SNARC – A Semantic Social News Aggregator

Ahmad Assaf[†] , Aline Senart[†] and Raphaël Troncy[‡]

[†]SAP Research, SAP Labs France SAS
805 avenue du Dr. Maurice Donat, BP 1216, 06254 Mougins Cedex, France
first.last@sap.com

[‡]EURECOM
2229 route des crêtes, 06560 Sophia Antipolis, France
raphael.troncy@eurecom.fr

Abstract. The Internet has created a paradigm shift in how we consume and disseminate information. Data nowadays is spread over heterogeneous silos of archived and live data. People willingly share data on social media by posting news, views, presentations, pictures and videos. Social media has become a source of live and up-to-date knowledge; a property that was lacking in traditional sources. SNARC is a service that uses semantic web technology and combines services available on the web to aggregate social news. SNARC brings live and archived information to the user that is directly related to his active page. The key advantage is an instantaneous access to complementary information without the need to dig for it. Information appears when it is relevant enabling the user to focus on what is really important.

Keywords: Social news, social networks, aggregator, data mining, mashup, automatic tagging, named entity recognition

1 Introduction

With the rapid advances of the Internet, social media becomes more and more intertwined with our daily lives. The ubiquitous nature of Web-enabled devices especially mobile phones, enables users to participate and interact on many different forms like photo and video sharing platforms, forums, newsgroups, blogs, micro-blogs, bookmarking services, and location-based services. Social networks are not just gathering Internet users into groups of common interests, they are also helping people follow breaking news, contribute to online debates or learn from others. They are transforming Web usage in terms of users' initial entry point, search, browsing and purchasing behavior [7].

A common scenario that happens often while reading an interesting article, coming across a nice video or participating in a discussion in a forum is the growing interest to check related material around the information read. To do so; users go to Twitter[1], Google+ [2] or YouTube[3]. They can try several times with several keywords to obtain the desired results. In the end, we might end up with several browser tabs opened and get often distracted by information overload from all these resources. The same happens in companies when business users are interested in information provided by corporate web application. SNARC is a semantic social news aggregator that leverages live rich data that social networks provide to build an interactive rich experience on the Internet. The service retrieves news related to the current page from popular platforms like Twitter, Google+, YouTube, Vimeo[4], Slideshare[5], Stackoverflow[6] and the Web.

[1] http://www.twitter.com
[2] http://plus.google.com
[3] http://www.youtube.com
[4] http://www.vimeo.com
[5] http://www.slideshare.com
[6] http://www.stackoverflow.com

2 Underlying Mechanism

SNARC is a web service that can be called through AJAX from any Web-enabled application. The URL of the active browser page is the only parameter sent and the results are returned in the JSON format. A Chrome extension and a Wordpress[7] plugin have been implemented as two possible front-ends that call the web service and display results in a side panel of the current page.
The back-end of SNARC consists of three major components: a document handler, a query layer and a data parser which processes the search results to generate the desired output (see Figure 1).

