Linked Data Based Library Web Services for Economics

Joachim Neubert
ZBW German National Library of Economics
Leibniz Information Centre for Economics
j.neubert@zbw.eu

Abstract

A large number of library metadata resources have become available as Linked Open Data (LOD) in the last two years. We see, however, not much re-use of this data in library applications.

The paper discusses hurdles for a broader adoption of such resources. It suggests building lightweight REST-oriented web service interfaces which fit well in the Web 2.0/Mash-up mindset of the majority of application programmers. Exemplifying this approach in the field of economics, we built and published *Web Services for Economics* based on a thesaurus, a classification, a personal and a corporate names authority file, all on economics, and interdisciplinary mappings to other terminological resources. Furthermore, we demonstrate how these services are integrated in real-life library applications and how authoring and publishing platforms can be enhanced to make use of them.

Keywords: economics; web services; REST; API; Linked Data; LOD; Semantic Web; thesaurus; classification; authority file; authorities

1. Motivation

Many libraries have published resources as Linked Open Data. As of June 2012, the Linked Library Data group of the Data Hub[1] spans 57 datasets, on bibliographic resources as well as on authorities. Converting datasets to Linked Data basically means two things: assigning web addresses (Uniform Resource Identifiers or URIs) to each entity, and expressing the properties of these entities by the means of the Resource Description Framework (RDF) in subject/predicate/object triples. The predicates of these triples consist of URIs themselves and are taken from one of the vocabularies designed for this purpose, such as Dublin Metadata Terms or Simple Knowledge Organization System (SKOS). The object of each triple may be another URI, especially one outside the originating dataset. This allows for easy linkage between datasets of completely different and unrelated providers, forming a giant and virtually worldwide graph.

However, the downside of this approach is that the traditional metadata record, which holds all related information about an entity together, dissolves into a set of triples. Within a larger dataset, these triples are indistinguishably merged with triples from other records, possibly of different record types and originating from different datasets. Library datasets typically range from tens of thousands to billions of triples. Therefore such RDF datasets, like highly normalized relational databases, have to be queried with elaborate queries to reassemble triples and to construct meaningful information that can be delivered to users. For this purpose, SPARQL (a recursive acronym for SPARQL Protocol and RDF Query Language) has been developed, resembling the Structured Query Language (SQL) for relational databases.

An immense advantage over relational databases, however, is the fact that the URI provides a worldwide unique "primary key" for each entity, from which (following the Linked Data rules) further information should be obtainable through HTTP lookup, most basically by putting the URI into a semantic or even conventional web browser. This builds the technical basis for the enormous potential LOD offers for opening up and interlinking data. But it requires new and hard-to-achieve skills to work with this kind of data.

[1] see http://thedatahub.org/group/lld

LOD data sets are currently provided mainly in three forms:

1) as interlinked "single record" RDFa (RDF embedded in HTML) pages or other RDF files (and in future certainly HTML5 microdata) with a predefined subset of triples,
2) as massive download files in RDF/XML and/or N-Triples and/or Turtle format,
3) as publicly accessible SPARQL query interfaces, so called endpoints.

Interlinked "record" files are well suited for follow-your-nose exploration of datasets, but normally need some data conversion before the data can be integrated into applications, and they cannot be queried at scale.

Bulk downloads require a Linked Data/Semantic Web infrastructure in place to be useful: Triplestores and SPARQL engines don't fit easily in library infrastructures (especially since the larger datasets come with high memory and processing power requirements). Their setup demands skills that cannot be taken for granted among current library IT staff. Thus, outside large and dedicated projects, the hurdle for the adoption of such datasets in library applications may be prohibitively high.

SPARQL endpoints are very flexible in delivering the required pieces of data. On the side of the application programmer, they require however a basic knowledge of RDF and graphs, there is a learning curve for the syntax, too, and dataset-specific know-how is indispensable: Which classes and properties are used for what? This requires thorough analysis of each and every dataset used. In practice, implementation-specific know how is often demanded as well. Some ways to query an endpoint are more efficient than others (we saw equivalent queries taking 40 or 1400 milliseconds).

For the data providers, unrestricted SPARQL endpoints are like open SQL interfaces: They are vulnerable to hapless queries, to ignorant batch runs or to all kind of denial-of-service attacks and as-yet-unknown risks in their security architecture. Therefore, we are not aware of any SPARQL endpoints with production status (defined service levels with regard to availability, response times, etc.) "in the wild".

