SKOS Sources Transformations for Ontology Engineering

Fabien Amarger, Jean-Pierre Chanet, Ollivier Haemmerlé, Nathalie Jane Hernandez, Catherine Roussey

SKOS Sources Transformations for Ontology Engineering: Agronomical Taxonomy Use Case

Fabien Amarger[1,2], Jean-Pierre Chanet[2], Ollivier Haemmerlé[1], Nathalie Hernandez[1], and Catherine Roussey[2]

[1] IRIT, UMR 5505, UT2J, Département de Mathématiques-Informatique, 5 allées Antonio Machado, F-31058 Toulouse Cedex, France - firstname.lastname@univ-tlse2.fr

[2] TSCF, Irstea de Clermont Ferrand, 9 av. Blaise Pascal CS 20085, 63172 Aubière, France - firstname.lastname@irstea.fr

Abstract. Sources like thesauri or taxonomies are already used as input in ontology development process. Some of them are also published on the LOD using the SKOS format. Reusing this type of sources to build an ontology is not an easy task. The ontology developer has to face different syntax and different modelling goals. We propose in this paper a new methodology to transform several non-ontological sources into a single ontology. We take into account: the redundancy of the knowledge extracted from sources in order to discover the consensual knowledge and Ontology Design Patterns (ODPs) to guide the transformation process. We have evaluated our methodology by creating an ontology on wheat taxonomy from three sources: Agrovoc thesaurus, TaxRef taxonomy, NCBI taxonomy.

Keywords: Ontology Development, Ontology Design Pattern, Non-Ontological Sources, SKOS, Trust, Agriculture

1 Introduction

The French Ministry of Agriculture has launched the Ecophyto plan [3] in order to reduce drastically pesticide use. Ecophyto includes several monitoring systems of agricultural practices. One of those is based on alert bulletins that inform farmers of pest attacks on crops. Thus, farmers adapt their crop treatements based on these alerts. These bulletins are called "Bulletin de Santé du Végétal" (BSV) [4]. In order to follow the evolution of pest attacks over several decades, these bulletins need to be gathered, analysed and annotated. The first step to help the annotating process is to build a reference source on any organism that could appear in the fields (crop plant, crop auxiliary, crop aggressor). This reference source is stored as Knowledge Base (KB) in OWL format [5].

[3] `http://agriculture.gouv.fr/ecophyto`
[4] `http://agriculture.gouv.fr/ecophyto-BSV`
[5] http://www.w3.org/TR/owl2-overview/

In agriculture, many data are available in various electronic formats about crops: thesauri, databases... The next challenge is to make these data available to all stakeholders (farmers, agronomist researchers) so that they can use the data in decision support and analysis tools. Linked Open Data (LOD) is an opportunity to accelerate the sharing of data. Thus we want to publish on the LOD the annotations of alert bulletins.

In this paper we describe a method to build a Knowledge Base (an ontology populated with individuals) from various sources. Unfortunately, we cannot trust all the extracted knowledge with the same confidence, because some errors appear in some sources [17]. Thus we propose a new method based on redundancy and trust scores to filter trustable knowledge.

This paper is organised as follows: Section 2 presents a state of the art about ontology engineering and trust. Then our proposition is explained in section 3. Some experiments are presented and discussed in section 4. We conclude and present our future works in section 5.

2 State of the Art

2.1 Reusing Non-Ontological Sources

Most part of ontology engineering methods use non-ontological sources during knowledge extraction processes. We can cite for example the MethOntology [7], the method [20] of the Neon methodology [18] or the SMOL methodology [9]. In our work we focus only on ontology engineering methods using Knowledge Organisation System (KOS) like thesauri, taxonomies and classification schemes because they are the most current to describe and classify organisms. Many knowledge organisation systems share a similar structure, and are used in similar applications. KOS can be defined as a hierarchical organisation of normalised terms used to classify any real entities. Some of the KOS are available on the LOD using the Simple Knowledge Organisation System (SKOS) [6] format. The figure 1 presents a SKOS example which comes from the Agrovoc thesaurus. We studied ten methods able to create a knowledge base in OWL format using KOS [1]. These methods can be classified as manual, semi-automatic or automatic.

manual The more recent methods [14] [3] and [13] are manual methods. This can be explained by the difficulties to translate a KOS conceptual structure into a knowledge base, due to the fact that the semantics of the KOS structure do not imply any logical formalisation. Thus the KOS conceptual structure is ambiguous for a logical point of view. For example, the figure 1 contains two hierarchies using the $skos : broader$ links. The left one defines different kinds of taxa (kingdom and phylum). The right one defines a taxonomy about plant organisms.

