Exploiting Freebase to Obtain GoodRelations-based Product Ontologies

Marek Dudáš, Ondřej Zamazal, Jindřich Mynarz, and Vojtěch Svátek

University of Economics, Prague,
xdudm12|ondrej.zamazal|jindrich.mynarz|svatek@vse.cz

Abstract. Application of semantic web technologies in e-commerce depends on the availability of product ontologies. However, such ontologies are not yet available for many industries and developing them from scratch is costly. We present a method of reusing parts of the Freebase schema by transforming it into a GoodRelations-based product ontology, using our Pattern-based Ontology Transformation Framework. We demonstrate our method on a part of the Freebase Medicine schema, which we transformed into a product ontology covering prescription drugs.

Key words: ontology development, ontology transformation, non-ontological resources, GoodRelations, product ontology, Freebase

1 Introduction

It can be expected that a better level of automation in e-commerce can be achieved by annotating product and service data with RDF. Appropriate ontologies or vocabularies, typically expressed in the standard OWL language,[1] are needed for that. GoodRelations (GR) [1] is an ontology covering the relationships between a possible seller, buyer and the product/service offered.[2] GR is supposed to serve as a basis for specific product ontologies that can be created as its vertical extensions.[3] One of the advantages of this approach is that the core structure of a product ontology does not have to be developed from scratch. Another, probably more important, advantage is that if several product ontologies covering different products share the GR basis they can be integrated into an application much easier than if each was created using different modeling style. A software tool working with data annotated with GR-based product ontologies can benefit from the unified modeling style: their code can be simpler when the developers can rely on the fact that all ontologies have the same basic structure (this is explained in more detail in Section 2.1). As the GR itself is already quite widely adopted [2], GR-based product ontologies and data annotated with them

[1] http://www.w3.org/TR/owl2-overview/

[2] There is also Schema.org, which has now GR basically integrated as its e-commerce core.

[3] In this paper we call such ontologies "GR-based."

can also be expected to have a better chance of adoption or exploitation than proprietary product ontologies and data. The problem is that such ontologies are not yet available for many industries; it seems that their only reliable sources are currently the 'specific industries' section of the GR cookbook[4] and the ontology collection developed in the OPDM project.[5] Developing a new product ontology from scratch might be very time consuming. Transforming shallow non-ontological models into OWL only saves a fraction of the effort, even if leveraging a specialized framework such as that from [3]. Yet, there are already knowledge models that are both thematically suitable and exhibit a relatively rich structure, and thus are worth considering for this purpose. However, since they do not comply with all Semantic Web best practices, never mind assuring compatibility with pivotal vocabularies such as GoodRelations, a certain kind of transformation is still needed. An outstanding example of this resource is the Freebase schema [4]. In this paper we argue that it is possible to obtain a GR-based product ontology from a part of Freebase with the help of our pattern-based ontology transformation framework [5]; this task can be viewed as a solution of *import-based extrinsic incoherence* ('adapting legacy ontologies to a canonical modeling style') as mentioned in [6]. Specifically, we demonstrate how a part of the Freebase Medicine schema can be transformed into a GR-based prescription drug ontology (the domain was chosen because drugs are usually well-documented in public contracts). The resulting ontology was evaluated by experimental usage in another project, involving matchmaking of public contracts with suppliers [7], where we hope to improve the matchmaking by exploiting product ontologies.

Section 2 contains a summary of principles of GR-based product ontologies and the employed transformation framework. In Section 3 we explain how the transformation has been performed. Section 4 describes the evaluation of the ontology. Section 5 surveys related research, and Section 6 concludes the paper.

2 Exploited Technologies and Resources

2.1 GoodRelations-based Product Ontologies

Available guidelines for the design of a GR-based product ontology[6] prescribe, among other things, the following:

- all product classes should be subclasses of the *gr:ProductOrService* class
- object properties for specifying product parameters should be subproperties of either *gr:quantitative-* or *gr:qualitativeProductOrServiceProperty*
- subproperties of *gr:quantitativeProductOrServiceProperty* should have range specified to gr:QuantitativeValueFloat or gr:QuantitativeValueInteger
- data properties should be subproperties of *gr:datatypeProductOrServiceProperty* and used only if there is a good reason; even properties with numerical range should be modeled as object properties.