To overcome these obstacles and to hide the complexity of Linked Data from its consumers, we built Web Services for Economics[2]. Web services provide access to resources by programs communicating over the Internet and are widely used in Web-2.0 style mash-up applications, exploiting services by Amazon, Google and numerous other providers. They expose an application programming interface (API) and allow the encapsulation of sensitive infrastructure components and the standardization and implementation-specific optimization of queries, which pleases the operating requirements on the provider side. On the consumer side, programmers can stick to well-accustomed Web-2.0 techniques and can integrate Linked Data based services like any other web services into their mash-up web applications.

In our approach, we wanted, firstly, to exemplify this concept for STW ("Standard-Thesaurus Wirtschaft" – Thesaurus for Economics[3]) and other economics resources. Secondly, we wanted to verify that semantic web technologies – especially SPARQL endpoints – are mature enough to power such basic infrastructures.

It should be stressed here that building these services *could* have been done without any of the Semantic Web and Linked Data technologies described in this article, relying on traditional infrastructure such as MySQL databases or Solr/Lucene indexes. However, it *would* not have been built: Essential parts of the data wouldn't have been available at all, let alone easily accessible in a common (RDF) format, and redistributable under an open license. And the vision of a scalable and sustainable Web of Linked Open Data was a strong motivating force for starting to explore the new prospects and possibilities in our very domain, information science for economics.

[2] http://zbw.eu/beta/econ-ws
[3] http://zbw.eu/stw

2. Related work

Terminology web services have been well investigated over the last ten years. Binding & Tudhope (2010) give an overview over web service approaches on terminologies so far. Most of the early services were based on the Simple Object Access Protocol (SOAP), which requires a considerable amount of infrastructure and intellectual effort to master the resulting APIs. The SKOS API[4], an early protocol to offer access to SKOS structured terminologies, tried to cover most of the underlying SKOS data model by a set of relatively fine-grained operations. It has seen a few implementations, but seems to have not been further developed since 2004. The Agrovoc web services[5] are an example of these early services, providing access to the corresponding worldwide used agricultural thesaurus (Lauser et al., 2006).

More recent implementations mostly rely on a "RESTful" approach. REST (Representational State Transfer) is an architectural style of web programming which focuses on resources and their representations (for example, RDF or HTML files). It is deeply integrated into the overall architecture of the web, using its basic protocol, HTTP. For GET (read-only) access, REST applications work very similar to looking up resources in an ordinary web browser. This RESTful way of interaction, which is far easier to implement than elaborated SOAP interfaces, was broadly adopted in Web-2.0 applications. As Linked Data is exposing resources on the web, which may be represented in different formats, it fits well into a REST architecture.

The HIVE (Helping Interdisciplinary Vocabulary Engineering) project developed a comprehensive REST API[6] to serve multiple SKOS vocabularies (Greenberg et al., 2011). Other services, such as the Authorities and Vocabularies service[7] of the Library of Congress, one of the major terminology web services in production, are currently offering very selected services, namely known-label retrieval and OpenSearch support, as well as an Atom update feed. The Virtual International Authority File (VIAF)[8] offers OpenSearch, too, as well as the library-specific SRU (Search/Retrieval via URL) protocol and an autosuggest service.

Autocomplete interfaces for resource lookup, implemented not as web services, but as components of larger systems, have been explored by Hyvönen & Mäkelä (2006) – with a focus on exploiting semantic relations –, and by Hildebrand et al. (2007). The latter describe dimensions of a design space for such a component and its implementation in the ClioPatria open source framework[9].

As our approach is focused on providing vocabularies for easy consumption by applications, the maintenance of vocabularies and alignments, as addressed in Meij, Isaac, & Zinn (2010), is regarded as out of scope here.

3. Services and API

Service and API design of Web Services for Economics, as provided by ZBW currently in beta state, aim to support easy, mash-up style use. Simple REST-style interaction seemed to fit best for the vast majority of anticipated use cases. After a short introduction of the underlying datasets, the design of the services is discussed in the remainder of this section.

The datasets strive to cover the whole field of economics and currently include STW, extended by mappings to thesauri of neighboring fields, the widely used Journal of Economics Literature Classification (JEL)[10], and sections of the authority files of the German libraries (Gemeinsame

[4] http://www.w3.org/2001/sw/Europe/reports/thes/skosapi.html
[5] ftp://ftp.fao.org/gi/gil/gilws/aims/kos/webservices/agrovoc_webservices_2.5_documentation.pdf
[6] http://code.google.com/p/hive-mrc/wiki/AboutHiveRestService
[7] http://id.loc.gov/techcenter/searching.html
[8] http://www.oclc.org/developer/documentation/virtual-international-authority-file-viaf/using-api
[9] http://e-culture.multimedian.nl/software.shtml
[10] http://www.aeaweb.org/jel/jel_class_system.php

Normdatei, GND)[11], providing URIs for persons and institutions. Despite the different resources referenced, the interface is basically the same.

