INTERSYND: A WEB SYNDICATION INTERMEDIARY THAT MAKES RECOMMENDATIONS

Adrian P. O’Riordan ¹), M. Oliver O’Mahoney ²)

Abstract
This paper describes the design and development of a Web syndication intermediary that works with various syndication technologies, specifically the RSS family and Atom. We developed format-neutral middleware that sits between content sources and the user. Another objective was to add a recommendation component transparently to the intermediary. Implicit collaborative recommendations of new feeds are made. The system, called InterSynd, isn’t tied to any particular feed reader; we developed a simple Web client for proof of concept. The extensible syndication software was built using the open standards ROME library and JDOM.

1. Introduction

Syndication has become a popular means of delivering relevant timely information to people. In general, syndication is the supply of information for re-use, for example print syndication, where newspapers or magazines license articles or comic strips. Web syndication is based on a publish-subscribe paradigm where XML-based formats such as RSS are used for the syndication of Web content such as blogs and news to Websites as well as directly to users. The basic Web syndication concept was developed at Apple in 1995 in the form of the MCF (Meta Content Framework) [1]. Web syndication is now used for a myriad of purposes, including publishing, marketing, news updates, bug-reports, and other activities involving updates. An update is published through a web “feed”, and notifications are sent to each subscribed user. A single notification is commonly referred to as a feed; a feed source is multiple instances of feeds in one location.

Feed readers or aggregators amalgamate a collection of subscribed web feeds, usually contacting different sources to do so, and allow users to read the content. An aggregator might be desktop software or a web application. Web-based feed readers such as Google Reader or Bloglines allow users to read content with a browser. Standalone e-mail-style applications such as Thunderbird are also popular. Client libraries process feeds programmatically.

There was been a proliferation of different feed formats and technologies proposed and in operation. One source of motivation for this work is the onus this places on the programmer. Many content providers publish information using only one technology or the other. For example the BBC and CNN both publish on-line using RSS but not Atom (whereas Google use both). A more detailed discussion of the main technologies is given in the next section. This proliferation has led to the development of different programs and libraries that can publish and process these formats, e.g. Userland’s Manila [27]. Many websites and software programs are written to support a particular

¹ Department of Computer Science, University College Cork, Cork, Ireland, a.oriondo@cs.ucc.ie
² Department of Computer Science, University College Cork, Cork, Ireland, moom1@student.cs.ucc.ie
flavour of RSS or Atom. Libraries for processing all the RSS versions and Atom include the ROME and Jakarta Feedparser [24].

In this project we build upon the ROME library [16], an open source Java library for processing feeds in a format-neutral way. We implemented a simple Web client that can read feeds of different formats transparently. All details of the syndication technology are hidden from the user. We also extended the functionality of the system by adding a recommendation component.

The paper is structured as follows. Section 2 provides context with background technical material on Web syndication and recommender systems. Section 3 presents our overall solution, a syndication interchange that makes recommendations and illustrates the overall architecture of our approach. Based on this overview, section 4 provides a detailed analysis of the implementation. Section 5 gives an overview of related work. Finally, section 6 summarizes the main achievements of the paper and suggests areas of future work.

2. Background

We begin this section by providing an overview of the main Web syndication technologies in use today, RSS and Atom. We then introduce recommender systems; information-based systems that recommend or suggest content to users.

2.1. RSS and Atom

RSS is the most popular form of Web syndication at present [12]. It is considered to be a “lightweight” syndication format. It has an XML specification; filename extensions are .xml or .rss. RSS is actually a family of related technologies. There has been a proliferation of different versions of RSS: RSS 0.90, RSS 0.91 Netscape, RSS 0.91 Userland, RSS 0.92, RSS 0.93, RSS 0.94, RSS 1.0, and RSS 2.0. RDF Site Summary (RSS 0.9) was created at Netscape for the My.Netscape.com portal site. A simplified version, RSS 0.91, was also released by Netscape and later versions such as 0.92 through 0.94 are based on this. The RSS 2.0 specification, the basis for many extensions, is copyrighted by Harvard University and is frozen so that no significant changes can be made to it [13]. Extensions via modules allow RSS to carry multimedia payload (RSS enclosures), and support ecommerce (ecommerce RSS). A module is a standard way of extending the core RSS specification using XML Namespaces. Briefly stated, an RSS feed consists of a channel with a number of items (content) within this channel. The compulsory <rss> tag element delimits the root element in the XML document structure. Channels have three required fields: <title>, <link> and <description>.

