Interoperable Metadata Semantics with Meta-Metadata: A Use Case Integrating Search Engines

Yin Qu, Andruid Kerne, Andrew M. Webb and Aaron Herstein
Interface Ecology Lab, Texas A&M University
College Station, Texas
{yin, andruid, andrew, aaron}@ecologylab.net

ABSTRACT

A use case involving integrating results from search engines illustrates how the meta-metadata language facilitates interoperable metadata semantics. Formal semantics can be hard to obtain directly. For example, search engines may only present results through web pages; even if they do provide web services, they don't provide them according to a mutually interoperable standard.

We show how to use the open source meta-metadata language to define a common base class for search results, and how to extend the base class to create polymorphic variants that include engine-specific fields. We develop wrappers to extract data from HTML search results from engines including Google, Bing, Delicious, and Slashdot. We write a short meta-search program for integrating the search results, reranking them, and providing formatted HTML output. This provides an extensible formal and functional semantics for search. Meta-metadata also directly enables representing the same integrated search results as XML or JSON. This research can profoundly transform the derivation and representation of interoperable metadata semantics from a multitude of heterogeneous wild web sources.

Categories and Subject Descriptors

H.1.m [**Information Systems: Models and Principles**]

General Terms

Design, Documentation, Human Factors, Management

1. INTRODUCTION

We use the meta-metadata language and architecture to integrate results from search engines. *Metadata* can be employed to describe the numerous documents published online by businesses, organizations, and individuals. A *metadata schema* describes the vocabulary and structure used to represent and communicate a type of metadata. *Meta-metadata* is a language for describing schemas, wrappers that map information sources to schemas, and more. We define *metadata semantics* as the conjunction between a document and its metadata, including content, descriptions, citations, and afforded operations. *Interoperable* metadata semantics are obtained from multiple sources, translated into consistent representations, and then processed independently of the initial sources. Interoperability can be impaired by the fact that publishers often use their own metadata schemas and representations. For example, a user may find it difficult to search across multiple engines since they use different structures and interfaces to present results.

We build on *meta-metadata* [10] as a structural basis for obtaining and using interoperable metadata semantics in applications across system boundaries. Meta-metadata is an open source, cross-platform formal language, architecture, and wrapper repository addressing metadata structure, extraction, translation, bridging, and presentation [8]. The heart of meta-metadata is a type system for *information sources*, each published with a consistent DOM structure and addressing scheme. With meta-metadata, developers author wrappers to represent metadata vocabularies and structures, information source selectors to automatically select wrappers, rules to extract metadata into program objects, and semantic actions to connect metadata with business logic. Meta-metadata does not impose particular schemas on metadata-consuming applications, allowing for flexibility. It comes with a repository of inheritable wrappers, encouraging re-use across contexts.

This paper develops a use case that derives interoperable metadata semantics to integrate results from multiple search engines. We show how meta-metadata helps developers invoke and re-use interoperable metadata semantics.

2. PRIOR WORK

Standard metadata schemas, such as Dublin Core [14], were developed to capture common needs for interoperable metadata semantics across systems. Metadata semantics that use the same standard schema are directly interoperable, since no structural or vocabular ambiguities prevent unified processing. Sometimes data represented with one schema can be translated to a more standard one by associating synonymous fields across schemas, achieving interoperability. For example, the Library of Congress developed a crosswalk between Dublin Core and MARC [11].

However, as the ecosystem of needs and tasks grows in complexity, the pure standards approach to metadata is inadequate. Diverse metadata is a fact of life. It is impos-

DocEng'11, September 19–22, 2011, Mountain View, California, USA.

sible for standards to keep up or attain sufficient flexibility. The cost of creating translations is high. In contrast, meta-metadata does not rely on external standard metadata schemas. Instead, it allows developers to extend existing schemas, or start anew, according to their needs.