It is oriented on the SKOS data model. In particular, we exploit the `skos:prefLabel/altLabel/hiddenLabel` properties' characteristics even for resources such as personal name authorities which do not provide these properties originally. As the SKOS label properties do not imply the `skos:Concept` domain, this comes without ontological burdens.

Web Services for Economics (econ-ws, formerly stw-ws) focus on three use cases:

1) resource lookup (autosuggest services)
2) retrieval support
3) mapping between datasets

For these use cases, the services aim to support applications efficiently, normally by delivering all needed information for fulfilling the use case in one single call. In the restriction to a few coarse-grained services, tailored to specific use cases, our approach differs significantly from most of the web services referenced above. It should be noted that this does not impose hard limits on the data consumers: As all underlying data is published as LOD, it is possible for application programs to additionally or alternatively access every bit of the data for extended use, however with higher costs in parsing RDF and possibly multiple round-trips over the network.

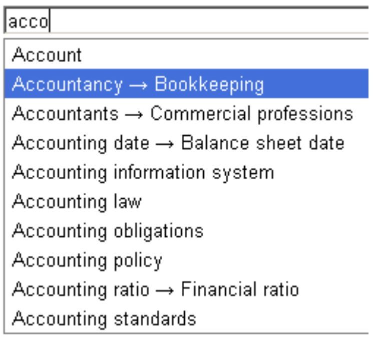
Generally, the results are returned in the formats SPARQL-XML[12], SPARQL-JSON[13] or HTML (SPARQL-XML + XSLT style sheet), as produced by the underlying Joseki[14] SPARQL endpoint. Although less expressive than other possible result set formats (lacking the original query and the number of hits), these formats require minimal post-processing on the web service layer and allow easy integration with query results from other, possibly institution-internal SPARQL endpoints.

Though the implementation is currently driven completely by LOD/Semantic Web technology (RDF datasets and a Joseki SPARQL server with accompanying Apache Jena infrastructure), it should be noted that this implementation can be replaced if the endpoint does not scale according to future needs, without any disruption for the users of the services.

One common, and until recently hard-to-overcome, hurdle for service use in web applications is that browsers generally disallow javascript access to third party sites ("same origin policy"). To solve this, econ-ws supports the Cross Origin Resource Sharing (CORS) specification[15].

3.1. Autosuggest Services

The lookup of resources by autosuggest services is ubiquitous in today's web applications, mostly based on local databases or Solr indexes. Regarding LOD resources, which are identified by HTTP URIs, autosuggest services are perhaps the most important instrument for the proliferation of these URIs. The econ-ws /suggest service, which at first was developed for STW (Neubert, 2009), exploits the SKOS data model in its multi-lingual as well as in its prefLabel/altLabel features. It returns the preferred label of a resource, e.g. the descriptor of a thesaurus, in the desired language together with the language-independent concept URI. Since it also returns the matched label (which may be an alternate or hidden label), this allows the construction of


acco

Account
Accountancy → Bookkeeping
Accountants → Commercial professions
Accounting date → Balance sheet date
Accounting information system
Accounting law
Accounting obligations
Accounting policy
Accounting ratio → Financial ratio
Accounting standards

FIG. 1. Autosuggest hints to preferred labels

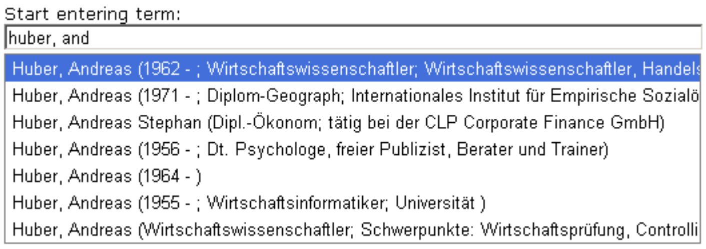
[11] http://www.dnb.de/EN/Standardisierung/Normdaten/GND/gnd_node.html
[12] http://www.w3.org/TR/rdf-sparql-XMLres
[13] http://www.w3.org/TR/rdf-sparql-json-res
[14] http://www.joseki.org/
[15] http://www.w3.org/TR/cors. (In order to work properly, besides the "Access-Control-Allow-Origin: *" we had to add an "Access-Control-Allow-Headers: X-Requested-With" header.)

user interfaces which hint from synonyms to the authoritative preferred label (see FIG.1).