automatic In methods proposing some automatic processes, some of them follows the same strategy. They generate an $owl : class$ for each normalised

[6] http://www.w3.org/TR/2009/REC-skos-reference-20090818/

term. Each hierarchical relation is transformed into an $owl : subClassOf$ relation. In this category we can cite [~~21, 11, 20,~~ 10, ~~4~~]. The figure 2 illustrates the automatic transformation of the Agrovoc example of figure 1. Let us point out that some of the $owl : subClassOf$ relationships are false: A Phylum taxon is not a Kingdom one. To overcome this drawback, [~~20, 11~~] includes a disambiguation process to validate the $owl : subClassOf$ relationship.

semi automatic Others proposed to associate a specific $owl : object\ property$ that can be the $owl : subClassOf$ one with the hierarchical relation of the KOS [17, 10].

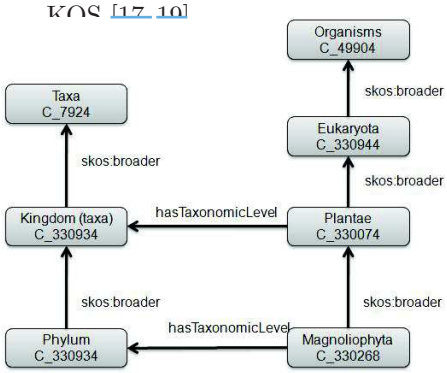

Fig. 1: example of agrovoc in SKOS format

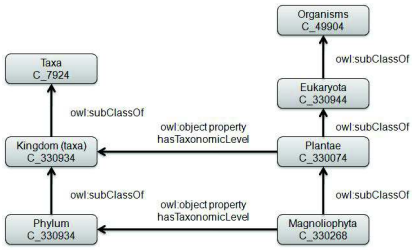

Fig. 2: Automatic transformation of agrovoc example

All these methods show that the KOS transformation should be guided in order to build a valuable knowledge base. Thus we decided to reuse Ontology Design Patterns [8] to guide the transformation of a KOS. An Ontology Design Pattern (ODP) is defined as a modelling solution to a recurrent ontology design problem [8]. ODPs are normally generated by experienced ontology engineers, who submit them to online repositories[7]. These patterns are evaluated by the Ontology Engineering community and generally accepted as good practices.

2.2 Ontological Object Trust

Extracting ontological objects from various non-ontological sources with different qualities requires a consideration of trust on these objects. Several trust definitions in computer science and semantic web are presented in [2]. The one which corresponds the most to our purpose is:

"Trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependable for a specified period within a specified context (in relation to service X)."

Consider A as the user who wants to create a knowledge base, B as a source and X the extraction process. In this definition, trust is about a source B, with the extraction process X, which generates ontological objects with a trust

[7] For instance, the repository at http://ontologydesignpatterns.org

score associated. This definition is very suitable for our purpose because they consider that a trust score is specific for a period, a context and a service. This corresponds to the fact that the trust on a source is variable depending on the objective of the project, the time and the source itself.

Using multiple sources to extract ontological objects leads to an aggregation of trust scores: Finding the same ontological object in several sources will increase the trust score of this object. As shown on [5] the aggregation of trust scores is more effective than classic approaches.

2.3 Synthesis

The method we propose can be seen as the combination of two Neon ontology engineering methods [18]: the one based on ODP [15] and the one based on non-ontological source transformations with NOR2O [20]. Note that in the SMOL methodology, the authors include a "knowledge structure construction" method in order to reorganise and harmonise the conceptual structures inherited from different sources. Moreover the Hepp method [11] includes an ODP to transform the KOS, because for 2 terms linked by a hierarchical relation, 4 $owl : classes$ and 3 $owl : subClasOf$ properties are build. Our method can be seen as a generalisation of [11] where different ODPs can be used depending of the domain of the ontology.

Moreover we take in consideration the consensus about each ontological object, to determine if we want to keep it or not. To do so, we use a trust score computed between all the sources used to extract ontological objects. As fas as we know, there is no ontology engineering method ables to transform KOS using consensus and ODPs. About consensus, we have to define a function to compute trust score and a way to aggregate them.

3 Our General Approach

Due to its completeness we selected the Neon methodology to build a KB. Neon proposes a set of nine methods for collaboratively building ontologies. Our ontology engineering method consists of adapting and merging two Neon methods.

Our method is composed of three processes detailled in next sections:

1 - Source analysing: During this process, the domain expert and the ontologist work together to select the most appropriate sources to build their KB. They inspect each source to evaluate its coverage and to have a broad idea if the source can be transformed to an KB or not.

2 - Source Transformation This process transforms each source into an KB in OWL format. It is based on Neon methods.

