Building up Ontologies with Property Axioms from Wikipedia

Tokio Kawakami[1], Takeshi Morita[1], and Takahira Yamaguchi[1]

[1]Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223-8522, Japan
kawakami0412@keio.jp,{t_morita,yamaguti}@ae.keio.ac.jp

Abstract. Wikipedia has been recently drawing attention as a semistructured information resource for the automatic construction of ontology. We proposed a method of constructing a general-purpose ontology by automatically extracting the is-a relations (rdfs:subClassOf), class-instance relations (rdf:type), property relations and types. Methods to automatically construct ontologies with numerous property relations and types from Wikipedia are discussed here. The ontologies include is-a relations, class-instance relations and property relations and types. The property relations are triples, property domain (rdfs:domain), property range (rdfs:range), and property hypernymy-hyponymy relations (rdfs:subPropertyOf). The property types are object (owl:ObjectProperty), data (owl:DatatypeProperty), symmetric (owl:SymmetricProperty), transitive (owl:TransitiveProperty), functional (owl:FunctionalProperty), and inverse functional (owl:InverseFunctionalProperty).

Keywords: Ontology Learning · Property Axioms · Wikipedia

1 Introduction

It is effective to construct large-scale ontologies for information searching and data integration. Among popular ontologies are WordNet [1] and Cyc [2]. However, it is expensive to construct these ontologies manually. Moreover, the manual ontology engineering process results in numerous bugs, and its maintenance and update are challenging. Therefore, more attention comes to build the automatic or semi-automatic creation of ontologies on research, ontology learning.

Wikipedia, the web-based open encyclopedia, is increasing in popularity as a new information resource. Because Wikipedia has rich vocabulary, reasonable updatability, and semistructuredness, there is less differences between Wikipedia and ontologies when compared with free text. Thus, ontology learning from Wikipedia is becoming popular.

We proposed a large-scale and general-purpose ontology learning method for extracting the is-a relations (rdfs:subClassOf), class-instance (rdf:type), and property relations and types by using Wikipedia as the resources [3]. However, there are certain challenges. For example, we did not define the property domain, propety range and types. Therefore, we extract the property names (that are including following types: object, data, symmetric, transitive, functional, and

inverse functional) and relations (that are triples, property domain and property range) by adding certain techniques to current technique and construct a large-scale and general-purpose ontology including numerous properties.

This paper is structured as follows: We introduce related works on deriving ontology from Wikipedia in Section 2. In Section 3, we explain the definition of property as described in our previous research and details on extraction techniques applied to Wikipedia. In Section 4, we present the result of the experiment wherein we applied the extraction techniques to Wikipedia. Finally, we present the conclusion of this paper and our future work.

2 Related Work

Auer et al.'s DBpedia [4] constructed a large information database to extract RDF from semi-structured information resources of Wikipedia. They used information resources such as Infobox, external link, categories the article belongs to. However, properties and classes are constructed manually, and there are 170 classes and 720 properties. We use not only Infobox and Wikipedia categories but also text information such as list structures, list articles and definition texts and extract relations automatically. It is also different in that it is working on automatic extraction of property axioms.

Fabian et al. [5] proposed YAGO, which enhanced WordNet by using the Conceptual Category. Conceptual Category is a category of Wikipedia English, whose head part has the form of a plural noun suchas "American singers of German origin". They defined a Conceptual Category as a class and defined the articles that belong to the Conceptual Category as instances of the class. This method permits all articles using categories to be set as an instance. However, as the information in the main text is not used, in the cases where the article is absent or where information not reflected in the categories is present in the main text, it is unfeasible to extract such information as the instance.

YAGO2 [6] aimed for further expansion of ontology with the knowledge base expansion of YAGO both by the previous linkage between WordNet and Wikipedia category and by the extraction of spatiotemporal information from Wikipedia and GeoNames3. Moreover, in the expansion version, YAGO3 [7], Wikipedia (in other languages as well as English) is used to permit multilingual expansion. YAGO2 focused on nonhierarchical relations and constructed an advanced ontology, incorporating a spatiotemporal relation that is not based only on the hierarchical relation. However, it has not utilized Wikipedia's unique structural aspects such as information in the main text, define statements, and Wikipedia lists.

