SNAP: THE SOCIAL NETWORK ADAPTIVE PORTAL

Alexiei Dingli,Mark Scerri, Brendan Cutajar, Kristian Galea, Saviour Agius, Mark Anthony Cachia, Justin Saliba, Jeffrey Cassar, Erica Tanti,
Sarah Cassar, Shirley Cini, Mariya Koleva
Faculty of Information and Communication Technology (ICT)
University of Malta
Msida, MSD 2080, MALTA

ABSTRACT

Since the boom of social networking lead to people using multiple account on many platforms in order to keep in touch with hundreds of contacts, managing one's contacts risks becoming a burden for many users. Following and finding information about friends and family has become an issue too. Guided by these observations and by careful research of existing adaptive web technologies, our team worked on the development of SNAP - an adaptive social network integrator which aimed to amalgamate four social networks (Facebook, Twitter, Flickr and Buzz) in one adaptive environment, which to unobtrusively sort the users' feed according to his/her preference. To achieve data transfer and authorisation, SNAP used APIs and the newest version of the OAuth protocol. Adaptivity was achieved through statistical filtering. Despite efforts, the initial field tests show that the system is not as yet ready to be launched for wider use. However, there is room for improvement in terms of Social Network Integration, and tester users expressed an interest in the idea of using an adaptive social integrator such as SNAP.

KEYWORDS

social networking, adaptive systems, adaptive hypermedia systems, social network portal, social network integration

1. INTRODUCTION

More and more people use social networking sites (SNS) everyday to interact with their family and friends, and for various reasons a number of these people use more than one social network. People maintain a large number of contacts (friends) on their chosen social networks and as most social networks today offer many functionalities, such as status updates, media sharing (text, photo, audio, video), third party applications and games, the users may experience difficulties finding the content in which they are interested.

Another characteristics of SNS usage is that users are not a homogeneous mass but individuals with distinct preferences over which social network is the best for them. These preferences are not static either, and over the time they tend to change in favour of one network or another. Managing several accounts over different portals may become a burden for many users.

For this purpose the SNAP adaptive web portal was developed, allowing the amalgamation of several popular social networks in one integrated environment. SNAP monitors and learns the browsing behaviour of the users and adapts to it, giving prominence to their preferred social networks.

In the following section we are going to discuss previous research on adaptive systems and some issues with their evaluation. In section 3 we present the implementation of SNAP. Section 4 outlines the results from the testing of SNAP and section 5 concludes with some possible future developments.

2. LITERATURE REVIEW

The function of an adaptive system is to offer personalized experience, analyzing the data from the user's interaction with the system (Brusilovsky and Milan, 2007),. It aims to improve the organization and

presentation of websites (Perkowitz, M. and Etzioni,O., 1997 cited in Mican and Tomai, 2010, p. 86) by adaptively selecting, prioritizing and manipulating links and content (Brusilovsky and Milan, 2007). The reasons which gave rise to the development of adaptive systems are varied, for instance, to give personalized recommendations (Balabanović, 1997), to personalize learning experience in tutoring systems (Baena, et al., 2000) or even to and even to adapt information for terminally-ill patients (Bental, et al., 2000). Furthermore, there are many approaches to adaptivity. In this review we are only aiming to present overlay user modelling, which is particularly pertinent to our work on SNAP.

The user model (a model of a user's behavior) is at the heart of an adaptive system. The problem receives prominence in Brusilovsky and Milan (2007) in a discussion about adaptive educational systems (AES), but the same principles can be used for other adaptive systems as well. Brusilovsky and Milan (2007) describe an overlay model as a structural model, which presents the domain knowledge as a set, and the user knowledge - as its subset. The elements of the user knowledge are assigned values (boolean, numeric or a probability) to represent the level of knowledge. When the domain knowledge is modelled, what is modelled can broadly be called concepts. The latter can be facts, rules or constraints. Depending on whether these concepts are independent or related, the domain models are vector (set) models or network models respectively.

