DBTropes—a linked data wrapper approach incorporating community feedback

Malte Kiesel
DFKI GmbH, Kaiserslautern, Germany
malte.kiesel@dfki.de

Gunnar Aastrand Grimnes
DFKI GmbH, Kaiserslautern, Germany
gunnar.grimnes@dfki.de

ABSTRACT

A common approach for serving Linked Data is to modify existing services to translate and export the underlying data as RDF. However, for many existing data sources on the web such an approach is not feasible: large installations might not be suitable for the changes necessary, programmers possibly are not able to adapt the software, or the data might not be suited for direct translation to RDF.
DBTropes.org is a wrapper to TV Tropes, a wiki describing works of fiction by associating features—known as “Tropes”. DBTropes is an independent service only using public data available via HTTP and translating it to RDF. Since the TV Tropes wiki does not provide structured data, the extracted data is noisy, and the interpretation of the data is sometimes ambiguous. DBTropes features a user interface that allows correcting and amending the data extracted from TV Tropes. This allows the extracted data to stay in sync with the original wiki, while also allowing the linked-data community to fix extraction errors.

Introduction

The *Web of Data* is a model for publishing structured data online. Dereferenceable URIs are used as identifiers, and either human- or machine-readable content is served over HTTP, using HTTP redirection to provide content-negotiation [2]. The motivation for the Web of Data is to solve the chicken and egg problem for the semantic web: As long as no interesting data is available, no-one will write services that consume it.

We describe our experience of making the data from an existing collaborative wiki available as linked data. Although our end-result is specific to the TV Tropes wiki, the challenges and solutions we used are general. In particular, our choice of a live wrapper of the wiki, as well as allowing the community to fine-tune the wrapping process, raise many interesting issues that are applicable outside this use-case.

The Need for Wrappers

Most data on the World Wide Web is available as websites for human consumption, marked up using the HTML. Rendering this content is easy for machines—however, machine support in using the data is limited to simple tasks such as keyword search. Some services such as Flickr or Delicious provide access to structured data underlying the HTML representation of their data through programming APIs. Very few services publish their data according to the linked data principles. However, most services do not expose their data in a machine-readable form at all.
The lack of linked data-exposing services is due to a multitude of reasons stemming from technical, social, but also economical reasons:

- Some services are very large and complex—extending the software running it to also serve linked data is difficult.
- For complicated domains, mapping the underlying data representation to linked data formats is nontrivial, and additional ontological information is needed for linked data representation.
- The data contained in the service is not available as structured data but only as plain text or other media.
- Making data available as linked data is just not a priority for the website’s community or administrative people.

Fortunately, in case the data is available under a liberal license, such as the GNU Free Documentation License[1] (GFDL) or most of the Creative Commons (CC) licenses[2], *wrapping* the data into a service separate from the original website might be possible. Wrapping solves some of the problems explained above:

- Even large services can be wrapped since the linked data service is independent from the original website, not obstructing the original service or imposing transition problems.
- Extraction and data enrichment methods that are not (yet) available as off-the-shelf solutions can be employed in the wrapper, not jeopardizing the original service’s availability or integrity.
- Specialized communities can form: The community behind the original service typically has other priorities and expertise than the community using the data exposed as linked data.

[1]`http://www.gnu.org/copyleft/fdl.html`
[2]`http://creativecommons.org/choose/`—in general any CC-license that allows derived works are suitable

Online Wrapping: DBTropes

As a case study, we built an online wrapper to the TV Tropes wiki, resulting in the DBTropes.org linked data source[3]. Unique to our wrapping approach, the DBTropes site also has a HTML front-end for end-users. This allows users to tweak the way resources are processed, removing incorrectly extracted facts and linking our pages to the rest of the web of data.

TV Tropes is a catalog of *tricks of the trade* for writing fiction, known as *tropes.* According to the tvtropes.org introduction page:

Tropes are devices and conventions that a writer can reasonably rely on as being present in the audience members' minds and expectations.

The wiki includes dozens of thousands of tropes and items. Each trope-page contains a description of the trope, as well as links to related tropes and links to example items (movies, games, etc.) where this trope occurs, almost always with a comment explaining why that trope is relevant in the context.

