ActiveTags: Making Tags More Useful Anywhere on the Web

Stephan Hagemann **Gottfried Vossen**

European Research Center for Information Systems (ERCIS)
University of Münster
Leonardo-Campus 3, 48149 Münster, Germany,
Email: {shageman,vossen}@uni-muenster.de

Abstract

Tags in social tagging systems store meaning for the taggers who have entered them, and other users often share this understanding. The result of this, a *folksonomy*, is typically used in several ways, including information retrieval and clustering, serendipitous information access, or visualization of folksonomic characteristics. For these uses tags work pretty well; however, the ambiguity of tags makes it difficult to use them for more than searching and browsing. This paper introduces examples of current programmatic support in the form of mashups and highlights its shortcomings. It identifies several types of tags based on their structure and language, and discusses how these types support programmatic uses. The main part is the presentation of the ActiveTags system, a browser extension with supporting server infrastructure. Using it the same community process that creates a folksonomy can be used to enhance tags with programmatic meaning. Users are enabled to create reliable mashups based on tags. Effectively, this leads to customized views of Web pages with tagged content. ActiveTags naturally increases the usability of social tagging systems and further extends the notion of user-generated content.

Keywords: Social tagging; mashups; data as a service; information retrieval, filtering, and dissemination.

1 Introduction

Social tagging systems are part of numerous sites all over the Internet (Hammond et al. 2005, Lund et al. 2005, Marlow et al. 2006) and they are a central component in current Web 2.0 developments of the social Web (Hinchcliffe 2006). The objects that are tagged are quite diverse, ranging from bookmarks, photos and videos to products. The systems all share the fact that users can freely choose tags to be attached to objects, they can typically see tags that others have attached, and they can subsequently use all these to search and browse through the resources on a site. While this works quite well, the ambiguity of tags generally prevents more specific uses – the meaning of a tag is obvious to its creator but not necessarily to others. There are some examples of tags being used as a basis for mashups, which then specifically and effectively support linkages between information source and operations on their contents. Yet currently, these examples are few, have a very limited scope, and are implemented and enforced by the site operator. The ActiveTags system we propose in this paper allows users to implement their own mashups based on tags and use mashups of other users; tags can be used regardless of their Web location, thereby effectively allowing Web pages to be customized and enhanced by their users.

Two examples of support for specific tags are present on Flickr, a photo sharing site which allows the owners of photos to tag pictures, and to share them with friends depending on settings. So-called *machine tags* were introduced in January 2007 (Straup Cope 2007), which marked the beginning of special treatment of tags of the form "`namespace:predicate=value`." More functions are supported for machine tags when the Flickr API is used (`flickr.com/services/api/flickr.photos.search.html`). Earlier, the site had begun to support two sets of specific tags: geotags and event tags: (1) Geotags are a combination of three tags that together indicate the presence of a geotag and encode a geographic location. Two of the tags are machine tags, which encode the longitude and the latitude and can thus be used to position objects on a map. The site allows these two tags to be exported into structured data fields for geographic information, so that the pictures are automatically shown on maps also. (2) Tags with the prefix "`upcoming:event=`" are interpreted as links to events on Upcoming.org, with the effect that a link to the corresponding event is included in the page.

These two tag types essentially create mashups of Flickr content with related external content, and they are great features of tagging on the Flickr site. Similar examples are present on other sites, e.g., Delicious.com uses a special tag to send bookmarks from one user to another. While these specific forms of tag support are reasonable steps towards a use of tags for accessing relevant related information, they have several shortcomings:

Nature of tagging. The meaning of the tags is effectively enforced by the functions connected. An essential aspect of folksonomic tagging is that the vocabulary is free. In general, it should be possible for different understandings to coexist.

Flexibility. The tags that were chosen to be supported have been chosen by the site; extensions are only possible if the site operator implements them. Yet new understandings continuously arise in social tagging systems, and they should not depend on site operators to be effectively supported.

Note that for Flickr and Delicious this does not hold for uses through their APIs, where new functions can be implemented by users, although not within the confines of the original site and not without certain programming skills.

Scope. The tags we have discussed above only work on individual sites, although the outlined forms of tagging occur on other Web sites as well. It should be left to the users to decide what the scope of their tag usage and subsequent mashups is.

Users. As it is the users who generate tags, define their meanings, and make use of them, they have to be taken into account when it comes to programmatic support.

It is these shortcomings that the ActiveTags system is designed to overcome; corresponding support is the guideline for our development. Users should have a wide range of possibilities when it comes to selecting which interpretations they subscribe to. This includes which mashups they want to use, the support should be flexible so that new interpretations (mashups) can be included as the use of tags evolves, and it should be up to the users to define in which contexts interpretations should lead to actions (and mashups be shown). These contexts may, for example, include settings based on Web pages, time, or a user's social network. We have implemented ActiveTags as a browser extension to enhance a user's browsing experience based on the above principles.