The application which calls the service can use the concept URI for unambiguous internal and external linking, and also for updating the preferred labels in all desired (and available) languages for multilingual user interfaces.

The JEL classification, for which ZBW published a beta LOD version[16], is offered in a very similar way. Two things are special, however: The preferred label is built by a concatenation of the notation and the label of the entry. And for multi-thematic entries like "C53 – Forecasting Methods; Simulation Methods" distinct `skos:altLabels` ("Forecasting Methods", "Simulation Methods") are added for easier lookup.

Authority files such as the German GND form a different use case. In large personal name authorities (GND contains more than 2 million differentiated persons), names are hardly unique. On the other hand, lots of variant names for one single person may exist – some completely different (such as birth names), but more often only slight variations (with or without middle initial, for example). Therefore it is challenging for users to select the correct person. The econ-ws /suggest2 service tries to ease this by, firstly, providing only a subset of economists and related professions. The subset was selected by (a) all persons (identified by their GND id) with publications in our portal EconBiz[17], and (b) persons from the set of all GND persons with professions that make it likely that they will publish in economics (such as sociologists or geographers). This reduces the number of persons from 2 million to 150,000 more relevant ones. Secondly, the service does not only deliver the preferred name and the URI, but also an extended preferred label which is constructed from birth and death year, professions, affiliations and other information available in the authority file (see FIG. 2), and which is supposed, but not formally guaranteed, to be unique (a highly valuable characteristic of `skos:prefLabel`). The construction of these labels is expensive and done while building the dataset. Thirdly, the entries delivered in this service are ordered by the presumed "relevance" in the field of economics (which is simply derived from the count of their publications in EconBiz). The alternate names from the GND dataset are mapped to `skos:hiddenLabels`. They are used for resource lookup, but (unlike the /suggest service described above) not returned, in order to avoid cluttering the result list with lots of variations of similar names which all point to the same person.



Start entering term:

huber, and

Huber, Andreas (1962 - ; Wirtschaftswissenschaftler; Wirtschaftswissenschaftler, Handels
Huber, Andreas (1971 - ; Diplom-Geograph; Internationales Institut für Empirische SoziaIö
Huber, Andreas Stephan (Dipl.-Ökonom; tätig bei der CLP Corporate Finance GmbH)
Huber, Andreas (1956 - ; Dt. Psychologe, freier Publizist, Berater und Trainer)
Huber, Andreas (1964 -)
Huber, Andreas (1955 - ; Wirtschaftsinformatiker; Universität)
Huber, Andreas (Wirtschaftswissenschaftler; Schwerpunkte: Wirtschaftsprüfung, Controlli

FIG. 2. Extended preferred labels for personal name authority

The implementation of this service had to work around the fact that, for large numbers of entries, substring matching in the used triple store (Jena MemoryModel – and TDB as well) is slow. Therefore we introduced an additional property, `zbwext:shortLabel`, for which we generate the leading 2-5 characters of all labels at dataset generation time. As the shortlabel clause is executed at the beginning of the SPARQL query, the expensive regular expression filtering is only required for a subset of entries, which makes execution reasonably fast. Because such highly implementation-specific details are hidden from the users of the service, they can be dropped when SPARQL full-text indexes become available and the hack is no longer necessary.

One disadvantage of the composite labels for persons is that their lengths often exceed reasonable sizes for suggest boxes or may even overshoot the maximum field length in the target system. In a first attempt, we had computed (during preprocessing) the minimum of additional information required to make the preferred labels unique. But we realized that we thus may miss the bits of information which are crucial to the user to select the right person (e.g. out of the 15

[16] http://zbw.eu/beta/external_identifiers/jel
[17] http://www.econbiz.de, with about 10 million publications in economics

"Peter Schmidt" in the dataset). Since it is not possible to predict which bits of information are most significant for a given set of persons sharing the same name (and since the field usage in the authority data is not completely consistent), we sometimes ended up with very long labels. An interesting solution to this problem has been developed by Jon Scott (Washington State University), who provides an overlay with additional information on mouse-over within the autosuggest list (Coombs, 2010). This information could even include bibliographic data and could be looked up through a second service call to a still-to-be-developed /info service.

