

Semantics of Social Web Search^{*}

Pierre-Antoine Champin^{1,2}, Peter Briggs²,
Maurice Coyle², and Barry Smyth²

¹ LIRIS, Université de Lyon, CNRS, UMR5205,
Université Claude Bernard Lyon 1 / F-69622, Villeurbanne, France
pchampin@liris.cnrs.fr
<http://liris.cnrs.fr/silex>

² CLARITY: Centre for Sensor Web Technologies
School of Computer Science and Informatics
University College Dublin, Ireland.
{first.last}@ucd.ie
<http://www.www.clarity-centre.org/>

Abstract. Social bookmarking and social search are attempts to leverage the search experience of Web users in order to help themselves or other like-minded users in future searches. HeyStaks is one of those applications, and in this paper we show how recent developments in HeyStaks allow it to leverage even more search experiences from the “regular” Web, as well as the Semantic Web, in order to enhance the service offered to its users. We also show how HeyStaks uses de facto standard ontologies to republish this information as linked data, thus enabling even more uses of the collected knowledge.

1 Introduction

The now familiar *Social Web* reflects an important change in the nature of the Web and its content. Since 1999, the rapid growth of blogs, as a simple way for users to express their views and opinions, ushered in this new era of *user-generated content* (UGC) as many sites quickly began to offer a whole host of UGC alternatives including the ability to leave comments and write reviews, as well as the ability to rate or vote on the comments/opinions of others. The result has been an evolution of the Web from a repository of information to a repository of *experiences*, and an increased emphasis on people rather than content. In combination with social networking services, this has precipitated the growth of the *Social Web* as a platform for communication, sharing, recommendation, and collaboration.

At the same time, the open linked data initiative (linkeddata.org) [8] has been contributing to realize the vision of the Semantic Web, by consolidating the growing repository of machine readable data that will enable a new era

^{*} This work is supported by Science Foundation Ireland under grant 07/CE/I1147, the French National Center for Scientific Research (CNRS), and HeyStaks Technologies Ltd.

of Semantic Web applications. This trend, combined with the development of similar effort such as microformats [13] and Web-based APIs [12], has amplified the ubiquity of the Social Web, by allowing the data to be shared and combined across multiple applications and modalities.

In this evolving online world, Web search has continued to play a vital role and there is no doubting the success of the mainstream Web search engines as a key information tool for hundreds of millions of users everyday. Given the importance of Web search it is no surprise that researchers continue to look for new ways to improve upon the mainstream search engine [26]. However, new tools are also needed to gather, harness, reuse and share, in the most efficient and enjoyable way, the experiences captured by UGC and linked data [19,23]. Semantic Web search engines [7,10] use linked data to provide more relevant search results, focusing on semantic relations rather than ambiguous textual contents. Another line of research has focused on using recommendation technologies in an effort to make Web search more personal: by learning about the preferences and interests of individual searchers, personalized Web search systems can influence search results in a manner that better suits the individual searcher [5,25]. Recently, another complementary research direction has seen researchers explore the *collaborative* potential of Web search by proposing that the conventional *solitary* nature of Web search can be enhanced in many search scenarios by recognising and supporting the sharing of search experiences to facilitate synchronous or asynchronous collaboration among searchers [20,17]. Indeed, the work of [21,3] has shown that collaborative Web search can lead to a more personalized search experience by harnessing recommendations from the search experiences of communities of like-minded searchers.

Our recent work has led to the development of a new system to support collaborative Web search. This system is called HeyStaks (heystaks.com) and it benefits from providing a collaborative search experience that is fully integrated with mainstream search engines such as Google. HeyStaks comes in the form of a browser toolbar and, as users search as normal, HeyStaks captures their search experiences and promotes results based on their search experiences and the experiences of friends, colleagues, and other like-minded searchers. HeyStaks introduces the key concept of a *search stak* which serves as a repository for search experiences. Users can create search staks to represent their search interests and they can share their staks with others to create pools of search experiences. At search-time, recommendations are generated from the user's staks and presented alongside mainstream search results. In this way, HeyStaks harnesses the shared experiences of searchers to deliver an improved search experience by working with, rather than competing against, mainstream search engines. With HeyStaks, users search as normal but enjoy the benefits of being able to easily share their search experiences and the advantages of a new form of search collaboration.