3 - KBs Merging This process builds the final KB based on all KBs extracted from sources. As far as we known, this process is not proposed in any ontology engineering method. Usually ontology engineering method uses several sources separately in order to enrich the KB in a incremental way. The merging process uses several KBs at the same time in order to extract consensual knowledge.

3.1 Source Transformation

The figure 3 present our "source transformation" process, which contains several other processes. The "module construction" is the Neon scenario 7 [8]. The scenario 7 proposes to build a module re-using Ontology Design Patterns (ODPs) and competency questions. We fully apply this method to build modules. The module is built once and is used for all the "source automatic transformation" processes. The "syntactic transformation" is an adaptation of the Neon scenario 2 [20]. The scenario 2 proposes to build a KB reusing non ontological source. We adapt this method to enrich modules previously build by scenario 7.

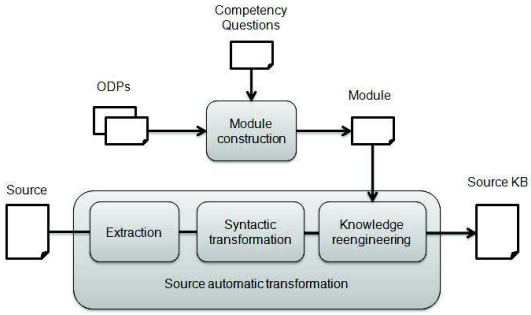

Fig. 3: Source transformation process

Module Construction The current best practice to create an ontology is to reuse ODPs. We follow the method [8] in order to generate modules. An example of one of our modules is Agronomic Taxon [16] (c.f. appendix 6.1) which has been manually built for a specific task (representing organism scientific name using taxonomy). The module is composed of $owl : Classes$ and defines the set of $owl : Properties$ that may exist between them. It re-uses some ODPs coming from Neon project and two vocabularies already published on the LOD [16].

The Agronomic Taxon module models living organism taxonomy. All the taxon types wanted in our knowledge base are defined in the module as $owl : class$, child of the $neon : Taxon$ class. For example we only focus on the seven most known taxon types: kingdom, phylum, class, order, family, genus and species. The next process will use the sources to enrich automatically this module. The final KB should contains several taxa, individuals of the class $neon : Taxon$ that are used to describe organisms appearing in fields.

Source Automatic Transformation As generally each non-ontological source follows some modelling principles and is implemented in a specific format. For example Agrovoc follows the modelling principles of multilingual thesaurus and is available in the SKOS format. The [20] [17] methods proposes transformation using patterns. The [20] method takes in account the modelling and implementation choices and applies the same transformation pattern on the source. The [17] method takes in account that the modelling choices may change over the same source and that the same pattern can not be applied on the whole source.

We will take advantage on these two methods and apply transformation based on pattern.

As shown in Figure 3, we extract first from the source, the parts that seem to follow the same modelling principles and that meet our requirements. The previous "source analysing" process has defined that these parts exist in the source. Secondly we apply a syntactic transformation using [20] method and tools in order to have a file following the OWL syntax. The "knowledge engineering" activity produces a new owl file which is an enrichment of the module that is to say a KB. To do so, the module is mapped to the first owl file. The output is a set of mappings. Then the module is expanded using the owl file and following new pattern that re-engineered the owl file. Thus the new OWL axioms are compatibles with the module.

For example, if Agrovoc was selected during the "source analyzing" process. The experts decide that it is a good source to build a KB about plant taxonomy. They decide to work on the SKOS file of the Agrovoc thesaurus. Based on the module Agronomic Taxon, we will illustrate the "source automatic transformation" process. First we extract from Agrovoc all the data related to plant taxonomy (see figure 1), that is to say all the $skos : concepts$ under Taxa and under Plantae. Then we apply a transformation pattern based on thesaurus and SKOS format as presented in the section 2. We obtain an owl file like the figure 2. We first mapped manually the owl file to Agronomic Taxon module. The $owl : classes\ neon : Taxon$, $neon : Kingdom$, $neon : Phylum$, and so one are mapped to $owl : class$ of Agrovoc file. Now we apply a re-engineering pattern (c.f. the appendix 6.3). This algorithm creates a new individual for each class representing a plant taxon. Using the $hasTaxonomicLevel$ link of Agrovoc the individual is typed by the corresponding rank type (Kingdom, Phylum and so one). Then the hierarchy between taxa is depicted using the $neon : hasHigherRank$ property of the Agronomic Taxon module. At the end of the re-engineered pattern the source KB contain the module plus new individuals as proposed in figure 4.