Sánchez [8] presented the approach which exploits the Web to extract evidences that sustain a particular property of a given relationship. Specifically, the axioms studied in this work are symmetrical, reflexive, functional, transitive, inverse and inverse functional properties. They don't extract property domains and ranges and don't extract using each property types mutually.

Fleischhacker et al. [9] proposed a method of creating an RDF dataset with property axioms. They extracted property subsumption, disjointness, transitivity, domain, range, symmetry, asymmetry, inverse, functionality, inverse functionality, reflexivity and irreflexivity. They used DBpedia for evaluating their approach. They don't extract using each property types mutually.

3 Property Definition for Our Wikipedia Ontology

3.1 Extraction of Property Names

We extract properties by using the following two methods.

Extraction by Scraping Infobox This method has been proposed in the past [3]. We extract a three-piece set of "article-item-value" in the Infobox as "instance-property-property value". In case the triple set is extracted directly from the dump data, the meaning of these properties is likely to appear challenging to understand at a glance, or the properties may not be integrated. Therefore, we convert properties in Infobox into HTML by Java Wikipedia API (Bliki engine)[1] here. Hereby, we can understand the properties' meaning at a glance and integrate properties.

Extraction by Scraping List Structure A large number of Wikipedia articles have list structures. This method extracts triples from list structures as "article name - heading name - each value". At that time, we examine the categories that each article belongs to and collect a number of heading names that appear in each category. This enables the extraction of the category that the article belongs to as a property domain. The procedure of this method is illustrated by steps 1 - 4 below.

1. Extract categories and heading name from each article of Wikipedia dump data.
2. Check occurrence rate of heading name from each category using (1).
3. Remove heading names that have a low occurrence rate from (2).
4. Extract heading name as property and each value of list structures as value of the property from each article.

3.2 Extraction of Property Domain

A subject of the triple discussed in 3.1 is an article name as an instance. Therefore, examining a category to which an article that is a subject belongs enables the definition a domain of a property. Therefore, the Infobox template name is extracted as a domain of each property in Infobox. Furthermore, we extract the Wikipedia categories to which the article using the Infobox template belongs, as

[1] http://code.google.com/p/gwtwiki

a domain. Moreover, for articles for which Infobox is not defined, a hyper concept that is extracted in "Extracting hypernymy-hyponymy from textual definition" in [3] is extracted as a domain. Collating the domain that is extracted so far with the class hierarchies extracted in [3], we extract the domain (considered as classes) included in the class hierarchy. As a result, we can eliminate domains that are not appropriate as classes. Finally, we remove a few occurrence rate domain (five or less), and we extract the remaining as a domain.

3.3 Extraction of Property Range

It is comparatively more convenient to define a property domain because an instance name as a subject of a property corresponds to an article name in a property triple, and the Infobox template name of an article can be considered to be a property domain. However, with regard to property ranges, an instance name as a object of a property cannot conclude to an article name; therefore, it is challenging to define the property ranges of all properties similar to property domain. Thus, for property ranges, we use following two methods:

1. Extraction using class-instance relations
2. Extraction using is-a relations

They are generally linked if a word has already had an article in Wikipedia and an article name corresponds to an instance name. Therefore, we match an object of a triple (instance) and instances of class-instance relations. Then, we extract classes as property ranges that the instance belongs.

Next, in order to extract property ranges that cannot be extracted by a previous technique, we match the categories that an article (which has a name identical to that of an object of a triple) belongs to and the classes of is-a relations in [3]. Then, we extract the classes as property ranges.

3.4 Extraction of Property Hypernymy-Hyponymy Relations

We attempt extracting property hypernymy-hyponymy relations. We use the following two methods.

Extracting by String Matching We extract the propety hypernymy-hyponymy relations by using "backward string matching" in [3]. The procedure of this method is shown following 1 - 4.

1. When a property A is set as "any string + property B", we extract them as property B rdfs:subPropertyOf property A.
2. In case they are set as "property A = any string + preposition + any string + property B", we eliminate them.
3. The domains of properties A and B, the ranges of properties A and B must be identical to or approximate the class hierarchy. Using this, we filter them.
4. We extract the remaining as property hypernymy-hyponymy relations.

Figure 1 shows an example. In the figure, the properties "Partner" and "Domestic partner" are matched, and their domains and ranges are identical. Therefore, we extract them as property hypernymy-hyponymy relations.