The key contribution of this paper is to combine the advantages of HeyStaks as a social search recommender system with those of the Semantic Web. After describing the original architecture of HeyStaks, we will present recent and near-future enhancements that leverage experiences collected from the "regular"

Web as well as the Semantic Web. Then we will describe how HeyStaks not only imports information from the Semantic Web but republishes aggregated information in a reusable way. To this end we will describe how standard ontologies can be used to accomodate the semantic representation of search experiences.

2 HeyStaks

HeyStaks adds two important collaboration features to any mainstream search engine. First, it allows users to create *search staks* as a type of folder for their search experiences at search time. Staks can be shared with others so that their own searches will also be added to the stak. Second, HeyStaks uses staks to generate recommendations that are added to the underlying search results that come from the mainstream search engine. These recommendations are results that stak members have previously found to be relevant for similar queries and help the searcher to discover results that friends or colleagues have found interesting, results that may otherwise be buried deep within the engine's result-list.

In designing HeyStaks, our primary goal is to help improve upon the search experience offered by mainstream search engines, while at the same time allowing searchers to continue to use their favourite search engine. As such, a key component of the HeyStaks architecture is a browser toolbar that permits tight integration with search engines such as Google, allowing searchers to search as normal while providing a more collaborative search experience via targeted recommendations. In this section we will outline the basic HeyStaks system architecture and summarize how result recommendations are made during search. In addition we will make this discussion more concrete by briefly summarizing a worked example of HeyStaks in action.

2.1 System Architecture

As per Fig. 1, HeyStaks takes the form of two basic components: a client-side *browser toolbar* and a back-end *server*. The toolbar allows users to create and share staks and provides a range of ancillary services, such as the ability to tag or vote for pages. The toolbar also captures search result click-thrus and manages the integration of HeyStaks recommendations with the default result-list. The back-end server manages the individual stak indexes (indexing individual pages against query/tag terms and positive/negative votes), the stak database (stak titles, members, descriptions, status, etc.), the HeyStaks social networking service and the recommendation engine. In the following sections we will outline the basic operation of HeyStaks and then focus on some of the detail behind the recommendation engine.

To make things more concrete, consider the following example. Steve, Bill and some friends were planning a European vacation and they knew that during the course of their research they would use Web search as their primary source of information about what to do and where to visit. Steve created a (private)

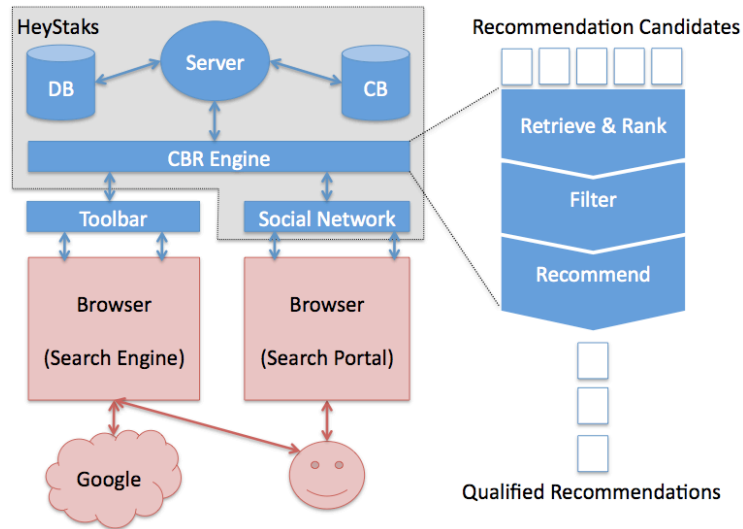


Fig. 1. The HeyStaks system architecture and outline recommendation model.