The Semantic Web [4] approaches interoperable metadata semantics with *ontology alignment* [1] and practice guidelines like Linked Data [3]. An ontology is a set of related metadata schemas (i.e. classes) and instances (i.e. resources) described by *formal semantics*, typically represented as a triple store (e.g. RDF or N3). A triple consists of a subject, predicate, and object. Any relationship between any pair of objects can be described. Ontology alignments map classes, fields and resources from one ontology to equivalents in another, resolving ambiguities. Linked Data identifies resources by URIs. Related resources in the same or other systems are bi-directionally linked using formal semantics.

Semantic Web triples constitute a general solution for resolving ambiguities. However, the situated and evolving nature of knowledge makes formal semantics costly to create and maintain [12]. In practice, they can involve extensive development of source specific "mediators" and optimizations [5]. As a result, The Semantic Web's availability and scalability are limited. Again, meta-metadata does not rely on rigid formal semantics from publishers. It allows developers to author schemas based on their own situated contexts, and use wrappers to automatically extract metadata semantics from informal representations into schematized structures.

Like meta-metadata, Piggy Bank [6] addresses scraping metadata into structured forms. It has been used to support Exhibit [7], a structured data presentation framework. The resulting system facilitates document publishing. One important difference between Piggy Bank / Exhibit and meta-metadata is that meta-metadata provides an object-oriented type system for wrappers. This helps reduce the effort of authoring domain models and extraction rules. Wrappers authored by independent developers can be re-used with inheritance and polymorphism, propagating interoperability across contexts.

3. INTEGRATING SEARCH ENGINES

Developers and power users author ***meta-metadata wrappers*** to specify metadata structures, extraction rules, operations, and presentation rules. The following wrapper (abbreviated for space) specifies a metadata schema `search` with a collection of `search_result`s inside, as well as semantic actions (which we will talk about soon):

```
<meta_metadata name="search" extends="compound_document">
  <collection name="search_results" child_type="search_result"/>
  <semantic_actions>
    <get_field name="search_results" />
    <for_each collection="search_results" as="search_result">
      <get_field object="search_result" name="link" />
      <parse_document now="true">
        <arg name="location" value="link" />
      </parse_document>
    </for_each>
  </semantic_actions>
</meta_metadata>
```

while `search_result` is defined in another wrapper, consisting of properties (or fields) commonly seen in search results:

```
<meta_metadata name="search_result" extends="metadata">
  <scalar name="heading" scalar_type="String"
          navigates_to="link" layer="10.0" />
  <scalar name="snippet" scalar_type="String" />
  <scalar name="link" scalar_type="ParsedURL" />
  ...
</meta_metadata>
```

In the above samples, attribute `extends` indicates inheritance, as the reserved word "extends" does in Java. Wrappers `compound_document` and `metadata` are primitive structures defined by the system. Wrapper `search` will inherit fields and attributes from `compound_document`. Wrapper innheritance represents an *is-a* relationship, which means support for polymorphism. For example, in a semantic action where `search` is expected, using a `google_search` is permissible. By inheritance and polymorphism, different metadata schemas can be translated into a consistent representation containing common fields, e.g. `google_search` and `slashdot_search` (which we will define later) being used as `search`, reconciling schematic differences. Meta-metadata maintains implicit mappings between fields initially defined in the base wrapper and inherited in derived wrappers for polymorphism and translation of schemas.

Attributes `navigates_to` and `layer` define *presentation rules*. The former makes a field navigable, using another field as the underlying target (`link` in the above sample); the latter indicates visual priority, to sort fields for presentation. Other supported presentation rules include hiding a field, or emphasizing it with a style.

The example specifies a group of *semantic actions* in wrapper `search`. These are inherited by derived wrappers that do not override and define their own. Semantic actions can include variable definitions, control structures, and bridge functions that connect metadata with applications. The semantic actions here take the collection of `search_result`s and iterate over each, forming and parsing `document` objects with `parse_document`, with the potential to derive further semantics. Semantic actions form a high-level abstraction of afforded operations, re-usable across applications and platforms. Developers extend or re-define bridge functions by simply overriding corresponding methods.