The contribution of this paper is twofold: First, we will show how the ActiveTags system is designed in order to fulfill the principles mentioned above, and second we will show how the ActiveTags browser extension technically works, explaining how mashups can be created based on tags. To this end, the organization of this paper is as follows: Section 2 analyzes the current practice of tagging. Section 3 outlines the ActiveTags system design and introduces our prototype. Section 4 discusses which research findings and which practical developments are relevant for our work. We give our conclusions as well as an outlook in Section 5.

We are currently conducting an evaluation of our approach and the prototype implementation. Since that work is still ongoing, we have opted for presenting our concept independently here. We will report on the results of the evaluation in an upcoming paper.

2 On the Practice of Tagging

We have analyzed random tags from Flickr to evaluate what types of tags occur and now discuss whether these can be used to support mashups effectively. As opposed to other analyses such as Golder & Huberman (2006) we have not looked at the roles tags play, but at their syntax and whether they are from natural language dictionaries.

Flickr has several ways of storing tags. We have accessed the so-called *raw* version of a tag, which can be easily extracted via the Flickr API. Its disadvantage is that whitespaces contained in tags are not present in this form; thus, if tags are comprised of multiple words, these cannot be detected. 45,924 photos with at least one tag were randomly extracted. This has returned the tags of photos from 24,375 distinct users, with 490,561 tags in total or 4.43 tags per photo on average. We have analyzed whether tags belong to one of the four languages English, French, Spanish, or German, which we believe are currently the most popular languages on Flickr. This has been done by checking them against the respective aspell dictionaries (see `aspell.net`). We have divided all remaining tags into "machine tags" and "other tags." The results are summarized in Table 1.

The largest group, almost 50% of all tags, contains tags from natural languages. This percentage is likely to be even higher (as we will see below) when one takes into account more languages and treats multiple-word tags differently. Machine tags make up for only 1.63% of the tags found. Together with the number for tags per photo, we can assume that machine tags are present roughly for 1 out of every 14 photos.

Type	Absolute	Percentage
Language tags	244,692	49.88
English	222,145	45.28
French	68,236	13.91
German	61,069	12.45
Spanish	48,172	9.82
Machine tags	7,978	1.63
Other tags	237,891	48.49
All Tags	490,561	100.00

Table 1: Types of tags (based on all occurrences).

Table 2 contains the five most popular tags from each of the three categories. Due to their structure, machine tags, as we have seen above, can be used effectively for storing structured data in tags. In Table 2 we have based our frequency calculation on the name space and property parts of the tag only. As can be seen, geotags are by far the most popular machine tags, which is most likely due to the fact that these tags are supported by the Flickr site. We take this as evidence that programmatic support increases the incentive to use tags.

The question now is: Are we dealing with less than two percent of all tags, or can any of the other tags be used for building mashups as well?

An example of a mashup that uses ordinary tags to achieve something similar to geotagging is fotoland.us. It uses tags as the basis for a mashup of photos from Flickr and the maps provided by Google (`maps.google.com`). Instead of geotags it uses the names of countries and cities, but can also handle additional tags added by its users. Although the site often achieves good results, there tend to be photos in result sets where the tags have been interpreted the wrong way: For example, for the German city of *Essen*, which is also the German word for *food*, it is quite common to find photos that were actually taken in other cities, but that show food, people having food, or something similar.

Although this may be rare, problems with cities that have the same name are very common. As anecdotal evidence, take Athens, Texas. If this town were only tagged with "Athens" most pictures that get connected will actually be from Greece. If one tries to disambiguate the tag by tagging "Athens, US", the results get better, leading to some pictures from the city in Texas, but also pictures from Greece with people on them. Here the meaning of "US" was obviously not the country but *us* as in "we are in the picture." Finally, tagging "Athens, Texas" yields very good results. So, with proper disambiguation even common terms can be used as a reliable basis for mashups.

The five most popular tags in our sample which belong into the "other" category show what types of tags this category contains: Obviously there are abbreviations (bw - short for black & white photography), portmanteau words (geotagged - see above), tags composed of multiple words ("a big fave" – with "fave" being a short form of favorite, and "black and white"), and numbers (2006 - most probably denoting the year). Portmonteau XXX? Portmanteau???

#	Dictionary	Machine	Other
1	portrait (1333)	geo:lat (2455)	bw (3089)
2	canon (1250)	geo:lon (2396)	geotagged (1852)
3	me (1238)	flickr:user (422)	abigfave (1476)
4	nikon (1218)	dc:identifier (418)	blackandwhite (1110)
5	bravo (1120)	xmlns:dc (418)	2006 (1070)

Table 2: Five most popular tags by category.

words are commonly used to uniquely identify events. Often used for conferences, examples include "barcampcologne", "barcampsydney07", and "drupalconbarcelona2007." For example, searching for the last term yields photos, slides, blog entries etc. related to the Drupal Conference 2007 in Barcelona.

We conclude that there are several groups of tags based on syntax which can reliably be used as metadata. The quality of this data is in the hands of the users. It is our hope that with the support of ActiveTags the awareness for good metadata will increase, as will the quality of the metadata.