Evaluating adaptive websites can be a difficult task, which was also pertinent to our research.. Bauer and Scharl (2000) investigate the possibilities for automatic content and structure evaluation of websites and raise important concerns about the validity of manual (subjective) website evaluation, but their techniques were not immediately applicable on the work on SNAP, one of the reasons being that Braun and Scharl's research is aimed at non-adaptive websites.

Sadat and Ghorbani (2004) propose a hierarchy of features, specifically aimed at the evaluation of adaptive hypermedia systems. The authors classify the features in three main groups: runtime features, technology, and software engineering. Runtime features refer to the way the system works and behaves, technology features include all the algorithms and techniques employed in the system, and software engineering refers to features of the software development. These have been useful in the process of evaluating SNAP.

3. METHODOLOGY

3.1 Basic Information and Functionalities

The SNAP project was developed adopting a spiral software development life cycle in an agile development framework. The front-end layer, which takes care of the user requests, was designed using layouts. The back-end communicates with the social networks via specialized classes. There is one parent class and all other specialized social network classes extend it. The general functionalities of SNAP are defined in this parent class and the network classes can also present unique functionalities. In its final form, SNAP is designed to support different social networks. For the prototype, we attempted to implement Facebook, Flickr, Twitter and Buzz. At the time of writing not all planned functionalities are available to the users. The planned functionalities are as follows:

The parent class retrieves user's posts, user's photos and user's inbox. It also has functionalities to search (e.g. for other users). The Facebook class should retrieve: the user's feed, wall, message inbox, message thread, albums, user's picture. It should also allow the user to post on the Wall. The Twitter class should retrieve user and home timeline, and the user's received direct messages. It has the functionality to send a direct message to one or more recipients, to search for a user, to update and delete a tweet, to retweet and to delete a tweet. The Flickr class has the functionalities to: retrieve photos, display photos in four formats, retrieve information about photos, and search for photos. Activities on Flicker are: commenting, tagging, adding to favourites, adding to gallery, adding to notes. The Buzz functionalities which were attempted are: retrieve posts, comment on posts, link posts, post text,post text and links, post text and photos.

3.2 Front End Specifications

When first coming across the site, a new user registers a new account with the system in a very straight forward procedure. As the new user logs in for the first time, they are redirected to a page , where they can manage social networks. Here the user can add the social network accounts they would like to integrate into SNAP (at least one). In the user's homepage, the user's feed is displayed in an ordered fashion. The social network that has the most priority will be displayed first, then the next in priority, and so on until all the spaces on the screen are filled. The user can also choose which particular social network to see using the filter button. At any time the user can delete a social network or add new ones to the system, making SNAP very useful for people who have multiple account on different social networking sites.
In the homepage one can find a share section. Here the user can share a status with one or multiple social networks. Messages, photos, and profile edit pages are deprecated in the current version.

3.3 Database and Adaptation Implementation

The database is the foundation of this solution. This project depends entirely on storing the user's social network credentials only once throughout their entire use of the website. Each of the user's account access IDs and tokens are stored within the database for quick access, thus giving the user the need to login only once - and the system handles the rest. The database uses stored procedures to perform functions on itself.

The adaptation is provided by a class. The filter class is designed to interface the front end with the back end of the system, the front end being all the displayable sections of the solution while the back-end concerns itself with data gathering and representation. It is designed to satisfy the requests from the front end by processing the request made and then executing all the necessary back-end method calls and object creation. Once this is done, the results are gathered and using some simple adaptivity algorithms are filtered to relay the correct set of elements to the front end for display. This means that this class effectively handles the adaptive section of this solution.

The filter is specifically intended to use statistical data collected from the use of the site, and use it to make assumptions of the user's preferences. Currently, the filters only ensure that it sorts the social networks into the order of preference when displayed on front end and attempts to allocate more space to the more preferred social network accounts.

The following concerns emerged when it came to displaying features to the front end. There are some social networks which do not receive the same bulk updates that others do. If all the space available is given to the heaviest social network then all other social networks are likely to be given much less prominence than they should. We had to ensure that the user will see most of their preferred material while not omitting other material entirely. To solve this we implemented a filter that retrieves the user's social network accounts from the database and iterates through each, requesting data. How much of the returned data is added to the final result depends entirely on the user's preferences. Thus a filtering effect is achieved: allowing a lot of one particular data to pass while blocking some of another.