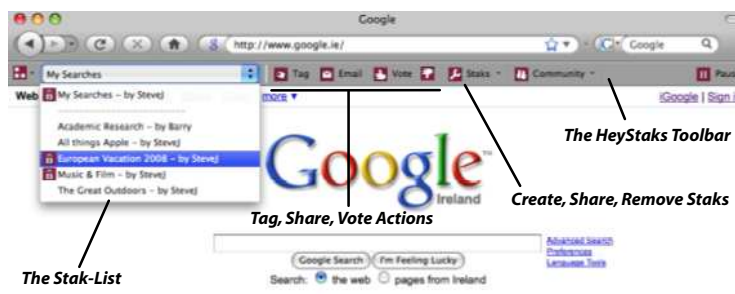


Fig. 2. Selecting a new active stak.

search stak called “European Vacation 2008” and shared this with Bill and friends, encouraging them to use this stak for their vacation-related searches.

Fig. 2 shows Steve selecting this stak as he embarks on a new search for “Dublin hotels”, and Fig. 3 shows the results of this search. The usual Google results are shown, but in addition HeyStaks has made two promotions. These were promoted because other members of the “European Vacation 2008” stak had recently found these results to be relevant; perhaps they selected them for *similar* queries, or voted for them, or tagged them with related terms. These recommendations may have been promoted from much deeper within the Google result-list, or they may not even be present in Google’s default results. Other relevant results may also be highlighted by HeyStaks, but left in their default Google position. In this way Steve and Bill benefit from promotions that are based on their previous similar searches. In addition, HeyStaks can recommend results from Steve and Bill’s other staks, helping them to benefit from the search knowledge that other groups and communities have created.

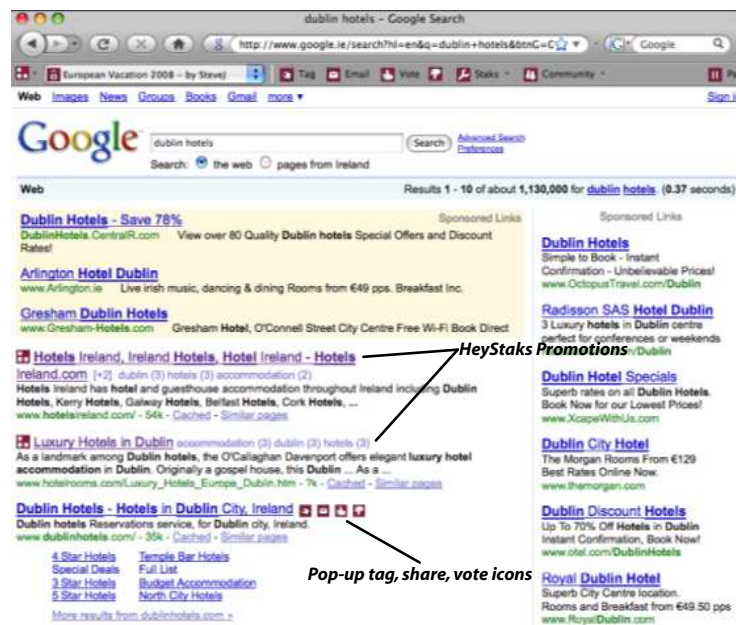


Fig. 3. Google search results with HeyStaks promotions.

Separately from the toolbar, HeyStaks users also benefit from the HeyStaks *search portal*, which provides a social networking service built around people’s search histories. For example, Fig. 4 shows the portal page for the “European Vacation 2008” stak, which is available to all stak members. It presents an activity feed of recent search history and a query cloud that makes it easy for the user to find out about what others have been searching for. The search portal

also provides users with a wide range of features such as stak maintenance (e.g., editing, moving, copying results in staks and between staks), various search and filtering tools, and a variety of features to manage their own search profiles and find new search partners.

