PODCAST PINPOINTER: A MULTIMEDIA SEMANTIC WEB APPLICATION

Aidan Hogan, Andreas Harth, John G. Breslin

Digital Enterprise Research Institute, National University of Ireland Galway,
Galway, Ireland

firstname.lastname@deri.org

Keywords: Semantic Web, Podcast, Multimedia.

Abstract

In late 2004, a new method of publishing multimedia broadcasts on the Internet became popular called 'Podcasting'. Podcasting incorporates existing feed description formats, namely RSS 2.0 (Really Simple Syndication), to deliver various enclosed files which allows users to subscribe to feeds, receiving updates periodically. Originally intended for self-publishing and syndication of audio files, usage of Podcasting for video files has become quite popular. Indeed, thousands of Podcast feeds are now available, reaching a wide range of listeners and viewers. The rapid development of such a technology proves the demand for structured formats of describing multimedia data, facilitating location and retrieval of desirable media for consumers. This paper proposes a specification to allow Podcast feeds be integrated into the Semantic Web framework. We are thus able to combine data from Podcast feeds with instance data from other sources and also able to reuse existing Semantic Web technologies such as repositories and user interface applications.

1 Introduction

As bandwidth increases and new internet-friendly compressed multimedia formats become popular, demand for, and consequently supply of multimedia files on the Web has experienced a surge in volume. However, such a proliferation of multimedia data corresponds in difficulties with regards data retrieval. The information overload problem is not specific to the multimedia domain however, and is a more general symptom of the expansion of the Web. Many researchers are focusing their efforts on solving said data retrieval problem, one such venture being in the area of the Semantic Web [1].

Podcasts have illustrated the demand for structured formats to describe multimedia by having experienced a phenomenal escalation in popularity (through radio station Podcasts or through individuals "audio blogging" and "video blogging" / "vlogging"). As is, Podcasts are described in RSS 2.0 format files, with supplementary specifications cropping up to allow multimedia-specific descriptions to be stipulated. Unfortunately, however, the RSS 2.0 format of feed description does not exploit the capabilities of RDF (Resource Description Framework) and is not a participant of the Semantic Web venture.

RDF, as a W3C recommendation, is a framework for providing the Web with structured data. There is a considerable amount of research being pursued in developing the technology and applications pertaining to RDF and the Semantic Web. Currently, many applications exist to handle and analyse such data and ultimately provide services of utility to users, and to bring order to a rather chaotic World Wide Web [2,3].

RSS 1.0[1] is a format for creating feeds based on RDF/XML. As such, it is within the realm of the Semantic Web.
This paper illustrates work with the following contributions

- We propose a Podcast vocabulary in RDF Schema and provide a method to convert RSS 2.0 Podcast feeds to RDF.
- We present a prototype that gathers or harvests RSS 2.0 Podcast feeds, uses the said conversion and then applies existing SW tools to create an application to search and browse Podcast descriptions which we entitle "Podcast Pinpointer".

2 Podcasting Revolution

Podcasting is very much a booming technology[2]. From humble beginnings, it has become a prevalent force in multimedia syndication and distribution.

Its origins lay in the introduction by Dave Winer, responding to demand from users, of an enclosure element in his company's RSS format, allowing the introduction of audio and video syndication through the model. Such an idea inspired a natural progression towards the contemporary Podcast feed format with strong grass roots support by blogging enthusiasts.

Part of its strength lies in its relative simplicity, allowing casual users to create and publish *online radio shows* and get them to a wide audience. All a user needs to create a Podcast is some simple recording equipment, a basic understanding of

[1] http://purl.org/rss/1.0/
[2] http://www.tdgresearch.com/press044.htm

RSS and some web space. It is also convenient for consumers, who can use traditional feed-catching methods to subscribe to a Podcast feed and receive automatic intermittent updates.

However, it is not only casual users that are publishing Podcasts, as larger organisations have begun to see the positive aspects of such technologies. Many companies are publishing media via Podcasts, ranging from the BBC and ABC News to NASA and Disney. Indeed, there has been much interest shown by many American radio stations, who have begun making Podcasts of their programmes available online (e.g. NPR's Science Friday).

