From RDF to RSS and Atom: Content Syndication with Linked Data

Alex Stolz
Universität der Bundeswehr München
E-Business and Web Science Research Group
85577 Neubiberg, Germany
+49-89-6004-4277
alex.stolz@unibw.de

Martin Hepp
Universität der Bundeswehr München
E-Business and Web Science Research Group
85577 Neubiberg, Germany
+49-89-6004-4217
mhepp@computer.org

ABSTRACT

For typical Web developers, it is complicated to integrate content from the Semantic Web to an existing Web site. On the contrary, most software packages for blogs, content management, and shop applications support the simple syndication of content from external sources via data feed formats, namely RSS and Atom. In this paper, we describe a novel technique for consuming useful data from the Semantic Web in the form of RSS or Atom feeds. Our approach combines (1) the simplicity and broad tooling support of existing feed formats, (2) the precision of queries against structured data built upon common Web vocabularies like schema.org, GoodRelations, FOAF, SIOC, or VCard, and (3) the ease of integrating content from a large number of Web sites and other data sources of RDF in general. We also (4) provide a pattern for embedding RDFa into the feed content in a "viral" way so that the original URIs of entities are included in all Web pages that republish the original content and that those pages will link back to the original content. This helps prevent the proliferation of identifiers for entities and provides a simple means for tracking the document URI at which particular content reappears.

Categories and Subject Descriptors

I.2.4 **[Artificial Intelligence]:** Knowledge Representation Formalisms and Methods

General Terms

Experimentation, Human Factors, Standardization, Languages

Keywords

RDF, RDFa, Microdata, RSS, Atom, Content Syndication, Affiliate Marketing, GoodRelations, Pipes

1. INTRODUCTION

In the past four years, the amount of useful, non-toy RDF content on the Web has grown tremendously: In the field of e-commerce, major retail sites like bestbuy.com, overstock.com, oreilly.com, and countless smaller shops have added RDFa markup to their page templates, exposing at least 25 million offer entities on a daily basis, and services like productdb.org and linkedopencommerce.com contain tens of thousands of rich product datasheets. In parallel, the various activities in the Linked Open Data community [1] contribute a lot of data, in particular about places, artists, brands, geography, transportation, and related facts. While there are still quality issues in many sources, there is little doubt that much of the data could add some value when integrated into external Web sites. For example, think of a Beatles fan page that wants to display the most recent offers for Beatles-related products for less than 10 dollars, a hotel that wants to display store and opening hours information about the neighborhood on its Web page, or a shopping site enriched by external reviews or feature information related to a particular product.

Unfortunately, consuming content from the growing amount of Semantic Web data is burdensome (if not prohibitively difficult) for average Web developers and site owners, for several reasons: Even in the ideal case of having a single SPARQL endpoint [2] at hand that already collates the data, one has to (1) craft a suitable SPARQL query, (2) connect to the SPARQL endpoint, and (3) extract and render the results from a variety of formats (e.g. RDF/XML, Turtle, or N3). Even in the case of an endpoint that offers a developer-friendly JSON representation of the results [3-6], processing the output requires an advanced understanding of RDF and the vocabularies used in the original data. Crafting suitable SPARQL queries is in practice also not as easy as in textbook examples, since one has to be aware of (1) alternative modeling patterns, (2) competing vocabularies for the same aspects, and (3) typical data quality problems. For instance, even in the simple domain of contact information, a potential consumer faces at least five different vocabularies, because such data is being modeled in schema.org [24], FOAF [7] and at least three variants of vCard ontologies [8-10]. Relatively simple geo-position information can be modeled e.g. using either schema.org [24], the WGS84 vocabulary [11] or via vCard [8]. This means that a consumer of data must have very advanced knowledge of relevant vocabularies, their usage, and potential structural alternatives within the same vocabulary. On top of that, a data consumer has to expect and handle popular modeling mistakes and data quality issues, like omitted datatype suffixes for RDF literals, for good recall. In conclusion, it is currently difficult if not impossible for the millions of "normal" Web developers to tap the growing amount of Semantic Web data for their purposes, while convincing those developers is important for broad adoption. Their adoption is in turn critical for creating incentives to publish structured data on the Web.

