As a platform at this level, we have chosen the Firefox web browser, whose advantages are in the direction of wide platform availability, extended support for current web standards and the encouragement of extension development through the clear structure imposed for the sub-application, the large development community and documentation and the possibilities of communicating with external modules (native libraries, Java programs).

The implementation of the collecting and display module takes some advantages from a browser embedded approach. Data collecting is done using the web documents downloaded by the browser; no new Internet connection and distinct request to the server are required for this activity. Extracting the

data from HTML documents takes place at the DOM (Document Object Model) tree level, built by the browser for its internal use but made available to the extensions developers. Displaying the results takes place in the same context of preprocessed data, thus decreasing the running time and the complexity of the programs.

We can now summarize the activity of the agent as follows: when a new web page is accessed, the DOM tree is processed in search of microformatted content and a list of blocks is built. This list is then sent to the data storage and prediction module. When the response from this module arrives (which represents a list of preference scores associated with the blocks), the display module is being called to modify the properties of the DOM tree associated with the page to highlight the recommended blocks.

4.2 Data Storage and Prediction Module

The data storage module relies on the physical storage engine which is, in our case, a native XML database engine, chosen for the management of concurrent access, the advantages in storing unstructured data and the programming interface based on actual XML technologies. Our choice was *Berkeley DB XML* (Brian, 2006) which is a fully embeddable database engine, available on multiple platforms and which provides programming interfaces in a large variety of languages.

The prediction sub-module implements the algorithm presented in the section 3.2, as a standalone module – most liable to future improvement –, in a platform independent language to increase portability. A solution for this implementation is the Java language that offers platform independency, compatibility with the XML documents management libraries (the native XML database, Saxon XML processing library), and moderate requirements for installed interpreters on the client machine.

Thus the activity of this module can be summarized as follows: when the data manager receives – from the collecting module – a list of blocks, it sends it to the recommending module whose results are stored in order to be sent to the display module. Then, the blocks are sent to the update module and finally, the response for the browser is elaborated from the recommending module's results and sent.

4.3 Scenario

A usage scenario for such an application is as follows: when the user accesses a new webpage, the tool analyzes the content of the page and discovers the blocks of data the user might be interested in.

Then, it highlights these blocks, and attaches tool-tips containing related preferred data – see Figure 3.

The blocks are marked in the place where they appear on the webpage, thus minimizing the interference of the agent in the navigation process.

Figure 3: Highlighting the information of interest.

The results of such an activity are a decrease in the user effort to parse large documents in search of the interesting content, an increased efficiency in discovering preferred content (blocks that would have been missed otherwise can be emphasized through the use of such a tool), and better semantic comprehension in the process of discovering new data on the web, provided by the permanent connections between current data and previously accessed information.

5 RELATED APPROACHES

5.1 Tools

We will briefly describe in the following some of the tools that serve purposes related to this application:

- *Tails* is a Firefox extension that collects microformats from the web page and allows users to execute miscellaneous actions through the Tails scripts.
- *Operator* is a Firefox extension as well, that brings as improvement the combination of the microformats with other services aligned to the social web: Del.icio.us, Flickr, Google Maps, Google Calendar or Upcoming.org.
- *Greasemonkey* is a Firefox extension that allows the execution of user scripts, offering – via JavaScript programs like *microformat-find-gm5* and *XFN Viewer* – support for microformats extracting and processing.
- The Firefox browser considers, for future versions, the detection and parsing of microformats.
- *Magpie* (Domingue, Dzbor & Motta, 2004) is a tool that proposes a semantic navigation by detecting, from a web page, all the items (words) that correspond to a user specified ontology.
- *WebIC* represents a recommender system which, using the words from the visited documents, determines the user preferences and helps the user achieve his goal by retrieving similar documents containing similar content.

5.2 Websites

There are various websites that use microformats in the generated markup, and their number is continuously rising (the expansion of microformats is facilitated by their fast assimilation by all web developers with basic XHTML knowledge), thus contributing to the success of microformats a source of semantics for the web. A list of implementations is online available – for example, *Upcoming.org* and *Last.fm* which use *hCalendar* to mark events, *Yahoo! Tech* and *Cork'd* use *hReview* for products and services review, many social websites (including *Last.fm*, *LinkedIn*, *Flickr*) use *hCard* to mark the profiles of the users.