3 The ActiveTags System

3.1 Tag-mashup lifecycle

The introduction of tag support for maps and events on Flickr was preceded by discussions in the communities of Flickr and other sites (Straup Cope 2007), and by external developments of services supporting the tags (Andrews 2006). As such, the implementation of terms depended on the process of the term development from inhibition to common understanding. The mashup development was then taken care of by the developers behind the site. In order to support a community in going through similar processes and the system to react flexibly, users have to be empowered to do so. They have to be able to define their understandings of tags, must be able to implement mashups, and to control the connection of the two.

The goal of ActiveTags is to support what could be called a "tag-mashup lifecycle" as depicted in Figure 1: (1) The starting point is an individual interest in expressing information which is not explicitly available. In this stage there is no common understanding of pragmatics. (2) The ActiveTags system allows the understanding to be expatiated through the implementation of a mashup. (3) The understanding spreads from individual to common, i.e., over the social network of a user. (4) This stimulates new terms to arise, which in turn spawns the creation of new mashups by restarting the cycle for other terms. This cycle is not to be made mandatory, but transitions between the steps are to be made as efficient as possible.

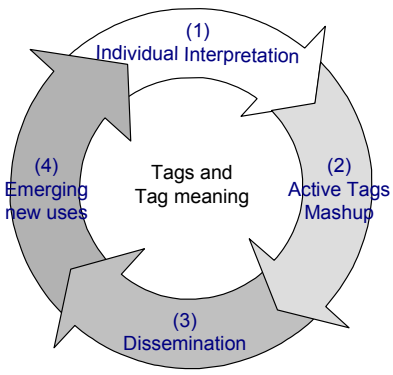

Figure 1: ActiveTags dissemination of mashups.

3.2 Building a tag-based mashup system

The four principles we have laid out above guide the design and implementation of the ActiveTags system; these were accordance to the nature of tagging, flexibility in control, user-defined scope, and explicit user modeling. The scope requirement has two aspects: to technically allow for a maximal scope, while enabling the user to restrict it. At this point we want to enable users to employ tags over the entire Web. Focusing on the Web does leave out programs that also use tagging (such as file tagging in Mac OS X and Microsoft Vista), but are not part of the Web. However, in doing so, we can rely on HTML as the dominant language for transporting content, and on the browser as the dominant way of access to the Web. Because of this, we have implemented ActiveTags as a Firefox extension that interacts with the Web pages a user visits to enhance them with tag-based mashup support. As a consequence, the design that is presented below is concerned with the Web context, but could be easily formulated for other contexts as well. Let us first look at an example of the intended user experience before we discuss the design of the ActiveTags system and its browser extension.

3.2.1 A sample page enhanced by ActiveTags

To exemplify the feasibility of our approach we have re-implemented the two functions which, as we have mentioned earlier, are available on Flickr as ActiveTags mashups. Figure 2 shows the screenshot of a Web page viewed in Firefox with the ActiveTags extension enabled and its results highlighted.

The Web page is a typical detailed photo page on the Flickr site. The dashed red lines indicate what the ActiveTags extension is working on. Tags are present on this page in the right column (1). They have been highlighted, indicating that ActiveTags has recognized them. As can be seen, the photo has been geotagged and tagged with an Upcoming.org Id. Flickr neither shows a map nor a link to one because geotags have to be specifically imported first; only thereafter a link to a map containing the photo will be shown. The link to Upcoming.org that is provided by Flickr is visible under "Additional Information" in the right column (2).

The mashups of ActiveTags that are available for this page are visible in the large highlighted section titled "ActiveTags MergeSpace" in the lower half of the image (3). First, a map mashup shows where the photo was taken, the map itself being a fully navigable Google Maps object included as an IFrame (4). Second, there is also a link to the event at Upcoming (5).

In effect, there is not much difference between the functionalities provided by Flickr's native methods and ActiveTags' mashups on this page. However, scope and implementation are very different: Indeed, the ActiveTags mashups were created by users and they not only work on this sample page, but on arbitrary Web pages as well.

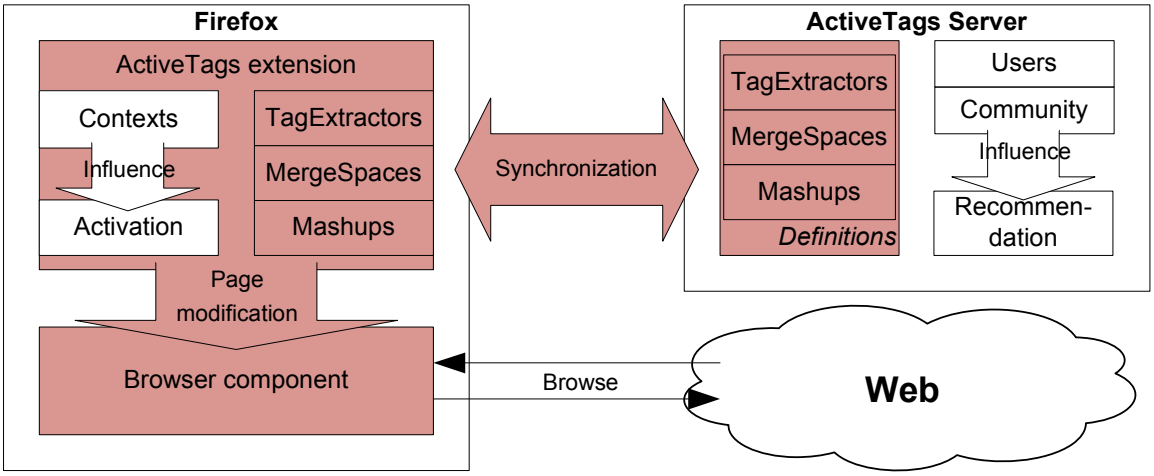

Figure 3: ActiveTags components.