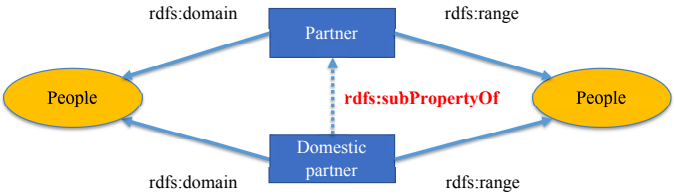

Fig. 1. Example of extraction of property hypernymy-hyponymy by string match

Extraction Using Triple from List Structures We match values of the extracted propety names by scraping the Infobox triple and the list structure in Wikipedia articles in each subject of the triples (instances). Then, when at least one matched value exists, we extract the property name by scraping the list structure and the property name by scraping the Infobox. Next, we compare the number of instances of the propeties; the larger one is extracted as the hypernymy, whereas, the smaller one is extracted as the hyponymy. Finally, we match the domains and ranges in these candidate properties. Then, when at least one matched properties exists, we extract these propeties as property hyperhymy-hyporymy relations.

Figure 2 shows an example. In this case, a subject of the triple "Nevada County, California" has a property "Communities" (which was extracted by scraping a list structure) and its values "Nevada City" and a property "Largest city" (which was extracted by scraping Infobox) and its values "Nevada City". Thus, we obtain "Community" as an upper property candidate and "Largest city" as a lower property candidate. Subsequently, when we match the domains and ranges of these properties, we know that these properties have identical domain and same range. Thus, we extract "Community - Largest city" as a property hypernymy-hyponymy relation.

3.5 Estimation of Property Types

Extracted properties by scraping Infobox have already classified Object type and Data type. In this method, adding to Object type and Data type, we attempt to estimate symmetric, transitive, functional, and inverse functional property types using the triples whose extraction is described in Section 3.1.

Extraction of Symmetric Property First, we attempt to estimate the symmetric propety. We consider a subject X and a value Y of each property P;

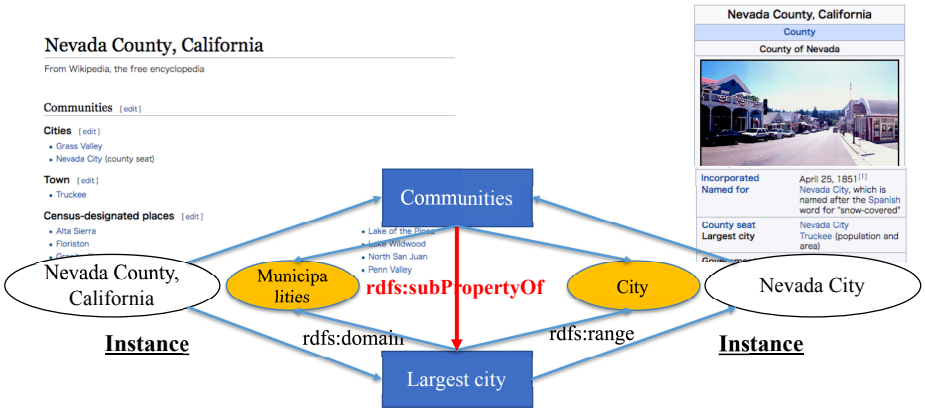

Fig. 2. Example of extraction of property hypernymy-hyponymy using triple from list structures

if a triple "$Y - P - X$" of P exist, we extract the propety P as a symmetric property candidate α. In addition, we attempt to estimate the ratio of all the triples of P to the triples extracted triples as symmetric relationship candidates.

Extraction of Transitive Property Next, we attempt to estimate transitive property. We consider a subject X, a value Y and a value Z of a subject Y of each propety P; if a triple "$X - P - Z$" of P exists, we extract the propety P as a transitive propety candidate. In addition, we attempt to estimate the ratio of all the triples of P to the triples extracted as transitive relationship candidates α.

Extraction of Functional Property Next, we attempt to estimate the functional property. We condider a subject X, a value Y of each property P; if P has a value Y for each instance X, we extract the property P as a functional property.

Extraction of Inverse Functional Property Moreover, we attempt to estimate the inverse functional property. We consider a subject X, a value Y of each property P; if P has only one instance X for each value Y, we extract the property P as the inverse functional property.

