Linguistically-Evaluated Integration of Semantic Web Data into Domain Ontologies

Alessandro Aichmann
Alpen Adria Universitaet Klagenfurt, Department of Applied Computer Science
Universitaetsstraße 65-67, A-9020 Klagenfurt
Alessandro.Aichmann@gmail.com

Günther Fliedl
Alpen Adria Universitaet Klagenfurt, Department of Applied Computer Science
Universitaetsstraße 65-67, A-9020 Klagenfurt
Tel: +43 (463) 2700 3733 – Fax: +43 (463) 2700 993733 – E-mail: Fliedl@ifit.uni-klu.ac.at

Christian Winkler
Alpen Adria Universität Klagenfurt, Department of Linguistics and Computational Linguistics
Universitaetsstraße 65-67, A-9020 Klagenfurt
Tel: +43 (463) 2700 2814 – Fax: +43 (463) 2700 992814 – E-mail: Christian.Winkler@aau.at

Abstract

The present paper reports on an experiment aimed at optimizing Semantic Web data for the construction of domain-specific ontologies. For this purpose, an Ontology Engineering Tool (The Semantic Graph Tool, SGT) was developed[1]. SGT facilitates the integration of RDF-Data-structures into a domain ontology. Exemplary source for data integration is delivered by Freebase, a major source for the Google Knowledge Graph. As for the linguistic workflow employed, it allows for removing trivial and repetitive facts contained in Freebase as well as adding more detailed information necessary for subsequent domain ontology engineering. Regarding the ontology engineering workflow, basic NLP (natural language processing) strategies like tagging, chunking, keyword weighting, and co-occurrence identification is carried out. As a result, text snippets for building a domain ontology can be identified. The following step is to integrate those Freebase terms, which are similar to the respective concept nodes. Similarity calculation is done with the help of the Lesk algorithm [1]. The labelling method follows the guidelines described in [2].

1. INTRODUCTION

With the evolution of the Semantic Web a series of Web Appliances and Data bases came to light, such as Dbpedia, FactForge, OpenCYC, YAGO [3], or FreeBase, which is of interest in the present paper.

In 2010, Google acquired Freebase with its 500 millions of named entities, linked to 3,5 billion of attributes and connections among them. In Mai 2012, Google's Knowledge Graph was launched first in the U.S., then spreading across the English speaking world, and nowadays comprising languages such as French, Spanish, and many more.

Recently, Google has announced to launch a Wikidata import review tool, and to announce a transition plan for the Freebase Search API & Suggest Widget to a Knowledge Graph-based solution before the end of March 2015. By March 31, 2015, Freebase as a service will become read-only, and the website will no longer accept edits. Moreover, the MQL write API will be retired.

[1] All programming was done during the internship of Alessandro Aichmann at Econob, Lakeside Park, Klagenfurt. http://www.econob.com/index.php/en/company/about-us. The authors wish to express their gratitude.

On June 30, 2015, the Freebase website and APIs will be retired, the last Freebase data dump however will remain available, see http://freebase-easy.cs.uni-freiburg.de/browse/
Optimizing this dump is certainly a good chance to continue using Freebase data for Ontology Engineering. This is particularly important insofar as iterative and graph based Ontology Engineering along with developing relational data bases has become a main pillar in Computer Science and Information Technology.

As for optimizing the engineering process, we completely agree with [4] in that *deeper linguistic analysis is likely to lead to better results,* referring to previous work presented at ICSSEA [5], [6], [7].

Furthermore, for professionally populating domain ontologies we adapt strategies as described in [8], which defines *ontology engineering as an iterative process involving concept learning (Ontolearn), machine-supported concept validation (ConSys), and management (SymOntoX).*

Further steps like e.g. creating an augmented domain ontology E+ should be performed best by using mapping rules as described in [9].

2. THE GOOGLE KNOWLEDGE GRAPH

Google‘s Knowledge Graph is used to provide better answers to queries sent to search engines. A Semantic knowledge base enables the user to search not only for strings but also for *things*. If for instance a search for Taj Mahal is carried out, either the respective building, or a musician, a casino or perhaps an Indian restaurant could be meant. With the help of the knowledge graph, it is possible to distinguish these queries, so that the relevant information concerning one *thing/concept/notion* can be identified much better, and eventually presented as a search result.

If a query (term) can mean several *things*, it is possible to choose the one intended by the search term. Herby, Google provides suggestions as brought out in illustration 1 (suggesting things/entries for restricting the search process). In the case of Taj Mahal for instance one can select whether information about the musician Taj Mahal or the Trump Taj Mahal Casino Resort in Atlantic City instead of content on the Taj Mahal Building should be displayed.

Once a *thing* has been identified, an info-box containing relevant information is displayed. The respective information is based on the frequency of queries regarding these facts. Furthermore, *things* (somehow) related to the entry and therefore being looked up frequently as well.