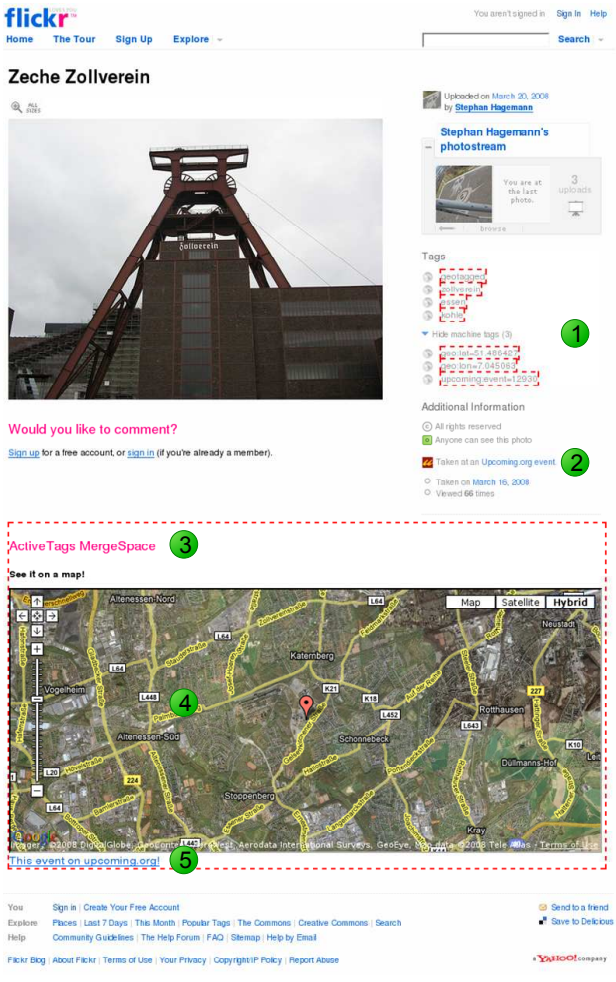

Figure 2: Sample page with ActiveTags mashups.

3.2.2 Design

Figure 3 contains an overview of the main components of the ActiveTags system. Depicted on the left is the user-side component of the ActiveTags system: a browser with the ActiveTags extension. The ActiveTags server (depicted on the right) stores the global database of definitions and social aspects, that is, users and community.

The *ActiveTags extension* is the hub for client interaction and it fulfills several functions. In executing the *ActiveTags routine*, it handles the visible part of ActiveTags' operations. The routine consists of the following steps:

1. Monitoring page loads,
2. extracting tags from the Web page,
3. checking for applicable mashups based on tags, URL, and settings,
4. finding a place on the Web page to merge mashups, and
5. including mashups into the Web page.

The process is aborted after steps 2, 3, or 4 if the previous step yields an empty result, resp. If, however, the process continues all the way to step 5, it leads to a situation similar to the one depicted in Figure 2, with tags detected, mashup integration space found, and mashups shown.

The extension regularly synchronizes its database of definitions with the server to support sharing among users. This encompasses TagExtractors (used in step 2), MergeSpaces (used in step 4), and mashups (used in steps 3 and 5). Finally, the extension offers users the possibility of creating new instances of these definitions. Contexts, the last portion of the extension, allow a user to specify his or her browsing context, which can be used to control the activation of different mashups in varying circumstances.

The *ActiveTags server* stores the global database of definitions. Through this database the extension instances exchange their definitions. Users can form links among each other that are equal to "friend" connections in social networks. Together with activation statistics the communities thus created influence the suggestion and ranking of definitions when they are proposed to the user.

3.2.3 Client implementation

The shaded parts of Figure 3 are those that have been implemented so far. We have opted to not implement contexts at this time, as a similar effect can

be achieved through profiles in the Firefox browser. Also, the community features have not yet been released, because we first want to grow the user base and the number of definitions. Even with good ranking algorithms, a small user base that has only a few definitions to work on may lead to arbitrary and useless results, which is why we currently take care of quality control by closely monitoring all newly created definitions.

The details of our implementation of the **ActiveTags extension** will follow the steps from the five-step procedure outlined above:

Monitoring page loads. The extension registers an event listener that waits for `DOMContentLoaded` events and thus implements step 1. These events are a Mozilla-specific and they fire after the content of a page, but before images are fully loaded. It is checked whether such an event is from an IFrame that was generated by ActiveTags. If this is not the case, the process continues.

