

# Shoehicken: An Intelligent System for Recommending RSS/Atom Content

James Horsley, Michael Wooten, and Eman El-Sheikh

Department of Computer Science

University of West Florida

11000 University Parkway

Pensacola, FL 32514 USA

james.horsley@gmail.com, mwooten111@gmail.com, eelsheikh@cs.uwf.edu

## Abstract

We present Shoehicken, an intelligent agent-based system for rating RSS and Atom news content based on user preferences. The system builds a user profile implicitly by observing a user's interactions with news content. Shoehicken rates incoming content based on the user's preferences, and adapts to both long term and short term changes in these preferences. The system is interoperable such that it can supply rated content for use by any news aggregator. This paper describes the Shoehicken system, its architecture, and its implementation.

## 1 Introduction

The World Wide Web (WWW) has become the predominant source for news and information for many people. To address the vast amount of content, a new delivery system has emerged in the form of "channels" or "feeds." These feeds, which are supplied by websites such as CNN [6] and BBC News [2], can be read using traditional web browsers or specialized software, called "news aggregators." The two main formats for these feeds are RSS (Really Simple Syndication or Rich Site Summary) [10] and Atom [1]. Both feed types provide an XML based summary of a website's content, with brief descriptions of articles and links to the actual article content.

Feeds provide easier access to content, but present users with more content that they must filter through. For example, a user subscribing to 10 feeds each supplying 15 articles would have to sift through 150 articles. Current news aggregators do not provide the user with an efficient means, beyond a simple keyword search, of locating relevant content. Over a span of time, users will repetitively consider and discard content that does not match their interests. In order to decide an article's relevance users must either read the

entire article or base their decision on a summary. Due to these limitations, users will either spend more time identifying, or make less informed decisions about, an article's relevancy.

We propose an intelligent article recommendation agent, called Shoehicken, to overcome the limitations of traditional feed aggregation. Incoming articles are rated based on learned user preferences. Shoehicken considers both an article's summary and content when deciding the article's rating. Aggregators can use the ratings to aid the user in determining desirable content. Users are spared the time of manually locating the most relevant articles by browsing highly rated articles first. Shoehicken's intelligent analysis replaces the repetitive task of evaluating irrelevant content.

In the rest of the paper we describe the design and implementation of the system. Section 2 describes related work on intelligent news aggregation. Section 3 describes the architecture of the system and its components. Section 4 outlines the implementation of the Shoehicken agent and the rating system. In section 5, we discuss our conclusions and potential future work.

## 2 Related Work

Shoehicken may be the first agent for RSS and Atom aggregation, but it is not the first system for recommending news content. Other agents and systems, such as [4], [5], [8], and [3] provide the service of rating or recommending news content based on a learned user model. The recommendation may come in the order that the news articles are presented to the user, such as [4], [5], and [8] do, or change the way the content is presented to make more relevant information have greater screen precedence, as [3] does.

There are two forms of feedback used for developing a user model: implicit and explicit. Work by [4] uses implicit user feedback to develop the user model.

Using implicit feedback, the system infers a user's preferences through observation. In contrast, [5], [8], and [3] use explicit feedback, or a combination of explicit and implicit feedback, when evolving a user model. The explicit feedback systems require the user to personally rate the relevancy of the articles they are presented with.

Both [4] and [5] use two user models for recommending news content: a short term and a long term user model. The short term user model keeps track of a user's immediate interests, such as major world events. The long term user model maintains a representation of the interests a user tracks over an extended period of time, such as stock information.

Agents often use feedback to make decisions. The feedback does not have to come from a user, but when recommending news content, it normally does. Feedback is typically categorized as either positive or negative. Positive feedback implies that the agent's actions are useful or correct, and negative feedback implies that they are incorrect. The problem for the agent is determining the cause of the positive or negative feedback. Most applications use both positive and negative feedback in decision making. However, [15] has shown that a user preference model can be built entirely from positive feedback.

Another approach to news content recommendation is collaborative filtering. Collaborative filtering works well in a client/server environment in which each client's agents co-reside on a single server. The user agents exchange information to help each other filter content. An example of this can be found in [14], in which the authors provided experimental data showing the technique's effectiveness.

### 3 System Architecture

One of the goals of the Shoechicken project is to provide a modular design that will allow for flexibility and expandability. The Shoechicken architecture incorporates multiple, smaller components, that can each be modified or exchanged without changing other system components. Shoechicken itself is intended to serve as a module that can be used by other independent systems. Internally, the Shoechicken system is broken down into five modules, which are illustrated in Figure 1.