Extracting Using Property Hypernymy-Hyponymy Relations In addition to extraction by the above four methods, we estimate property types by using property hypernymy-hyponymy relations. We add property types to the subproperty of the property that have types.

3.6 Refining Extracted Relations

In this section, we propose refining methods of extracted property domain, property range, symmetric property, transitive property, functional property, and inverse functional property. Here, we attempt to improve the accuracy by mutually using each relation extracted by each method.

Refinement of Property Domain and Range In the method of Sections 3.2 and 3.3, multiple domains and ranges are linked to a property. In that case, the property "BornPlace" links to the property domain "People" and "Japanese people". "Japanese people" is a subclass of "People" and can be regarded as a redundant domain. Therefore, we consider removing such redundant domains and ranges. Hereby, using the is-a relation already extracted, if the hyper class of the extracted domain is also extracted as a domain and the latter has a high inclusion rate, the former is removed.

Refinement of Symmetric Property and Transitive Property We refine the symmetric and transitive proeprties extracted earlier. In the symmetric and transitive properties, the domain and range of the property must match. Therefore, we can filter them by using property domain and property range extracted earlier.

4 Experimental Results and Observations

Now, we discuss the results and observations extracted from the Wikipedia dump data as a source as of January 2017.

4.1 Results and Observations of Property Names

Results and Observations by Scraping Infobox From the Wikipedia dump data, we extracted 8,311,427 infoboxes, 12,309 infobox templates, and 22,767,071 triples. The number of property names was 12,088.

According to the property type estimation, approximately 60% of the properties were classified as either owl:ObjectProperty or owl:DatatypeProperty. The reason 40% of the properties were classified as Unknown was that the rule of classification was insufficient. In the future, we need to take into consideration that property values such as "small" can be literal. we extracted 1,000 samples from all triples to estimate the precision. The precision was $93.1 \pm 1.57\%$.

Hereby, We used the following expression (1) as a formula for 95% confidence interval. In formula (1), n represents the number of samples, N represents population, and $\hat{p}$ represents the estimated amount which is the number of accuracy samples divided by the total number of samples.

$$[\hat{p} - 1.96\sqrt{(1 - \frac{n}{N})\frac{\hat{p}(1-\hat{p})}{n-1}}, \hat{p} + 1.96\sqrt{(1 - \frac{n}{N})\frac{\hat{p}(1-\hat{p})}{n-1}}] \quad (1)$$

Results and Observations by Scraping List Structure From the Wikipedia dump data, we extracted 7,410,819 triples by scraping the list structure. The number of property names was 2394. We can extract properties that cannot be extracted by scraping Infobox, and they have numerous triples.

We extracted 1000 samples from all the triples to estimate the precision. The precision was $81.6 \pm 2.40\%$. A number of the errors were scraping errors, and they were extracted when a large quantity of information was described in each line of the list structures. For example, the property "Track listings" can be observed in numerous Wikipedia albums or single articles of singers; however, the list structures in these articles has generally described information such as Writers, Released Date, and Length, apart from the Track listings. Thus, Writers, Released Date and Length was extracted as values of the proeprty "Track listings". We consider it feasible to remove these errors to add more detailed rules of list structures with each category that the articles belong to.

4.2 Results and Observations of Property Domain

We were able to extract 500,967 property domains for 12,088 extracted properties. The properties extracted by scraping Infobox has the Infobox template as domains. Thus, we could define the property domain of all properties. In addition, we define the property domain of 100 properties by scraping the list structure in the Wikipedia articles. The property domains of 12,188 properties were defined. Thus, 84.2% properties have property domains.

We used the property domain defined in DBpedia Ontology for the evaluation. DBpedia Ontology defines properties, classes and property domains manually. Therefore, there are numerous upper classes. However, our ontology uses the Wikipedia category etc. Therefore, there are numerous lower classes. Consequently, because it does not match in a straightforward comparison, we assumed it to be correct to considered that the extracted property domain is included in the property domain of DBpedia Ontology. In addition, the notation of properties is different from that of DBpedia's properties in certain cases. By using the Infobox template, we compare them with the Infobox items before being mapped to the DBpedia ontology, to achieve consistency.