Other companies have taken an interest in promoting the technology. Apple has become heavily involved in the area of Podcasting. Being implicitly involved from the start (Podcasting being a portmanteau involving a reference to their portable MP3 player, iPod, and broadcasting), they took an active role in June '05. Some of their products such as iPod and iTunes were already Podcast friendly, featuring music file synchronisation features exploited by Podcasting origins. They began providing Podcatching software, which allows users to browse and subscribe to feeds, and also a Podcast directory, a categorised listing of broadcasts. They upgraded the firmware on iPods to display Podcasts in the top-level music menu. They also built a specification for describing aspects of Podcasts which would appear in a predefined format on iTunes and iPod displays[3]. This *itunes* namespace[4] is intended for use within an RSS 2.0 environment.

Besides this specification, Yahoo! have also created a namespace for syndicating media items[5]. This specification is intended as a replacement for the RSS enclosure element, offering a more expressive means of describing media items.

3 Podcast Specification in RDFS

In the following we describe the classes and properties that participate in our specification. We do not create a whole new vocabulary, but invent new terms where appropriate and use properties and classes from existing vocabularies such as RSS 1.0[6], DC[7] (Dublin Core) and FOAF[8] (Friend of a Friend). Such terms are explained here as they form an integral part of the format. Not all valid terms are listed here, for instance many properties of the FOAF specification which would be legitimate within the `foaf:Person` class are omitted for brevity. The new terms use the new local prefix `pod` and the complete list as it currently exists is defined in the proceeding subsections. In addition a specification[9] with accompanying RDF Schema[10] currently reside on the Web.

[3] http://phobos.apple.com/static/podcast_specifications.pdf
[4] http://www.itunes.com/DTDs/Podcast-1.0.dtd
[5] http://search.yahoo.com/mrss
[6] http://purl.org/rss/1.0/
[7] http://purl.org/dc/elements/1.1/
[8] http://xmlns.com/foaf/0.1/
[9] http://sw.deri.org/2005/07/podcast/
[10] http://sw.deri.org/2005/07/podcast/spec.rdfs

3.1 Classes

In keeping with the typical RSS 1.0 specification, we preserve the use of the `rss:channel` elements and introduce a subclass of the `rss:item` element.

`foaf:Person` denotes a person and is in the domain of `pod:credit` in this application.

`rss:channel` is an overview of the feed itself, providing a virtual table of contents for the show items and also the generic author, date, topic etc. of the proceeding items.

`rdf:Seq` is a class which acts as a container for `pod:Podcast` item listings within the `rss:channel`.

`pod:Category` is an allowance for input data demands and is a class used in conjunction with DC properties to denote a category description and the taxonomy it refers to.

`pod:File` represents a physical multimedia file and has many optional properties described later.

`pod:Podcast` is a container for a single Podcast show, which is described using properties described later including title, creator etc. `pod:Podcast` is a subclass of `rss:item`. Where it contains multiple `pod:File` entries of the same content, the `rdf:about` attribute should refer to the default file.

3.2 Properties

`dc:creator` can be a property of an `rss:channel` or a `pod:podcast` which provides the name of the creator of said class or classes.

`dc:title` is a property of the `pod:category` element which offers the word or phrase to denote its topic.

`dc:subject` is a property of a channel or item which it links to a pod:category element.

`foaf:mbox` provides the email of the person to whom the parent `foaf:Person` instance refers.

`foaf:name` is the name of the person to whom the parent `foaf:Person` instance refers.

`rdf:about` provides the subject for all the properties of the class it describes and usually refers to the URL of the resource.

`rdf:li` is a property of the `rdf:Seq` denoting individual items.