[4] `http://wiki.goodrelations-vocabulary.org/Cookbook`
[5] `http://purl.org/opdm`
[6] `http://wiki.goodrelations-vocabulary.org/Documentation/Extensions`

The task of turning a generic ontology covering a product and its properties into a GR-based product ontology is thus mainly in identifying relevant classes and properties, making them subclasses and subproperties of appropriate GR entities, and modifying their range so that the rules such as the above-mentioned are fulfilled.

Such alignment of a product ontology with GR allows for example an easy design of an automatic GUI generator. GUI elements for entering property values can be created automatically according to the parent property: a selection of possible values for subproperties of *gr:qualitativeProductOrServiceProperty*, a number field for subproperties of *gr:quantitativeProductOrServiceProperty*. The latter can also be accompanied with a field for measurement units as there is already a property for specifying those integrated in GR. GR also allows to assign an interval instead of an exact value to a quantitative property. We are developing such GUI generator for an application that will allow entering product specification data to a public contract, but a similar GUI generation principles could be used for e.g. a product search application. In the matchmaking scenario, the property structure can be exploited in comparing product specifications. E.g. different values of a qualitative property indicates different products while a slight difference between two quantitative values might mean that the two products are interchangeable.

2.2 Transformation Framework

The involved transformation framework can be viewed as a more sophisticated variant of using SPARQL updates. It is based on transformation patterns (expressed in XML[7]) consisting of three parts: source ontology pattern, target ontology pattern and pattern transformation. The source pattern describes a fragment that is to be found in the ontology and transformed into the fragment described by the target pattern. The pattern transformation describes how the two fragments are connected. Based on this information the transformation framework computes and applies the necessary ontology updates: axioms and entities are either added or removed. Unlike SPARQL update, however, the transformation can be user-assisted. After the first phase of the transformation, the occurrences (instances) of the source pattern are displayed to the user and it is possible to select only a subset of the instances that are then transformed while the unselected rest are kept intact. We employed this mechanism to select classes representing GR product classes and qualitative classes based on manual analysis of the schema, as we currently do not have an automated way of determining which classes in the ontology have such a meaning.

The framework is available as a Java library.[8] To make its application easier, we developed a graphical user interface, called GUIPOT.

[7] The schema for the files is available at `http://nb.vse.cz/~svabo/patomat/tp/tp-schema.xsd`

[8] `http://owl.vse.cz:8080/releases.html`

2.3 GUIPOT

GUIPOT is a plugin for Protégé 4.[9] It allows the user to load a transformation pattern and display a list of pattern instances detected in the given ontology. If one or more instances from the list are selected, they are highlighted in a classical Protégé hierarchy view and also visualized in a node-link view on the left part of the plugin window (Figure 1). The visualization can be switched to four different levels of detail. The default one uses OntoGraf[10] and displays classes as nodes with properties as links between them (according to domain/range relationships). A more detailed view where properties and complex relationships between classes (e.g. complementOf) are displayed as separate nodes is implemented using SOVA[11] visualization plugin. SOVA is also used for less detailed overviews of class hierarchy where the class nodes are laid out with spring layout or tree layout algorithms. The detailed views show fragments of the ontology involved in the selected pattern instances. The overviews show the whole ontology with entities involved in the selected pattern instances highlighted.

When the user is satisfied with the selection, s/he just clicks the "Apply Transformation" button. The right part of the window shows the ontology after transformation, with affected entities indicated by red arrows and highlighted or focused in the visualization (analogically to the visualization of selected pattern instances). The transformed ontology can then be loaded into Protégé with a single click and then either edited manually or transformed with another pattern.