On the contrary to those hassles with consuming RDF data, content syndication, i.e. the integration of remote content into a Web site, has gained widespread popularity; for an overview, see e.g. [12] or [13]. Content syndication in the narrower sense means that one site provides access to a frequently updated extract of site content in a standardized, structured syntax and that another site consumes and integrates the respective "feed" into an existing

24th ACM Conference on Hypertext and Social Media
1–3 May 2013, Paris, France.

page layout [12]. Sites can either provide one or multiple *predefined* feeds (e.g. the most recent entries from a particular blog) or allow configuring *customized* feeds (e.g. news entries containing a particular keyword).

Content syndication has incentives for both publishers and consumers: Publishers can increase the reach of their content, and that into very qualified audiences, since feeds are often consumed by sites very related to the original content. Subscribers can augment their pages by fresh, quality content that they could not produce themselves. Large shopping sites like Amazon and eBay also support feeds that are tied to a registered affiliate account, who will earn a commission on purchases initiated via this affiliate's site, thus providing a monetary incentive for including their content [14]. Also, specific and fresh content is honored by search engines and may thus lead to a better ranking in the organic search results.

The most prominent feed formats that can be found on the Web are RSS 2.0 ("Really Simple Syndication", [15]) and ASF or Atom ("Atom Syndication Format", [12, 16]). Both the multiple RSS versions (e.g. [17] and [15]) as well as Atom are actively used on the Web, but despite technical advantages of the Atom format, the RSS family is still the most popular feed with a market share of ca. 70-80% [18; 25]. An interesting aspect of content syndication via feed formats is the high degree of currency. Already back in 2005, i.e. several years before "real-time" services like Twitter appeared, a study [18] showed that 55% of RSS feeds were updated on an hourly basis.

In this paper, we describe the use of RSS and Atom feed syntaxes for accessing the growing amount of useful data from the Semantic Web. The main idea is to leverage the consumption of Linked Data by exploiting widely standardized data formats as a carrier. For this purpose, query results from single or federated SPARQL endpoints are transformed into RSS or Atom feeds, which can then be consumed by feed readers and feed aggregators as desktop applications, or respective modules for blog, shop, or CMS software. Our approach combines the simplicity and broad tooling support of existing data feed formats with the precision of queries against structured data built upon common Web vocabularies like schema.org [24], GoodRelations [19], FOAF [7], SIOC [20], or VCard [8], and the ease of integrating content from a large number of Web sites and other data sources of RDF in general. This is difficult with traditional feeds, because their content is difficult to consolidate (e.g. to list the cheapest ten offers from the union of an Amazon and an eBay feed), filter (e.g. show only shops in New York), or convert (e.g. show prices in Euro, no matter which currency the original feed uses). We also show how a subset of the original RDF data can be included in the feed as non-intrusive payload in the form of RDFa in escaped HTML markup ("Viral RDFa"), so that the original URIs of data objects remain available and will be contained in any Web page that republishes the data. By including a `foaf:page` property we can even link back from the URIs of the Web pages republishing the feed content to the original entities. These two approaches help prevent the proliferation of identifiers for entities and provide a simple means for tracking the document URI at which specific content reappears.

As a proof-of-concept, we developed a prototype[1] that (1) translates the definition of desirable e-commerce content, from a site-owner's perspective, into non-trivial SPARQL queries for GoodRelations-based data, (2) pre-fetches and caches respective content for fast delivery, and (3) embeds the original URIs for data entities in escaped RDFa inside the resulting feeds.

The structure of the paper is as follows: In Section 2, we describe the approach at the conceptual level. In Section 3, we evaluate our work and compare it with existing approaches, namely Yahoo Pipes [21] and DERI Pipes [22]. Section 4 discusses the work and summarizes our contribution. Section 5 concludes the paper.