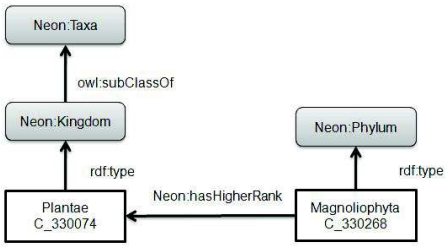

Fig. 4: Enrichment of Agronomic Taxon module using Agrovoc

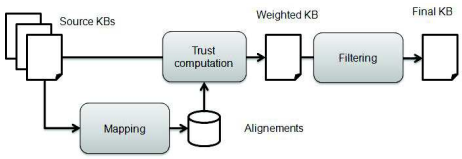

Fig. 5: KB Engineering process

3.2 KBs Merging

The figure 5 shows that the merging of KBs previously built, named *Source KB*, is composed of three activities:

Mapping: this activity computes Alignments between all the *Source KBs*. Mapping activity identifies similar ontological objects contained in distinct *Source KBs*.

Trust computation: This activity identifies candidates and computes their trust score. A candidate contains a set of ontological objects that are found similar by the mapping activity.

Filtering: This activity filters candidates according to their trust score and add them in the final KB.

(1) Mapping Aligning KB_1 and KB_2 consists in computing all the mappings between objects of KB_1 and objects of KB_2. This is a large research area [6] and a lot of methods have been proposed and implemented in tools. We choose to use LogMap [12] because it can map any ontological objects, it obtains good results in OAEI Challenge[8] and its source code is available online[9]. Let us define a mapping m as a triplet $< e_i, e_j, s_{ij} >$ such as:

$e_i \in KB_i$**:** is an ontological object belonging to KB_i ,
$e_j \in KB_j$**:** is another ontological object belonging to KB_j ($KB_j \neq KB_i$),
s_{ij}**:** is the similarity degree between e_i and e_j.

We define a function called $degree(e_i, e_j)$ from $KB_i \times KB_j$ to $[0, 1]$. Where $s_{ij} = degree(e_i, e_j)$ is the similarity score between e_i and e_j given by the mapping tool and 0 if there is no mapping.

(2) Trust Computation Due to space limitation of the article, we present two kinds of candidate, but more candidate types are taken in account in our method.

Individual Candidate (ic): Individuals are instances of classes. We define an individual candidate ic as a set of mappings that share common individuals. Each individual, belonging to a candidate, should belong to a distinct knowledge base.

Relation Candidate (rc): the instances of property that links two individual candidates. We define a relation candidate rc as a pair of individual candidates, such as there exist some instance of the same properties that link components of individual candidates.

We define $dim(c)$ as the number of KBs involved in a candidate c.

Let consider the example in Figure 6 with three knowledge bases KB_1, KB_2 and KB_3. KB_1 and KB_3 contains two individuals a_i, b_i linked by the same property p. The dash line represent mapping between individuals. There are two individual candidates ic_1 and ic_2 and one relation candidate rc_1.

[8] Ontology Alignment Evaluation Initiative - http://oaei.ontologymatching.org/2013/
[9] https://code.google.com/p/logmap-matcher/

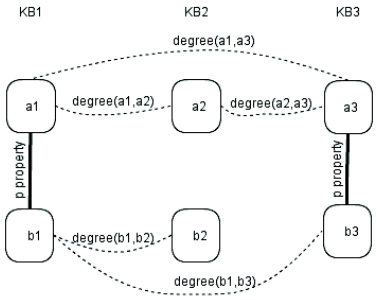

Fig. 6: Example of candidates

$$
\begin{aligned}
ic_1 &= \begin{cases} [< a_1, a_2, s_{12} >, < a_2, a_3, s_{23} >, \\ \qquad\qquad < a_1, a_3, s_{13} >] \\ a_1 \in KB_1, a_2 \in KB_2, a_3 \in KB_3 \\ s_{12} = degree(a_1, a_2) \\ s_{23} = degree(a_2, a_3) \\ s_{13} = degree(a_1, a_3) \end{cases} \\
ic_2 &= \begin{cases} [< b_1, b_2, s_{12} >, < b_1, b_3, s_{23} >] \\ b_1 \in KB_1, b_2 \in KB_2, b_3 \in KB_3 \\ s_{12} = degree(b_1, b_2), s_{13} = degree(b_1, b_3) \end{cases} \\
rc_1 &= \begin{cases} [ic_1, ic_2] \\ [p(a_1, b_1), p(a_3, b_3)] \\ p(a_1, b_1) \in KB_1, p(a_3, b_3) \in KB_3 \end{cases}
\end{aligned} \tag{1}
$$

In the example figure 6 $dim(ic_1) = 3$, $dim(ic_2) = 3$ and $dim(rc_1) = 2$.