3 Transforming the Freebase Schema

3.1 Overview of the Process

The translation of Freebase Schema into RDF is already solved, since Google offers a Freebase RDF dump.[12] To obtain a GR-based product ontology from the dump, we have to select relevant classes and properties and align their modelling style with that indicated in Section 2.1. This can be done in a semi-automated way using the transformation framework in GUIPOT.

Simple pre-processing was needed before the Freebase schema could be loaded as an ontology into Protégé. As the dump is too big (>200GB), we filtered the relevant part of the file using the Linux grep command (exploiting the naming conventions of Freebase – all entities related to one topic have the same prefix in their name). Furthermore, we had to retype the instances of *rdf:Property* to *owl:ObjectProperty* (since all Freebase properties are, syntactically, conceived as object properties, possibly valued by a custom class of, e.g., 'all integers'), using a simple SPARQL update query.

[9] `http://owl.vse.cz:8080/GUIPOT/`
[10] `http://protegewiki.stanford.edu/wiki/OntoGraf`
[11] `http://protegewiki.stanford.edu/wiki/SOVA`
[12] `https://developers.google.com/freebase/data`

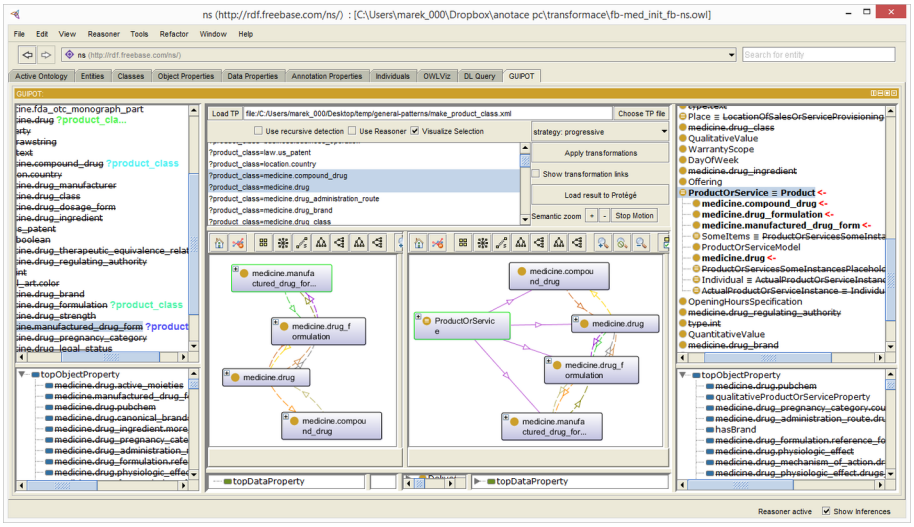

Fig. 1. A screenshot of GUIPOT where the transformation pattern for making selected classes subclasses of gr:ProductOrService is being applied: the left part shows selected instances of the pattern, the right part shows the transformed ontology.

We designed 14 transformation patterns[13] for aligning the modelling style of an ontology obtained from Freebase. When they are applied sequentially, they transform the preprocessed (as described above) ontology obtained from Freebase into a GR-based product ontology. The patterns are designed to be reusable on other parts of Freebase schema than the one described in this paper: they should allow the transformation of virtually any Freebase schema part related to a selected product into a GR-based product ontology in a few hours (including the initial schema analysis and the above described preprocessing).

The whole transformation process is shown in Fig. 2. The resulting Drug ontology (DON) has 27 classes, 34 object properties, 17 data properties and 6 individuals, altogether 84 entities described with 715 axioms.

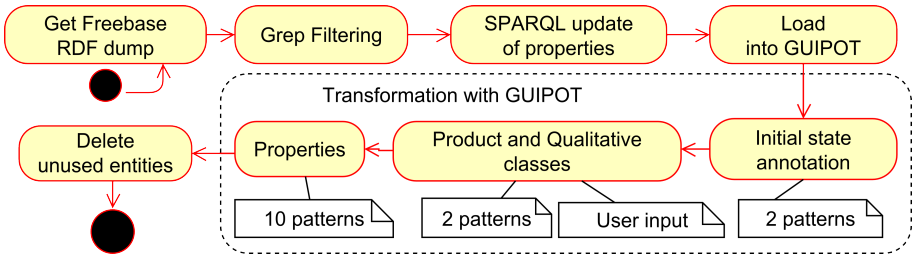

Fig. 2. Diagram of the transformation process

[13] All patterns, the transformed ontology and details about the evaluation can be found at `http://pages.vse.cz/~xdudm12/ecweb2014/` (or `http://bit.ly/1eEP3yL`).