2. RDF DATA AS RSS OR ATOM FEEDS

In this section, we describe our approach, in particular the viral use of RDFa in feed content. We also discuss techniques for making the consumption of Linked Data via syndicated feeds more useful for site owners, e.g. by adding geo-position information in a format supported by popular site software packages.

2.1 Approach

The basic idea is to combine the following components: (1) a query builder that generates SPARQL queries for the desired content based on user input and takes into account alternative RDF data patterns found in the wild and (2) a service that executes the queries against a public SPARQL endpoint or federation of SPARQL endpoints, enriches and caches the results, and makes each custom query result available as both an Atom and a RSS feed at a persistent URI. The URI of the feed can than be simply pasted into a feed-based content syndication component of an existing Web site. Since we use SPARQL endpoints as input and popular feed formats as the output syntax, our approach can also be chained up with RDF-oriented middleware (e.g. DERI Pipes [22]) and tools for aggregating or transforming content in traditional feed formats (e.g. Yahoo Pipes [21]). That means that one could use DERI Pipes to combine and pre-process multiple RDF-based data sources, expose the result as a SPARQL endpoint, use our component to filter and transform the result into a RSS or Atom, and then use Yahoo Pipes to post-process the resulting feed.

2.2 Viral Use of RDFa

Republishing data from the Semantic Web via feed formats in Web content has a few important caveats: First, it strips off the valuable meta-data found in the original content. So a product price that can be understood by a browser extension at the origin of the information would be turned into plain text or HTML when reappearing on a syndicating site. Second, crawlers finding the content on the syndicating site have no direct means of discovering the original data source (other than following each and every HTML link). Third, there is no RDF triple that links the original locations with the URIs on which content is being republished. All this is undesirable, for it weakens the idea of the Semantic Web, because it takes RDF data and makes it less usable for machines.

Thus, we add a technique, which we call **Viral RDFa:** When composing the feed contents, we embed additional information as invisible RDFa (RDFa Snippet Style, see [26]) to suitable feed fields. In order to keep the respective markup intact during the processing of the feed payload, we use HTML entity encoding. The same technique is often used by feed publishers to include basic HTML formatting in text content (e.g. `<b>` for boldface or `<br />` for line-breaks) and is commonly supported by feed-consuming applications.

This allows (1) adding proper RDFa markup to the content when republished on a syndicating site, (2) enforcing the consistent use

[1] http://www.stalsoft.com/gr2rss/

of one canonical URI for the same entity across republishing sites, and (3) adding a machine-readable link from the document URIs of the pages with the syndicated content to the subject URIs.

Here is an example: Assume that the raw feed would encode a product as follows:

```
<item>
 <title>Camcorder with Widescreen LCD</title>
  <description>The best camcorder...</description>
  <link>http://www.example.com/camcorder</link>
  <pubDate>Tue, 04 Dec 2012 23:33:07
+0100</pubDate>
  <guid isPermaLink="false">...omitted...</guid>
</item>
```

Then, in RDFa, the global URI of the product and the product name and description would be encoded as:

```
<span typeof="gr:Offering"
 about="http://www.example.com/camcorder#product">
 <span property="gr:name"
  content="Camcorder with Widescreen LCD"></span>
 <span property="gr:description">The best
  camcorder...</span>
 <span rel="foaf:page" resource=""></span>
</span>
```

Note that the line

```
<span rel="foaf:page" resource=""></span>
```

will create a `foaf:page` triple from the subject entity to the URI of the page in which the RDFa parser will find the markup. This allows adding the respective link without knowing the URI of the final target page in advance (which is not available in anonymous publish-subscribe scenarios anyway).