`rss:description` is a brief summary of the content of the `rss:channel` or `pod:Podcast.`

rss:items is a property relating the rss:channel class to the rdf:Seq class of item listings.

rss:title describes the title of the parent rss:channel or pod:Podcast

pod:bitrate is an optional attribute of pod:File which provides the file's kilobits per second ratio.

pod:credit contains a foaf:Person class used to describe a person with involvement in the Podcast.

pod:domain is an optional attribute of the pod:Category class. It provides a URL to a taxonomy or classification to which the title of the category refers.

pod:duration provides the length in seconds of the file.

pod:explicit is a property of either rss:channel or pod:podcast. If the content of the class is unsuitable for minors, the property's value will be true.

pod:expression is an attribute of pod:File and states whether the file is a sample excerpt, a full edition or a continuous stream

pod:fileSize is an attribute of pod:File and indicates the size of the media file in bytes.

pod:framerate provides the frames per second ratio of a video file.

pod:hasMedia is a property of pod:Podcast referring to a single or multiple pod:File elements contained within a pod:Podcast instance. In the case of multiple instances, each pod:File within the property must have the same content. Such usage is intended for files which are either identical bar physical formats or where some of the files are abridged or excerpts of the default file.

pod:isDefault is useful where multiple pod:File instances occur within the one pod:hasMedia property and thus have identical or abridged content. This value is true when the parent pod:File instance is the default file format of the pod:Podcast item.

pod:length represents the length of the file in pixels if visual content is present.

pod:role is a property used to describe the role of the person, to whom the parent foaf:Person instance refers, in the Podcast's creation.

pod:type refers to the MIME type of the file.

pod:width represents the width of the file in pixels if visual content is present.

3.3 Example Usage

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:pod="http://sw.deri.org/2005/07/podcast#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns="http://purl.org/rss/1.0/">

<channel rdf:about="http://example.com/podcasts/index.html">
 <title>New Specification</title>
 <description>An example Podcast feed describing this new
specification.</description>
 <pod:credit>
  <foaf:Person>
   <foaf:name>Aidan Hogan</foaf:name>
   <foaf:mbox rdf:resource="mailto:aidan.hogan@deri.org"/>
   <pod:role>Owner</pod:role>
  </foaf:Person>
 </pod:credit>
 <dc:subject>
  <pod:category>
   <pod:domain rdf:resource="http://example.com/Taxonomy" />
   <dc:title>Semantic Web</dc:title>
  </pod:category>
 </dc:subject>
 <items>
  <rdf:Seq>
   <rdf:li rdf:resource="http://example.com/podcasts/newShow1.mp3" />
   <rdf:li rdf:resource="http://example.com/podcasts/newShow2.mp3" />
  </rdf:Seq>
 </items>
</channel>

<pod:Podcast rdf:about="http://example.com/podcasts/newShow1.mp3">
 <dc:creator>Andreas Harth</dc:creator>
 <title>Item #1</title>
 <pod:explicit>true</pod:explicit>
 <description>This week Andreas talks all about the new RDF Podcast
Specification</description>
 <pod:hasMedia>
  <pod:File rdf:about="http://example.com/podcasts/newShow1.mp3"
   pod:fileSize="16490812"
   pod:type="audio/mpeg"
   pod:expression="full"
   pod:isDefault="true"
  />
  <pod:File rdf:about="http://example.com/podcasts/newShow1.ra"
   pod:fileSize="482342"
   pod:type="audio/x-realaudio"
   pod:expression="sample"
   pod:isDefault="false"
  />
 </pod:hasMedia>
</pod:Podcast>