Each candidate has a trust score to define how much we can trust this candidate. There are several way to compute this score. In the experiments we will test several trust functions.

4 Experiments

The goal of our experiments is to test different functions to compute the trust score. We want to determine which function obtain the best results. To do so, we build a use case about wheat crops. We want to build a KB describing wheat taxonomy.

4.1 Source Analysing Process

For the project of wheat taxonomy, our experts select the following sources:

Agrovoc[10]**:** a multilingual thesaurus with more than 40,000 terms,
TaxRef[11]**:** a french taxonomic referential with 80,000 taxa created by the "Muséeum national d'histoire naturelle",
NCBI Taxonomy[12]**:** a taxonomy created by the National Center for Biotechnology Information (NCBI) of the United States with 1,000,000 taxa.

We chose these three sources because of their complementarity. First NCBI is the source with the most taxa. It is considered by experts to be the most up-to-date source but this include potential errors and there are only few labels. Alongside, Agrovoc contains labels in several languages with distinctions between scientific labels and vernacular ones but less taxa than NCBI and with a quality often criticized [17]. TaxRef overcomes this drawback and is considered as a national reference in agronomic classification. But its number of taxa is limited

by the data verification process. Combining these three sources is very suitable because we combine the taxa quantity (NCBI), with labels quantity (Agrovoc) and the assurance of quality (TaxRef).

4.2 Source Transformation Process

We start the "source transformation" process by building a module about plant classification (*Agronomic Taxon* [16], see appendix 6.1 for more details. From each sources, we extract automatically subparts of the wheat taxonomic classification. We focus the extraction on the Triticum taxa. We create an OWL file corresponding at the "syntactic transformation" of each source using NOR20 patterns. Then we define re-engineering patterns to extract instances of $neon : Taxon$ from the three different OWL files. For each individual we type them and link them using the $neon : hasHigherRank$ object property.

KBs Merging Process For this process we reused LogMap tool for the mapping activity. Then we define several trust functions to compute trust score of individual candidates and relation candidates. Then we apply a threshold empirically fixed at 0.6 to filter candidates. Thus a candidate becomes a component of the final KB if its trust score is above 0.6 otherwise it is rejected.

Simple Trust Function The simple way to extract consensual ontological objects is to determine in how many KBs the candidate appears. We consider that a candidate is consensual if it appears in at least two KBs. Otherwise the candidate should not belong to the final KB. We defined a function called $trust_{simple}$ to implement the simple consensus. $trust_{simple}$ is defined by the following formula :

$$trust_{simple}(c) = \begin{cases} 1 & \text{if } dim(c) >= 2 \\ 0 & \text{if } dim(c) < 2 \end{cases} \tag{2}$$

Degree Trust Function We can also use the mapping degree (provided by LogMap) to compute the trust score. We consider that a candidate with higher mapping degrees implies more trust. For the degree consensus implementation there is a different formula for each kind of candidate.

The instance candidate trust function is defined by the formula:

$$trust_{degree}(ic) = \frac{\sum_{i=1}^{dim(ic)} \sum_{j=i+1}^{dim(ic)} degree(a_i, a_j)}{\frac{nb_{Sources}(nb_{Sources}-1)}{2}} \quad such\ as (a_i, a_j) \in ic \tag{3}$$

This function sum all mapping degree involved in the candidate. We normalised the result with the maximum number of individuals mappings possible in an

individual candidate (We have 3 KBs thus we can have at most 3 mappings in an individual candidate). Here, $nb_{sources}$ is the total number of KBs involved on the merging process.

The relation candidate trust function is defined by the formula:

$$trust(rc) = \frac{dim(rc) + \frac{trust(ic1)+trust(ic2)}{2}}{nb_{Sources} + 1} \quad \text{such as } ic1 \in rc, ic2 \in rc \tag{4}$$

This formula takes in account the $dim(rc)$ and the average of trust scores of individual candidates, components of the relation candidate. We do so to simulate a mapping degree between object properties instances. Note that LogMap do not match object property instances. We normalised this result with the $nb_{Sources}$, which is the maximum value that $dim(rc)$ could be, plus 1, which is the maximum value that the average of the two ic trust score could be.

4.3 Experiment Set up

To build our Gold Standard KB in order to compare the output of the different KBs merging process, we ask to three agronomists to validate manually the three KBs, outputs of the "source transformation" process. We consider that an ontological object is validated by the experts if at least two experts validated it and the third one vote for the "don't know" option (more precision on the appendix 6.2). The baseline is composed of the union of all ontological objects validated by the experts. Two final KBs are generating using the different trust functions ($trust_{simple}$ and $trust_{degree}$). Precision, recall and f-measure are computing to evaluate the quality of the final KBs. The precision is the ratio between the number of ontological objects of the final KB validated by experts and the total number of ontological objects of the final KB. The recall is the ratio between the number of ontological objects validated by experts which appear on the final KB and the number of ontological objects validated by experts.