As a next step, we use HTML entity encoding for this RDFa snippet and insert it into the `<description>` field of the feed:

```
<item>
 <title>Camcorder with Widescreen LCD</title>
  <description>&lt;span
typeof=&quot;gr:Offering&quot;
about=&quot;http://www.example.com/camcorder#produ
ct&quot;&gt;
     &lt;span property=&quot;gr:name&quot;
content=&quot;Digital Camcorder with Widescreen
LCD&quot;&gt;&lt;/span&gt;
     &lt;span
property=&quot;gr:description&quot;&gt;The best
camcorder...&lt;/span&gt;
     &lt;span rel=&quot;foaf:page&quot;
resource=&quot;&quot;&gt;&lt;/span&gt;
&lt;/span&gt;</description>
  <link>http://www.example.com/camcorder</link>
  <pubDate>Tue, 04 Dec 2012 23:33:07
+0100</pubDate>
  <guid isPermaLink="false">...omitted...</guid>
</item>
```

A typical feed consuming component will decode the HTML entity sequences like (e.g. `<`) into the original characters (<) before generating the HTML of the page including the syndicated content. Then, the RDFa snippet shown above with the canonical entity URI (via the `about` keyword) and the link to the current page (via the `foaf:page` statement) will be included in the final HTML without the need for any intervention by the operator of the syndicating site. Suitable fields in feed formats to carry entity-encoded HTML are the `<content>` and `<summary>` fields in Atom and the `<description>` field in RSS. In a practical scenario, one will likely add a little bit more granular RDFa, e.g. including strong product identifiers for products (UPC, ISBN, GTIN13, …), structured price information, and image meta-data. However, it is not necessary to pack the entire data into the RDFa snippet, since a client can fetch additional data easily by dereferencing the entity URI, because that will always point to the original resource.

The `foaf:page` links will contribute to a growing graph of RDF data, because they create triples from each entity URI to each and every page mentioning it, assumed that the crawler used will discover and process those pages. So we add a mechanism that prevents the proliferation of entity identifiers and at the same time provide a simple means for tracking the document URI at which a particular piece of content reappears. Of course, the mechanism does not help finding URIs of syndicating sites. Instead, it preserves the link between related data. Examples of this approach are available from http://www.stalsoft.com/gr2rss/examples.

Our RDFa-based approach can also be used to encode additional meta-data into feeds for which no standard structures exist in the respective feed formats. In fact, this allows using all schema.org types and properties for publishing additional meta-data about feeds and the entities contained therein. The same pattern can also be used with microdata syntax [30].

2.3 Geo-Location Information

Information about the geo-position of places, events, or objects is often available in public RDF data. Also, for many subscription scenarios, geographic proximity will be a relevant filter. For instance, a hotel may want to list restaurants or events in its vicinity on its Web site. The geo-location is also a popular starting point for establishing relations between data from different sources, since physical proximity is often a good indicator of relatedness. With traditional content syndication via feed formats, geo-position information is often used to automatically show the position of an entity on a Web-based map service like Google Maps. A popular way to convey geo-location data via RSS and Atom feeds is the so-called "geotagging" with GeoRSS[2]. Respective tags can be attached to items inside RSS or Atom feeds, while interoperability between feed consumers is typically ensured. Embedding geo information for each entry into a content feed allows applications like Google Maps to parse the relevant location data and display them in one single map. The importance of embedding geo information into RSS or Atom is also indicated by the wealth of modules for CMS that provide GeoRSS support, namely GeoPress[3] for WordPress and the GeoRSS module[4] for Drupal. For geotagging locations in RSS or Atom feeds, there exist three relatively similar standards, namely GeoRSS Simple[5], GeoRSS GML[6], a more sophisticated geo-data markup language than GeoRSS Simple, and GeoRSS[7] by the W3C (now deprecated, yet still popular).

In its simplest form, GeoRSS tagging means embedding geometry *points* with longitude and latitude coordinates, which is sufficient for our proposal. A point describing the city center of Munich could e.g. be written in GeoRSS Simple syntax as

```
<georss:point>48.13695 11.57540</georss:point>
```

where the first floating point parameter determines the latitude and the second describes the longitude.