By May 16 in 2012, when the Knowledge Graph was announced, it already comprised over 500 million objects and more than 3,5 billon facts and relations to the different objects. A more up-to-date figure is currently not available, however, in the meantime the Graph itself is available in many languages, not only in English, as opposed to its start.

Freebase, a user driven Web portal, embodies the freely accessible core of the Knowledge Graph, covering 43 million things and more than 2,5 billion facts (by April 4, 2014). Free access is provided (until June 30, 2015) in order to enhance the quality of automatic quality assurance by means of manual quality assurance.

Until March 31 of 2015, users are enabled to enter data into the general Freebase namespace, to change them, and to suggest their deletion. The Freebase Namespace is composed by 100 different domains, 76 of them are domains where data is organized, the remaining are system internal domains for gathering e.g. external sources and system types such as languages and primitive data types. Furthermore, users can create and organize data in a user specific namespace called /base, where each user has his own domain, which can be read by the whole community. That way, it is possible for instance to add facts to already existing things in the Freebase namespace, even if these data are stored in the user's domain.

If a user enters data that are interesting for the community, it might be suggested that these be added to the Freebase namespace. If such a proposal is accepted by the admins, the data are included in the respective namespace, thus serving also the Knowledge Graph. The latter however disposes of even more knowledge from different sources which cannot be retrieved via Freebase.

3. FREEBASE

Freebase features a special query language: MQL, however, the common RDF (Resource Description Framework) is available for single instances)

The Freebase API, according to the Google API Project Key

- Allows 100.000 queries per day
- Allows 10.000 writing operations per day

Instance query is carried out via keyword and filter, in addition to that, Freebase Suggest Widget
Regarding the topic API – all information on an instance are rendered by MQL Read, MQL Write, and data dumps.

Deficits of MQL Read III

Frequently occurring errors can be solved by the following steps:

- Backend error -> renew query
- [NO RESULT] -> renew query
- Daily Limit Exceeded -> counter is set back at 9:00 MEZ
- Request URI too long -> HTTP GET request too long, shorten query
  Invalid -> Type ID wrong, wrong data type expected
  NULL -> renew query; not optimal attribute has not attribute value

3.1 Some basic information on the structure of RDF (Resource Description Framework)

The fundamental concepts of RDF are *Resources, Properties,* and *Statements*. As for *Resources*, one might think of a resource as an object, a *thing* we want to talk about, e.g. *authors, books, publishers, places, people, and hotels.* Every resource has a URI, a Universal Resource Identifier. A URI can be a URL (Web address) or some other kind of unique identifier.

Properties describe relations between resources, e.g. *written by, age, title,* etc. They are identified by URIs as well.

The advantage of using URIs is that a global, worldwide, unique naming scheme reduces the homonym problem of distributed data representation

Fundamental Concepts RDFs

- rdfs:subClassOf - Members of subclass are also member of superclass
- rdfs:subPropertyOf – Relations described by sub property also hold for super property
- rdfs:domain – The subject of a triple is classified into the domain of the predicate
- rdfs:range – The object of a triple is classified into the rage of the predicate

Triples & URIs (following the above mentioned *Subject – Predicate – Object* pattern)

These describe a single fact. Generally, URI's are used for the subject and the predicate. The object is either another URI or a literal such as a number or string. Literals can have a type (which is also a URI), and they can also have a language.

3.2 Optimizing the Freebase output

All information available in Freebase, like in most semantic web appliances, is standardized according to the linguistic base structure Subject – Predicate – Object.

Though, the enormous variety of semantic types for each slot of this triple structure allows a manifold description of many aspects of our real world. This is exactly what the Google Knowledge Graph takes advantage of.

Indeed, this approach is problematic if a particular world view of a person X in a domain Y exceeds these standards. It might be well the case that certain requirements merely concern particular aspects of *things* (entities). Moreover, certain predicates/properties that are relevant for specifying a domain might not be provided by Freebase.

Last but not least, objects should be represented much more fine grained as it is the case in Freebase.

All these deficits bring along that Freebase can serve only as a starting point for describing a particular world or ontology, never satisfying exhaustively all demands. The fact that anything can be declared to be a subject, a predicate, or an object, thus is only a make-believe flexibility which in many cases turns out to be an arbitrary assumption based on a linguistic structure.

In other words, this triple relation is a syntactic norm that cannot fulfill the manifold semantic and conceptual demands of a micro-domain.

This is not the place to question the RDF structure on the whole, however, it might be quite useful to complete the suggested repertoire of concept and property types by means of a flexible, text-based linguistic process.