TagExtractors are definitions that perform step 2 of the ActiveTags procedure. They consist of a regular expression for URLs, an XPath expression for element extraction, and an optional separator string.

The following example is an extractor for tags attached to photos on a Facebook photo page:

URL pattern: `^.*www\.facebook\.com\/photo\.php.*$`
XPath: `//*[@id='phototags']/span`
Separator: –

A TagExtractor's XPath is executed on any browsed page that matches the URL pattern, where, if present, the separator string is used to split up the contents of elements into tags (otherwise the entire element is considered a single tag). The results of all applicable TagExtractors are combined and fed into the next step of the process.

There are certain extractors that are not site specific, since they make use of standardized formats, which can encode tags. Therefore they are applied to all Web pages. Relevant formats include microformats, eRDF, and RDFa. In particular, we use the rel-tag microformat (Celik & Marks 2005). It specifies tags by annotating links with a *rel* attribute whose value is "tag". The XPath to extract tags with this microformat is `//a[contains(concat(' ',@rel,' '),' tag ')]`. Microformats are widely used: e.g., blogs created using major blogging tools such as those from `Blogger.com` or `Wordpress.com` encode tags for blog posts using this microformat.

The other formats are listed for completeness; they are currently not activated, since they are hardly used in practice. eRDF and RDFa allow RDF to be embedded into Web pages. GRDLL (W3C 2007) is a specification detailing how semantic data can be extracted from XML content. In the semantic data we would look for tag definitions as specified by the Newman tag ontology (Newman 2004).

If the tags on a page are not recognized by ActiveTags, users can create a new TagExtractor for it. TagExtractors are built as CSS selectors (W3C 2005) and converted into equivalent XPath expressions prior to storage and execution. Patterns are often easy to create, as tags are commonly presented on Web pages using lists or tables of links, or plain text versions of tags. Therefore, ActiveTags offers simple visual extractor creation with which users can point and click to create TagExtractors. Although this process can create only a very limited subset of selectors it is sufficient for many Web pages. Currently, only about 12% of TagExtractors in use have been manually corrected to cope with (partial) false positives.

The main part of the TagExtractor creation dialog is depicted in Figure 4: after a user opens the dialog, he or she starts creating a selector by moving the pointer over the copy of the Web page, which is displayed in the center (1). While hovering selectable elements under the pointer are highlighted. Once the user finds the tags and clicks on one of them a selector is created. Now, all elements that are covered by this selector are highlighted. The selector is also shown in the field *Selector* (2). Using the buttons that make up the selector, the selector expression can be generalized, i.e., parts can be generalized, or left out completely. This allows the user to widen the selector. Should multiple tags be captured by one HTML element, a separator can optionally be specified to separate them (3). Finally, the URL pattern on which the extractor should be working can be modified like the selector by clicking on the boxes (4). Submitting the extractor makes it available for immediate use and uploads it to the ActiveTags server so that it is available to other users as well.

Mashup definitions check for the applicability of mashups (step 3) in two ways: via tags and via URLs. The latter is an optional list of regular expressions on URLs that, if present, needs to contain at least one item that matches the current URL. The former is a list of patterns for tags, where each item specifies what tag pattern is looked for and what part of this pattern (if any) is to be used as a mashup parameter.

The following example, which checks for the latitude component of geotags, is such an item. It defines the expected tag pattern as a regular expression. Part of the tag is to be used as a parameter. By default, it is the outermost parentheses in the regular expression that define which part of the tag is to be used as a parameter: in this case it is the numerical component of the tag. Finally, the extracted part can be referred to using the parameter name, which here is "lat."

Tag pattern: `^geo:lat=(-?\d*[0-9](\.\d*[0-9])?)$`
Param? yes
Param name: lat

The other parts of a mashup definition are relevant for its execution and will be explained below.

Finding a place on Web page. Step 4 is performed with the help of MergeSpace definitions. These are very similar to TagExtractors, the difference being that MergeSpaces do not have separators. The rule to go from selected elements to the space where to include mashups is the following: Mashups are incorporated into a page after the last element that is selected by the XPath of the definition. If multiple MergeSpaces are applicable on a page, the first MergeSpace found will be used.