In this context we must also mention that there are various tools dedicated to the microformats authoring and publishing – from editors to content management systems of blogs (for example, *WordPress*) and wikis like *XWiki* (Dumitriu, Girdea & Buraga, 2007; XWiki, 2008) –, thus widening the set of possible microformatted content authors.

6 CONCLUSIONS

6.1 Contributions

One of the main goals of the microformats initiative is to facilitate the machine access to the information published by humans, to enable them to assist humans in the web browsing process.

Although tools that use microformats have been developed with the emergence of microformats, the attempts have settled in extracting – manually or semi-automated – the (meta)data marked through microformats, leaving the comprehension and semantic parsing to the human users. Our paper takes a step further and tries to emphasize automated semantic detection as the first usage of the semantic markup, in the context of human computer interaction.

Such an approach has real applicability in better web browsing experience (by increasing efficacy), information retrieval, products or services recommendation, social networks recognition, user assistance for various tasks and others.

6.2 Further Development

An important direction to follow is towards collaborative recommending: the application can automatically correlate two users based on the detected browsing preferences, and can use these correlations to improve its recommendations – using collaborative filtering (Chakrabarti, 2003) or association rules.

Also, the recommending principles presented in this paper can be improved by elaborating superior category building techniques (by taking into account new properties or new similarity measures).

Since microformats are in continuous evolution and new microformats proposal is open to the web community, the specification of new structures that would encapsulate new semantics represent a possibility of improvement for the dedicated tools, by offering access to data which is presently "hidden" by the lack of appropriate markup.

By combining this type of instrument – that uses exclusively microformats – with the applications focused on the "classical" semantic web methods and with standard text classification methods, we can go further in the direction of processing all kind of information on the web: practically, any page on the web could be understood by the navigation assistant, thus achieving one of the goals of the "new web" – according to (Khare & Celik, 2006) and

(Shadbolt, Hall & Berners-Lee, 2006): the equality of humans and machines as information consumers.

REFERENCES

Adomavicius, G., Tuzhilin, A., 2005. 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*, 17(6).

Allsopp, J., 2007. *Microformats: Empowering Your Markup for Web 2.0*, Friends of ED.

Brian, D., 2006. *The Definitive Guide to Berkeley DB XML*, APress.

Celik, T., Marks, K., 2004. Real World Semantics. *ETech Conference*, O'Reilly.

Chakrabarti, S., 2003. *Mining the Web – Discovering Knowledge from Hypertext Data*. Morgan Kaufmann.

Domingue, J.B., Dzbor, M., Motta, E, 2004. Collaborative Semantic Web Browsing With Magpie. In *Proceedings of the First European Semantic Web Symposium (ESWS), Lecture Notes in Computer Science – LNCS 3053*, Springer.

Dumitriu, S., Girdea, M., Buraga, S., 2007. Knowledge Management in a Wiki Platform via Microformats. In *FLAIRS 2007 Conference Proceedings*, AAAI Press.

Haine, P., 2006. *HTML Mastery. Semantics, Standards, and Styling*. Apress.

Khare, R., Celik, T., 2006. Microformats: a Pragmatic Path to Semantic Web. *Proceedings of the 15th International Conference on World Wide Web*, ACM Press.

Mitchell, T., 1997. *Machine Learning*, McGraw-Hill.

O'Reilly, T., 2005. *What is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software*. O'Reilly.

Shadbolt, N., Hall, W., Berners-Lee, T., 2006. The Semantic Web Revisited. *IEEE Intelligent Systems*, 3(21).

Suda, B., 2006. Using Microformats. *O'Reilly Short Cuts*.

Atom Specifications, 2008. http://www.atomenabled.org/.

Berkeley DB XML, 2008. http://www.oracle.com/database/berkeleydb/xml.

Firefox Extensions, 2008. https://addons.mozilla.org/firefox/extensions/.

Magpie, 2006. http://kmi.open.ac.uk/projects/magpie/main.html.

Microformats Support in Firefox. http://wiki.mozilla.org/Firefox/Feature Brainstorming:Microformat Handling.

Saxon, 2008. http://saxon.sourceforge.net/.

Microformats Initiative, 2008. http://microformats.org/.

WebIC, 2007. http://www.web-ic.com/.

WordPress, 2008. http://www.wordpress.org/.

XWiki, 2008. http://www.xwiki.org/.