<pod:Podcast rdf:about="http://example.com/podcasts/newShow2.mp3">
 <dc:creator>John Breslin</dc:creator>
 <title>Item #2</title>
 <description>This week John gives his two cents on the new RSS 1.0
Podcast Specification</description>
 <pod:hasMedia>
  <pod:File rdf:about="http://example.com/podcasts/newShow2.mp3"
   pod:fileSize="8727310"
   pod:type="audio/x-mpeg"
   pod:bitrate="128"
   pod:duration="1058"
  />
 </pod:hasMedia>
</pod:Podcast>
</rdf:RDF>
```

Figure 1: An example usage of the new specification

Fig. 1 illustrates an example Podcast feed using the already described properties and classes. As can be seen, the RSS 1.0 format is still evident in the structure of the document, with channel acting as a source of information and table of contents for all the items following it. The items in this format are of type `pod:Podcast`, a subclass of the traditional `rss:item` element.

3.4 Data Conversion

To introduce Podcasts to the Semantic Web, the thousands of feeds residing on the Web would have to be converted to the RDF specification. Whilst RSS 2.0 can quite neatly convert over to RSS 1.0 (both having comparable elements) with the exception of a few features, the current non-RDF vocabularies associated with Podcasting are incompatible with a simple transformation.

To implement such a conversion, we commissioned a stylesheet to map the original element structures from the RSS 2.0 template model which is commonly interspersed with elements from Yahoo!, Apple and other specifications. Many difficulties arose in the conversion operation.
With several designs for publishing Podcast metadata, there existed much duplication between the various methods. For instance, both Yahoo! and Apple introduce a `<category>` element, `<author>` element, `<description>`, `<keywords>` etc. Not only are such elements and others present twice within these documents but many equivalent elements already existed in Dublin Core (`<author>` = `<dc:creator>`, `<description` = `<dc:description>`) and RSS 1.0 schemas (`<description>`, `<title>`) etc.

Also both Yahoo! and Apple use conflicting methods of describing multimedia files within their "RSS 2.0 format" models. Whilst Apple advocate the use of an existing RSS 2.0 element, namely the `<enclosure>` tag, Yahoo! have created a new class called `<media:content>` with different applicable properties.

Another challenge raised was by the continuous updating of the Yahoo! model which has been widely adopted for use. The original publication of the specification has been radically updated twice without versioning data appearing in the namespace rendering instance data of multiple versions unwieldy.

Whilst the above obstacles lead to difficulties in developing software agents to interpret current Podcast feeds it furthermore gives rise to human error. The complexity and duplication present in the model constructs traps which users new to RSS and feed creation are vulnerable to. Such users attempting to use the various specifications are prone to creating feeds with invalid XML or RSS. For instance one common error encountered in user created Podcasts was to syndicate multiple shows within the one RSS item element.

In order to make our specification interoperable and practically usable in our intended application (searching and browsing any Podcast feed), we designed an XSLT document to resolve the differences present in the various XML RSS 2.0 Podcast feeds into a uniform RDF representation. The stylesheet consists of almost 1000 lines of code, which shows that XML-based solutions have shortcomings in terms of integrating data from different sources. After applying the XSLT to the native XML Podcast feeds, we end up with RDF/XML which is subject to further processing using existing tools.

4 Multimedia Search Engine and Browser

In pursuance of the idea of being able to use existing SW tools on top of the converted data, we present a system of browsing and searching Podcast instance data. This system incorporates pre-existing components. We entitle it "Podcast Pinpointer".

4.1 Architecture