ATOM is the main alternative to RSS currently, and was born out of the limitations of RSS. Atom is a proposed IETF standard [14]. Essentially it contains <feed> elements after the XML declaration, stating meta-data about the feed source. There are many technical differences between the RSS family and Atom [3]. Here are listed a couple of the principal differences: (1) RSS 2.0 may contain either plain text or escaped HTML, with no way to indicate which of the two is provided whereas Atom allows detailed payload metadata; and (2) Atom allows standalone entries whereas RSS does not.

2.2. Recommender Systems
Recommender systems or collaborative filtering systems produce personal recommendations by predicting items of interest based on users’ behaviour. The opinions of users can be obtained explicitly from the users or by using some implicit measures. Collaborative approaches to filtering or recommendation exploit the profiles of a community of users [20]. Recommendations for a user are generated from the profiles of other users who are deemed to be related. In the case of feed subscriptions, these are feeds that are not already in the user’s subscription list but which are in the subscription lists of users with similar profiles. Collaborative recommendation can be user-based (memory-based) or item-based (model-based) depending on whether users are rated and used as parameters or items have such ratings attached [6].

3. Syndication Interchange

This section motivates the syndication interchange project by showing how it fits with existent technological solutions. An overview of the user interface is given. A brief description of the system architecture ends this section.

3.1. InterSynd Overview

InterSynd is middleware placed between the content source and the user application. The situation is complicated by the existence of blog or ping servers that sit between the source and the reader or aggregator. Weblogs.com (Verisign) and blo.gs (Yahoo!) are two such servers. The “pinging” is done by an XML-RPC-based push mechanism. Server side readers such as Bloglines and portals such as MyYahoo! can also sit between content source and user. InterSynd is positioned as shown in Figure 1.

![Figure 1. Syndication Middleware](image)

Many aggregators/feed reader systems also incorporate a filtering system and search capabilities. However, with a recommender system in place you can both filter and recommend/present (effectively search) syndication feeds based on user interests. InterSynd was not intended to replace a feed reader. It is middleware that could be utilized in conjunction with a feed reader. Presently we have a simple InterSynd client that serves as reader and interchange. The system allows an end-user to register by providing personal information such as name, username, and password. The system maintains user records and feed histories. Each user is able to add to or remove from his/her subscriptions. The system also provides recommendations.
Figure 2 shows a screenshot of the Web client interface, implemented in DHTML and Javascript/Ajax. The top right hand corner of the page has the following links: Home | Manage Feeds | Menu | Register (| logout). When a user is logged in, his/her subscriptions, sorted by date, are shown in the main body of the web page. The recommendations are shown on the right hand side of the screen. These are displayed by the feed source’s title and the titles of each sub-entry in that feed document.

For recommendations we devised a simple implicit hybrid user/item-based collaborative algorithm. This employs statistical techniques to find a set of users, known as neighbours that have a history of agreement with the target user. A neighbour is defined as any other user who has 20% plus feeds in common. This parameter is adjustable. It then calculates the number of occurrences of each feed source in the neighbourhood set that is not in the user’s subscription list. A weighted average of the occurrence data for the subscriptions for that subgroup is calculated where the weights are based on the inverse of the feed list sizes. The resulting function is used to recommend feeds for which the user has expressed no personal interest as yet.

We also include a diversity feature. Recommender systems can lack diversity if based entirely on similarity algorithms [20]. This enables the inclusion of items that wouldn’t otherwise be recommended to broaden profiles. We endeavour to recommend feeds that are similar to the user’s existing subscriptions but that are different from each other. We implement a simple algorithm that introduces items not at the top choice as defined by the function outlined above but which users may be interested in. We modify the recommendation function to factor in a diversity term, which is a measure of candidate specificity across all subscriptions. A bias is given to a less common feed. Basically this introduces some diversity so that the most popular feed sources are not always presented.

3.2. System Architecture

Our system architecture, shown in Figure 3, is based on Sun’s Java 2 Enterprise Edition (J2EE). J2EE provides several useful technologies such as Servlets, JSP (Java Server Pages), and Java Beans that can be glued together in a modular fashion using an MVC (Model View Controller)
architecture. Java Beans encapsulate the Model, accessing a MYSQL database using the Java Database Connectivity (JDBC) API. The View of the application system is implemented in JSP. The Controller consists of syndication, utilities, user, and recommender components for data manipulation. These essentially break-up the Servlets for manipulation of the model level data.