4. THE LINGUISTIC WORKFLOW

Starting point for the linguistic procedure is the textual description of a domain or a so called mini-world with respect to those criteria that according to the domain expert are crucial. For that purpose, an expert is provided with those RDF statements (triples) that Freebase offers for the respective domain, in order to evaluate them. In this process, relevance is hierarchized with respect to importance, implausible statements are deleted, new subjects are introduced, missing association types (properties) added and objects are made concrete, unified, or disambiguated, if necessary.

Problems like missing distinctiveness on the one hand or too big a distance concerning the meaning of concepts on the other hand which might arise in that context are solved with the help of common methods of similarity calculation. As for the calculation of similarity between concept terms (commonly nouns) used as subjects or objects, the Lesk algorithm is operated.

For the calculation of the similarity of association types (properties, in our case primarily relational, say bivalent verbs) a strategy is employed that would best be called *lexicon-based method of term-list comparison* [3].

In a further step, the existing concepts are visualized, and tree structures are substituted by graphs, which are closer to reality.

As for labeling, only those terms are used, whose linguistic category can assume a subject, predicate, or object function also in natural language [2]. This reality driven approach eventually facilitates a user friendly modeling of mini-worlds.

In the following section, the tool developed for this purpose is described in detail.

5. THE SEMANTIC GRAPH TOOL (SGT)

The Semantic Graph Tool (SGT) is a tool developed especially for the administration and visualization of relations between semantic concepts. As a result of its compact representation it is possible to manage ontologies by visualizing only a small section in variable dimensions.

Visualization in SGT is done by means of a radial graph with a central node and a circular arrangement of related nodes, where each level is represented on a circle. Thus, the farer the distance to the central node, the larger the circle. Relations are visualized by edges between the nodes, resulting in a tree with a central root element. Readability is enhanced by the fact that concepts are represented by circles, and association types by asterisks. Since semantic relationships comprise unidirectional and bidirectional association types, the edges for bidirectional types are represented by lines whereas unidirectional types are given by lines with an arrow pointing at the respective direction. A clear advantage over conventional ontology visualizations is the merger of associations over identical or identically directed association types based on one concept. That way, an association type for a central node is connected by one single edge, even if it is used for several associations. That is to say, only the edges between the association type and the associated concepts are split.

An association type node can thus have only one entering edge, but several outgoing edges (meaning the direction from the root to the leaves, not the direction of the edge itself). Such a node can be represented twice on one level, say as an applied association type of a concept, if it is unidirectional and if employed in both directions.

As a consequence, an *is-father-of* association type for instance can be used both coming from a concept A (*A is-father-of B*) and entering a concept A (C is-father-of A). This reduction of edges and nodes brings forward a clear arrangement of the represented part-ontology.

5.1 The workflow and its functionality

In the following, some of the different operations and functions of the Semantic Graph Tools are explained more in detail, including the respective illustrations.

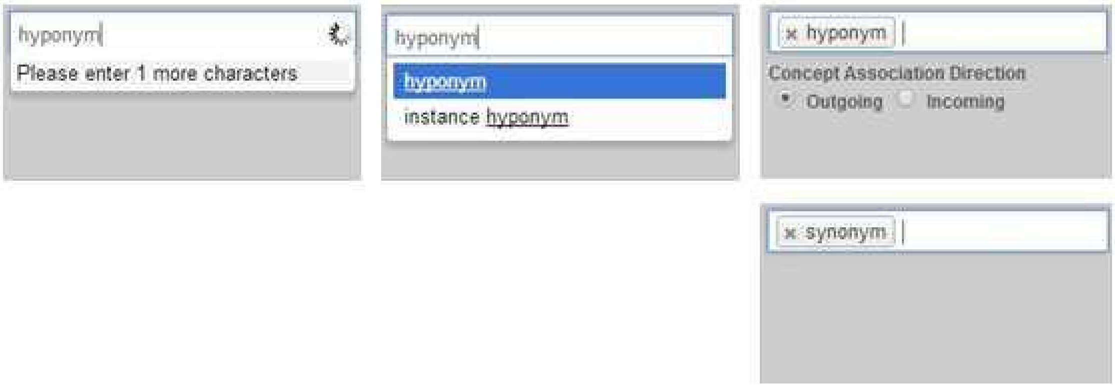
5.1.1 Creating associations

For creating a new association, an association type is required. If the latter does not yet exist, it has first to be linked manually to a concept. This is a preliminary step which does not by itself create a valuable association. The actual creation occurs by adding a concept to an association type, since that way an association is being completed and taken into the data base.

Adding an association type to a concept can be done with the help of the query window on top right of the SGT, or by using the [+]-symbol of the desired concept. However, none of the two alternatives performs a change in the data base, since the association has first to be completed by an associated concept. As a consequence, on reloading the page, an association type which is a leaf node, i.e. an incomplete association, will not be displayed, and thus the association type has to be added to the desired concept in the tree according to requirements.