To attach *extraction rules*, we derive a new wrapper from `search` for Google Search:

```
<meta_metadata name="google_search"
               type="search" parser="xpath">
  <selector url_stripped="http://www.google.com/search" />
  <collection name="search_results"
              xpath="//div[@id='res']//div//ol//li[@*]" />
    <scalar name="heading" xpath=".//h3/a" />
    <scalar name="snippet" xpath=".//div[@class='s']" />
    <scalar name="link" xpath=".//h3/a/@href" />
  </collection>
</meta_metadata>
```

We use `type` to re-use an existing wrapper without defining new fields. We refer to fields defined in the base wrapper by name, e.g. `search_results` and `heading`, to attach extraction rules. Currently supported extraction rules include XPath and regular expressions, and direct binding of XML or JSON to objects through the S.IM.PL (Support for Information Mapping in Programming Languages) de/serialization engine [13]. Relative XPaths are supported inside nested structures, as shown for `heading`, `snippet` and `link`. Selectors specify the URL pattern or MIME type for a particular source, which the runtime uses to find an appropriate wrapper given a location and mime type.

We define similar wrappers for other publishers' search engines, including Yahoo Buzz, Bing, Slashdot, Delicious, and Tumblr (some wrappers and XPaths are omitted for space):

```
<meta_metadata name="yahoo_buzz_search"
               type="search" parser="xpath">
  <selector url_path_tree="http://buzz.yahoo.com/search"/>
  <collection name="search_results" xpath="//div...//dl">
    <scalar name="heading" xpath="./dt/a" />
    <scalar name="snippet" xpath="./dd[...]/a" />
    <scalar name="link" xpath="./dt/a/@href" />
  </collection>
</meta_metadata>
```

```
<meta_metadata name="bing_search_xpath"
               type="search" parser="xpath">
  <selector url_stripped="http://www.bing.com/search" />
  <collection name="search_results" xpath="//div.../ul/li[*]">
    <scalar name="heading" xpath=".//h3/a" />
    <scalar name="snippet" xpath=".//div[@class='sa_cc']/p" />
    <scalar name="link" xpath=".//h3/a/@href" />
  </collection>
</meta_metadata>
```

```
<meta_metadata name="slashdot_search"
               type="search" parser="xpath">
  <selector url_stripped="http://slashdot.org/index2.pl" />
  <collection name="search_results" xpath="//div...">
    <scalar name="heading" xpath=".//a" />
    <scalar name="link" xpath=".//a/@href" />
    <scalar name="snippet" xpath="..." />
    <scalar name="author" xpath="..." />
    <collection name="tags" xpath="...">
      <scalar name="tag_name" xpath="." />
      <scalar name="link" xpath="./@href" />
    </collection>
  </collection>
</meta_metadata>
```

The *meta-metadata compiler* automatically translates wrapper specifications into ***metadata classes***, in the form of Abstract Data Types in target programming languages, including Java, C#, and Objective C. A metadata class serves as a mapping between an information source and its internal representation in program. The resulting metadata classes in Java for wrapper `search` and `search_result` are:

```
@simpl_inherit public class Search extends CompoundDocument
{
  @simpl_collection("search_result")
  private ArrayList<SearchResult>  searchResults;
  ...
}

@simpl_inherit public class SearchResult extends Metadata
{
  @simpl_scalar private MetadataString    heading;
  @simpl_scalar private MetadataString    snippet;
  @simpl_scalar private MetadataParsedURL link;
  ...
}
```

```
<search mm_name="google_search"
 location="http://www.google.com/search?q=japan+earthquake">
 <search_result heading="BBC News - Japan earthquake"
  snippet="Japan quake relief budget passed ..."
  link="http://www.bbc.co.uk/news/world-asia-pacific-12711226">
 </search_result>

 <search_result heading="Japan Quake Map"
  snippet="Time-lapse visualisation of the March 11, 2011 ..."
  link="http://www.japanquakemap.com/">
 </search_result>

 <search_result heading="Powerful Quake and Tsunami ..."
  snippet="Mar 11, 2011 ... Japan was filled with ..."
  link="http://www.nytimes.com/2011/03/12/world/asia/...">
 </search_result>

 <search_result heading="Japan Earthquake: New Explosion ..."
  snippet="Mar 13, 2011 ... A hydrogen explosion reportedly ..."
  link="http://abcnews.go.com/International/japan-earthqua...">
 </search_result>

 <search_result heading="Widespread destruction from Japan ..."
  snippet="Mar 11, 2011 ... The morning after Japan was ..."
  link="http://articles.cnn.com/2011-03-11/world/japan.qua...">
 </search_result>
 ...
</search>
```