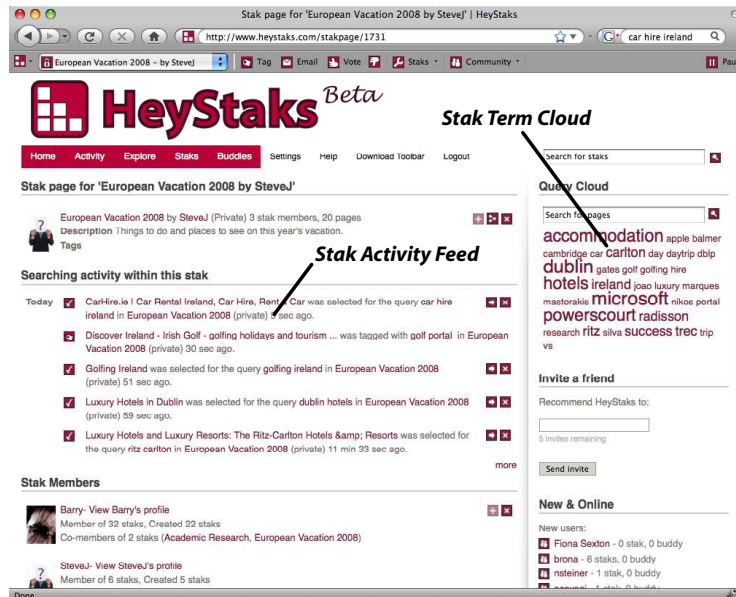


Fig. 4. The HeyStaks search portal provides direct access to staks and past searches.

2.2 Generating Recommendations

In HeyStaks each search stak (S) serves as a profile of the search activities of the stak members. Each stak is made up of a set of result pages ($S = \{p_1, \dots, p_k\}$) and each page is anonymously associated with a number of implicit and explicit interest indicators, including the total number of times a result has been selected (sel), the query terms (q_1, \dots, q_n) that led to its selection, the number of times a result has been tagged (tag), the terms used to tag it (t_1, \dots, t_m), the votes it has received (v^+, v^-), and the number of people it has been shared with ($share$).

In this way, each page is associated with a set of *term data* (query terms and/or tag terms) and a set of *usage data* (the selection, tag, share, and voting counts). The term data is stored as a Lucene (lucene.apache.org) index, with each page indexed under its associated query and tag terms, and provides the basis for retrieving and ranking *promotion candidates*. The usage data provides an additional source of evidence that can be used to filter results and to generate

a final set of recommendations. At search time, recommendations are produced in a number of stages: first, relevant results are retrieved and ranked from the *stak* index; next, these promotion candidates are filtered based on the usage evidence to eliminate noisy recommendations; and, finally, the remaining results are added to the Google result-list according to a set of presentation rules.

Briefly, HeyStaks uses a number of different recommendation rules to determine how and where a promotion should be added. Space restrictions prevent a detailed account of this component but, for example, up to 3 *primary* promotions are added to the top of the Google result-list and labelled using the HeyStaks promotion icons. If a remaining promotion is also in the default Google result-list then this is labeled in place. If there are still remaining promotions then these are added to the *secondary* promotion list, which is sorted according to TF*IDF scores. These recommendations are then added to the Google result-list as an optional, expandable list of recommendations. The interested reader can refer to [22] for more details.

3 Collecting Experiences from the Web

As an isolated system, HeyStaks suffers from the sparsity problem which plagues recommender systems [4]. Before a search *stak* reaches a critical mass, it is hard for the system to provide relevant recommendations related to that *stak*. On the other hand, recommendations are one of the main incentives for using a *stak*, so sparse *staks* may never be used enough to become useful.

Another cause of sparsity in HeyStaks, which is also faced by many social and “Web 2.0” applications, is the functionality overlap with other popular applications. Users are generally using several such applications, because sometimes they need a set of features that are not all present in a single system, and very often have their friends or colleagues scattered across different social network applications. Despite a number of efforts to enhance the portability of one’s data in Web applications³, their interoperability is still very limited. This forces users either to duplicate their data in multiple applications, or to maintain only parts of their information in each application, making all of them less efficient.

For HeyStaks, this functionality overlap occurs with applications that allow people to express interest in Web pages, and share this information with their social network: social bookmarking (such as delicious.com or simpy.com), social news (such as digg.com or reddit.com), social citations (such as citeulike.org, connotea.org or zotero.org). This of course comes in addition to similar services available as standalone applications (bookmarks, news aggregators, reference management), without the social dimension. Some people are nevertheless more comfortable with them, for privacy reasons, for having their data available offline, or simply because they are used to them.