3.2. Retrieval Support Services

For retrieval support, econ-ws offers a simple /synonyms service as well as a /combined1 service, which make use of semantic relations. The /synonyms service is aimed at basic full text search scenarios and simply tries to match a given query string with some STW preferred or alt label and returns a flat list of all preferred and alternative labels in all languages, e.g. for

```
http://zbw.eu/beta/econ-ws/synonyms?query=managers&output=sparql-json
```

the service returns (greatly abbreviated – compare FIG. 4 for full example in an application)

```
 { "synonym": { "type": "literal" , "xml:lang": "en" , "value": "managers" } }
 ,
 { "synonym": { "type": "literal" , "xml:lang": "en" , "value": "corporate
executives" } } ,
 { "synonym": { "type": "literal" , "xml:lang": "de" , "value": "direktoren" }
} ,
 { "synonym": { "type": "literal" , "xml:lang": "de" , "value": "leitende
angestellte" } },
 { "synonym": { "type": "literal" , "xml:lang": "en" , "value": "chief
executive officers" } },
```

This can be extended further by a parameter `vocabs=ALL`, which extends lookup and term expansion to other vocabularies (see below), making use of `skos:exactMatch` relationships.

The /combined1 service works in a much more sophisticated way. It matches the query string against a full text (Lucene) index and is therefore tolerant against differences in the exact wording of labels, such as different word sequence, as well as the combination of multiple concepts in a single search (e.g. "telework wages for women"). This is a key difference to our /synonyms and similar services, which are aimed at the extension of a single concept. Real live search queries most often consist of multiple words, and there is no easy way to find out if these words map to one or more concepts. Thus, showing what comes up from a full text search on the labels of a terminology proved an efficient way to propose matching concepts. As a recent study (Borst, 2012) shows, for about two third of the EconStor queries such concepts could be suggested.

For the matching concepts found, the service returns the concept itself and, by default, the narrower and related concepts. For each of these concepts, all available labels are returned. This happens in *one* call, and typically in less than half a second. The result can be used to expand search results automatically, or to offer users a clickable interface for search expansion (see EconStor example below, FIG. 3). The single round trip is essential for this kind of use – more fine-grained services (get the matching concepts, get the narrower/related concepts, and get the labels for each of them) would be conceptually cleaner, but take a multiple of the current response times.

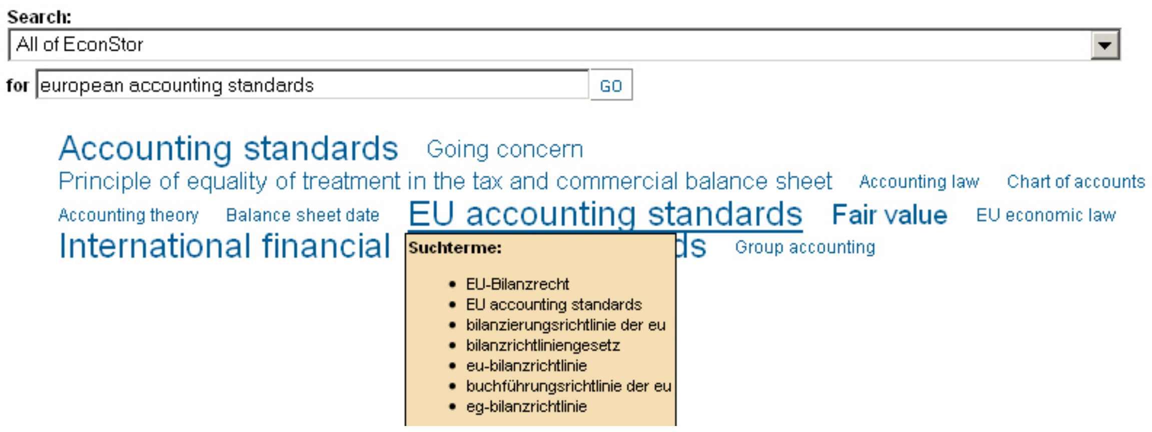


FIG. 3. Search support in EconStor through /combined1 service

However, to work well (i.e., to return a reasonable number of concepts, avoid semi-duplicate concepts), the service had to be optimized heavily. Unlike the other services, it relies on a Lucene/LARQ index, an option provided by the Jena-based Sparqlite[18] server. Results can be restricted by a relevancy threshold or to the top *n* terms matched. We found the best balance using the latter, with *n* = number of words in the query (i.e., the maximum number of intended different concepts). When we started to integrate additional vocabularies, it turned out that the straightforward approach—using SPARQL queries which spanned the different `skos:exactMatch` related concepts—gave too many similar results: The full text search in the first step of the query matched labels of different concepts, which could not be reduced ex post. So we flattened the dataset by directly attaching all unique (lowercased) labels to STW concepts and avoiding any other literals within this specially crafted dataset. Again, this highly application-specific setting may change in future, when we evaluate the full text search extensions for Fuseki (successor of the Joseki server, also part of Apache Jena), or maybe other RDF servers. As all of this is hidden from the users of the service, we are free, again, to change the implementation.