Figure 3. System Architecture

4. InterSynd Implementation

We wanted to work with an API that was lightweight, extensible, efficient, and also supported feed parsing. ROME (RSS and atOM utilitiEs) is an extensible open source Java library developed by Sun/Java.net for reading and publishing feeds in a format-neutral way. It defines a simple pluggable architecture for extensions; see the documentation for details on the plug-in mechanism. Rome supports the Dublin Core [4] and Syndication [5] metadata element sets. Rome used JDOM 1.0 [17] for parsing. We used ROME API version 0.9 and ROME Fetcher version 0.9. ROME (version 0.9) itself consists of 61 interfaces and classes grouped into four packages. Most of the interfaces (e.g. SynContent) have a corresponding default implementation (SynContentImpl) that can be used or extended.

In ROME feeds are manipulated by implementing the SyndFeed interface. The use of SyndFeed, a format independent Java object, makes ROME independent of any particular syndication format. Any particular format (such as Atom) is converted into a generic SyndFeed. Any future XML formats could be implemented as plug-ins for ROME. The following paragraph gives an outline of the Java objects that are created in the conversion. Note that ROME does not attempt a parse-at-all-costs approach unlike some implementations such as Mark Pilgrim’s Universal Feed Parser [24]. Instead it throws exceptions if the XML is ill-formed.

Each SyndFeed contains a number of SyndEntries – data that will be utilised to display a feed Title, URL and Description at the very minimum. The SyndFeedInput class parses a feed (object of class SyndFeed) using it’s build method. The URL is wrapped using an XmlReader. SyndFeedInput then delegates to WireFeedInput to do the parsing. WireFeedInput uses a FeedParser, which in turn employs JDOM to parse the feed into a WireFeed. If it is an RSS feed, the WireFeed is of class Channel and contains other elements defined in the package com.sun.syndication.feed.rss. If it is an Atom feed the WireFeed is of class Feed and assigns the values from the com.sun.syndication.atom package. The WireFeedInput returns a WireFeed either way. SyndFeedInput creates a SyndFeedImpl from the WireFeed. SyndFeedImpl converts WireFeed (RSS or Atom format) to a SyndFeed (no particular format) and returns a SyndFeed.
The ROME Fetcher provides a simple means of retrieving feeds using HTTP conditional GET handling HTTP response codes (e.g. 404 Not found) including unrecognized HTTP response codes. It can be used with or without a cache. The HttpURLFeedFetcher class does the actual HTTP request. The retrieveFeed method then creates a SyndFeed. We cache values using a Hash Map implementation (HashMapFeedInfoCache) for efficiency. Before the feed is retrieved from the source, it examines the last modified date. If this has changed it retrieves the feed, otherwise it ignores the unchanged content, saving bandwidth.

Following is a brief summary of how the recommendations take place. The Recommender cycles though the user sessions to determine who is logged in. A common list of popular feeds is also generated. A SyndFeed object is constructed for each user. Recommendations in the form of lists of recommended feeds are produced for each user via the user-based algorithm. If there are not enough recommendations, the list is populated with the most popular feeds to a maximum of five. All recommended FeedIDs are converted to their specific URLs so as to be retrieved. Next feeds are retrieved using ROME’s Fetcher and stored as an ATOM feed. All feed entries are stored and delivered (printed). The Fetcher is multi-threaded, and can fetch many feeds at once.

![Figure 4. Package Diagram](image)

Figure 4 shows how InterSynd’s classes are partitioned in namespaces in a UML Class Diagram. Table 1 below gives summary metrics on the number of lines of code for this implementation. Note that these numbers exclude front-end and library code which is part of ROME, the Fetcher or JDOM.
<table>
<thead>
<tr>
<th>Package</th>
<th>#Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>88 java; 1069 jsp</td>
</tr>
<tr>
<td>bean</td>
<td>202 java</td>
</tr>
<tr>
<td>db</td>
<td>660 java</td>
</tr>
<tr>
<td>exception</td>
<td>26 java</td>
</tr>
<tr>
<td>recommender</td>
<td>171 java</td>
</tr>
<tr>
<td>syndication</td>
<td>756 java</td>
</tr>
<tr>
<td>user</td>
<td>434 java</td>
</tr>
<tr>
<td>util</td>
<td>368 java</td>
</tr>
<tr>
<td>total:</td>
<td>2705 java; 1069 jsp</td>
</tr>
</tbody>
</table>