To handle such data, return types as well as method names of each requested feature had to be uniform. This is to produce code efficiently, reduce maintenance effort and improve readability and overall quality of the code. The methods that are absolutely necessary are dictated by the parent class which all social networks must extend. Then the return objects are dictated via a set of "generic" classes that ensure that returned objects give the necessary data to the front end of the system.

3.4 Authorization

Sharing content and data between SNAP and the social network sites requires authentication, as to ensure confidentiality and integrity of communications. This has been achieved by utilizing the specific APIs of the social networks, when such mechanisms were available, together with the OAuth protocol (Hammer-Lahav, n.d.), which handles the access control. In the prototype the following APIs were used:
The Facebook class uses the Graph API - Facebook's social graph view, which contains objects, such as people, photos and events, and the connections between them (e.g., friend relationships, tags, etc) (Facebook, 2011). The Twitter class uses the Twitterizer API (Twitterizer, 2011). The Flickr class uses the FlickrAPI

(Yahoo! Inc., 2011), which consists of methods and API endpoints(Yahoo! Inc., 2011), to send and retrieve data in all of the following formats: REST, XML-RPC, SOAP, JSON, and PHP.

There was no Google Buzz API for C#, which was an issue, because SNAP was implemented in the .NET framework. An attempt was made to implement a Google Buzz API based on the Java client found in the Google Code page (Google, 2011), but had to be abandoned due to problems with signing requests.

4. EVALUATION

The tests which were planned for SNAP were Black box testing (testing the system with an external view), White box testing (testing whether the correct data constructs are being created during the parsing and receiving of data from social networks), and Field (Alpha) testing, where users were to be asked to fill in a questionnaire to determine their feedback from use of the website. Due to time constraints, the Alpha testing performed was brief and with fewer tester users than would have been desired. However, the developers' team tested thoroughly the functionalities of the website using the Black box and White box techniques. In the process, some of the planned features were deprecated in this version of the website.

During the test period, the system viewed with the following browsers: Microsoft Internet Explorer, Mozilla Firefox, Opera and Google Chrome. The system performed without problems on Opera and with some issues in displaying Twitter images in Firefox. Google Chrome displayed the website and the implemented social network functionalities appeared to be working, but adaptation did not seem to be possible with this browser at the time of testing. At this moment in time, Microsoft Internet Explorer did not work with the system.

Black box tests were planned on five groups of features of the system: initial log in and registration, integration of a social network account, deletion of a social network account, posting, and photo functionalities. Four out of the planned aspects were tested, with exception of the photo functionalities, since these had to be deprecated from the system.

The log-in testing involved testing the registration, first-time log in and subsequent log ins. In the final version of the website, the output of the system matched the expected output, with exception of the registration procedure, when during the return to the login page, some formatting errors occurred.

During the next tests (to add a social network and permissions), Twitter and Facebook accounts were successfully integrated. Flickr, however, could not be. It was known from earlier development stages that a Buzz account cannot yet be implemented without some compromising of the privacy and security of the system. Tests for deleting a social network account ended up unsuccessful, as the accounts appeared deleted, but in reality persisted. The test was performed on Facebook and Twitter accounts only, for the reasons described in the paragraph above.

The posting tests meant to establish: whether the system checks for posts in all social networks added, if it checks for posts in a specific networks (chosen by the user), and if sharing a user status across all social networks is visible on all of them. When checking if all posts from a user's social networks are displayed in order of preference, the test was successful (Chrome had some issues with accounting for preference). With regards to specific social networks, the system displayed all posts from the user's social network account in order of most recent with comments, as it was meant to. Sharing a status across all social networks was successful with regards to Facebook and Twitter was successful.