[2] http://www.georss.org/
[3] http://georss.org/geopress
[4] http://drupal.org/project/georss
[5] http://www.georss.org/simple
[6] http://www.georss.org/gml
[7] http://esw.w3.org/GeoInfo

Table 1. Qualitative comparison of our approach with similar services

	Multiple Sources	Consuming SPARQL Endpoints	Exposing RSS or Atom	Query Builder	Usage of structured data across multiple sources	Maintaining URIs for Entities
Our Approach[8]	Yes	Yes	Yes	Yes	Yes	Yes
Yahoo Pipes[9]	Yes	No (maybe possible via with Fetch Data module)	Yes	No	Limited, only via feed elements or regular expressions	No
DERI Pipes[10]	Yes	Yes	No	No	Yes	Yes, but limited to RDF/XML output
eBay RSS Feed Generator[11]	No	No	Yes	Yes	No	No, only links to product offer pages
Amazon RSS feeds[12]	No	No	Yes	No	No	No
Manual integration of multiple feeds in PHP etc.	Yes, but integration effort	Yes, but difficult and time-consuming	Yes	No	Only via regular expressions	Depends on integrated sources

The same information as with GeoRSS Simple above can be expressed in GeoRSS by the W3C as follows:

```
<geo:Point>
 <geo:lat>48.13695</geo:lat>
 <geo:long>11.57540</geo:long>
</geo:Point>
```

There are multiple ways of using geo-data in our proposed technique. The most straightforward one is translating relevant geo-position information from the RDF world into GeoRSS Simple and GeoRSS by the W3C syntax so that consuming clients can easily show the position of entities in dynamically generated maps etc. In addition, one can use the "Viral RDFa" technique to preserve geo-information for RDFa-aware clients in ways not supported by the feed syntax standards. In our prototype, store information can be ordered by their proximity to a given location. Then, the prototype outputs a map containing markers for every store that could be found. Furthermore, links to Google Maps are provided, where the GeoRSS information will be interpreted by the map rendering engine and displayed accordingly.

3. Evaluation

In this section, we describe the method and results of our evaluation. In particular, we give a qualitative comparison of our approach to the most relevant existing services, like Yahoo Pipes, DERI Pipes, and Amazon and eBay feed services.

3.1 Qualitative Comparison

Table 1 shows the comparison of our prototype with available alternatives. We used the following dimensions:

Multiple Sources: Is it possible to integrate data from multiple sources (e.g. multiple sites)?

Consuming SPARQL Endpoints: Is it possible to integrate RDF data from SPARQL endpoints?

Exposing RSS or Atom: Can the result be made available in standard feed syntaxes?

Query Builder: Does the technology provide a user interface for specifying the desired content?

Accessing structured data across multiple sources: Can the query address individual data elements from multiple sources (e.g. can it sort the union of offers from multiple vendors by price)?

Maintaining URIs for Entities: Are the canonical identifiers (URIs, product identifiers, etc.) of entities preserved so that clients can link the information with external data easily?

3.2 Yahoo Pipes and DERI Pipes

In addition to the evaluation described so far, we tried to integrate two different endpoints in DERI Pipes [22]. Since its codebase has not been updated for some time, it comes as no surprise that it does not support JSON syntaxes for RDF. Thus, we could not easily build a use-case on DERI Pipes and abandoned the idea.

Next, we tried the same integration task in Yahoo Pipes, but failed again with integrating the results of the two endpoints: We were not able to perform string matching and comparison between labels (which are in fact not recognized as strings) from dbpedia.org and linkeddata.uriburner.com. We did not manage to consolidate entities on the basis of their URI. We finally managed to integrate two feeds, namely an eBay RSS feed and an RSS feed

[8] http://www.stalsoft.com/gr2rss/
[9] http://pipes.yahoo.com/pipes/
[10] http://pipes.deri.org/
[11] http://pages.ebay.com/affiliates/tools/rssgenerator/index.html; this service has been recently restricted to registered affiliates.
[12] http://www.amazon.com/gp/tagging/rss-help.html

of from our tool in Yahoo Pipes. We were able to compare prices on the level of price values (disregarding different exchange rates for currencies), so we could sort items based on the price value. Figure 3 shows our experiment.