When search requests are processed at runtime, the correct `Search` subclass and associated meta-metadata will be selected. Wrapper `google_search` and others that specify `type="search"` will be mapped to the data structure of Java class `Search`. Wrappers extending `search` will be mapped to the appropriate metadata subclasses extending Java class `Search`. Instances will be constructed and populated. Semantic actions will be invoked.

S.IM.PL annotations generated by the meta-metadata compiler (see above) enable de/serialization of metadata objects from / to XML or JSON for storage and communication. For example, here results from a Google Search are serialized:

Using these wrappers, we wrote a Java program in less than 50 statements to mix search results from the engines with meta-metadata and present the results (Figure 1). It takes a query as input, uses meta-metadata to make requests to the search engines. All resulting metadata objects are of Java class `Search`, allowing the program to simply iterate over the collection field `search_results` to mix search results. The program then uses the meta-metadata runtime's built-in DHTML rendering capabilities to present the re-ranked integrated results to the user. The support for polymorphism preserves specialized fields such as "tags" and "author" from Slashdot in the integrated results. The program and results, along with other examples, are avaiable online [8].

4. CONCLUSION AND FUTURE WORK

Meta-metadata functions as a basis for interoperable metadata semantics for developing applications connecting published information sources, metadata schemas, and users. Meta-metadata wrappers integrate metadata structure, extraction rules, semantic actions, and presentation rules. The meta-metadata runtime automates the process of structurally extracting, translating, and connecting metadata semantics to applications, supporting software development. Inheritance and polymorphism encourage re-use of the repository and sharing of wrappers, transferring interoperability across time and space.

Meta-metadata has been used to develop a variety of applications using metadata semantics. There are presently wrappers for digital libraries, products, restaurant reviews, and social media, including Google Books, the ACM Portal, Amazon, Urbanspoon, Wikipedia, and Flickr. One student uses it to integrate RSS feeds in a novel interface. Another uses it to scrape artwork data with semantics from a museum web site for an exhibition planning tool. Our creativ-