³ <http://opensocial.org/>, <http://dataportability.org/>

3.1 Importing Search Experiences

In order to tackle those problems, HeyStaks must be able to aggregate into a stak all the experiences related to its topic, so as to reduce the sparsity and provide better recommendations to the users —without requiring them to duplicate their information manually. For this purpose, we have enhanced HeyStaks with a generic import functionality, which is illustrated in Figure 5.

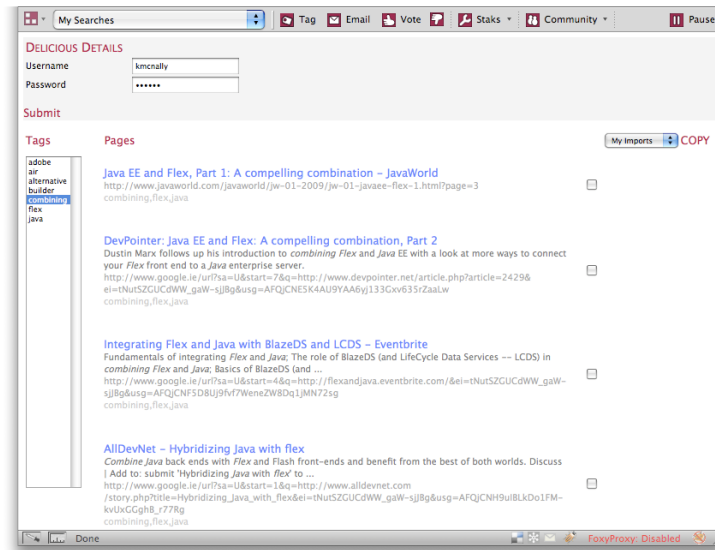


Fig. 5. Import interface on the HeyStaks portal

This feature currently only supports delicious.com bookmarks, but is designed in such a way that it can easily (and will) be extended to other online bookmarks services as well as Firefox’s built-in bookmarks. It allows users to import the content of a bookmark folder or tag into a newly created stak, “seeding” it with search experiences previously captured in other systems. In our previous example, Steve would have imported in the “European Vacation 2008” all his delicious bookmarks tagged with “hotel”, taking advantage of searches he had made before even using HeyStaks. Bill, who is using Firefox bookmarks, could soon do the same with his “Vacation” folder.

The next step in this direction will be to allow staks to be *continuously* fed by the import sources. For example, if Steve was now to tag a new page with the tag “hotel”, (or Bill to file it under his “Vacation” bookmark folder), this new page must be dynamically added to the “European Vacation 2008” stak. This allows users to continue with their browsing habits (an important design

concern in HeyStaks), but more importantly, to combine the benefits of several applications. For example, bookmarks make a page easier to find when the users *knowingly* seek it, while recommendation brings up the page for relevant queries, even if the user didn't recall that page.

Another possible way to use the import feature is to import published data from other users, like somebody else's delicious bookmarks, or a blog or microblog syndication feed. Here, the relevance assessment is indirect: the stak owner states implicitly that they trust the author(s) of the imported source to provide links that are relevant to that stak. For example, Steve decides to import an RSS feed from `booking.com` into the "Europe Vacation 2008" stak. This spares him the trouble of reviewing manually this feed, but lets HeyStaks recommend its pages whenever a user of the stak issues a related query.

3.2 Semantic Web Import

Although the import features described above increases the quantity of search experiences (thus reducing sparsity), this may come to this expense of quality. Although we have already studied methods to cope with noisy staks [24], limiting the amount of input noise at import-time is still the best option.

Semantic Web technologies offer us ways to limit the noise by using semantically richer information. Ontologies such as SIOC [2] or Common Tag allow users to describe the topic of a page (or even a page fragment) using unambiguous concepts. These descriptions can then be found by Semantic Web crawlers such as `sindice.com`. In principle, importing data from a Semantic Web crawler is not different from importing an RSS feed, as long as the query provides a set of pages, possibly annotated with tags. However, it has three major advantages. First, it spans the whole Semantic Web rather than the limited set of sources that stak users are aware of.