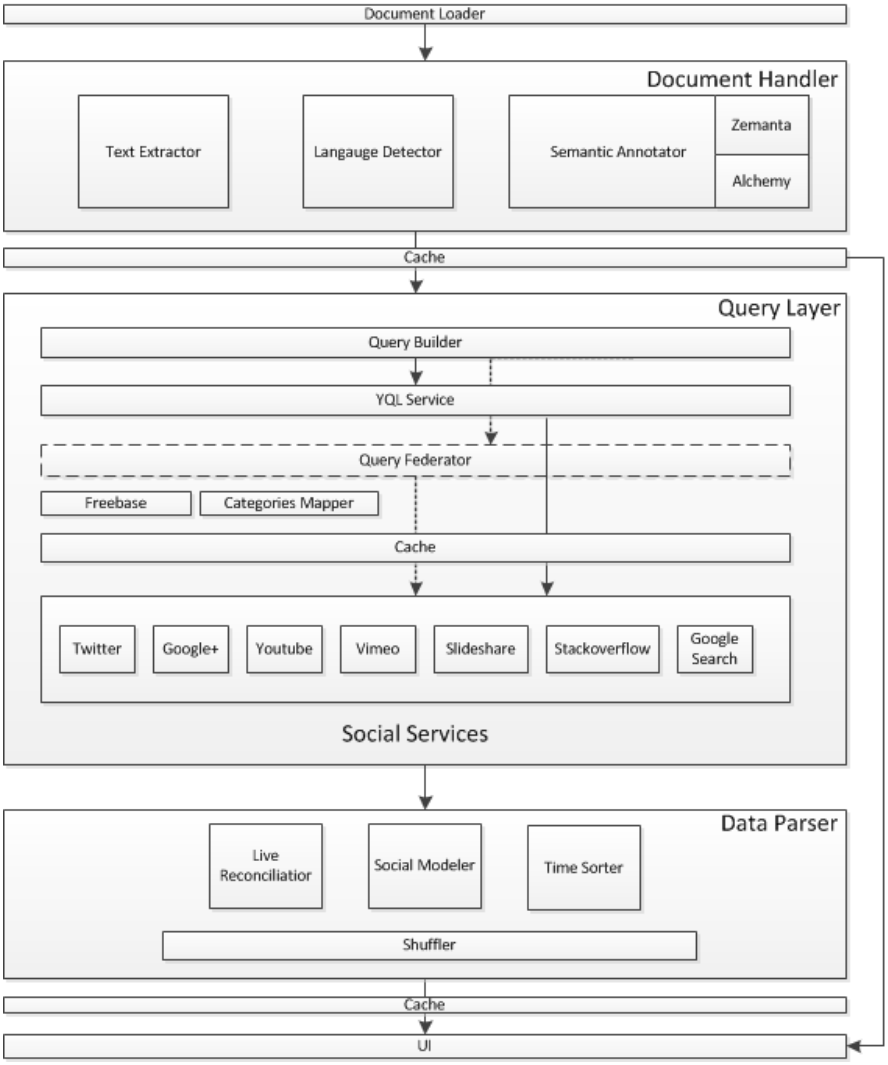


Figure 1 SNARC Architecture

2.1 Document Handler

The call to the SNARC service with the URL of the active browser is received by the Document Loader. It loads the corresponding HTML document and passes it to the Document Handler. After this, there are three main steps to process the HTML document.

1. Fetch the webpage that corresponds to the parameterized URL and extract the textual content using a set of heuristics. These latter identify the main content of the page by stripping unwanted HTML tags and rank the different sections based on their semantics, class names and order. In the beginning we have used Alchemy API[8] to perform text extraction; but we have chosen to implement a simpler method ourselves which saved us an extra API call.

[7] http://www.wordpress.org
[8] http://www.alchemyapi.com/api/text/

2. Detect the webpage language using the Language Detection service of Alchemy API. This is necessary to match the desired language with compatible services like Twitter.
3. Annotate the extracted text is the most important step in this phase. We need to identify the most important keywords (tags) that describe best the extracted text. These keywords are used as the base query on the different sources. Keyword extraction is done using the Zemanta Suggest[9] and Alchemy's Keyword Extraction APIs[10]. Fundamentally, they employ sophisticated statistical and semantic algorithms as well as natural language processing technologies to analyze the text and extract keywords. In addition to the "tagging" process, we also need to map the text to high level taxonomies; this is important when querying services like YouTube where we can execute our query on specific categories. This process is done using Alchemy's Text Categorization API.

At the end of this process, we will have the needed input elements (keywords and high level categories) to be passed to the query generator.

2.2 Query Layer

This is the place where the calls to the social services are made. The set of services are defined in an array and are passed as a parameter. Adding a new search service is easy; we just need to extend the *Social_Search* class and add the service name to the parameters list.

The query builder is responsible for building tailored queries for each service. To enhance the performance we tried to reduce the number of external calls to the minimum. For this we used the Yahoo Query Language (YQL)[11] . It is an expressive SQL-like language that lets you query, filter, and join data across Web services. It helped us minimize the number of calls and batch them into a single one. However, we have found that we cannot fully rely on YQL due to their API calls limit and the restriction on the query's execution time that is set to 30 seconds. To overcome this, we have implemented a fallback mechanism that federates the queries to all the social services and groups the result to be passed afterwards to the parser.

To minimize the number of calls as well, we have decided to take the top two ranked keywords. Keywords are not gathered in a query by any logical operator (AND/OR); a one-to-one mapping is done with each keyword and query. We have decided to do so, as we have found that gathering keywords even if semantically related might bring up noise in the results.

The main setback in the query layer was the variable limited number of calls we can make to external APIs. To overcome this, we have implemented a simple cache mechanism that saves the results on disk up to an hour. There are several cache levels; the first is a URL level one where the results of the parsed queries are cached. For example, if a user visited a certain article on the CNN webpage the results might take up to 15 seconds to appear, whereas a second user visiting the same article minutes afterwards will have the results shown in a matter of few seconds. The second level is keyword and service specific. This can be very helpful as users generally browse articles of related topics or interests (semantic concepts), so for each user we can end up with the same high level concepts being requested frequently. An important thing to note is that the caching is done on the server side and is disk-based.