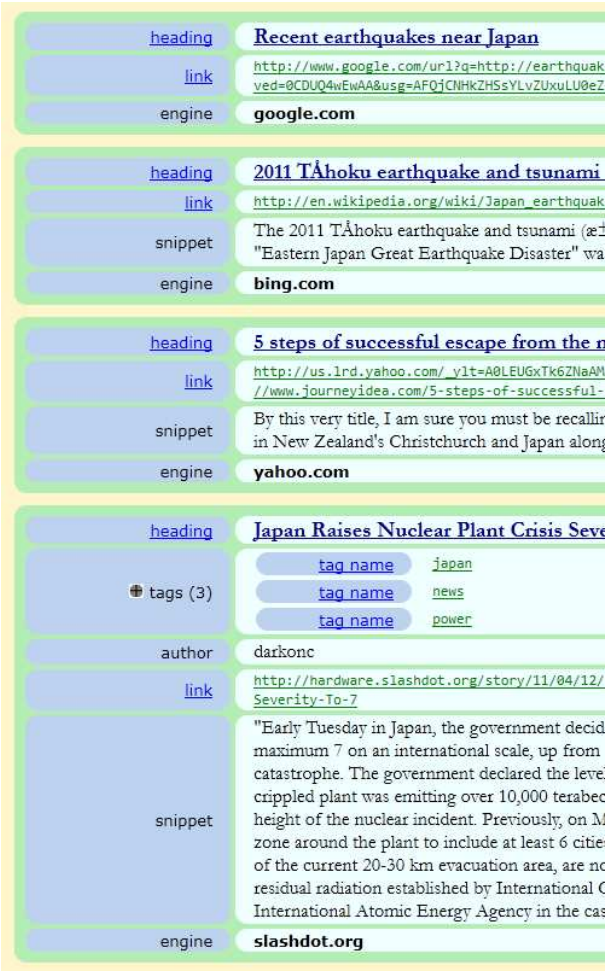

Figure 1: Meta-search for "japan earthquake", from Google, Bing, Yahoo Buzz and Slashdot. Note that specialized fields such as "tags" and "author" from Slashdot are preserved.

ity support tool, *combinFormation* [9], uses meta-metadata to operate on and present rich semantics to users amidst a holistic visual information presentation.

We envision meta-metadata as a foundation for building applications operating on metadata semantics from one or many sources. As it builds on common mechanisms like HTTP and XPath, and does not impose rigid external semantics, any template-based web site or collection can be wrapped and turned into a situated source of metadata semantics, making semantics immediately available and usable without waiting for publishers to adopt semantic web standards. Thus, meta-metadata translates the wild web into an ecosystem of interoperable semantic information. The repository allows sharing and re-use of metadata schemas / wrappers for each source. The runtime hides the complexity of handling network connections, extracting metadata semantics, and connecting to applications. As a result, applications that need metadata semantics buried in XML or HTML documents on the web, especially those that integrate multiple sources, can benefit from meta-metadata. These include browsing and searching applications [15] that use metadata fields as facets, bibliography management tools that support citation chaining and berry picking [2], planning assistants that depend on rich metadata semantics (e.g. weather, schedules, locations, etc.), and visualizations that use metadata semantics to convey stories.

In the future, we will work on connecting meta-metadata with other standards and systems such as the Semantic Web, Linked Data, and Exhibit, to extend its applicability. We will continuously extend the expressiveness of the meta-metadata language based on emerging needs and use cases. We seek collaborations with developers, curators, and publishers to enhance future derivation of metadata semantics to make good use of the world's information resources.

5. REFERENCES

[1] G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, 2004.

[2] M. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Information Review*, 13:407–424, 1993.

[3] T. Berners-Lee. Linked data. *International Journal on Semantic Web and Information Systems*, 4, 2006.

[4] T. Berners-Lee et al. The semantic web. *Scientific American*, 284:34–43, 2001.

[5] H. Glaser et al. CS AKTive Space: building a semantic web application. In *Proc. of ESWS*, pages 417–432. Springer Verlag, 2004.

[6] D. Huynh et al. Piggy bank: Experience the semantic web inside your web browser. *Proc. of ISWC*, 2005.

[7] D. Huynh et al. Exhibit: lightweight structured data publishing. In *Proc. of WWW*, 2007.

[8] Interface Ecology Lab. Meta-Metadata Guide. `http://ecologylab.net/research/simplGuide/metaMetadata/index.html`.

[9] A. Kerne et al. combinformation: Mixed-initiative composition of image and text surrogates promotes information discovery. *ACM TOIS*, 27:5:1–5:45, 2008.

[10] A. Kerne et al. Meta-metadata: a metadata semantics language for collection representation applications. In *Proc. of CIKM*, 2010.

[11] Library of Congress. Dublin Core/MARC/GILS crosswalk. `http://www.loc.gov/marc/dccross.html`, 2008.

[12] C. C. Marshall and F. M. Shipman. Which semantic web? In *Proc. of HYPERTEXT*, 2003.

[13] N. Shahzad. S.IM.PL serialization: Translation scopes encapsulate cross-platform, multi-format information binding. Master's thesis, Texas A&M University, 2011.

[14] S. Weibel et al. RFC 2413: Dublin core metadata for resource discovery. *RFC*, 1998.

[15] K.-P. Yee et al. Faceted metadata for image search and browsing. In *Proc. of SIGCHI*, 2003.