In contrast to information-sources like wikipedia, TV Tropes does not attempt to be objectively correct and detailed. However, the plot devices employed in a movie or the adherence to realism exhibited in a book might be much more relevant to a human than the purely objective information such as release dates or movie casts. Thus, DBTropes nicely complements the information contained in Wikipedia/DBpedia and might be used for recommendation and clustering functionality. In fact, DBTropes data is used in the Skipforward project[4] exactly for these purposes.

Building Blocks of the Wrapper Software

In Figure 1, the components of an online wrapper are shown along with the data flow between them. The *input HTML cache* helps relieving the wrapped website from unnecessary load. For example, if a user of the wrapper tries different settings, we do not want the wrapper to retrieve the wrapped web page multiple times. Also, in case the wrapper website gets crawled by a search engine bot or a linked data browser, we need to make sure the load imposed on the wrapped website stays as low as possible. The *HTML parser* extracts information from the fetched HTML pages. In the case of DBTropes, we used a set of XPath[5] expressions for this step. For other use cases, general screen scraping techniques can be employed (see *Piggy Bank* [1], an RDF screen scraping framework). The *interpreter* generates RDF from these data snippets. Typically, additional information for generating RDF is needed—this information is fetched and updated in the *processing information store.* The *dependency manager* controls updating wrapped pages in case metadata/processing information data changes. The *RDF filter* hides RDF statements marked as invalid by *user feedback.* Users can also correct some other information such as the page type in the TV Tropes scenario.

[3]`http://dbtropes.org/`
[4]`http://skipforward.opendfki.de/`
[5]`http://www.w3.org/TR/xpath`

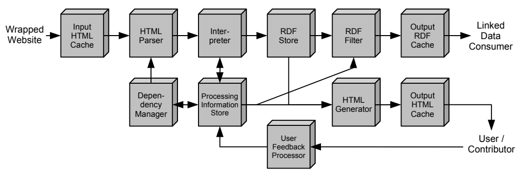

Figure 1: The components of the online wrapper. Arrows show information flow. Caching management not shown.

Statistics and Discussion

As of July 2010, DBTropes contains information about more than 13,000 movies and other items, about 18,000 tropes, and almost 1,200,000 trope occurrences[6]. The full RDF dump contains about 7,000,000 statements in almost 1.5 GBytes of RDF data (N-TRIPLES format).

We did an analysis of precision and recall of the trope extraction process. This covered randomly selected item and trope pages with about 580 trope occurrences all in all.

In the test set's *trope pages*, 460 trope occurrences were counted manually. 100 of these were deemed not to be extractable automatically in any case (because the items mentioned were present only as plain text and not represented as a wiki link, etc.). DBTropes extracted 300 trope occurrences. Most trope occurrences identified but not extracted (58) were due to DBTropes not having enough data to estimate the type of the page linked to—in this case, DBTropes errs on the side of caution and drops the statement, giving a notice. This leads to a recall of about 83%. Of the extracted occurrences, 14 were invalid, yielding 95.3% precision.

In the test set's *item pages*, 120 trope occurrences were counted manually. Apart from 4 of them, all seemed extractable with reasonable effort. DBTropes extracted 104 trope occurrences, having dropped 11 occurrences due to missing type data. Of the extracted occurrences, 4 were invalid. This leads to 89.7% recall and 96.2% precision.

We were able to remove all invalid occurrences using the interactive DBTropes user feedback features after measurements, resulting in 100% precision. Adding a feature that allows users to add new information would also be possible, potentially increasing recall. However, this is something we expect to be done much better through the wrapped service (editing the TV Tropes wiki in this case) since, as the evaluation shows, most misses are due to missing primary information in the original wiki.

1. REFERENCES

[1] D. Huynh, S. Mazzocchi, and D. Karger. Piggy bank: Experience the semantic web inside your web browser. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):16–27, 2007.

[2] Leo Sauermann and Richard Cyganiak. Cool uris for the semantic web. w3c interest group note 03. `http://www.w3.org/TR/cooluris/`, December 2008.

[6]A feature instance/trope occurrence is a statement of the type “item X features trope Y” or vice versa.