A key issue with the Shoechicken project is the source of the content. In order to provide the most utility, Shoechicken must support as many content sources as possible. At the time of this writing, there are eight versions of the RSS standard[?] and multiple versions of the proposed Atom standard[1]. The first module, the Feed Subscription Service, is responsible for down-

loading feed content and parsing it into an intermediary form. The module can transform any feed format into a form usable by the rest of the system. Using this approach, the module can be expanded in the future to include support for new feed formats without affecting other modules. The second module is called the Bufunkalo. The Bufunkalo is responsible for accepting the intermediary form, downloading additional content related to it, and preparing the content for the Recommendation Agent by breaking it apart and storing it in the article database. The Bufunkalo is responsible for content at the feed level, and delegates individual articles to the Shredder module.

The third module, called the Shredder, processes individual articles and saves their content to the article database. The article's content is stripped of HTML tags, broken into keyword tokens, and the tokens are categorized by the context (title, summary, or body text) that they are found in. The Shredder includes a stop list filter that is used to remove common keywords (such as the, and, is, etc), leaving words that reflect the article's true content. The Shredder is designed to function as a sequence of operations, and will allow for new operations, such as keyword chaining or stemming, to be added anywhere in the process.

The fourth and most important module is the Recommendation Agent. The Recommendation Agent maintains a user model representing each user's preferences. When the agent receives new articles, it applies knowledge from the user model to articles, and produces a rating that represents the articles' relevance to the user. The Recommendation Agent has a clearly designed interface, such that its implementation can be expanded over time without affecting the rest of the system. Section 4 describes the Recommendation Agent in more detail.

The fifth module is the User Agent. The User Agent is responsible for observing the actions of users and reporting their preferences to the Recommendation Agent, as well as recommending articles rated by the Recommendation Agent to the user. The User Agent provides an interface, called the Gubernatorial Leader (Governor), which the user's software is required to implement. From the perspective of the system, the Gubernatorial Leader represents a user. The User Agent presents rated articles to Gubernatorial Leaders, and accepts feedback from them. Using this approach, any software that chooses to become a Gubernatorial Leader can interact with the Shoechicken system through the User Agent. Potentially, existing RSS/Atom news readers such as Mozilla Thunderbird [9] could be modified to assume the Gubernatorial Leader role.

The Shoechicken modules are designed to work together to form a coherent system. Figure 1 depicts

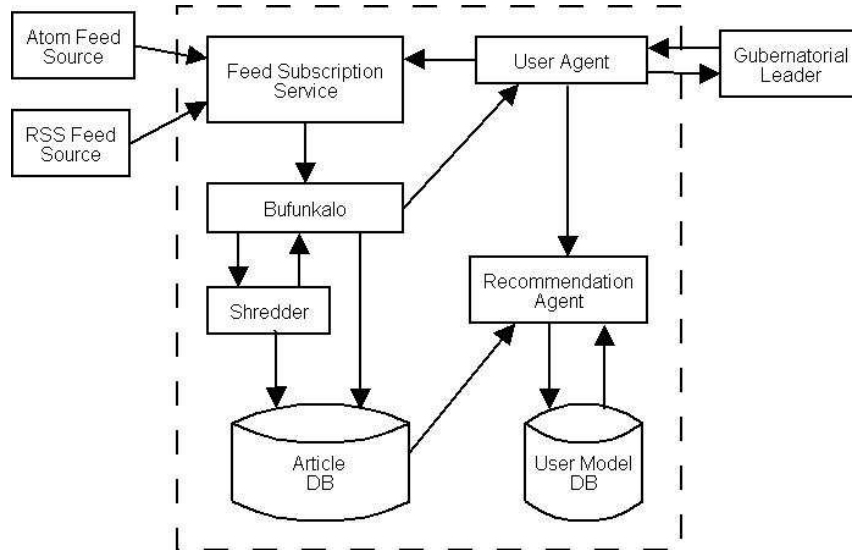


Figure 1: Shoehicken Architecture

the interactions between the modules. The Feed Subscription Service provides the Bufunkalo with parsed feed content in an intermediary form. The Bufunkalo uses the Shredder to prepare the content for the Recommendation Agent and store it in a database. The Bufunkalo notifies the User Agent of the new content after it is processed. When a Gubernatorial Leader requests new articles, the User Agent contacts the Recommendation Agent for the user and requests rated articles. The Recommendation Agent searches the article database for unrated content, and uses the new content and the user model to determine a rating for the content. The Recommendation Agent passes the rated content to the User Agent, which distributes it to Gubernatorial Leaders. In return, the User Agent accepts feedback from the Gubernatorial Leaders and passes it to the Recommendation Agent so that the user model can be updated.

In addition to the Shoehicken modules, an aggregator called the Frying Pan will be developed for the project. The aggregator will implement the Gubernatorial Leader interface mentioned above, and will allow for experimentation and data gathering with the Shoehicken system.