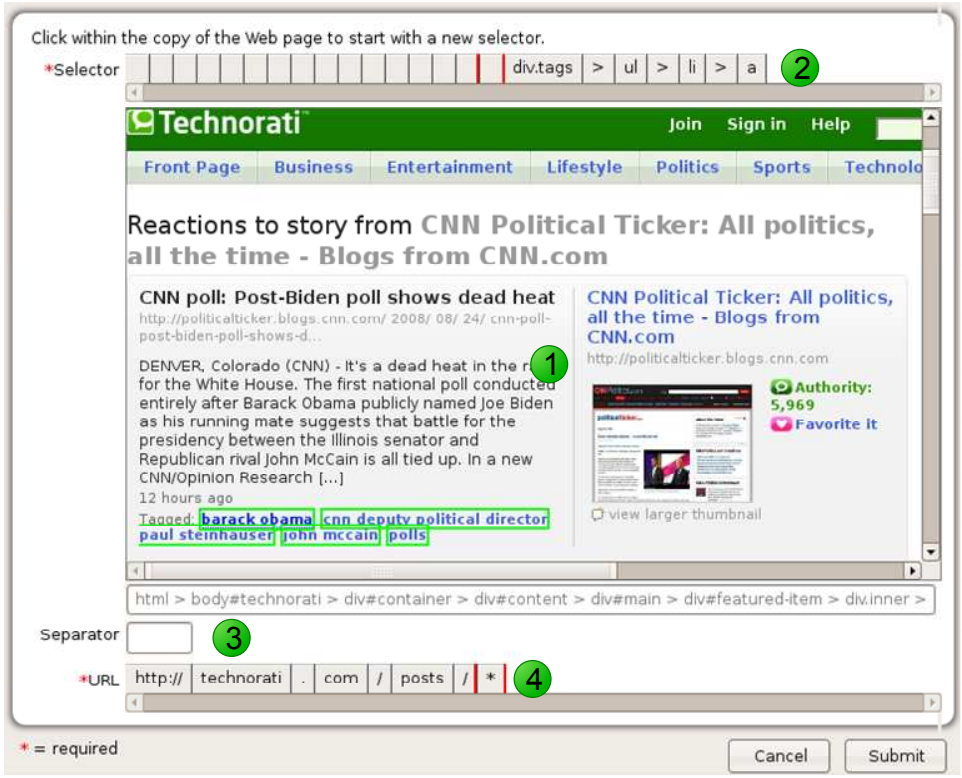

Figure 4: TagExtractor creation dialog.

Mashup execution. As seen above, mashups extract parameters from tags. In order to be able to execute, they also define where the Web procedure call (WPC, see (Vossen & Hagemann 2007)) endpoint implementing the mashup is located. For the example introduced above the URL is `www.bananathinking.net/ActiveTags/webServices/google-map/map.php?lat=%lat%&lon=%lon%`. At the end of this URL, enclosed in percent signs, the parameter names for "lat" and analogously for "lon" can be seen. For an execution of this WPC, these parameter names are replaced with the current values. Mashups are executed by a piece of activating code that is embedded into the Web page.

There are currently two types of mashups, which differ in their embed code: link and IFrame mashups. The former creates a link pointing to the URL discussed above, while the latter creates an IFrame for which the source is that URL.

3.2.4 Server implementation

The **ActiveTags server** has been implemented using CakePHP (`cakephp.org`). It currently resides at `bananathinking.net/ActiveTags/server`. It offers a management console for enabling, disabling, and modifying definitions. While access to most parts of the server is restricted, definitions can be up- and downloaded by ActiveTags extensions. A running ActiveTags extension will usually download new definitions from the server once after every browser start. There are options to change this default behavior and users can trigger this manually. The server delivers definitions as a JSON string, which is cached by the extension so that it is operable even should the server be unreachable. A newly created definition is uploaded to the server and will spread to other users when they restart their browsers or manually update their definitions.

The current implementation is running proof that the outlined system is functional; however, there are aspects of large, distributed systems which need to be addressed. The server implementation is performing all storage and distribution functions of the system. This centralized component will eventually become a bottleneck when the number of users grows. Therefore, the next step in the development will include an extension of the server towards a peer-to-peer architecture. We are currently investigating the use of the BitTorrent protocol. In this scenario, the server will continue to handle definition uploads, but distribute them not only directly, but also via the BitTorrent protocol. To this end, the server will provide tracker information for the definition files and be the seed of these files, while the Firefox extensions will act as BitTorrent peers, helping with the distribution of definitions. An approach of combined short-term individual updates and regular cumulative ones will balance timeliness and generated network traffic.

While TagExtractors and MergeSpaces are needed by every extension, the utility of mashups is very much dependent on what a user wants the functionality to be. Therefore, a second way to distribute mashups will be implemented in the future. Similar to the way Ubiquity distributes actions (see Section 4), it will be possible to include mashup definitions into arbitrary Web pages, highlighted in such a way that ActiveTags extensions recognize them and allow installation. This approach would simultaneously tackle conflicts between different requirements towards mashups and scalability of mashup distribution. At the same time it allows mashups to remain private, e.g., a business mashup using company internal sources.

Conflicts between definitions are also an issue with TagExtractors and MergeSpaces. They are tackled by a fine-granular recommendation system. Based on the numbers of users activating certain definitions, and explicitly stated inconsistencies, activations will be proposed to other users.

4 Related Work

The projects closest to ActiveTags are Piggy Bank (Huynh et al. 2007) and Intel Mash Maker (`www.mashmaker.intel.com/`), which both are also Firefox extensions and quite similar in their features. Piggy Bank allows Web pages to be connected after the pages have been scraped. Scrapers are available for nine Web sites and three generic formats. The scraped data is transformed into Semantic Web languages, and then used for mashups. Intel's project covers features similar to Piggy Bank, yet without focusing on Semantic Web languages. The scraped data has to be transferred to Intel servers before it can be used. Both projects do not have an integrated notion of communities. In order for mashups to be used they have to be manually installed. This gives users control over which mashups they want to use. We aim at retaining control while increasing the ease of sharing mashups. An essential part of ActiveTags, namely supporting the development of meaning in folksonomies, is not within the scope of these two projects.