3.3. Mapping Services

In 2011, econ-ws was extended by a /mappings service. It is based on `skos:*Match` relationships between STW and Agrovoc, DBpedia, the German subject authority file (part of GND) and the social sciences thesaurus TheSoz.

The data for this service, which was obtained from the United Nations Food and Agriculture Organization, the German National Library and GESIS – Leibniz Institute for the Social Sciences, was already in SKOS format. It had however to be preprocessed, too, especially in order to flatten skos-xl label properties and some special constructs for language mappings used in TheSoz.

This service takes a list of concept URIs and returns the URIs of all matching concepts, along with their preferred labels and the mapping relations, e.g. for

```
  http://zbw.eu/beta/econ-
ws/mappings?concept=http://zbw.eu/stw/descriptor/19218-6&lang=en
```

it returns statements such as (abbreviated)

```
<result>
    <binding name="concept">
      <uri>http://zbw.eu/stw/descriptor/19218-6</uri>
    </binding>
    <binding name="prefLabel">
      <literal xml:lang="en">Marginal analysis</literal>
```

[18] http://code.google.com/p/sparqlite/

```
      </binding>
      <binding name="relation">
        <uri>http://www.w3.org/2004/02/skos/core#exactMatch</uri>
      </binding>
      <binding name="targetPrefLabel">
        <literal xml:lang="en">marginal productivity</literal>
      </binding>
      <binding name="targetConcept">
        <uri>http://lod.gesis.org/thesoz/concept/10045917</uri>
      </binding>
      <binding name="target">
        <uri>http://zbw.eu/stw/mapping/thesoz/target</uri>
      </binding>
  </result>
```

Other than retrieval support services (as described above), the /mappings service aims at "translating" concepts from one vocabulary to concepts from another. For a given book, for example, this may apply to multiple concepts. Again, for use in applications it is essential to get all related information in one round trip. A use case for this could be suggesting descriptors to indexers, based on indexing already undertaken in another vocabulary. This use has been demonstrated in an example application,[19] which takes an ISBN number as input, looks up the existing STW descriptors for it, and suggests GND keywords. (A prototype evaluation for a similar, but much more comprehensive system is described in Isaac et al., 2009)

4. Usage of Services

A design goal for Web Services for Economics was to make its use easy in web applications – particularly in javascript mash-ups. Therefore, results in Javascript Object Notation (JSON) and a callback parameter are provided. This allows for very shallow integration on a web page level, without putting stress on the application backend. Example HTML pages with embedded javascript code, using the autosuggest component of the Yahoo! User Interface library, are provided.[20] Usage examples in library applications and authoring environments are discussed in the remainder of this section.

4.1. Library applications

In a similar way as in the example pages, but making use of the Google Web Toolkit javascript library, these services are integrated into the self-upload pages of the ZBW subject repository, EconStor,[21] for adding support on the entry of STW descriptors, JEL codes and personal names (and amending names with identifiers behind the scenes). This integration does not touch the underlying DSpace repository software at all, but is simply handled on the page level.

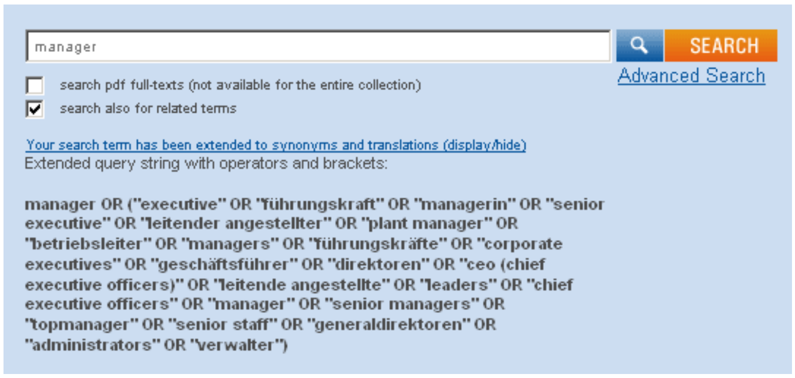
More publicly visible is the search support in EconStor: Since the repository collects papers from many diverse sources, most of them are not indexed with controlled vocabularies. Therefore, the searches to support are full text searches, spanning German and English text. Search phrases entered by the users are looked up in the background at the /combined1 service. Its data is rendered as suggestions of relevant subjects and asynchronously added to the result page (see FIG. 3). When the user follows a suggestion and clicks a term, all existing English and German synonyms are concatenated to a query string (e.g., "European accounting standards" OR "EU-Bilanzrecht" OR "bilanzrichtliniengesetz" OR …), which is inserted into the query form field and submitted to the DSpace server. In this way, much more complete full text searches can be triggered, and users are offered related subjects (like "EU economic law" or "Fair value" in the example), which may be relevant for their particular search interest.