White box testing was designed to examine the functions of the system as they were coded. Five functions were examined: Facebook Get Photos, Filter get Posts, Filter get user posts, Facebook getInbox. All the tests seemed to be successful, with exception of the Filter get user posts. With the SNAP user as input it was supposed to give a sorted list according to preference of post objects containing post data. The test failed with an empty array.

To complement the developers' testing, 24 users (11 male, 13 female) were asked to review their initial experience with the system in an informal questionnaire. They rated the success of social network integration and various site attributes, namely: overall experience, ease of use, social networks functionality, feed, initial account setup, and privacy and security.

The integration Facebook was rated the best and Twitter, Flickr and Buzz – neutral to negative. The site attributes received neutral scores, with exception of the design and layout which received high scores, and the

site speed, which received low scores. The user also left open-worded comments, in which they most frequently reported problems with displaying the Twitter timeline. Part of the testers indicated in their comments that they would be interested to use a social network integrator of this kind.

5. CONCLUSION

The SNAP adaptive web portal was developed out of the need to have a system which relieves the user of the burden to visit different social network sites separately in order to maintain contact with their friends, family and colleagues. While amalgamating portals exist, SNAP is designed to also respond to the changing interests and information needs of the users. In other words, SNAP was designed as an adaptive portal which displays the user's social networks in an order of preference that the system learns unobtrusively on its own, with the help of statistical filtering.

As SNAP was developed within a tight time frame, only basic testing could be done before launching the website. Only a brief initial test with participants was possible. During the final testing by the developers it was discovered that while the core functionalities are in order, there is still room for improvement in terms of social networks functionalities which had to be deprecated, as well as some browser discrepancies.

Future work on the project is still a viable possibility, since we have retained a number of methods by which we can further improve our solution in terms of features, reliability and flexibility. Some possible new avenues include photo management and uploading, photo album management, and further adaptability by layout changing to emphasize specific functionality. Efforts need to be made to resolve the issues with the integration of Buzz in a way which does not compromise the privacy and security of our users.

REFERENCES

Baena, A. et al., 2000. An Intelligent Tutor for a Web-Based Chess Course. AH '00 Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Springer-Verlag London, UK, pp. 17-26

Balabanović, M., 1997. An Adaptive Web Page Recommendation Service. Proceedings of the First International Conference on Autonomous Agents. Marina del Rey, California, United States, pp.378-385.

Bauer, C and Scharl, A., 2000. Quantitive Evaluation of Web Site Content and Structure'. Internet Research: Electronic Networking Applications and Policy pp. 31-44.

Bental, D. et. al., 2000. AH '00 Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Springer-Verlag London, UK,pp. 27-37

Brusilovsky, P. and Milan, E., 2007. User Models for Adaptive Hypermedia and Adaptive Educational Systems. In: Brusilovsky, P. et. al.,eds. The Adaptive Web. Springer-Verlag, Berlin Heidelberg, pp. 3-53.

Facebook, 2011. Graph API Reference Documentation. Available: http://developers.facebook.com/docs/reference/api/. [Accessed 24 March 2011].

Google, 2011. Google Buzz API. Available: http://code.google.com/apis/buzz/docs/libraries.html. [Accessed 24 March 2011].

Hammer-Lahav, E. et al., n.d. OAuth. Available: http://oauth. net. Last accessed 24th March 2011.

Mican, D. and Tomai, N., 2010. Association-Rules-Based Recommender System for Personalization in Adaptive Web-Based Applications. Proceedings of the 10th international conference on Current trends in web engineering. Vienna, Austria. Springer-Verlag Berlin Heidelberg, pp. 85-90.

Sadat, H. and Ghorbani, A. A., 2004. On the Evaluation of Adaptive Web Systems. Proceedings of the Workshop on Web-Based Support Systems. Beijing, China, pp. 127-136.

Twitterizer, 2011. Twitterizer: We Want to Give your App Twitter. (Updated 22 March 2011) Available: http://www.twitterizer.net/. [Accessed 24 March 2011].

Yahoo! Inc., 2011. The App Garden. Available: http://www.flickr.com/services/api/misc.overview.html.[Accessed 24 March 2011].