3.2 The Transformation Patterns

The first two patterns are 'helper' patterns used to annotate the initial state of the ontology. The annotation is removed from the entities as they are transformed, which allows leaving them out of further transformations and also to delete entities that have not been touched by any transformation at the end of the process; they are not supposed to be part of the product ontology as they have not been selected by any transformation.

The next two patterns require user assistance. A list of all classes is presented to the user and s/he decides which of them are to be transformed into subclasses of *gr:ProductOrService* and *gr:QualitativeValue*.

The application of the fifth pattern leads to transformation of qualitative properties to subproperties of gr:qualitativeProductOrServiceProperty.

The sixth pattern is used to transform qualitative properties in a similar way as the previous pattern. The difference is that in this case, properties with range of a subclass of gr:ProductOrService are transformed. Only two of them are selected, the rest of them represent inverse properties (e.g. "drugs with this formulation" linking a formulation to a drug when there is already a property linking a drug to a formulation) and are not included into the final product ontology to keep it lightweight.

The remaining patterns are designed to transform properties according to their range. They do not require any selection of the pattern instances: all detected instances are transformed. The range is changed from the custom Freebase class (like *type.int* or *type.text*) to either an appropriate GR class (e.g. *gr:QuantitativeValueInteger*) or a datatype. In the latter case, the object property is transformed into a datatype property.

Three of these patterns are more complex and thus are discussed in more detail. Pattern no. 7 finds properties whose range is a class that is a domain for two other properties with ranges of Freebase classes *type.text* and *type.float* where the text property is used to specify units of the value specified by the float property. The same situation is already modeled in GoodRelations and the additional two properties are unnecessary. The former property is transformed into a subproperty of *gr:quantitativeProductOrServiceProperty* with range of *gr:QuantitativeValueFloat* and the other two properties can be removed at the end of the transformation process. Pattern no. 8 transforms properties with range of *type.boolean* into instances of a newly created class *Feature* (subclass of *gr:QualitativeValue*). A property *hasFeature* is created with range set to *Feature*. (Follows modeling style used in Vehicle Sales Ontology[14] – a model example of a GR-based product ontology.) Pattern no. 10 is designed for properties with range of *type.enumeration*: they are made subproperties of *gr:qualitativeProductOrServiceProperty* and their range is changed to a newly created subclass of *gr:QualitativeValue*. A new datatype property is also created with that class as a domain. The intention is to make the values of the property dereferencable. A summary of all transformation patterns is shown in Table 1.

[14] `http://purl.org/vso`

Table 1. Simplified overview describing what entities are transformed by each pattern identified by its number, the last column specifies whether the user has to select the transformed entities manually. Abbreviations are used: 'float' object property means object property with range of type.float (a Freebase class), gr:qualit...Property means gr:qualitativeProductOrServiceProperty etc.

#	Transformed entities	Transformed into	U
1	classes	original state annotation	
2	object properties	original state annotation	
3	selected classes	subclasses of gr:ProductOrService	x
4	selected classes	subclasses of gr:QualitativeValue	x
5	object properties with range of gr:QualitativeClass subclass	subproperties of gr:qualit...Property	
6	object properties with range of gr:ProductOrService subclass	subproperties of gr:qualit...Property	x
7	'float' object properties with unit specification	subproperties of gr:quan...Property with range of gr:QuantitativeValueFloat	
8	'boolean' object properties	instances of class Feature	
9	'datetime' object properties	datatype subproperties of gr:data...Property	
10	'enumeration' object properties	subproperties of gr:qualit...Property	
11	'float' object properties	subproperties of gr:quan...Property with range of gr:QuantitativeValueFloat	
12	'int' object properties	subproperties of gr:quan...Property with range of gr:QuantitativeValueInteger	
13	'rawstring' object properties	datatype subproperties of gr:data...Property	
14	'text' object properties	datatype subproperties of gr:data...Property	