## 4 Implementation

Accurately rating an article for the user is a challenging task that depends on several criteria. To properly rate an article, a program must know what the user is looking for and then know which articles come closest to having what the user wants. To judge

whether an article has what the user is looking for, the article's content has to be understood in some rudimentary fashion. A simple solution could involve asking the user to enter keywords that he or she is interested in, searching all incoming articles for those keywords, and returning the articles with the highest frequency of those keywords. This solution works for applications such as search engines. Users typically query search engines to answer a question. For example, if a user wants to know the population of Texas he or she could just type "population Texas" into most search engines and get an answer. This model does not work for browsing news. By the very nature of news feeds, the user does not know what the most up-to-date news is. An idea to overcome this could be to enter broad topical keywords like "Politics", but this approach will fail in a simple keyword counting scenario because the frequency of the word "Politics" may be low or nonexistent in political articles because the topic is implied. Shoehicken will use a combination of approaches to overcome these issues.

The first approach is that of observing user behavior. This means that when a user performs actions such as reading, deleting, and bookmarking articles, Shoehicken will be notified. By observing what a user reads, Shoehicken is able to record detailed information about the content of the articles that interest the user. To start with, Shoehicken will only note the different words in the articles and their frequencies. By observing what the user deletes without reading, Shoehicken is able to identify what the user does not want to read about and adjust its model of the user's preferences accordingly. This means that

at first Shoechicken will not be able to filter incoming content as it will have no user information. However, as the user begins to pick articles that he or she wants to read, Shoechicken will use these articles as a basis for the user model to rate incoming articles.

Another technique that Shoechicken will employ is latent semantic indexing (LSI) [7]. LSI expands the traditional vector space model [12] by accounting for semantic connections between articles. Identifying these underlying conceptual connections between articles is vitally important to the recommendation agent's effectiveness. Through these connections, a user interested in Russian politics could be recommended articles about former Russian states, even if the articles have no mention of Russia whatsoever.

Since LSI is an extension of the vector space model, each article will be represented by a vector in that vector space. The learned user model is also represented as a vector in the same vector space. As is typically done, the conceptual distances are measured using the cosine of the angle between the vectors. Before applying LSI to the vectors, each vector component represents a term in that article. Typically the value of a component is a weighted value that is often computed using TFIDF [13], which is a statistical technique used to evaluate how important a word is to a document. However, in Shoechicken we want to expand the term weighting to include the context or location in which the term was used within the article. Basic examples of this for RSS and Atom feed parsing of HTML are the summary tag in the feed document, the author of the article, the title tag of the article, the meta keywords tag, and the body of the article. In a simple model of this, Shoechicken will maintain weights for the feed summary and body contexts when determining the total weight of the vector component:

$$w_{ij}(x) = \frac{TF_{ijx}}{\max_{w_k \in d_i} TF_{ikx}} * \frac{IDF_{jx}}{\max_{w_k \in d_i} IDF_{kx}}$$

$$w_{total_{ij}} = \alpha w_{ij}(summary) + \beta w_{ij}(body)$$

$$\beta = 1 - \alpha \text{ and } 0 \leq \alpha \leq 1$$

where  $w_{total_{ij}}$  is the weight of vector component  $j$  for article  $i$ ,  $TF_{ijx}$  is the term frequency value for term  $j$  in the given context, and  $IDF_{jx}$  is the inverse document frequency for the given context. Handling of  $\alpha$ , and thereby  $\beta$ , is discussed below.

Maintaining a user model is difficult. If the model is updated too quickly, short spikes of interest in new topics can overwrite long term interests. However, if the model is changed too slowly, the model does not keep up with a user's changing preferences. For example, a user may have a long term interest in UFOs, exotic travel locations, and his favorite football team.

If a major sporting event such as the World Cup takes place, the user may have a sudden spike of interest in that event. If the user model changes too quickly, an article about the user's football team being abducted by aliens and turning up naked on a remote desert island could receive a low rating when it should get a high one. To accurately keep track of a user's long term interests while adapting to short term ones, Shoechicken maintains two user models. One user model represents long term interests and changes very gradually. The other model corresponds to short term interests and will swing rapidly to new preferences. Each model corresponds to a vector in the vector space model. When calculating article's rating, the cosine similarity between the article vector and both the long and short term user vectors is calculated. A rating is determined for the article based on a weighted combination of both similarity values:

$$\cos(x, y) = \frac{x^T y}{\|x\| \|y\|}$$

$$rating(x) = \gamma \cos(U_{long}, x) + \epsilon \cos(U_{short}, x)$$

$$\epsilon = (1 - \gamma) \text{ and } 0 \leq \gamma \leq 1$$

where  $U_{long}$  is the long term user model vector, and  $U_{short}$  is the short term user model vector,  $x$  is the vector of the article being rated, and  $rating(x)$  is a rating value of that article which will range from zero to one. The handling of  $\alpha$ , and thereby  $\beta$ , is discussed next.