Fig. 2 illustrates the architecture of the system. The crawler component is responsible for locating and fetching Podcast feeds on the Web. The data integration component essentially applies the XSLT stylesheet to transform the various XML-based formats to RDF, which then can be stored in YARS, an RDF repository [4]. The user interacts with the system via a User Interface, where they can perform searches and browse the dataset. We elaborate on the structure and purpose of the various components in the subsequent subsections.

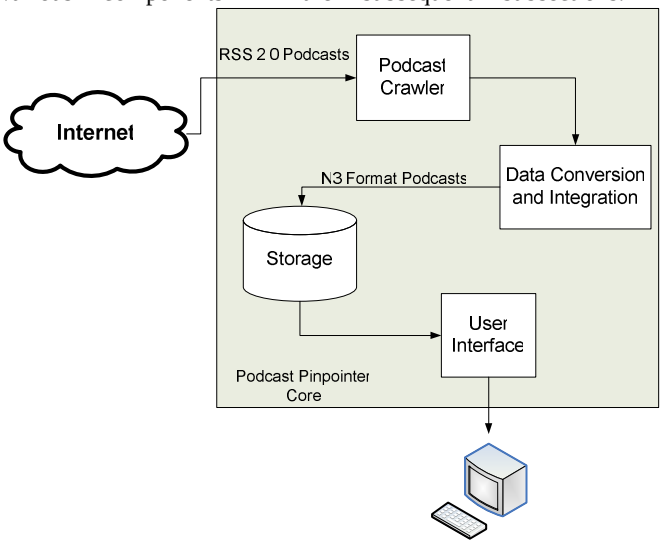

Figure 2: The architecture of the current Podcast Pinpointer application.

4.2 Locating & Crawling Multimedia on the Web

In order to create a central repository of multimedia data, we have to locate and crawl such data. However, Podcasts are not interlinked, and so achieving a complete set of data is demanding. The most complete method of data acquisition

would be a crawl of the entire Web, however the resources necessary to complete said task are quite substantial.
To attain an initial set of links to Podcasts feeds we visited various public Podcast Directories (e.g. iPodder) and achieved a combined list of about 6,000 Podcast pages, which was numerically greater than the amount claimed to be referenced by any of the main Podcast Directories.

Once we screen-scraped an initial set of URL's from Podcast directories, they were crawled and cached. Podcast feeds should not be neglected for more than a few hours however, as new shows are continually being added and local versions of the data would become obsolete. Currently, the crawler is run intermittently. As underlined in the future work section, plans exist to extend the crawler capabilities to allow for continuous crawling and user submission of feeds.

We completed a crawl of 6054 Podcast feeds towards the end of August 2005. The overall size of the crawled data was 120 MB. The dataset is available online[11]

4.3 Data Conversion and Integration

We applied the XSLT stylesheet previously outlined in Section 3.4 to the native Podcast feeds. The output of the transformation was RDF/XML in the Podcast vocabulary described in Section 3.

4.4 Storage

Podcast Pinpointer uses YARS to store and retrieve Podcast/RDF instance data. This provides a central repository for data which can then be queried. YARS offers optimized index structures. YARS uses Notation3[12] as a format for encoding facts and queries. Agents pose queries via an HTTP interface. YARS features advanced querying features, in line with the structured dataset it stores. It also features keyword search capabilities.

4.5 User Interface

Presently, a user interface exists to query and browse the integrated dataset. The user interface interacts with the Storage component via HTTP. A user can pose either single keyword queries or queries for matching literal. The storage component returns the results in RDF/NTRIPLES (Notation 3) format, which are then parsed, sorted, and serialized to RDF/XML. Finally, an XSLT generates the HTML page visualizing the result set, which can then be browsed by the user.

5 Future Work

5.1 Syndicating Other Multimedia

It would be beneficial to create RSS 1.0 descriptions for more generic multimedia files which do not possess a corresponding feed. In doing so, there would be a wealth of multimedia data available to various Semantic Web applications, enriching the services such applications provide to users and making such data more easily accessible and so benefiting both.

The expression of Podcasts in RSS 1.0 format is essentially a process of reorganising pre-structured data (see Fig. 3). In fact, it may also be possible to explicitly define metadata in an RSS 1.0 format for multimedia data where such metadata does not already exist.

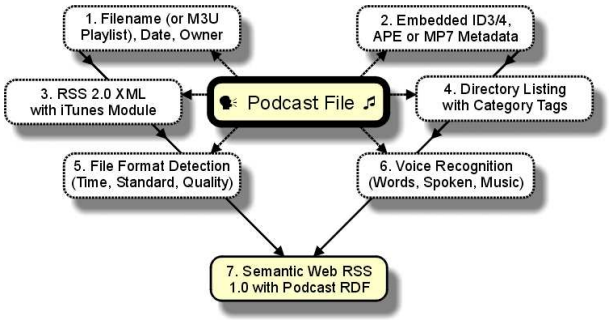

Figure 3: Some sources of metadata for a Semantic Web representation of a Podcast file.

One possibility would be to parse and convert metadata embedded in multimedia files. An example of such would be ID3/ID4/APE tags embedded in MP3 files. Such tags provide information relating to the file name, song or piece name, creator or artist, album, genre and year. Other multimedia metadata standards include the MPEG series of standards. Of particular interest would be MPEG-7, a means of expressing audio-visual metadata in XML. Upon parsing out such information, a pre-templated RSS 1.0 file could be filled with the available information. This would then be interpretable by the same tools as the modified Podcasts.