3.3 Example of a Transformation Pattern – Pattern No. 5 in More Detail

The *axioms* part of the source ontology pattern (Figure 3) describes the axioms which the entities and placeholders from *entity_declarations* must fulfill. After the pattern is loaded, the ontology is searched for combinations of entities that when used to instantiate the placeholders hold the axioms. The placeholder *?m* represents the qualitative object properties we are looking for. The first two axioms ensures that these properties have the range of a subclass of *gr:QualitativeValue*. Here we exploit the fact that we have already transformed relevant classes into subclasses of *gr:QualitativeValue* with the previous transformation pattern. The third axiom ensures that only the properties that were not yet transformed are taken into account.

The *pattern transformation* (<pt>) part of the target ontology pattern (Figure 4) defines the connection between the source and target ontology pattern. The placeholders are instantiated with the selected instances of the source ontology pattern. Axioms defined in the target ontology pattern (<op2>) are

```
<entity_declarations>
     <placeholder type="ObjectProperty">?m</placeholder>
     <placeholder type="Class">?range_class</placeholder>
     <entity type="Class">&gr;QualitativeValue</entity>
     <entity type="AnnotationProperty">&owl;deprecated</entity>
</entity_declarations>
<axioms>
     <axiom>ObjectProperty: ?m Range: ?range_class</axiom>
     <axiom>Class: ?range_class SubClassOf: QualitativeValue</axiom>
     <axiom>ObjectProperty: ?m Annotations: deprecated true</axiom>
</axioms>
```

Fig. 3. Source ontology pattern part of the pattern for transformation of qualitative properties.

```
<op2>
    <entity_declarations>
        <placeholder type="ObjectProperty">?OP2_m</placeholder>
        <placeholder type="Class">?OP2_range_class</placeholder>
        <entity type="ObjectProperty">
            &gr;qualitativeProductOrServiceProperty</entity>
        <entity type="Class">&gr;QualitativeValue</entity>
        <entity type="AnnotationProperty">&owl;deprecated</entity>
    </entity_declarations>
    <axioms>
        <axiom>ObjectProperty: ?OP2_m SubPropertyOf:
            qualitativeProductOrServiceProperty</axiom>
        <axiom>Class: ?OP2_range_class SubClassOf:
            QualitativeValue</axiom>
        <axiom>ObjectProperty: ?OP2_m Range:
            ?OP2_range_class</axiom>
        <axiom>ObjectProperty: ?OP2_m Annotations:
            deprecated false</axiom>
    </axioms>
</op2>
<pt>
    <eq op1="?m" op2="?OP2_m"/>
    <eq op1="?range_class" op2="?OP2_range_class" />
</pt>
```

Fig. 4. Target ontology pattern (op2) and pattern transformation part (pt) of the pattern for transformation of qualitative properties.

added into the ontology. The most important is the first axiom which makes the qualitative properties found using the source ontology pattern subproperties of *gr:qualitativeProductOrServiceProperty*. The second and third axioms are there only to keep the range class intact: the standard behaviour of the transformation framework is to remove axioms defined in the source ontology pattern from the ontology. The fourth axiom disables the deprecated annotation on the transformed object property.

3.4 Use Case Lessons Learned for the Transformation Framework

Keeping the Description of the Entity During Its Transformation When the type of the entity is not changed during its transformation, its annotations (labels, comments etc.) and other axioms describing it are kept intact. In case of *heterogeneous* transformation, when the entity is transformed into an entity of different type, e.g. in case of the pattern no. 8 where an object property is transformed into an instance, all annotations and other axioms related to that entity and not specified in the transformation pattern are removed from the ontology. That might be unpleasant – e.g. the label of a property may be still valid even though the property is transformed into an instance. The only current solution is to explicitly define all annotations and axioms that are to be kept into the transformation pattern. This is not a big issue in case of this Freebase schema transformation as most of the entities that are transformed into different types do not have any annotations. A possible future improvement of the transformation framework related to this issue is to automatically add annotations that would link the new (transformed) entity to the original one.