Table 1. InterSynd Code size

All the user logins are presently stored in the user_subscriptions MYSQL table. Feeds get added and returned by generating a SQL SELECT or UPDATE statements. Popular feeds for each user are defined as the most popular feeds that the current user is not subscribed to. Figure 5 shows the schemas and relationships of the database tables in an ER diagram. The global_feed_list and user_details only contain unique entries dictated by the primary keys FeedID and login, respectively. The tables’ user_subscriptions and user_preferences can contain many entries due to the fact that a user can subscribe to more than one feed and can have more than one preference. Preferences are not used in the current version but would allow for more detailed recommendations.

Figure 5. ER Diagram of Database Tables

5. Related Work

The most relevant emerging technologies and related work with respect to our application are now described.

Several projects involving RSS and recommendation have been published recently. Chen et al. developed a recommender system for personalized advertising in a RSS reader application [9]. This system creates private dynamic user profiles. Celma et al. describe a collaborative music recommendation system based on social networks where information is extracted from music related RSS feeds [8]. Kobayashi and Saito describe an RSS based information recommendation that does
not depend on user profile data, but on topical news information [19]. These three projects only cater for RSS feed formats. Jun and Ahamad describe a feed exchange system, where hosts can exchange feeds with similar neighbours [18].

The main alternative to ROME is Informa [25], a software library that predates ROME and is widely used. Informa is complicated and doesn’t support RSS 2.0 output or Atom 1.0. Informa does have other interesting features such as good support for persistence and search (via Lucene). Using lower-level APIs such as JDOM or SAX2 is another alternative, but would involve duplicating code already in ROME.

While Atom and RSS dominate on the Web, other XML-based formats that can be used for syndication exist, such as ICE [7] and XOXO [21]. ICE is a heavyweight protocol that aims to “automate the scheduled, reliable, secure redistribution of any content” [7]. XOXO enables publishing outlines and blogrolls. Simple Sharing Extension developed at Microsoft and available under a Creative Commons License extends the RSS and Atom feed formats to enable the aggregation of information by the “bi-directional, asynchronous synchronization of new and changed items amongst two or more cross-subscribed feeds” [23]. The Windows RSS Platform is an API for applications developers to access feeds and subscriptions supported by Internet Explorer 7. The platform includes a data store, a sync engine, and a feed list. The constituent Feed Download Engine downloads feeds and merges them into a feed store. Yahoo! Pipes [26] is a web application that provides a graphical editor for non-programmers to aggregate feeds. Users write rules for how that content should be filtered. Unlike free open-source systems such as ROME, Yahoo! does not allow unrestricted feed access by policy. Google Data API or GData [10] is a an HTTP-based protocol proposed by Google that combines XML-based syndication formats (Atom and RSS) with a feed-publishing system based on the Atom Publishing Protocol. GData unlike RSS and Atom have some support for queries and updates. Any web service can provide a GData feed. APML (Attention Profiling Mark-up Language) [ref] is an XML-based format supported by Bloglines that can be used to store a user's interests. This protocol is not widely supported at present. Google Reader [11] has a feed auto-discovery feature which allows users to search for new related feeds. Users of Google Reader can now as well make feeds visible to their Google Talk contacts.

6. Summary and future work

This paper describes middleware for developing applications that transparently use feed technologies like RSS and Atom. A collaborative recommender component was included. A goal of our approach was not to develop completely new software from scratch, but rather to base our work on already existing systems and standards. This we achieved with a fully operational client application.

A review of the background material and related work shows that a number of efforts are underway to process feeds with greater ease and flexibility. There is also considerable interest in applying ideas from collaborative filtering in this area. To demonstrate our extension of ROME a simple feed interchange application called InterSynd was developed. This is open source and extensible in itself and will serve as the basis for future work.

It remains to use InterSynd with an existing fully-featured feed reader such as Thunderbird. This is an area of current work. Another improvement would be to allow the import and export of subscription lists as OPML (Outline Processor Markup Language) [22], a proposed standard for the exchange of lists of web feeds between Web feed aggregators. ROME Aqueduct-Prevayler [15] or
some other mechanism could be included to enable persistence. SyndInter currently doesn’t process multimedia content. This could be added because ROME (as of version 0.9) supports enclosures and the mediaRSS module. Another area for future work is privacy issues that were not a focus of the work before now.

References