The precision was $94.9 \pm 1.93\%$. Table 1 presents examples of property domains. When a property has plurals property domains, we should integrate the common upper concept classes. For example, the property "Staff" has "TV program" and "TV drama" as property domain, and it has a number of classes as property domains apart from these (e.g. "Radio program", "Baseball team"). Therefore, we attempt to integrate a common upper concept class in Section 3.6.

4.3 Results and Observations of Property Range

We were able to 78,117 propety ranges for 12,088 extracted properties. Similar to the property domains, we evaluated property ranges using DBpedia Ontology. The precision was $76.9 \pm 3.68\%$. Table 2 presents examples of property range.

Table 1. Example of extracted property domains

Property	Property domain	#Triples
Length	Song	164,684
Nationality	Person	167,429
Spouse	Person	78,635

Table 2. Example of extracted property ranges

Property	Property range	#Triples
Starring	Actor	401,656
Place of birth	Town	890,271
Written by	Person	158,123

For example of errors, "Native dress" as property range of the property "nationality" was extracted. This is an error owing to an incorrect class-instance relation. Numerous errors by this method are because of this reason. Therefore, we consider that the precision of this method increases by improving the precision of the class-instance relations. Furthermore, "Article on mathematics" was extracted as a range of the property "victory Frequency" or "numberOfAccommodations". This is an error owing to an incorrect classificaion of owl:ObjectProperty or owl:DatatypeProperty. Generally, the property type becomes owl:DatatypeProperty. Thus, the range becomes literal (rdfs:Literal).

4.4 Results and Observations of Property Hypernymy-Hyponymy Relations

Results and Observations by String Matching We extracted 1297 property hypernymy-hyponymy relations. We extracted 500 samples from all the relations to estimate the precision. Thus, the precision was $89.0 \pm 2.36\%$. As examples of errors, "Movement - Tradition or movement" and "Period - Time period" are extracted. We can prevent the former error "Movement - Tradition or movement" by adding rules while removing unnecessary relations after string matching. The latter is an example in which the properties with identical meaning are extracted as hypernymy and hyponymy. In this case, it is feasible to assess whether they have identical meaning by using external resources.

Results and Observations by Using Triple from List Structure We were able to 242 property hyperhymy-hyponymy relations by using a triple from the list structure. We determined all 242 relations' truthfulness and falseness manually. The precision was 19.4%. As an example of errors, numerous upper properties and lower properties had identical meanings. Furthermore, numerous upper properties and lower properties were reversed. Because the triples extracted from Infobox are exceedingly large (although we had compared the properties by the total number of property values), the property extracted from

Infobox is an upper property, whereas the property extracted from the heading is a lower property.

4.5 Results and Observations of Property Types

We attempted to estimate by using the methods described in Section 3.2 with 12,088 extracted properties and 22,767,071 extracted triples.

First, we attempted to estimate symmetric property. We were able to 13,371 symmetric relation triples and 89 symmetric property candidates. We determined all 89 properties' truthfulness and falseness manually. Thus, the precision was 51.7%. Moreover, when we regard the inclusion rate as the threshold value, the precision is 92.3% (threshold 3%).

There are a number of properties by this method, including words such as "Related" ("Related case", "Related index"). However, this method could extract words such as "Former partner" and "Sister station" too.

Secondly, we attempted to estimate the transitive property. We were able to 48,035 transitive relation triples and 165 transitive properties candidates. Hereby, we excluded those which can be regarded as symmetric among the extracted relations because they are errors. As a results, We were able to 141 transitive property candidates excluding symmetric properties. We determined all the 141 properties' truthfulness and falseness manually. Thus, the precision was 19.6%.

We consider the covering problem in Wikipedia to be a reason for such a low accuracy. When we extract by this method, it is necessary to extract at least three triples that become the transitive relations. Thus, it is necessary to cover the information by Infobox or list structure as an identical property name in the Wikipedia articles; however, such covered information is negligible. For extracting transitive properties with high accuracy, we have to refine the property names, integrate the same one, and attempt to extract another method that use a part of the nostructuredness information in articles.

Thirdly, we attempted to estimate the functional property. We were able to 313,935 functional relation triples and 2,026 functional property candidates. We extracted 500 samples from all the triples to estimate the precision. The precision was $64.0 \pm 3.59\%$.

The largest errors are that the number of triples is small and properties that happen to have uniquely determined values of were extracted. Moreover, because the information in Infobox is incomplete, we extracted it erroneously as a functional property. We consider that errors owing to insufficient information can be prevented by using information such as Wikipedia texts.