Instances So far, we have been focusing on the transformation of the schema (i.e. TBox). The transformation does not deal with existing Freebase instances (i.e. ABox), even though it would obviously be useful to transform the data along with the schema so that the Freebase instances might be reused together with the schema. Adding such functionality to the transformation framework should not be technically difficult and we are currently considering it.

Namespaces The transformation patterns do not change the prefix of transformed entities - their URI is kept intact. As the transformed entity is obviously not the same as the original entity, its URI should be different. This can be solved easily with Protégé where the namespace of all entities can be changed by one command.

4 Evaluation

4.1 Evaluation by usage

As the main motivation and purpose of the ontology is its use in our matchmaking project, we made a preliminary evaluation of its coverage by using it

experimentally. We manually annotated[15] 10 public contracts (selected as having extensive documentation) focusing on specifications of demanded and offered drugs included in publicly available plain-text documentation of the contracts. Some contracts included more than one drug and each contract had several contenders – 79 instances of *gr:Offering* were created. Only 4 classes and 3 properties from DON were used in the annotation, i.e. approx. 8% of DON was needed to annotate the drugs as described in the documentation. One property was identified as missing and had to be added to DON. Hence, we can say that 8 concepts were needed to annotate the product data (the 4 classes, 3 properties from DON and the one added property) of which 7 were covered by the ontology as extracted from FB, i.e. the ontology covered 87.5% of the specific domain.

4.2 Evaluation by a group of test users

While the previous evaluation was done by the authors of this paper (i.e. expert users), we decided to also test the user-friendliness of DON to lay users in the domain of drug supply public contracts (as the above mentioned 8% exploitation of concepts from DON might indicate it is too complex for our use case). 15 students with basic knowledge of OWL were given a brief (one hour) introduction to GR, DON and Ontowiki and a written manual describing its usage. Then they were asked to annotate one instance of a *gr:Offering* including a drug specified in some public contract (the selection was up to the students) using the same classes and properties as above. 3 students made no apparent mistakes; 4 others only made one mistake in unit specification (they were asked to use UN/CEFACT codes as GR documentation recommends) and were also considered as successful; i.e. about 47% of the students were able to use DON correctly. Furthermore, the mistakes were mostly in the usage of GR; only two errors were in incorrect usage of DON. Each student spent more than an hour creating the annotation, which was mainly caused by technical difficulties with Ontowiki (for this reason, precise time measurement did not make sense). Similar annotations done by an expert user after solving most of the difficulties took about 5–10 minutes each.

4.3 Evaluation against a similar ontology

We also measured similarity to a reference ontology by counting precision and recall measures based on [8]. We took part of the RDF representation of Schema.org describing the Drug concept as the reference ontology. To abstract from structural differences between Schema.org and DON, we limited the comparison to properties having as their domain the Schema.org Drug class and properties having as their domain one of the classes representing a drug in DON. The equivalence of the properties was evaluated according to the similarity of their meaning, as defined in their descriptions and comments. Recall was computed as the proportion of properties from the drug part of Schema.org (DSchema) that have an equivalent in DON:

[15] Using Ontowiki: `http://aksw.org/Projects/OntoWiki.html`

$$\frac{(DSchema \cap DON)}{DSchema} = 14/38 \approx 37\%$$

Precision is analogical – the proportion of DON properties that have an equivalent in the drug part of Schema.org :

$$\frac{(DON \cap DSchema)}{DON} = 22/37 \approx 59\%$$

The comparison suggests that basic properties are covered in both schemas. However, the relatively high values might be influenced by the involvement of Google in both Schema.org and Freebase.