Ubiquity (Raskin 2008), an extension adding a command line interface to Firefox, has a complementary approach. Instead of focusing on the resources (of Web sites), its main concept are *commands*, which can be invoked on any Web page. If text is selected on a Web page this text functions as the parameter to commands automatically. The mapping example mentioned above is available in Ubiquity. Here, the workflow is as follows: The text of an address is selected, the Ubiquity shortcut is pressed (which opens the "command line"), the command "map" is typed in, which brings up a small map immediately. New commands can be developed by anyone. The code implementing a command is embedded into a Web page using special markup so that a browser with Ubiquity installed will inform the user of the command's presence and allow installation.

The online mail application Zimbra implements a feature called "Zimlets" (Zimbra 2006). These are programs that are activated on pre-specified patterns found in certain objects (mails, event entries, contacts, etc.). They are used to show data from other sources, thereby effectively mashing-up these pieces of content. The Operator extension (`https://addons.mozilla.org/de/firefox/addon/4106`) for the Firefox browser scans Web pages for microformat data, which it can extract and send to other programs that make use of this data. The extension Sxipper (`www.sxipper.com`) extends the Firefox password store and is able to automatically fill out forms by learning forms on Web pages through user interaction. The information on how a form is to be used is stored centrally, so of all Sxipper users using a site only the first one has to train a particular form. The ActiveTags system uses similar information extraction mechanisms as these two extensions. It employs a mashup execution mechanism similar to Zimlets, but allows more complex invocation criteria.

There is a lot of research on folksonomies and tagging, as well as on bringing these two together with Semantic Web concepts; see Voss (2007) for an overview. Ontologies are a central aspect of the Semantic Web. They provide machine-interpretable semantics and intend to allow machines to reason about the meaning of objects. In order to create an ontology, a design process is needed, which means upfront investment of knowledge engineers that will structure the respective field (Hüsemann & Vossen 2005).

There are several different attempts to bridge social tagging systems and the resulting folksonomies with ontologies. It has been argued by Christiaens (2006) that these two approaches are complementary and differ by their respective degree of freedom. The relationship between folksonomies and machine-extracted keywords was evaluated in Al-Khalifa & Davis (2007) in order to study the former's potential for being used to generate semantic metadata. How semantic knowledge can be used to enhance tags in folksonomies explicitly has been evaluated by Angeletou et al. (2007). Tags have been processed by Specia & Motta (2007) to infer meaning by using other Web resources. Their results are "faceted ontologies ... partial ontologies conceptualizing specific facets of knowledge." In contrast, statistical methods have been used by Zhang et al. (2006) to infer semantics from folksonomies.

Plug-in modules have been proposed by Wu et al. (2006) for social tagging systems that support the extraction of knowledge from those systems. They thus aim at systematizing the capabilities of the technical infrastructure to allow for further use in this direction. The possibility of defining relationships between tags into a social bookmarking system has been introduced by Hotho et al. (2006). All these contributions relate folksonomies and the Semantic Web in one way or another, but do not attempt to directly support the use of tags programmatically, which is the aim of our contribution.

5 Conclusions and Future Work

Social tagging systems obtain their capabilities from massive amounts of users who participate. The open structure and low specificity of tagging allows for many uses, without forcing a particular one or favoring another. The downside of this is that, although the tags allow it, only a few forms of use are efficiently supported in typical tagging systems. The ActiveTags system brings direct programmatic support for tags in the form of mashups right alongside the objects where the tagging takes place.

This paper has shown how the ActiveTags system has been designed and how it works. The system is able to copy popular tag-based mashups that are in use today. Moreover, it has been shown that right from the start the scope is greatly improved when compared to these examples. The evaluation period that is currently running has seen the creation of several new mashups. Other initial results from our evaluation are also promising: For example, Yahoo! Pipes has been used to connect related articles of big German news portals. The rel-tag microformat TagExtractor detects tags on many pages that have not been specifically trained for ActiveTags. Still, the largest growth in numbers has been observed for TagExtractors – most of which worked properly right away. As already mentioned, we are working on a detailed evaluation of the ActiveTags system. Nevertheless, we take the observations mentioned above as positive indicators from the first weeks of our evaluation. Although not all features of the system have been activated, the desired developments are already observable. While continuing the evaluation, we will complete the implementation of the system as outlined in Section 3.2.4.

The design of the ActiveTags system is inherently distributed: The more Web sites use tags (and these becoming detectable), the more common tag interpretations and the more ubiquitous mashups can become. ActiveTags leverages several dimensions of user-generated content and we expect it to influence the way in which tags are used.