[19] http://zbw.eu/beta/econ-ws/examples/indexing.html
[20] http://zbw.eu/beta/econ-ws/examples/suggest.html, http://zbw.eu/beta/econ-ws/examples/suggest2.html
[21] http://www.econstor.eu

The loose coupling on the HTML page level offers two more benefits: Responses to user requests are not delayed, but subsequently enriched. And if the web service, which is still in beta, is slow or completely down, this does not affect the availability of the production state EconStor application.

A shortcoming of the current implementation is that it does not offer a simple facility to combine two or more concepts. Although the most useful case is conceptually straightforward ((‘a’ OR ‘b’ …) AND (‘c’ OR ‘d’ …)), it is challenging to present this in catchy fashion in the user interface, and requires future work.

Another way to integrate a retrieval service shows up in EconBiz portal. If the user activates the “search also for related terms” checkbox, a call to /synonyms service with vocabs=ALL parameter is launched, in this case on the server side. Also on the server, the query is enriched with the retrieved synonyms, and the user gets the result of the extended search. Only afterwards and on demand the user may inspect the constructed query (FIG. 4).



☐ search pdf full-texts (not available for the entire collection)
☑ search also for related terms

Your search term has been extended to synonyms and translations (display/hide)
Extended query string with operators and brackets:

manager OR ("executive" OR "führungskraft" OR "managerin" OR "senior executive" OR "leitender angestellter" OR "plant manager" OR "betriebsleiter" OR "managers" OR "führungskräfte" OR "corporate executives" OR "geschäftsführer" OR "direktoren" OR "ceo (chief executive officers)" OR "leitende angestellte" OR "leaders" OR "chief executive officers" OR "manager" OR "senior managers" OR "topmanager" OR "senior staff" OR "generaldirektoren" OR "administrators" OR "verwalter")

FIG. 4. EconoBiz extended query through / synonyms service

In both cases, the programmers who made use of the services had no Semantic Web, RDF or SKOS training and didn’t know anything about the working of SPARQL queries. They were able to integrate the services—in quite different ways—simply based on the given web service API documentation.

4.2. Authoring environments

While integration on the javascript level requires coding by the provider of an application, we currently experiment with ready-made plugins for editorial and publishing systems. An early (alpha) approach for this is the “Economics Taxonomies” module[22] for the Drupal content management system, which allows for the integration of STW and economists autosuggest services. It works as a plugin for another alpha module, Web Taxonomy[23], which allows integration of external “taxonomies”. These taxonomies in fact can be arbitrary data sources providing autocomplete data in JSON – and can be in particular SPARQL endpoints which deliver sparql+json. The web taxonomy module saves selected entries as local Drupal taxonomy terms, along with a “web term id”, which may be the URI of a LOD resource. An `owl:sameAs` statement for the resource and the local taxonomy term can be added in the RDFa produced by Drupal 7. Other than the Drupal core taxonomy functions, which don’t allow for synonyms, the external services can support synonym lookup as well.

A batch update function is provided by web taxonomy that allows synchronizing terms that may have changed in the source terminology. This could be extended to functionalities such as

[22] http://drupal.org/sandbox/jneubert/1447918
[23] http://drupal.org/project/web_taxonomy

automatically get preferred labels for all languages supported by the local system (and provided by the external terminology, of course).

Our vision is that with a module like web_taxonomy, and plugins for different data sources provided by different developing parties, Drupal site builders can integrate terminologies through configuration, without programming efforts required. Updates would be handled automatically, superseding the need to follow release cycles in costly bulk data uploads. The burden of terminology building and maintenance for the authors working with such a system would be reduced to selecting the terms which fulfill their local requirements from larger, well-constructed terminologies. Thus, the usage of controlled vocabularies could become a commodity.

5. Conclusions

Generalizing the approach described here to SKOS and other value vocabularies, and resource discovery and lookup at large, is a future task. Currently, for each dataset added, we learned about new requirements, some changing or augmenting the interface. Also, implementations are still too sparse. However, some basic findings should hold: Firstly, Linked Data based web services should strive to reduce network round trips, and especially never return merely URIs but augment them consistently with labels which can be used in user interfaces without further lookup. Secondly, the SKOS labeling properties provide valuable patterns, namely, with `skos:prefLabel` an (almost) unique string representation, which is useful not only for selection in autosuggest interfaces, but also for unambiguous local display; `skos:altLabels` provide formalization for all kinds of synonyms; and `skos:hiddenLabels` allow arbitrary additional hints pointing to the resource. These patterns should be applicable beyond genuine SKOS vocabularies.