4.4 Results and Analyse

Table 1 presents our results. The column, called simple consensus, shows the results of the "KBs merging" process using the $trust_{simple}$ function. The column, called degree consensus, shows the results of the "KBs merging" process using the $trust_{degree}$ function.The fist line presents the results on individual candidates and the second line presents the results on relation candidates.

	Simple Consensus			Degree Consensus		
Candidate	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Individual	0.91	0.66	0.77	1	0.59	0.74
Relation	0.41	0.48	0.45	0.44	0.4	0.4

Table 1: Results

We can observe on the table 1 that the results are encouraging. All the individuals that our method is able to extract are valuable one (our individual

candidate results obtain a high precision); But our method is not yet able to extract all the valuable ontological objects. The degree consensus approach worka little bit better than the simple consensus one. More over our approach is able to extract more individual, than links between individuals.

5 Conclusion and Future Works

In this article, we propose a method to transform several KOS into a knowledge base using Ontology Design Patterns and consensus. Our method estimate consensus by computing a trust score for each ontological object extracted. We determined which trust formula is the more suitable for our use case. This method helps the validation at the end of the process because some candidates could be validated (or rejected) automatically, by using different filtering thresholds.

We will focus our next works on the results filtering to answer to several problems we observed. First we want to implement the same approach to extract different type of ontological objects (labels, classes, ...). We should also improve the extraction of links between individuals. Then we have to face the problem of contradictions between candidates. Currently all candidates are considered and can be accepted, even if there is a conflict. To solve such conflicts, it could be possible to use the argumentation theory associated with the trust score to manage the candidate selection. We want also to work on another sub-domain than the plants taxonomy classification. We planned to work on the attacks from bio-aggressors using the module CultivatedPlant[13] with a database from the Arvalis[14].

Acknowledgements

We want to special thanks the three experts who helped us to validate our results by generating the gold standard:

Franck Jabot from Irstea Clermont-Ferrand, France
Jacques Le Gouis from INRA Clermont-Ferrand, France
Vincent Soulignac from Irstea Clermont-Ferrand, France

References

1. F. Amarger, C. Roussey, J.P. Chanet, O. Haemmerlé, and N. Hernandez. Etat de l´art: Extraction d´information à partir de thésaurus pour générer une ontologie. *INFORSID*, pages 29–44, 2013.
2. D. Artz and Y. Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, pages 58–71, 2007.

[13] https://sites.google.com/site/agriontology/home/irstea/cultivatedplant
[14] http://www.arvalis-infos.fr

3. J. Charlet, G. Declerck, F. Dhombres, P. Gayet, P. Miroux, and P.Y. Vandenbussche. Construire une ontologie médicale pour la recherche d'information: problématiques terminologiques et de modélisation. *Ingénierie des connaissances*, pages 33–48, 2012.
4. C. Chrisment, O. Haemmerlé, N. Hernandez, and J. Mothe. Méthodologie de transformation d'un thesaurus en une ontologie de domaine. *Revue d'Intelligence Artificielle*, pages 7–37, 2008.
5. D. Downey, O. Etzioni, and S. Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI*, pages 1034–1041, 2005.
6. J. Euzenat and P. Shvaiko. *Ontology matching*. 2007.
7. M. Fernández-López, A. Gómez-Pérez, and N. Juristo. Methontology: from ontological art towards ontological engineering. *American Asociation for Artificial Intelligence*, 1997.
8. A. Gangemi and V. Presutti. Ontology Design Patterns. In *Handbook on Ontologies*, pages 221–243. 2009.
9. R. Gil and M. Martín-Bautista. Smol: a systemic methodology for ontology learning from heterogeneous sources. *Journal of Intelligent Information Systems*, pages 415–455, 2014.
10. U. Hahn. Turning informal thesauri into formal ontologies: a feasibility study on biomedical knowledge re-use. *Comparative and functional genomics*, pages 94–97, 2003.
11. M. Hepp and J. De Bruijn. GenTax: a generic methodology for deriving OWL and RDF-S ontologies from hierarchical classifications, thesauri, and inconsistent taxonomies. In *ESWC*, pages 129–144, 2007.
12. Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. Large-scale interactive ontology matching: Algorithms and implementation. In *European Conference on Artificial Intelligence*, pages 444–449, 2012.
13. D. Kless, L. Jansen, J. Lindenthal, and J. Wiebensohn. A method for re-engineering a thesaurus into an ontology. In *FOIS*, page 133, 2012.
14. P. Li and Y. Li. On transformation from the thesaurus into domain ontology. *Advanced Materials Research*, pages 2698–2704, 2013.
15. V. Presutti, E. Blomqvist, E. Daga, and A. Gangemi. Pattern-Based Ontology Design. In *Ontology Engineering in a Networked World*, pages 35–64. Springer, 2012.
16. C Roussey, J.P. Chanet, V. Cellier, and F. Amarger. Agronomic taxon. In *WOD*, page 5, 2013.
17. D. Soergel, B. Lauser, A. Liang, F. Fisseha, J. Keizer, and S. Katz. Reengineering thesauri for new applications: The AGROVOC example. *Journal of Digital Information*, pages 1–23, 2004.
18. M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi. *Ontology engineering in a networked world*. 2012.
19. M. Van Assem, M.R. Menken, G. Schreiber, J. Wielemaker, and B. Wielinga. A method for converting thesauri to RDF/OWL. *ISWC*, pages 17–31, 2004.
20. B. Villazón-Terrazas, M. C. Suárez-Figueroa, and A. Gómez-Pérez. A pattern-based method for re-engineering non-ontological resources into ontologies. *Int. J. Semantic Web Inf. Syst.*, pages 27–63, 2010.
21. B. J. Wielinga, A. Th. Schreiber, J. Wielemaker, and J. A. C. Sandberg. From thesaurus to ontology. In *K-CAP*.