The social services queried can be grouped as the following:

1. **Multimedia Services**: They include Slideshare, Vimeo and YouTube. Slideshare and YouTube allow the results to be fetched in a specific language that was detected earlier in the previous step. In addition to that, YouTube search services are called twice; the first call is done to the YouTube V2 API[12] where we specify in addition to the keywords a high level category to be targeted. To do so, we have implemented a Categories Mapper function that maps the high levels categories of Alchemy's API with those provided by YouTube. The second call is done to YouTube V3 API[13]. The new feature provided by Google in this version is the ability to search using a semantic concept that corresponds to a Freebase concept ID as it proved to retrieve better results that the normal search. Freebase concept calls are cached for longer periods as they are less prone to changes.
2. **Microposts Services**: They include Twitter, Google+ and Stackoverflow. Language filtering is done where applicable.

[9] http://developer.zemanta.com/docs/suggest/
[10] http://www.alchemyapi.com/api/keyword/
[11] http://developer.yahoo.com/yql/
[12] https://developers.google.com/youtube/2.0/developers_guide_protocol_category_keyword_browsing
[13] https://developers.google.com/youtube/v3/

3. **General Search**: This includes similar results found via Google search or those retrieved from the Zemanta API call. They are general articles or blog posts related to the current active page.

At the end of this process, we will end up with a relatively big JSON file containing all the results retrieved from all the services. They are grouped by the corresponding service name that identifies the parser which will handle them in the next step.

2.3 Data Parser

This is where we unify the results and wrap them in a single social model. Adding a new parser is easy, we just need to implement the *social_parser* interface which produces a *social_service* object.

Social (or folksonomic) tagging has become a trending method to describe, search and discover content on the web. Folksonomies empower users by giving them total freedom in choosing their categories and keywords that they think describe best the content. This contrasts with taxonomies that over-impose hierarchical categorization of content [1]. However, in services like Twitter and Google+; tagging has been abused in a way that increased noise in the stream of results. To overcome this problem we align the incoming stream of posts with the set of semantic concepts or keywords that describe the document. There are several approaches and tools like [1] [3] [4] [5] that aim at solving this problem. In SNARC we rely on two levels of reconciliation: one uses the keywords and the second one exploits the high level taxonomy extracted from Zemanta and Alchemy; this process is influenced by the reconciliation module implemented in [2].

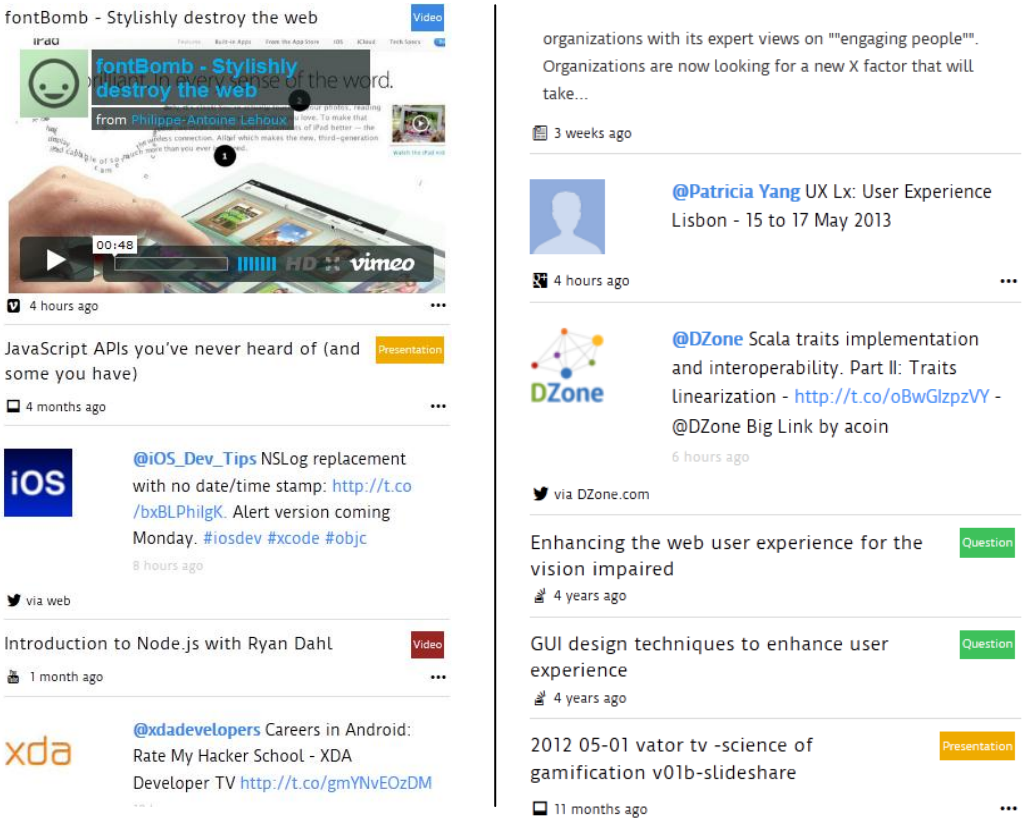
The social modeler is the process where the *social_service* object is created in order to capture all the information needed from the several services. To do so, we have created an optimized universal social model. It contains all the necessary data to model social information and can be reused in other projects. The model contains the following attributes:

1. **Service related information**: Service Name, Service Type.
2. **General post information**: Author Name, Author Profile Link, Author Image, Time, Geo-location, Program
3. **Post information**: Title, Thumbnail, Embed code, Content, Link

So far, we have filtered the results returned and wrapped them in a common social mode. Since the number of results can be quite large sometimes (although we have set a limit for the results returned in the query layer) we need to show the most recent results first. Therefore, we unify the time representation and sort the results based on it. afterwards we pick the top N results and shuffle them in order to view them nicely in the front-end.

3 Front-End

SNARC is a web service that generates a JSON file containing the results wrapped in our universal social model. We have implemented two separate front-ends that consume SNARC's service: a Wordpress plugin that displays related social news to the current page visited (i.e. http://goo.gl/dF2Hg) and a chrome extension that can be loaded on any web page or application. Figure 2 shows two screenshots of SNARC's UI in the Wordpress plugin.



SNARC is a perfect fit to be used within blogging platforms. The pre-assigned post tags and categories optimize the results returned by the service. As for the Wordpress plugin, we have created a special service that accepts an array of keywords and categories as an input. This service matches them with those generated by the Document Handler and picks the best matches.
The Chrome extension offers more flexibility for users to load related social news anytime on any webpage or application. The results are visualized using jQuery templates as a sliding panel on one of the screen edges. The items are scrolled automatically as an infinite carousel, and the animation is paused when the user is active on the reserved area. To minimize the load time, only the first rich multimedia object is embedded, while the others are only shown on request.

4 Conclusions and Future Work

Aggregating relevant social news is not an easy task. SNARC performs the task in a nice and intuitive way that allows the user to discover what is happening instantly and without the need to navigate away from the current page. One of the important things to consider for the future is the integration of better reconciliation features and tools to ensure the display of relevant social posts. Moreover, the introduction of real-time feature than can also push new related posts. We would also want to test SNARC on business web applications. It can be a good fit to perform brand monitoring especially after plugging a sentiment analysis component. We want also to evaluate the necessity to use a content scrapping API like Alchemy's[14] as a fallback for out text extraction mechanism. Finally, we plan to introduce improved recommended content; like people to follow on Twitter or Google+ [6] or video channels to subscribe to.

5 References

1. Zanardi, V and Capra, L (2008) **Social Ranking: Uncovering Relevant Content Using Tag-based Recommender Systems**. In: RECSYS'08: PROCEEDINGS OF THE 2008 ACM CONFERENCE ON RECOMMENDER SYSTEMS. (pp. 51 - 58). ASSOC COMPUTING MACHINERY
2. H Khrouf, G Atemezing, G Rizzo, R Troncy, T Steiner. **Aggregating Social Media for Enhancing Conference Experiences**. Proceedings of the 1st Int. Workshop on Real-Time Analysis and Mining of Social Streams 01/2012
3. Preotiuc-Pietro, Daniel, Samangooei, Sina, Cohn, Trevor, Gibbins, Nicholas and Niranjan, Mahesan (2012) **Trendminer: an architecture for real time analysis of social media text**. In, *6th International AAAI Conference on Weblogs and Social Media (ICWSM-12), Dublin, IE, 05 - 07 Jun 2012*. 5pp
4. Iván Cantador, Alejandro Bellogín, Ignacio Fernández-Tobías, Sergio López-Hernández edited by Christian Huemer, Thomas Setzer, Will Aalst, et al. **Semantic Contextualisation of Social Tag-Based Profiles and Item Recommendations E-Commerce and Web Technologies** Vol. 85 (2011), pp. 101-113
5. Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. 2012. **Real-time top-n recommendation in social streams**. In *Proceedings of the sixth ACM conference on Recommender systems* (RecSys '12). ACM, New York, NY, USA, 59-66.
6. Marco Pennacchiotti and Siva Gurumurthy. 2011. **Investigating topic models for social media user recommendation**. In *Proceedings of the 20th international conference companion on World wide web* (WWW '11). ACM, New York, NY, USA
7. Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. 2012. **The role of social networks in information diffusion**. In *Proceedings of the 21st international conference on World Wide Web*(WWW '12). ACM, New York, NY, USA

[14] http://www.alchemyapi.com/api/scrape/