Using the Select2 search box on the right, first a search term has to be entered, whereupon eligible results are displayed. If the chosen association type is unidirectional, a radio button allows for selecting an incoming or outgoing direction of association.

The search procedure is illustrated by a series of screenshots in Fig. 1 [Search process of an association type (first line unidirectional, second line bidirectional].



hyponym

Please enter 1 more characters

hyponym

hyponym

instance hyponym

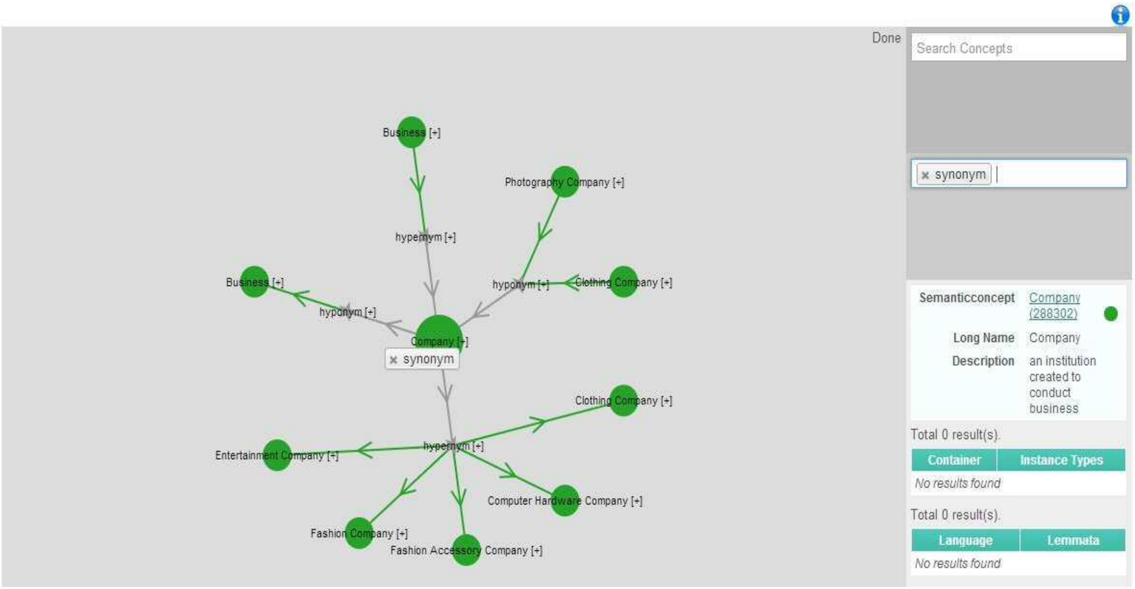
× hyponym

Concept Association Direction

● Outgoing ○ Incoming

× synonym

Fig. 1: Search process of an association type (first line unidirectional, second line bidirectional)



Search Concepts

× synonym

Semanticconcept	Company (288302)
Long Name	Company
Description	an institution created to conduct business

Total 0 result(s).

Container | Instance Types

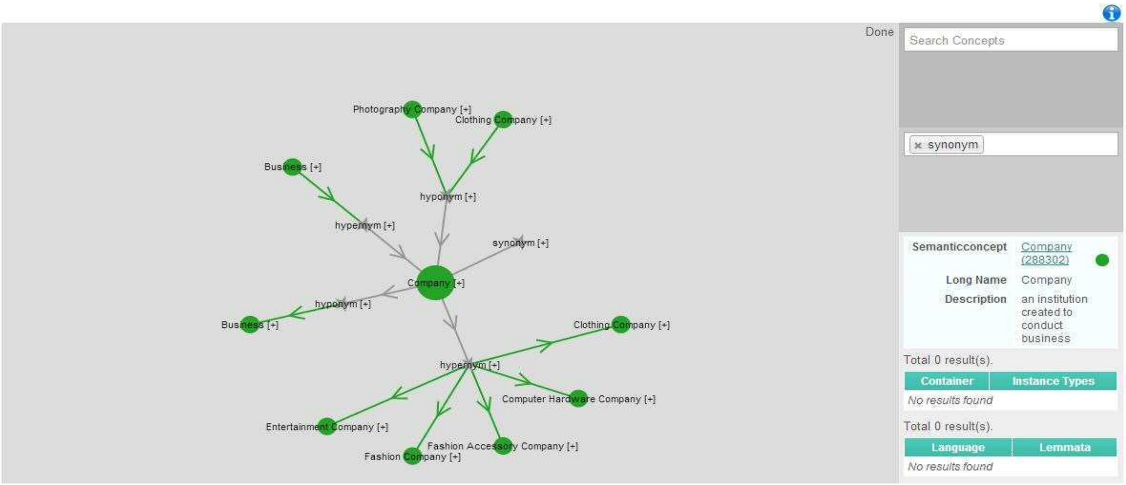
No results found

Total 0 result(s).