Second, it is semantically more focused, using precisely defined concepts instead of, for example, tags having possibly different meanings (homonyms). For example, the tag "python" may refer to a genus of snakes, a programming language or a mythological creature. A Semantic Web search engine, on the other hand, can be queried for resource related only to the snakes using the appropriate concept URI⁴.

Finally, since staks are meant to group experiences about one particular topic, it is possible to have the stak owner describe this topic once and for all in terms of Semantic Web concepts. It is worth noting that those concepts are linked, in many cases, to a human-readable description (e.g. `dbpedia.org` concepts are mapped to wikipedia pages). HeyStaks users could therefore be prompted to choose a concept based on these descriptions, without having any knowledge of Semantic Web technologies. Then, HeyStaks could use this information to define import settings automatically. Non technical users would then be relieved of the burden of defining import manually, and would mainly have to focus on the semantic description of the stak.

⁴ [http://dbpedia.org/resource/Python_\(genus\)](http://dbpedia.org/resource/Python_(genus))

These features are therefore a natural extensions to the current import functionality, and we are currently considering their integration to the HeyStaks portal.

4 Republishing Search Experiences for Reuse

The second integration of HeyStaks with the Semantic Web is the publishing, as linked data, of the experiences captured by staks. This information is already available (to human users) on the HeyStaks portal, as can be seen for example in figure 4. Our goal is now to make it available to machines as well, using RDFa [1] to embed machine-processable data into portal pages.

4.1 Representation

We are using the SIOC ontology⁵ [2] to describe the main elements of HeyStaks. SIOC is ubiquitous enough to represent various kinds of online community, and indeed HeyStaks structures map easily into the general concepts it defines: HeyStaks is a Site, each stak is a Container, containing pages which are its Items.

Each page must then be associated with a number of tags. Although SIOC supports basic tagging, we need a finer grained description here: some tags are put explicitly by users (either using the tagging functionality of Heystalks, or by importing tagged bookmarks), but others are attached implicitly by the system (for example, the query terms used before selecting this page). Common Tag⁶, another de facto standard ontology, provides this level of detail: it distinguishes author tags (attached by the author of a resource), reader tags (attached by any reader of the resource) and automatic tags (attached by an automated agent). In HeyStaks, only reader and automatic tags are considered (although in the future the toolbar could be instrumented to extract author tags from the pages themselves).

Figure 6 summarizes the links of HeyStaks resources (shaded nodes in the figure) with other reused terms. Dotted arrows represent links that are not implemented yet but mentioned along this paper as possible future developments. Note also that some of the informations from the stak described in section 2.2 are not represented for the moment in the linked-data export: vote counts, number of selections, number of times the page was shared.

4.2 Reusing Search Experiences

The immediate benefit of publishing as linked data the experiences captured by HeyStaks is to have them indexed by Semantic Web crawlers. The interest for HeyStaks as a community is to augment its visibility, hence attracting more users that will in turn share their own experiences in public staks. The interest

⁵ <http://sioc-project.org/>

⁶ <http://commontag.org/>

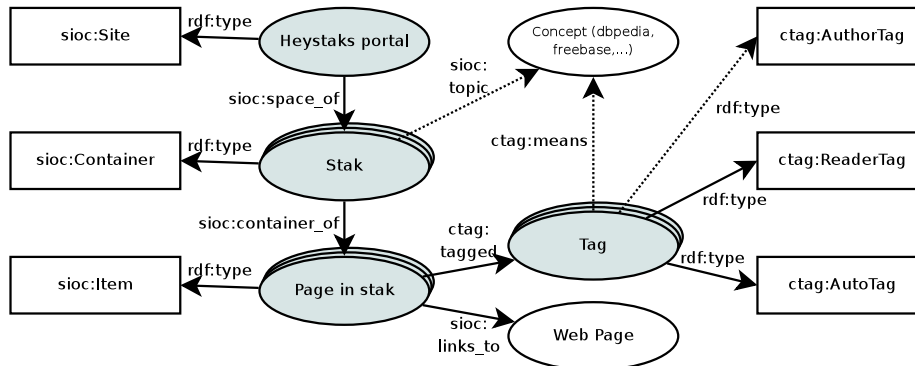


Fig. 6. HeyStaks representation reusing standard vocabularies

for other communities is that experiences of HeyStaks users are made available on the linked data cloud, available for reuse by other applications.