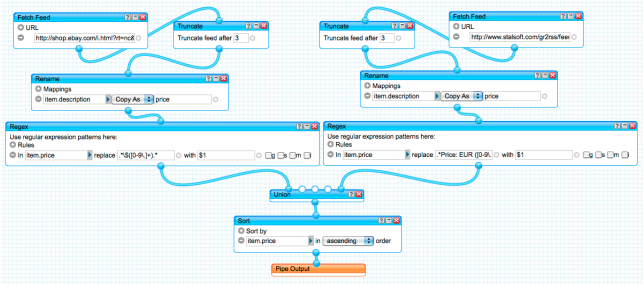

Figure 1. Integration experiment in Yahoo Pipes

4. Discussion

In this section, we discuss our contribution and compare it with existing alternatives.

4.1 Related Work

Besides the substantial body of work covering standard, XML-based feed syndication (e.g. [12-18]), the most relevant related work to our proposal are Yahoo Pipes [21] and DERI Pipes [22].

Yahoo Pipes is a Web-based tool for integrating Web content from multiple sources, including feeds, CSV files, Web services, and scraping content from HTML [21]. The source data can be filtered, augmented, and transformed using various pre-defined components, including string operations, regular expressions, database-style aggregates, arithmetic operations, and more [21]. The results can be exposed in various syntaxes, e.g. RSS, JSON, and KML for inclusion in a syndicating Web site [21]. Yahoo Pipes is in general a powerful environment for data-based mash-ups, in particular thanks to the mature user interface, rich library of operators, and good documentation. However, consuming Linked Data sources is not sufficiently supported. There is no direct module for including SPARQL endpoints or to request RDF syntaxes via HTTP content negotiation. It may be possible to employ the Data Fetch module to include SPARQL endpoint data, but that is even more burdensome than directly consuming such data in a programming language. Also, the results will not contain the original URIs of entities from data sources and thus break the idea of Linked Data when the content is being republished. It is also worth noting that, surprisingly, a team member from our group with a moderate PHP background found using Yahoo Pipes less productive that directly coding PHP scripts and mentioned a substantial learning effort, despite the mature graphical user interface. In our opinion, however, the main limitations of Yahoo Pipes are that (1) it is currently not suited for integrating content from the Semantic Web into Web sites and that (2) it breaks the idea of Linked Data, because it strips off meta-data and Web-scale identifiers.

DERI Pipes [22, 31] is very similar to Yahoo Pipes in its basic approach, but was designed for consuming RDF data**.** It directly supports the consumption of SPARQL endpoints [31] and offers several operators specifically targeted at processing RDF data. In comparison to our approach, it cannot produce RSS or Atom feeds for easy consumption by existing Web application software (except via complex, custom XSLT scripts). Also, the consumption of existing RSS or Atom feeds is limited to a raw XML import interface, which would require manual effort to extract feed content. Finally, the use of the SPARQL import interface requires considerable expertise for typical queries, e.g. in e-commerce scenarios. It is also worth noting that the codebase of DERI Pipes has not been updated since March 2009.

Neither Yahoo Pipes nor DERI Pipes support the preservation of entity URIs and structured data via embedded RDFa in feeds. While DERI Pipes provides substantial support for applying the idea of piping to the RDF world, it is not properly linked with popular RSS and Atom feed environments. Yahoo Pipes, on the other hand, breaks the link between data from the source to the syndicating site and thus further fragments the Web from a data perspective.