Language | Lemmata

No results found

Fig. 2: Dragging an association type from a search result to a concept in the tree



Search Concepts

× synonym

Semanticconcept	Company (288302)
Long Name	Company
Description	an institution created to conduct business

Total 0 result(s).

Container | Instance Types

No results found

Total 0 result(s).

Language | Lemmata

No results found

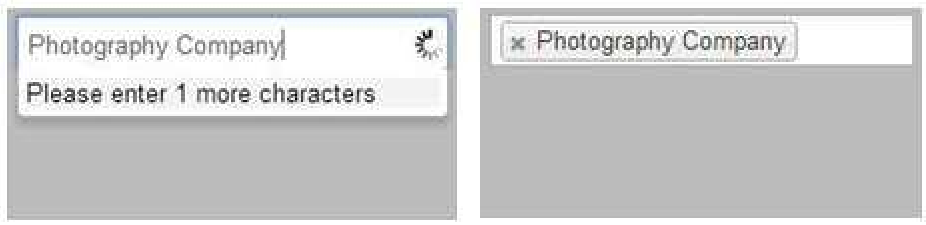
Fig. 3: Association type added by dropping to a concept

In the following, the process of adding a concept to an association type is explained. Only by accomplishing this step an association is completed and stored into the data base. As for this operation, there are three possible ways to go:

1) Searching a concept in the respective window at the right of the SGT with drag & drop confirmation on a potential association type in the graph.
2) Clicking the [+]-symbol of an association type after searching in the hereby opening pop up window and confirming the selected concept by pressing the button *Add Node*.
3) Duplicating an already existing concept in a tree by drag & drop and simultaneously pressing Strg / Ctrl.

The new association appears green if created by an expert, whereas it is white if drawn by a normal user in order to turn green after being confirmed by an expert.

With reference to 1), fig. 4 [Searching a concept] shows the Select2 search box already introduced when discussing the look-up of association types. Hereby, on entering a search term the matching items are shown so that desired one can be selected.



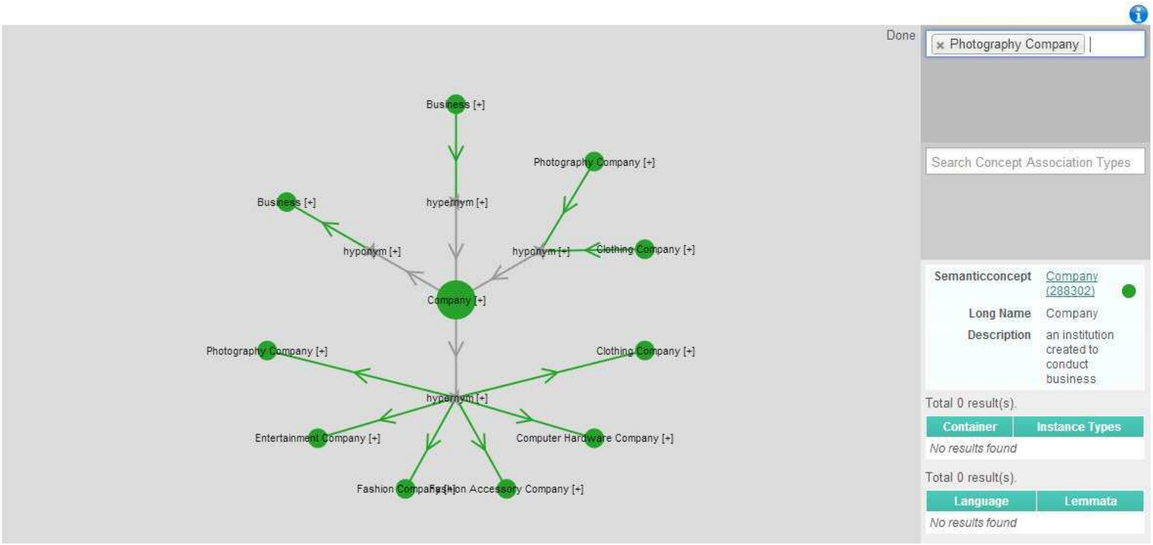
Photography Company

Please enter 1 more characters

× Photography Company

Fig. 4: Searching a concept

After successful selection (see right window of fig. 4 [Searching a concept]) this element can be dragged to the potential target node in the graph and dropped as an association type. This association type is only a possible target if the herby resulting association does not already exist. Dropping the concept on an association type and the resulting association (that *company* is hyperonym of *photography company*) is shown in fig. 5 [Dropping a concept on an association type and the resulting association (Company → Hypernym → Photography Company)]. As it is the case with association types, here also the last searched concept remains in the search box to be used again if needed.



× Photography Company

Search Concept Association Types

Semanticconcept: Company (288302)

Long Name: Company

Description: an institution created to conduct business

Total 0 result(s).