Considering the import features mentioned in section 3, it may seem that HeyStaks is only restating information that is already available on the Web. We argue however that HeyStaks brings added value in the following respects. First, the experiences directly collected by the toolbar are obviously novel. Second, HeyStaks import feature may extract information from legacy sources that are not currently accessible to Semantic Web crawlers, either because they just do not implement it yet, or because technical barriers apply to them, that do not apply to the HeyStaks toolbar. Finally, the fact that several sources are grouped together into a stak creates a link between them which was not in the original sources. This link, assessed by the user, hence deemed semantically relevant, is precisely what makes the value of linked data in general [8].

That link can even be made semantically stronger: the common tag ontology, that we used to describe tags, aims at eliciting the *meaning* of those tags, using Semantic Web concept URIs instead of plain labels. We already suggested in section 3.2 that such associations could be made at the stak level, but with Common Tag it could also be made at the page level. With HeyStaks data exported as linked data, this elicitation does not have to take place in HeyStaks itself⁷, but can be done in another application such as LODR⁸.

Finally, external applications could take advantage not only of the synthetic view of a stak as a container of pages, but also of the dynamic view of a stak as a stream of actions performed on those pages (selection, tagging, sharing, etc.). This view is available in human readable form on the Heystak portal, as *activity feeds*, but SIOC does not provide a standard way of representing this. Therefore, we are currently designing an ontology for HeyStaks actions, and we are also considering to propose the most generic part of it as an extension module for

⁷ Although at some point, it could be useful if the recommendation algorithm was made to use this information.

⁸ <http://lodr.info/>

SIOC. We believe that this kind of knowledge can indeed prove useful [6], not only for HeyStaks, but for many other online communities described with SIOC (blogs, content management systems, etc.).

5 Related Works and Discussion

While ontologies have already been used in recommender systems to model user profiles [16], Semantic Web technologies have rarely been used to model the actual recommendation knowledge in those systems ([28,11] being notable exceptions). There is indeed a defiance of the recommender system community, regarding the Semantic Web as a source of spam and attacks [15]. The problem of spam is indeed a concern for all kinds of social applications, whether they use recommendation techniques or not [18,27,14].

In HeyStaks, the notion of staks partially addresses this problem. Each stak being targetted at only a part of the community, the incentive for spamming is lowered. While popular public staks can nevertheless be joined by malicious users and subjected to spam, it is possible for staks to be restricted to invited users. In any case, the standard import feature described in section 3.1 does not significantly aggravate the problem: users are as much responsible for what they import as for the pages they select or tag directly while searching.

The trust issue becomes more serious, however, when we consider querying the whole Semantic Web, since all sources crawled by a Semantic Web search engine may not be equally trustworthy, at least from the point of view of the stak users. As pointed out by [9], tracking the provenance of an assertion is a crucial feature for the Semantic Web, as it is a prerequisite to any model of trust.

From a more technical point of view, tracking the provenance of Semantic Web information will also be useful in HeyStaks when importing data from the Semantic Web, in order to prevent redundancy. Indeed, HeyStaks may collect the same information via several channels, if for example a user tags a web page both in delicious and on their blog, and both of them are imported by the same stak. It might be desirable that this be considered as a single assertion, and be given less importance than if the delicious bookmark and the blog belonged to different persons.

6 Conclusion

In this paper, we have presented HeyStaks, a social search service using recommendation techniques to contextually promote and add links in the search results produced by mainstream search engines. This is achieved by storing in topic-specific *staks* the search experiences of users interested in that topic. We have presented recent improvements in HeyStaks aiming at opening it to the Semantic Web: by importing into staks other related experiences collected on the Web, and by republishing all collected experiences as linked data. We have discussed the immediate benefits of these new features (for HeyStaks users and other users of the Semantic Web) as well as their ongoing and future developments.