Fourthly, we attempted to estimate inverse functional property. We were able to 10,187 inverse functional relation triples and 526 inverse functional property candidates. We extracted 200 samples from all the triples to estimate the precision. The precision was $24.0 \pm 4.03\%$.

There are problems of mark difference or synonym of property name, such as "Main work" and the property "Notable work." Therefore, we have to refine the property names and integrate the same one to extract inverse functional

property with high precision. Moreover, there were numerous errors owing to mistakes caused by scraping of Infobox. Therefore, it is necessary to refine the triple extraction method to improve accuracy.

Finally, we attempted extraction by using property hypernymy-hyponymy relations. We were able to 4 symmetric properties, 129 transitive properties, 195 functional properties, and 108 inverse functional properties. We dertermined all properties' truthfulness and falseness manually. The precision was 75%, 7.8%, 68.7%, and 21.3% respectively. In order to increase the accuracy and extract numerous relations, it is necessary to reconsider the estimation method of property type and the extraction method of subproperties.

4.6 Results and Observations of Refining Extracted Relations

Results and Observations of Refining of Property Domain and Range We refined extracted domains and ranges. As a result, the average and variance of the domains and ranges per property changed, as shown in the Table 3. Even after refinement, the number of domains and ranges per property is large, and there are variations depending on the properties. This is because the hierarchical structure of the class hierarchy is incomplete, and it is necessary to refine the class hierarchy in the future.

Table 3. Comparison of average and variance before and after refining

	Before refinement		After refinement	
	Average	Variance	Average	Variance
rdfs:domain	75.9	329094.1	38.5	64317.6
rdfs:range	33.7	39786.9	26.8	21672.6

Results and Observations of Refinement of Symmetric Property and Transitive Property We refined the extracted symmetric properties and transitive properties. As a result, the number of extracted relations and accuracy changed, as illustrated in the Table 4. Although the precision marginally increased with respect to the transitive properties, the precision marginally decreased with respect to the symmetric properties. The reasons for this are insufficient refinement and insufficient extraction of domains and ranges. In the future, it is necessary to unify classes and improve the refinement methods of domains and ranges.

5 Conclusion

In this paper, we proposed and evaluated a method of constructing a large-scale and general-purpose Ontology with property axioms using Wikipedia as

Table 4. Comparison of number of extracted relations and precision

	Before refinement		After refinement	
	Number of relations	Precision	Number of relations	Precision
Symmetric	89	51.7%	47	51.5%
Transitive	141	19.6%	77	24.7%

the resource. Through this study, we demonstrated that Wikipedia is a valuable resource for ontology learning for property axioms. It is feasible to extract property domain, property range, and property hypernymy-hyponymy relations etc. from Wikipedia. A few of the proposed methods can also be applied to DBpedia. These can be considered effective for constructing costless and large-scale ontologies in ontology learning.

In the future, we plan to improve the precision of each method. Moreover, we plan to integrate properties and unify classes extracted in [3] by using a upper ontology such as WordNet. Also, we intend to provide our Wikipedia Ontology.

References

1. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41. (1995)
2. Lenat, D.B. and Guha, R.V.: Building Large Knowledge Based Systems. Addison-Wesley (1990)
3. Kawakami, T., Morita, T., Yamaguchi, T.: Building Wikipedia Ontology with More Semi-structured Information Resources. JIST2017, LNCS, vol.10675, pp.3-18
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp.722-735(2007)
5. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A Large Ontology from Wikipedia and WordNet. Elsevier Journal of Web Semantics (2008)
6. Hoffart, J., Suchanek, F., Berberich, K., Weikum, G.: YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Reserch Report MPI-I-2010-5007, Max-Planck-Institut fur Informatik (2010)
7. Mahdisoltani, F., Biega, J., Suchanek, F.M.: A Knowledge Base from Multilingual Wikipedias. In: CIDR (2015)
8. Sánchez, D., Moreno, A., Del Vasto-Terrientes, L. (2012): Learning relation axioms-from text: An automatic web-based approach. Expert Systems with Applications, 39, 5792 - 5805.
9. Fleischhacker, D., Völker, J., Stuckenschmidt, H.: Mining rdf data for property axioms. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 718 - 735. Springer, Heidelberg (2012)