Container	Instance Types
No results found	

Total 0 result(s).

Language	Lemmata
No results found	

Fig. 5: Dropping a concept on an association type and the resulting association
(Company → Hyperonym → Photography Company)

A further option for creating new associations is also based on the user friendly drag & drop, like option 1). In addition to that, already used concepts in the tree are taken advantage of. This is again an enhancement of userfriendliness, given that within the scope of one concept, quite often concepts have more associations through different association types. Thus, it is much more comfortable for the user not having to search again for concepts, but to copy them directly within the tree. Fig. 6 [Copying a concept to create a new association (a-d)] illustrates the graphical procedure of this operation. Starting point in any state is 6a. Here, a leaf node in the form of a concept is dragged to a potential target node in the form of an association type (6b). When the concept is dropped onto the target node (6c), a new association is created and an animation in the tree shows that the concept in question is being duplicated (6d).

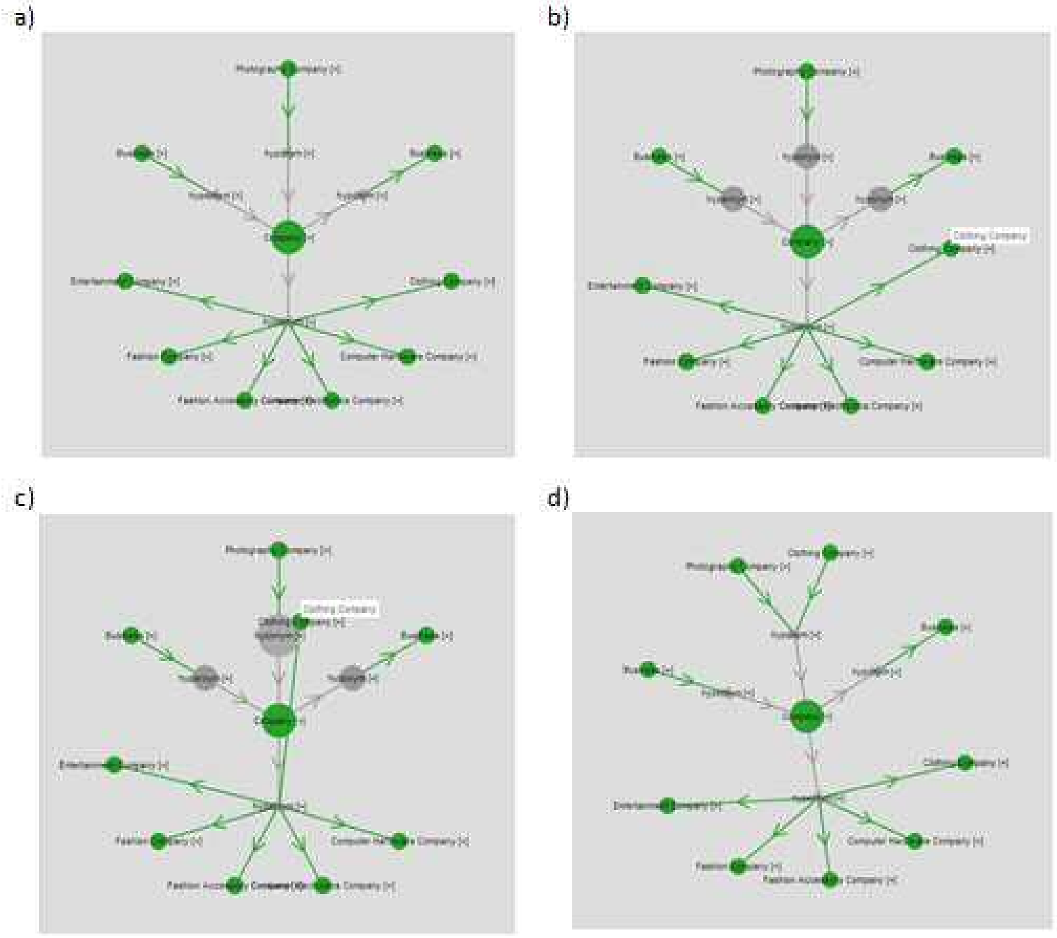


Fig. 6: Copying a concept to create a new association (a-d)