References

1. B. Adida and M. Birbeck. RDFa primer. W3C working group note, W3C, Oct. 2008.
2. J. Breslin and S. Decker. The future of social networks on the internet: The need for semantics. *IEEE Internet Computing magazine*, Nov. 2007.
3. P. Briggs and B. Smyth. Provenance, trust, and sharing in peer-to-peer case-based web search. In *Proceedings of the 9th European conference on Advances in Case-Based Reasoning*, pages 89–103, 2008.
4. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
5. P. A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using ODP metadata to personalize search. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA, 2005. ACM.
6. D. Cram, B. Fuchs, Y. Pri, and A. Mille. An approach to User-Centric Context-Aware assistance based on interaction traces. In *Modeling and Reasoning in Context (MRC 2008)*, 2008.
7. L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the 13th ACM international conference on Information and knowledge management*, pages 652–659. ACM New York, NY, USA, 2004.
8. W. Halb, Y. Raimond, and M. Hausenblas. Building linked data for both humans and machines. In *WWW 2008 Workshop: Linked Data on the Web (LDOW2008)*, Beijing, China, 2008.
9. H. Halpin. Provenance: The missing component of the semantic web for privacy and trust. In *Trust and Privacy on the Social and Semantic Web (SPOT2009), workshop of ESWC 2009*, Heraklion, Crete, Greece, June 2009.
10. A. Harth, J. Umbrich, and S. Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)*, page 258, Athens, GA, USA, Nov. 2006.
11. E. Hyvnen, E. Mkel, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. MuseumFinland - finnish museums on the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):224–241, 2005.
12. J. Jarvis. The API revolution. Blog post, June 2009. <http://www.buzzmachine.com/2009/06/17/the-api-revolution/>.
13. Khare. Microformats: The next (Small) thing on the semantic web? *Internet Computing, IEEE*, 10(1):68–75, 2006.
14. B. Krause, A. Hotho, and G. Stumme. The anti-social tagger - detecting spam in social bookmarking systems. In *Proc. of the Fourth International Workshop on Adversarial Information Retrieval on the Web*, 2008.
15. P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of the Federated International Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, pages 492–508. Springer, 2004.
16. S. E. Middleton, N. R. Shadbolt, and D. C. D. Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, 2004.

17. M. R. Morris. A survey of collaborative web search practices. In *CHI*, pages 1657–1660, 2008.
18. J. O’Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 167–174. ACM New York, NY, USA, 2005.
19. E. Plaza. Semantics and experience in the future web. In *Proceedings of the 9th European conference on Advances in Case-Based Reasoning*, pages 44–58. Springer, 2008.
20. M. C. Reddy and P. R. Spence. Collaborative information seeking: A field study of a multidisciplinary patient care team. *Inf. Process. Manage.*, 44(1):242–255, 2008.
21. B. Smyth. A community-based approach to personalizing web search. *IEEE Computer*, 40(8):42–50, 2007.
22. B. Smyth, P. Briggs, M. Coyle, and M. P. O’Mahony. Google? shared! a case-study in social search. In *User Modeling, Adaptation and Personalization*. Springer-Verlag, June 2009.
23. B. Smyth and P. Champin. The experience web: A Case-Based reasoning perspective. In *Grand Challenges for reasoning from experiences, Workshop at IJCAI’09*, July 2009.
24. B. Smyth, P.-A. Champin, P. Briggs, and M. Coyle. The Case-Based Experience Web. In N. W. Derek Bridge, Enric Plaza, editor, *WebCBR workshop at ICCBR’09*, July 2009.
25. J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. In *WWW ’05: Proceedings of the 14th international conference on World Wide Web*, pages 382–390, New York, NY, USA, 2005. ACM Press.
26. Yahoo. SearchMonkey guide. Technical report, Yahoo! Inc, 2008.
27. C. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4):337–358, Dec. 2005.
28. C. N. Ziegler. Semantic web recommender systems. In *Proceedings of the Joint ICDE/EDBT Ph. D. Workshop*. Springer, 2004.

Acknowledgements

The authors would like to thank Uldis Bojars and Alexandre Passant for their valuable feedback on SIOC.