6 Appendix

6.1 AgronomicTaxon

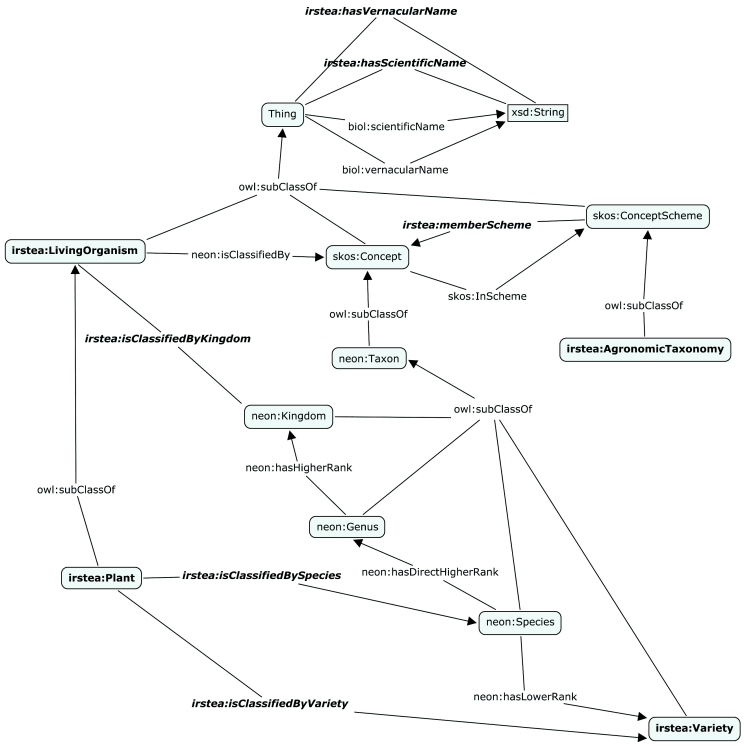

Fig. 7: AgronomicTaxon

6.2 Gold Standard

Experts validation To validate our approach, we asked to three domain experts to analyse the three knowledge bases extracted automatically from the three sources: Agrovoc, Taxref, NCBI. The experts have to determine which ontological objects are well represented and in the scope of the knowledge base.

An interface was implemented to let the experts validate the ontological objects. Here there was only instances of the $neon : Taxon$ class. For each individual four questions were asked to the expert :

1. **Does the taxon belong to the domain?** first we ask if the element presented is really a taxon (an element of a taxonomy). Also we want to know if the taxon is in the scope of the KB that is to say (Triticum or Aegilops).

2. **Do the labels designate the same entity?**
   There are several labels available in KB (especially in Agrovoc) but sometimes some labels are inexact, because there are not synonym or they are not the exact translation (if the source contains multilingual labels).
3. **Is the taxon more specific than "anotherTaxon" ?**
   When we can extract a "hasHigherRank" relation between two taxa from the source, we want to validate this link. So we want to know if the first taxon is a specialisation of the second.
4. **Is the rank of the taxon "aTaxonType" ?**
   Sometimes there is information about the rank type of the taxon in the source. We want here to validate this extraction and define the type of the taxon. Is it a specie, a family, a gender, ... ?