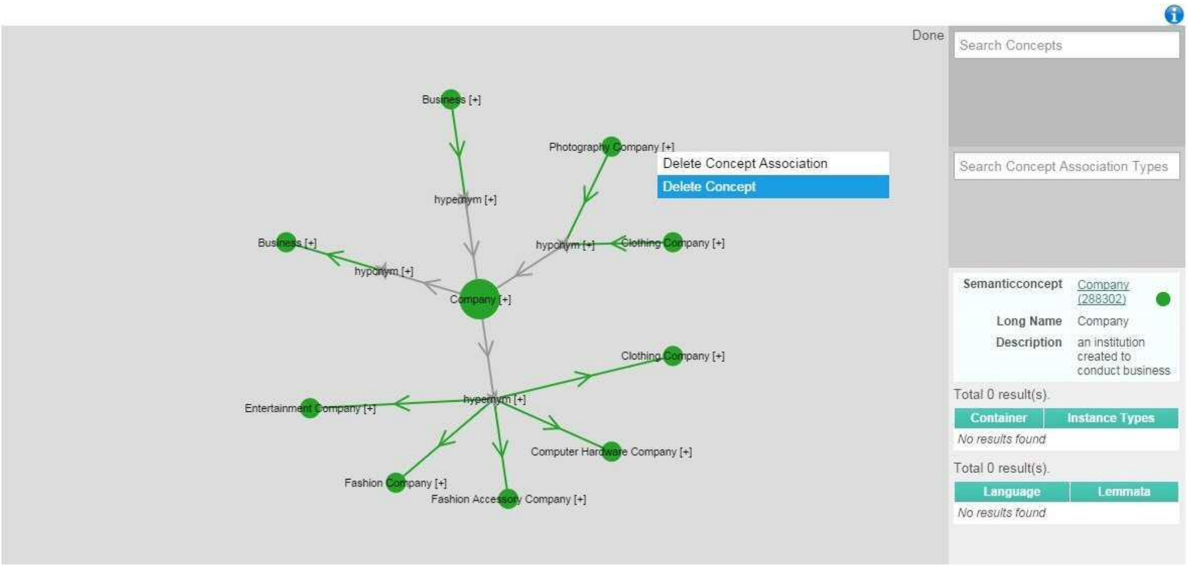
5.1.2 Changing existing associations

Changing existing associations is performed by simply reallocating concepts as leaf nodes by drag & drop. This is quite similar to the creation of new associations by copying existing concepts, with the difference that when copying, the Strg/Ctrl button has to be pressed during the transfer, which is not required when reallocating a concept.

When changing existing associations it is important to notice that even if in the database only one association is concerned, in its visualization the association in the case of unidirectional association types (with directed edges) can be displayed twice, so that two displayed edges might change.

5.1.3 Deleting associations and concepts

Deletion of associations and concepts can be performed with the help of a context menu which appears by right-clicking a node. Possible operations within the context menu depend on the chosen node and its state, since operations on nodes and edges that are already marked for deletion cannot be carried out (according to the definition of LingRep[2] logging). By right-clicking a concept, either the concept itself, or the association that is linked to it can be deleted. If the concept is deleted by a normal user, its state turns to red, and no more operations on this node will be possible. If however an expert deletes the concept, the tree is reloaded in order to suppress the concept and also the affected associations. The function *Delete Concept* is illustrated in fig. 7 [Deleting a concept with the help of the context menu].



Semanticconcept Company (288302)
Long Name Company
Description an institution created to conduct business

Total 0 result(s).

Container	Instance Types
No results found	

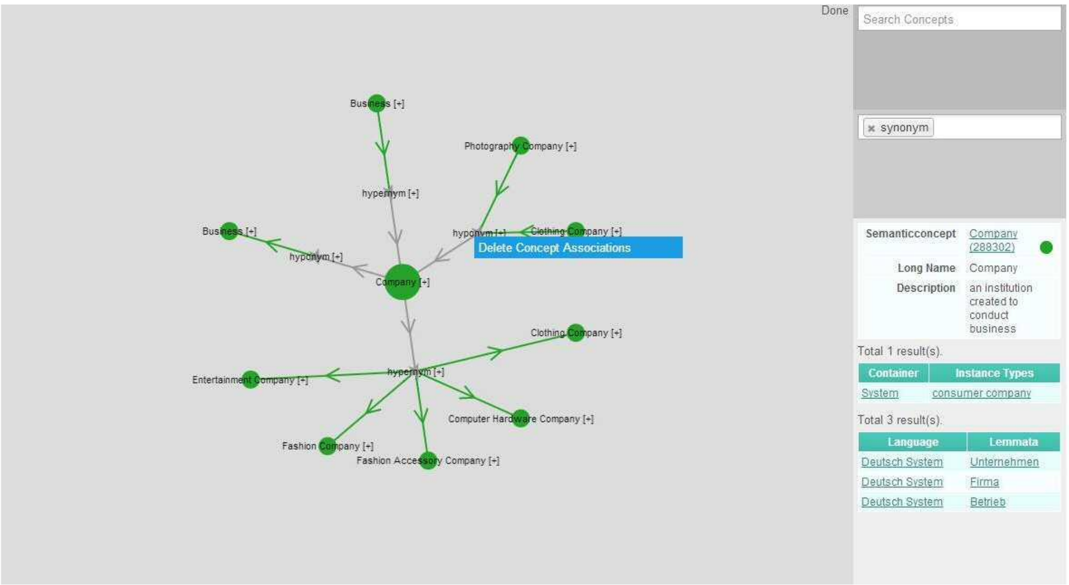
Total 0 result(s).