Another interesting application of the RDF Podcast specification is in relation to the Asterisk project[13], an open source Linux-based PBX application. A potential side use for Asterisk on online bulletin boards has already been touted as the next evolutionary step in web-based discussions by Drupal/CivicSpace, whereby phone conversations made through the Asterisk PBX would be recorded and stored as streamed or downloadable audio conversations for other readers of the bulletin board discussion.

Many sites have begun using voice recognition technology in the indexing of multimedia files, one such site being 'blinkx'[14]. Voice recognition software has seen many advances in recent years, and is becoming more and more accurate. Such sites use the ever-more advanced technology

[11] http://sw.deri.org/2005/07/podcast/podcast-crawl-2005-08.zip
[12] http://www.w3.org/2000/10/swap/Primer
[13] http://www.asterisk.org/
[14] http://www.blinkx.tv/downloads/blinkx_TV_White_Paper_v1.0.pdf

to create a transcript of spoken words in the audio of files. Indeed this would be quite useful in keyword searches.
On top of these transcripts then, HLT (Human Language Technology) could be implemented to derive a structure from the prose [5]. This structure could take the form of various elements within an RSS 1.0 document, accompanying all other metadata already located.

5.2 Extending the Crawler

The crawler could be extended in two ways: firstly, a more practical means of achieving new feeds would be to offer user submission options, whereby creators of new broadcasts can issue their feed address to the system and the crawler can pick it up. Also, it is soon planned to have the crawler run continually. Another planned feature for the crawling component includes a ping listener, where users can ping the crawler as notification that their feed has been updated.

Some sites such as Odeo[15] are making the Podcast crawling process easier by providing multi-format metadata about both Podcasts and site members. Odeo has made the Podcast categorisation process easier by allowing users to both create and subscribe to Podcasts that are tagged as belonging to a certain category. The site also provides links to RSS 2.0 metadata / M3U files corresponding to both a particular Podcast series and to the various series that a particular member has subscribed to. There exists the opportunity to use the advantages of such a system in the crawling of raw Podcast feeds.

6 Conclusions

This paper has described methods for representing Podcast information on the Semantic Web, beginning with the development of an ontology to represent Podcast metadata (and more generally, information on other multimedia audio and video files) using RDF. We have created a means of converting existing Podcast feeds to this specification by means of an XSLT. We have also created a Podcast Pinpointer application to aid in the intelligent search and retrieval of relevant Podcasts by combining Podcast metadata with other data that already exists in the Semantic Web framework. As such, this prototype demonstrates interesting possibilities for the use of audio and video metadata in future Semantic Web multimedia applications.

Acknowledgements

This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

References

[1] T. Berners Lee, “Semantic Web Road Map”, http://www.w3.org/DesignIssues/Semantic.html, September 1998.

[2] 5. L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, J. Sachs.” Swoogle: A Search and Metadata Engine for the Semantic Web”. In Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management, Washington, DC, Nov. 2004.

[3] A. Harth. “SECO: Mediation Services for Semantic Web Data”. IEEE Intelligent Systems, May/June 2004.

[4] A. Harth, S. Decker. “Optimized Index Structures for Querying RDF from the Web”, Proceedings of the 3rd Latin American Web Congress, Argentina, October 2005.

[5] W. Minker. “Semantic Analysis for Automatic Spoken Language Translation and Information Retrieval”, European-Japanese Conference on Information Modelling and Knowledge Bases (EJC), Iwate, Japan, May 1999.

[15] http://www.odeo.com/