There are two variable weights that have been discussed so far; for instance, the weight with which different user feedback events are interpreted. It is the intelligent agent's job to maintain these weights in light of feedback from the user. Shoechicken is designed to take chances by adjusting various weights. If a change results in positive feedback, then the change should be kept. If a change results in negative feedback, the change should be undone. How to adjust these weights largely falls upon how user feedback is interpreted. Shoechicken considers both positive and negative feedback when making decisions. An example of what Shoechicken considers negative feedback is a highly rated article being deleted without being read; particularly if much lower rated articles were read first. An example of positive feedback for Shoechicken is the user reading all the highly rated articles and not reading the lower rated articles. These cases are very simplistic, but interpreting their subtle variations needs to be handled carefully to properly model the user's preferences. Reacting to this feedback will be adjusted as the system enters a testable state from which data can be retrieved.

The Shoechicken system continues to be devel-

oped and evolve. The feed subscription service is completed and current work focuses on the Recommendation Agent, Shredder, and article database. Since a great deal of the focus and complexity lies in the intelligent agent, it will be developed in tandem with the other components.

## 5 Conclusions and Future Work

RSS and Atom feeds provide a vast information space, however little work has been published on applying intelligent techniques to help users locate relevant content. We have proposed an intelligent agent-based system, Shoechicken, for rating news feed content. We believe that Shoechicken will save users' time by allowing aggregators to present the most applicable articles first. The rating system will shift the burden of searching for relevant information from the user to the system.

Work continues on the Shoechicken modules. Additionally, we intend for the Feed Subscription Service to have full support for all versions of the RSS and Atom standards. Once the modules are complete, we would like to develop the Frying Pan aggregator to experiment with the effectiveness of the Shoechicken system. An eventual goal is to integrate the system with an independent aggregator in order to test its interoperability.

There are several additions that could be made to the Shoechicken system in order to improve its flexibility and effectiveness. All communication within the system is performed using the Java language, limiting its ability to interact with aggregators that are not implemented in Java. To overcome this drawback, we recommend that all external communication with the system be implemented as an XML service. To improve the accuracy of the system, the Shredder should include a stemming operation to reduce the number of unique words presented to the Recommendation Agent. Stemming is a recommended practice in information retrieval that reduces words to their root form [7]. The completed Shoechicken system will provide a more beneficial news reading experience for users. The system will save users time by providing them with a means for reading the content that is most relevant to them. The system will be interoperable with multiple new aggregators, which will allow users to choose their preferential aggregator and still receive rated news content. In conclusion, we believe that Shoechicken will provide a better and more efficient news reading experience for users.

## References

- [1] What is Atom?, <http://www.atomenabled.org>
- [2] BBC News, <http://news.bbc.co.uk>, 2005
- [3] Bharat, K., Kamba, T. and Albers, M., Personalized, Interactive News on the Web, *Multimedia Systems*, Vol. 6, Issue 5, pp. 349-358, 1998
- [4] Billsus, D., Pazzani, M. and Chen, J., A Learning Agent for Wireless News Access, *Proceedings of the 5th International Conference on Intelligent User Interfaces*, pp. 33-36, 2000
- [5] Billsus, D. and Pazzani, M.A Personal News Agent that Talks Learns and Explains., *Proceedings of the third annual conference on Autonomous Agents*, pp. 268-275, 1999
- [6] CNN Homepage, <http://www.cnn.com>, 2005.
- [7] Dumaïs, S. T, Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, pp. 23:229-236, 1991
- [8] Lang, K., Newsweeder: Learning to Filter Netnews, *Proceedings of ICML-95, 12th International Conference on Machine Learning*, pp. 331-339, 1995
- [9] Mozilla Thunderbird, <http://www.mozilla.org/products/thunderbird/>, 2005
- [10] RSS 2.0 Specification.(2005, January 30), <http://blogs.law.harvard.edu/tech/rss>, 2005.
- [11] Pais, What is Stemming?, <http://www.comp.lancs.ac.uk/computing/research/stemming/general/>
- [12] Salton, G, The SMART Retrieval System: Experiments in Automatic Document Processing, 1971
- [13] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., 1983
- [14] Sarwar, Konstan, Borchers, Herlocker, Miller, and Riedl, Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system, *In Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1998.
- [15] I. Schwab, W. Pohl, and I. Koychev, Learning to recommend from positive evidence', *in Proc. 2000 Int. Conf. on Intelligent User Interfaces*, pp. 241-247, New Orleans, (2000).