5 Related Research

[3] presents guidelines and a pattern-based framework for re-engineering non-ontological resources (NOR), such as classification schemas, into OWL ontologies. The framework is very versatile and might have been even applicable to Freebase proprietary data format if its OWL representation wasn't already available. Our OWL-to-OWL transformation framework might still be useful for the refinement of an ontology produced by the NOR-to-OWL transformation.

A similar project targeted directly on product ontologies is PCS2OWL [9] – a framework for transformation of non-ontological product classifications such as eCl@ss[16] into GR-based product ontologies. Although such classifications sometimes include product properties descriptions and PCS2OWL should allow to transform even those, it focuses mainly on transformation of the product taxonomy. Freebase schema contains richer description of product properties (e.g. including their range) but there is virtually no information about the product class hierarchy. We thus believe that our approach could be combined with PCS2OWL, e.g. by enriching the class taxonomy created with PCS2OWL by properties extracted from Freebase.

A method for transformation of existing knowledge sources into conceptual data models is proposed in [10]. Although it focuses on rather opposite direction of the transformation and shows an example of transformation of an OWL ontology into a Power Designer conceptual model, it also mentions Freebase as a possible resource.

6 Conclusion and Future Work

We have shown that a GR-based product ontology can be obtained from Freebase by extracting a relevant portion of the Freebase RDF dump (using simple text-based filtering with grep) and transforming its modelling style using the pattern-based ontology transformation framework. The preliminary evaluation

[16] `http://www.eclass.de`

suggests that the resulting ontology is not perfect and might need minor manual refinement. However, the usage of the ontology in our specific use case has so far been successful, the ontology seems to be quite easy to use and its similarity to part of Schema.org suggests it covers most of the important concepts from its domain. We plan to do more thorough testing in the future including transformation of more product ontologies and running matchmaking algorithms (from [7]) over the data. We have already extracted mobile phone and digital camera product ontologies from Freebase but these ontologies are yet to be evaluated.

Acknowledgements This research is supported by VŠE IGA project F4/34/2014 (IG407024) and EU ICT FP7 under No. 257943 (LOD2 project). Ondřej Zamazal has been supported by the CSF grant No. 14-14076P.

References

1. Hepp, M.: Goodrelations: An ontology for describing products and services offers on the web. In: Knowledge Engineering: Practice and Patterns. Springer (2008) 329–346
2. Ashraf, J., Cyganiak, R., O'Riain, S., Hadzic, M.: Open ebusiness ontology usage: Investigating community implementation of goodrelations. In: LDOW. (2011)
3. Villazón-Terrazas, B., Gómez-Pérez, A.: Reusing and re-engineering non-ontological resources for building ontologies. In: Ontology Engineering in a Networked World. Springer (2012) 107–145
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM (2008) 1247–1250
5. Šváb-Zamazal, O., Svátek, V., Iannone, L.: Pattern-based ontology transformation service exploiting OPPL and OWL-API. In: Knowledge Engineering and Knowledge Management by the Masses. EKAW-2010. (2010)
6. Dudas, M., Svatek, V., Torok, L., Zamazal, O., Rodriguez-Castro, B., Hepp, M.: Semi-automated structural adaptation of advanced e-commerce ontologies. In: E-Commerce and Web Technologies. Springer Berlin Heidelberg (2013) 51–58
7. Nečaský, M., Klímek, J., Mynarz, J., Knap, T., Svátek, V., Stárka, J.: Linked data support for filing public contracts. Computers in Industry (2014)
8. Maedche, A.: Ontology learning for the semantic Web. Kluwer Academic Publishers (2002)
9. Stolz, A., Rodriguez-Castro, B., Radinger, A., Hepp, M.: Pcs2owl: A generic approach for deriving web ontologies from product classification systems. In: The Semantic Web: Trends and Challenges. Springer (2014) 644–658
10. Trinkunas, J., Vasilecas, O.: Ontology transformation: From requirements to conceptual model. Scientific Papers, University of Latvia, Computer Science and Information Technologies **751** (2009) 52–64