For a few modern CMS systems (e.g. Drupal), there exists functionality for directly including RDF content via SPARQL queries, like the Drupal SPARQL Views module [32]. However, this is still a very advanced approach for typical Web developers, does not free them from a deep understanding of the SPARQL query language, vocabularies, and the popularity of data patterns. It is also unclear whether the *original* entity URIs can be preserved and re-exposed as RDFa or microdata within Drupal.

4.2 Our Contribution

In this paper, we proposed to use the popular RSS and Atom syntaxes for consuming the growing amount of Linked Data in real-world content syndication scenarios, while preserving the original Web-scale entity identifiers and formally representing the relations between original entities and the URIs of Web resources that republish the content by a viral pattern of RDFa or microdata syntax that survives typical transformations in Web applications. We have given very preliminary evidence that this direction simplifies the consumption of RDF data for Web developers. In comparison to the two existing approaches, our proposal serves the idea of Linked Data better than Yahoo Pipes and integrates better into RSS and Atom feed consumption infrastructure than DERI Pipes. As a proof-of-concept, we developed a mature prototype[13] that translates the definition of desirable e-commerce content from a site-owner's perspective into non-trivial SPARQL queries for GoodRelations-based data, (2) pre-fetches and caches respective content for fast delivery, and (3) embeds the original URIs for data entities in RDFa inside the resulting feeds.

4.3 Future Extensions

Our proposal could of course benefit from future extensions. In the particular, we plan to investigate adding learning components so that the component will automatically adapt to emerging data patterns. For specific e-commerce scenarios, we also want to add support for opening hours information from GoodRelations and add an option for exposing microdata using our "viral" pattern. Also, a deeper investigation of a consolidated ranking of results gained from multiple sources is desirable.

On the economic side, support for monetary compensations for traffic or transactions caused by content syndication by an affiliate will be a promising direction.

We will also further investigate the generalization of our idea to transport more structured data as "viral" RDFa or microdata embedded as escaped content in existing syntaxes, for this will allow using e.g. the broad coverage and granularity of the schema.org vocabulary for augmenting various rigid XML-based exchange syntaxes in a backwards-compatible way.

[13] http://www.stalsoft.com/gr2rss/

5. Conclusion

We have shown how popular RSS and Atom feed syntaxes and infrastructure can be used to reduce the barriers for syndicating Linked Data content in typical Web applications in a way that prevents a proliferation of entity identifiers and that support the growth of a giant graph of data. Taking away the barriers for integrating Linked Data will increase the number of sites that consume and republish the underlying information, thus creating stronger incentives for owners of content to offer SPARQL endpoints and Linked Data for their assets.

6. ACKNOWLEDGMENTS

The authors would like to thank Uwe Stoll for testing our implementation and for providing valuable feedback. The work described in this paper was partly sponsored by the German Federal Ministry of Education and Research (BMBF) under the project “IntelligentMatch”, grant identifier 01IS10022B.

7. REFERENCES

[1] Linked Data - Connect Distributed Data across the Web, available at http://linkeddata.org/, retrieved 2012-12-08.

[2] SPARQL Protocol for RDF. W3C Recommendation 15 January 2008, available at http://www.w3.org/TR/rdf-sparql-protocol/, retrieved 2012-12-08.

[3] Talis: RDF JSON, available at http://docs.api.talis.com/platform-api/output-types/rdf-json, retrieved 2012-12-08.

[4] Clark, K.; Feigenbaum, L.; Torres, E.: SPARQL 1.1 Query Results JSON Format, W3C Proposed Recommendation 08 November 2012, available at http://www.w3.org/TR/sparql11-results-json/, retrieved 2013-03-02.

[5] Alexander, K.: RDF/JSON: A Specification for serialising RDF in JSON In: Bizer, C., Auer, S., Grimnes, G.A., Heath, T. (eds.): Proceedings of the 4th Workshop on Scripting for the Semantic Web (SFSW '08), Vol. 368. CEUR-WS, Tenerife, Spain (2008) 1-6.

[6] JSON-LD - Expressing Linked Data in JSON, available at http://json-ld.org/, retrieved 2012-12-08.