At the end of the form, there is an input field to let the expert add a comment if the response was not obvious. We can see the validation interface on the figure 8.

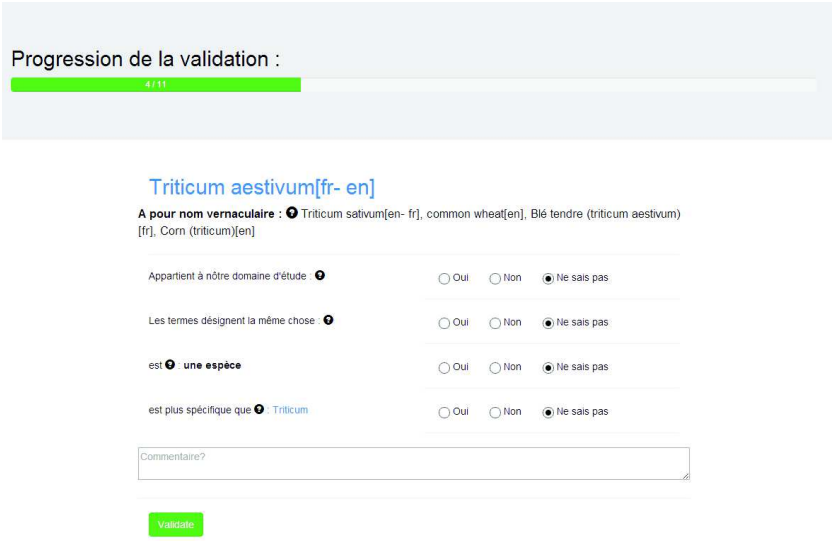

Fig. 8: Validation interface

This figure shows that for each question there are three available answer: Valid, Not Valid or Don't know.

The three experts have validated the $owl : KBs$ (NCBI/Triticum, NBCI/Aegilops, Agrovoc/Triticum, Agrovoc/Aegilops, TaxRef/Triticum, TaxRef/Aegilops). At the end of the validation we have a list of ontological objects validated by the experts. We can then compare them with the candidates generated by the prototype.

Evaluation of the consensus intuition To build our Gold Standard baseline, we first have to know if an agreement is possible between experts. To do so we computed a ratio between the number of experts and their number of validations. We consider that there is an agreement between experts when at least two experts validate the same ontological object and the third one select the *Don't know* option. We get a consensual ratio of **0.82**. We also computed the Fleiss Kappa score on the expert validations. Then we get the Fleiss Kappa of **0.69**. These two values show that the experts agree, most of the time, on the validation of the ontological objects. So we can use this consensus and use the experts validation as a gold standard to validate candidates.

6.3 Agrovoc algorithm

Algorithm 1 Transformation Pattern : AGROVOC for AgronomicTaxon

```
aModule: the AgronomicTaxon module
anOwlFile: agrovoc transformed in owl using transformation pattern
aModuleClassesList: the classes of AgronomicTaxon module that are mapped to anOwlFile
(neon:Taxon, neon:Kingdom, neon:Phylum etc...)
aKB: the AgronomicTaxon module enriched by the data from agrovoc owl file
aKB ← copy (aModule);
aClassLis() ← All subClasses of Plantae in anOwlFile
while aClassList() is not empty do
    aTaxonClass ← extract from(aClassList());
    anIndividualTaxon ← create an Owl Individual From(aTaxonClass)
    Add anIndividualTaxon in aKB
    if  Exist a property called hasTaxonomicLevel linking aTaxonClass in anOwlFile then
        aCurrentModuleClass ← Find a class linked to aTaxonClass by the hasTaxonomicLevel
        property in aModuleClassList()
        Add an rdf : type property between anIndividualTaxon and aCurrentModuleClass in
        aKB
    else
        aCurrentModuleClass ← neon : Taxon
        Add an rdf : type property between anIndividualTaxon and aCurrentModuleClass in
        aKB
    end if
end while
anIndividualList() ← All owl : individual type of neon : Taxa in aKB
while anIndividualList() is not empty do
    aTaxonIndividual ← extractFrom(anIndividualList());
    aTaxonClass ← Find a class equivalent to aTaxonIndividual in anOwlFile
    if Exist a class subClass of aTaxonClass in anOwlFile then
        aChildClass ← Find a class subClass of aTaxonClass in anOwlFile
        anotherTaxonIndividual ← Find an individual that is equivalent to aChildClass in aKB
        Add a neon : hasLowerRank object property between aTaxonIndividual and
        anotherTaxonIndividual in aKB
    end if
end while
```