References

Al-Khalifa, H. S. & Davis, H. C. (2007), 'Exploring The Value Of Folksonomies For Creating Semantic Metadata.', *International Journal on Semantic Web and Information Systems (IJSWIS)* **3**(1), 13–39.

Andrews, P. (2006), 'Flickr Upcoming Event Userscript'. http://userscripts.org/scripts/show/5464.

Angeletou, S., Sabou, M., Specia, L. & Motta, E. (2007), Bridging the Gap Between Folksonomies and the Semantic Web: An Experience Report., *in* 'Workshop: Bridging the Gap between Semantic Web and Web 2.0, European Semantic Web Conference'.

Celik, T. & Marks, K. (2005), 'Rel-tag microformat'. http://microformats.org/wiki/rel-tag.

Christiaens, S. (2006), Metadata Mechanisms: From Ontology to Folksonomy ... and Back, *in* R. Meersman, Z. Tari & P. Herrero, eds, 'OTM Workshops (1)', Vol. 4277 of *Lecture Notes in Computer Science*, Springer, pp. 199–207.

Golder, S. & Huberman, B. A. (2006), 'Usage patterns of collaborative tagging systems', *Journal of Information Science* **32**(2), 198–208.

Hammond, T., Hannay, T., Lund, B. & Scott, J. (2005), 'Social Bookmarking Tools (I)', *D-Lib Magazine* **11**(4). http://www.dlib.org//dlib/april05/hammond/04hammond.html.

Hinchcliffe, D. (2006), 'Continuing an Industry Discussion: The Co-Evolution of SOA and Web 2.0', Dion Hinchcliffe's Web 2.0 Blog. http://web2.wsj2.com/continuing_an_industry_discussion_the_coevolution_of_soa_and.htm.

Hotho, A., Jäschke, R., Schmitz, C. & Stumme, G. (2006), Emergent Semantics in BibSonomy, *in* C. Hochberger & R. Liskowsky, eds, 'GI Jahrestagung (2)', Vol. 94 of *LNI*, GI, pp. 305–312.

Hüsemann, B. & Vossen, G. (2005), Ontology Engineering from a Database Perspective, *in* S. Grumbach, L. Sui & V. Vianu, eds, 'ASIAN', Vol. 3818 of *Lecture Notes in Computer Science*, Springer, pp. 49–63.

Huynh, D., Mazzocchi, S. & Karger, D. (2007), 'Piggy bank: Experience the semantic web inside your web browser', *Web Semantics: Science, Services and Agents on the World Wide Web* **5**(1), 16–27. http://www.sciencedirect.com/science/article/B758F-4MVF4YB-1/2/38957cbb8d2a861463d6a77e35637bf6.

Lund, B., Hammond, T., Flack, M. & Hannay, T. (2005), 'Social Bookmarking Tools (II): A Case Study - Connotea', *D-Lib Magazine* **11**(4). http://www.dlib.org/dlib/april05/lund/04lund.html.

Marlow, C., Naaman, M., boyd, d. & Davis, M. (2006), HT06, tagging paper, taxonomy, Flickr, academic article, to read, *in* (Wiil et al. 2006), pp. 31–40.

Newman, R. (2004), 'Newman Tag Ontology'. http://www.holygoat.co.uk/projects/tags/.

Raskin, A. (2008), 'Introducing ubiquity', Mozilla Labs Blog. http://labs.mozilla.com/2008/08/introducing-ubiquity/.

Specia, L. & Motta, E. (2007), Integrating Folksonomies with the Semantic Web, *in* E. Franconi, M. Kifer & W. May, eds, 'ESWC', Vol. 4519 of *Lecture Notes in Computer Science*, Springer, pp. 624–639.

Straup Cope, A. (2007), 'Machine tags', Flickr API / Discuss. http://www.flickr.com/groups/api/discuss/72157594497877875/.

Voss, J. (2007), 'Tagging, Folksonomy & Co - Renaissance of Manual Indexing?'. http://arxiv.org/abs/cs/0701072.

Vossen, G. & Hagemann, S. (2007), *Unleashing Web 2.0 - From Concepts to Creativity*, Morgan Kaufmann, Burlington, MA.

W3C (2005), 'Selectors, W3C Working Draft'. http://www.w3.org/TR/css3-selectors/.

W3C (2007), 'W3C GRDLL Specification'. http://www.w3.org/2001/sw/grddl-wg/.

Wiil, U. K., Nürnberg, P. J. & Rubart, J., eds (2006), *HYPERTEXT 2006, Proceedings of the 17th ACM Conference on Hypertext and Hypermedia, August 22-25, 2006, Odense, Denmark*, ACM.

Wu, H., Zubair, M. & Maly, K. (2006), Harvesting social knowledge from folksonomies, *in* (Wiil et al. 2006), pp. 111–114.

Zhang, L., Wu, X. & Yu, Y. (2006), 'Emergent Semantics from Folksonomies: A Quantitative Study', *J. Data Semantics* **VI**, 168–186.

Zimbra (2006), *Zimlets - A Mechanism for Integrating Disparate Information Systems and Content with the Zimbra Collaboration Suite (ZCS)*, version 0.97 edn.