[7] FOAF Vocabulary Specification 0.98, available at http://xmlns.com/foaf/spec/, retrieved 2012-12-08.

[8] Halpin, H.; Iannella, R.; Suda, B.; Walsh, N.: Representing vCard Objects in RDF. W3C Member Submission 20 January 2010, available at http://www.w3.org/Submission/vcard-rdf/, retrieved 2012-12-08.

[9] Ianella, R.: Representing vCard Objects in RDF/XML. W3C Note 22 February 2001, available at http://www.w3.org/TR/2001/NOTE-vcard-rdf-20010222/, retrieved 2012-12-08.

[10] Yahoo Developer Network: VCard, available at http://developer.yahoo.com/searchmonkey/smguide/vcard.html, offline as of 2012-12-08.

[11] Brickley, D. (ed.): Basic Geo (WGS84 lat/long) Vocabulary, available at http://www.w3.org/2003/01/geo/, retrieved 2012-12-08.

[12] Sayre, R.: Atom: The Standard in Syndication. IEEE Internet Computing 9 (2005) 71-78.

[13] Hess, T.: Content Syndication. Wirtschaftsinformatik 43 (2001) 83-85.

[14] RSS web feeds for tags at Amazon.com, available at http://www.amazon.com/gp/tagging/rss-help.html, retrieved 2012-12-08.

[15] RSS 2.0 Specification, available at http://www.rssboard.org/rss-specification, retrieved 2012-12-08.

[16] The Atom Syndication Format, available at http://tools.ietf.org/html/rfc4287, retrieved 2012-12-08.

[17] RSS 0.90 Specification, available at http://www.rssboard.org/rss-0-9-0, retrieved 2012-12-08.

[18] Liu, H.; Ramasubramanian, V.; Sirer, E.: Client behavior and feed characteristics of RSS, a publish-subscribe system for web micronews. Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, Berkeley, CA, 2005.

[19] Hepp, M.: GoodRelations: An Ontology for Describing Products and Services Offers on the Web. 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Vol. 5268, Springer LNCS (2008), pp. 332-347.

[20] SIOC Core Ontology Specification. 25 March 2010, available at http://sioc-project.org/ontology, retrieved 2012-12-08.

[21] http://pipes.yahoo.com/pipes/, retrieved 2012-12-08.

[22] Phuoc, D.L., Polleres, A., Morbidoni, C., Hauswirth, M., Tummarello., G.: Rapid Semantic Web Mashup Development Through Semantic Web Pipes. Proceedings of the 18th World Wide Web Conference (WWW2009). ACM Press, Madrid, Spain (2009), pp. 581-590.

[23] Vocabulary of Interlinked Datasets (voiD), available at http://vocab.deri.ie/void, retrieved 2012-12-08.

[24] http://schema.org, retrieved 2012-12-08.

[25] http://www.syndic8.com/stats.php?Section=feeds#RSSVersion, retrieved 2012-12-08.

[26] Hepp, M.; García, R.; Radinger, A.: RDF2RDFa: Turning RDF into Snippets for Copy-and-Paste, Technical Report TR-2009-01, 2009. PDF available at http://www.heppnetz.de/files/RDF2RDFa-TR.pdf.

[27] Shvaiko, P.; Euzenat, J.: Ontology matching: State of the Art and Future Challenges. IEEE Transactions on Knowledge and Data Engineering, Vol. 25, Nr. 1, 2013, pp. 158-176.

[28] http://wiki.goodrelations-vocabulary.org/Axioms, retrieved 2012-12-08.

[29] Berners-Lee, T.: Cool URIs don’t change. http://www.w3.org/Provider/Style/URI.html, retrieved 2012-12-08.

[30] http://www.whatwg.org/specs/web-apps/current-work/multipage/microdata.html, retrieved 2012-12-18.

[31] http://pipes.deri.org/, retrieved 2012-12-18.

[32] http://drupal.org/project/sparql_views, retrieved 2012-12-18.