Language	Lemmata
No results found	

Fig. 7: Deleting a concept with the help of the context menu

Associations can be deleted or marked for deletion either one by one or collectively. Deleting a single association can be handled via the leaf node *Concept*. Herby, in the case of normal users, the association regarding the chosen concept is marked through its parents and grand-parents node. In the case of an expert, the node is deleted and thus suppressed immediately. Fig. 8 [Deleting several associations with the help of the context menu] shows the more powerful variant of the delete function of associations, which allows for marking or deleting all associations between a concept as a parent node and an association type as the selected node. Carried out by a normal user, the associations concerned are marked by a red edge, if done by an expert, the affected subtree is deleted as a whole.

[2] http://lingrep.com/index.php?r=overview/index



Semanticconcept	Company (288302) ●
Long Name	Company
Description	an institution created to conduct business

Total 1 result(s).

Container	Instance Types
System	consumer company

Total 3 result(s).

Language	Lemmata
Deutsch System	Unternehmen
Deutsch System	Firma
Deutsch System	Betrieb

Fig. 8: Deleting several associations with the help of the context menu

In both variants of deleting associations it has to be noticed that due to the possibly doubled visualization of a unidirectional association, its inversion can lead to the deletion or marking of associations shown as inverted.

6. SUMMARY

The present paper reports on a project on the integration of Freebase data into domain ontologies with the help of SGT. Generating an exemplary domain ontology requires several steps to be carried out, such as filling gaps, resolving contradictions, explaining parts of concepts with respect to requirement concepts of selected domains.

Problems like missing distinctiveness or too big a distance concerning the meaning of concepts are solved with the help of common methods of similarity calculation. Hereby, calculating the similarity between concept terms is operated by means of the Lesk algorithm. As for the calculation of the similarity of association types, the *lexicon-based method of term-list comparison* is employed.

In a further step, the existing concepts are visualized, and tree structures are substituted by graphs, which are closer to reality, thus guaranteeing a user friendly environment.

References:

[1] Jürgen Vöhringer, Günther Fliedl (2011). *Adapting the Lesk Algorithm for Calculating Term Similarity in the Context of Ontology Engineering*. Proceedings of ISD2010 Conference.

[2] Günther Fliedl, Christian Kop, Jürgen Vöhringer (2010). *Guideline based evaluation and verbalization of OWL class and property labels.* Data & Knowledge Engineering 04/2010; 69:331-342.

[3] N.N., Günther Fliedl, Christian Winkler. *A lexicon-based method of term-list comparison* (work in progress).

[4] Marta Sabou, Chris Wroe, Carole Goble, Heiner Stuckenschmidt (2005). *Learning domain ontologies for semantic web service descriptions*. Web Semantics: Science, Services and Agents on the World Wide Web, 3(4), 340-365. http://ac.els-cdn.com/S1570826805000296/1-s2.0-S1570826805000296-main.pdf?_tid=b4e8210e-f3d1-11e4-a13f-00000aab0f02&acdnat=1430904453_589dd7f304c610c8ab5bfdd28b01bd9f

[5] Günther Fliedl, Christian Kop, Heinrich C. Mayr, Georg Weber, Christian Winkler (2004) *Linguistic Analysis of Unstructured Text by Chunk-Parsing, Tagging, and Semantic Interpretation.* ICSSEA 2004.

[6] Günther Fliedl, Georg Weber, Christian Winkler (2005)
A morphosemantic tagger for German and the structure of its lexicon components. ICSSEA 2005.

[7] Manuel Warum, Horst A. Kandutsch, Günther Fliedl, Christian Winkler (2010) *Optimizing Software by Linguistically Enhanced Comparisons of Requirement Documents and Source Code Graphs.* ICSSEA 20010.

[8] Roberto Navigli, Paola Velardi (2004). *Learning domain ontologies from document warehouses and dedicated web sites*. Computational Linguistics, 30(2), 151-179.
http://www.mitpressjournals.org/doi/pdf/10.1162/089120104323093276

[9] Marvan Sabbouh, Joseph DeRosa. *Using Semantic Web Technologies to Integrate the Enterprise.*
http://www.mitre.org/sites/default/files/pdf/derosa_semantic.pdf

Freebase
https://www.freebase.com/
http://freebase-easy.cs.uni-freiburg.de/browse/

Google Knowledge Graph
http://www.google.com/insidesearch/features/search/knowledge.html
https://plus.google.com/109936836907132434202/posts/bu3z2wVqcQc

LingRep: http://lingrep.com/index.php?r=overview/index

Econob: http://www.econob.com/index.php/en/