Our general conclusions from the development and usage of Web Services for Economics are threefold so far:

1) Restricting terminology web services for libraries (and beyond) to such aimed at, and optimized for, common use cases in applications—resource lookup, retrieval, and mapping—and, offering simple RESTful interfaces for those web services proved to be instrumental for lightweight integration of Linked Data terminology resources into web applications.
2) The work proved that it is possible to build sufficiently efficient services on RDF data and SPARQL endpoints. However, sometimes this required highly optimized queries and transformations of the datasets to support them. Being able to keep the idiosyncrasies of these optimizations hidden and out of the API gives us freedom for future revision.
3) Building terminology web services for a special field such as economics enables the offering of a common interface not only for different domain-specific services, but also allows the exploiting of domain-specific knowledge outside the terminologies themselves for improving the services—particularly narrowing down subsets of persons or institutions relevant for the field, or ranking them according to the frequency of their occurrence in bibliographic datasets.

The demonstrated lightweight web service approach, amending more challenging RDF and SPARQL services, should lower the barrier for libraries entering the "Web of Data". We would be glad to see its transmission to other fields.

References

Binding, Ceri, & Tudhope, Douglas. (2010). Terminology Web Services. *Knowledge organization*, *37*(4), 287–298. Rt. from http://hypermedia.research.glam.ac.uk/media/files/documents/2010-12-16/cbinding_dtudhope_KO_rev7.pdf

Borst, Timo. (2012). Usage and impact of controlled vocabularies in a subject repository for indexing and retrieval. *LIBER Quarterly*, *21*(3). Retrieved from http://liber.library.uu.nl/index.php/lq/article/view/8035

Coombs, Karen (librarywebchic). (2010, November 11). AutoSuggest Implementations Abounds. *OCLC Developer Network News*. Retrieved from http://www.oclc.org/developer/news/autosuggest-implementations-abounds

Greenberg, Jane, Losee, Robert, Agüera, José Ramón Pérez, Scherle, Ryan, White, Hollie, & Willis, Craig. (2011). HIVE: Helping interdisciplinary vocabulary engineering. *Bulletin of the American Society for Information Science and Technology*, *37*(4), 23–26. Retrieved from https://asis.org/Bulletin/Apr-11/AprMay11_Greenberg_etAl.pdf

Hildebrand, M., Ossenbruggen, J. R. van, Amin, A. K., Aroyo, L. M., Wielemaker, J., & Hardman, L. (2007). *The design space of a configurable autocompletion component*. Retrieved from http://oai.cwi.nl/oai/asset/12297/12297D.pdf

Hyvönen, Eero, & Mäkelä, Eetu. (2006). Semantic autocompletion. *Proceedings* (pp. 739–751). Presented at the Asia Semantic Web Conference (ASWC), Beijing. Retrieved from http://www.cs.helsinki.fi/u/eahyvone/publications/2005/autocompletion.pdf

Isaac, A., Kramer, D., Van Der Meij, L., Wang, S., Schlobach, S., & Stapel, J. (2009). Vocabulary Matching for Book Indexing Suggestion in Linked Libraries–A Prototype Implementation and Evaluation. *The Semantic Web* (pp. 843–859). Presented at the ISWC. Retrieved from http://www.few.vu.nl/~aisaac/papers/stitch-iswc09.pdf

Lauser, Boris, Sini, Margherita, Salokhe, Gauri, Keizer, Johannes, & Katz, Stephen. (2006). Agrovoc Web Services: Improved, real-time access to an agricultural thesaurus. *IAALD Quarterly Bulletin*, 79 – 81. Retrieved from http://eprints.rclis.org/handle/10760/15681

Meij, Lourens, Isaac, Antoine, & Zinn, Claus. (2010). A Web-Based Repository Service for Vocabularies and Alignments in the Cultural Heritage Domain. *The Semantic Web: Research and Applications* (Vol. 6088, pp. 394–409). Presented at the ESWC, Heraklion, Greece: Springer Berlin Heidelberg. Retrieved from http://www.few.vu.nl/~aisaac/papers/STITCH-Repository-ESWC10.pdf

Neubert, Joachim. (2009). Bringing the "Thesaurus for Economics" on to the Web of Linked Data. *Proc. WWW Workshop on Linked Data on the Web (LDOW 2009), Madrid, Spain*. Retrieved from http://ceur-ws.org/Vol-538/ldow2009_paper7.pdf