WP7-D72

Interactive UI Design

Project full title:	Knowledge Driven Data Exploitation
Project acronym:	K-Drive
Grant agreement no.:	286348
Project instrument:	EU FP7/Maria-Curie IAPP/PEOPLE WP 2011
Document type:	D (deliverable)
Nature of document:	R (report)
Dissemination level:	PU (public)
Document number:	ISOCO, UNIABDN/WP7-D72/D/PU/b1
Responsible editors:	Panos Alexopoulos, Ronald Denaux, Boris Villazon-Terrazas, and Honghan Wu
Reviewers:	TBD
Contributing participants:	ISOCO, Expert System Iberia, UNIABDN
Contributing workpackages:	WP7
Contractual date of deliverable:	31 July 2015
Actual submission date:	31 July 2015

Abstract
In this deliverable we describe the design of three interactive user interfaces, each of which facilitates the execution of a different semantic data management task. The first interface enables data producers and ontology engineers to annotate vague ontological entities with explicit formal descriptions of the nature and characteristics of their vagueness. The second interface enables the selection and reuse of Linked Open Vocabularies during the ontology development process. Finally, the third interface allows users to derive personalized dataset summarization tasks that fit best to their data-related goals and requirements.

Keyword List
Interactive UI, Vagueness Annotation, Linked Open Vocabulary Discovery, Personalized Dataset Summarization

Project funded by the European Commission within the 7th Framework Programme/Maria Curie Industry-Academia Partnerships and Pathways schema/PEOPLE Work Programme 2011.

Interactive UI Design

Panos Alexopoulos[1], **Honghan Wu**[2], **Ronald Denaux**[1]

[1] iSOCO, Spain
[2] Department of Computing Science, Aberdeen University, UK
[3] Expert System Iberia, Spain

31 July 2015

Abstract
In this deliverable we describe the design of three interactive user interfaces, each of which facilitates the execution of a different semantic data management task. The first interface enables data producers and ontology engineers to annotate vague ontological entities with explicit formal descriptions of the nature and characteristics of their vagueness. The second interface enables the selection and reuse of Linked Open Vocabularies during the ontology development process. Finally, the third interface allows users to derive personalized dataset summarization tasks that fit best to their data-related goals and requirements.

Keyword List
Interactive UI, Vagueness Annotation, Linked Open Vocabulary Discovery, Personalized Dataset Summarization

Contents

1 Introduction

This deliverable consists of 3 sections, each describing the design of an interactive user interface for a particular semantic data management task. The first task is related to Ontology Authoring and, more specifically, to how ontology engineers can detect, analyze and make explicit the vagueness that may characterize their ontology and/or semantic data. The corresponding user interface we describe here allows such engineers to perform this task in a semi-automatic way, without being concerned with the complexity of the underlying representation. The second task is related to Ontology Reuse and focuses on enabling ontology engineers to discover and reuse Linked Open Vocabularies that best fit to the ontology they are developing. The corresponding interface is designed as a Protege plugin.

Finally, the third task is related to the semantic data summarization framework of WP4 and, in particular, to the personalized recommendations of dataset summarization tasks to (new and existing) users. In D7.1 we defined a User Model that captures the preferences of users with respect to the summarization tasks they use/prefer for particular goals. Given these preferences, collaboration filtering mechanisms can then be used to recommend summarizations tasks to (either new or existing) users, based on user and task similarity. In this deliverable, we focus on how this recommendation can be best facilitated in the UI level by describing a corresponding interactive user interface based on a novel interaction paradigm called Collaboration Spheres.

2 Authoring Vagueness-Aware Ontologies

The need to effectively support ontology developers in creating high quality ontologies that are able to facilitate interoperable knowledge representation, has long been recognized in the community Simperl et al. (2009), leading to the development of several ontology authoring frameworks Francescomarino et al. (2012) Keet et al. (2013) Ren et al. (2014). Each of these frameworks typically targets a particular aspect, dimension or subtask of the ontology authoring process such as, for example, the linking to upper ontologies Keet et al. (2013) or the facilitation of collaboration among ontology developers Francescomarino et al. (2012). Nevertheless, a dimension that, to the best of our knowledge, enjoys limited support from existing ontology authoring frameworks and tools is **vagueness**.

More specifically, vagueness is a common linguistic phenomenon, typically manifested by terms and concepts like *High*, *Expert*, *Bad*, *Near* etc., and related to our inability to precisely determine the extensions of such concepts in certain domains and contexts. That is because vague concepts have blurred boundaries which do not allow for a sharp distinction between the entities that fall within their extension and those that do not Hyde (2008) Shapiro (2006). For example, some people are borderline tall: not clearly *"tall"* and not clearly *"not tall"*.

When building ontologies, engineers and domain experts often use predicates that are vague. This is evident from several publicly available ontologies that contain entities (classes, relations, etc.) with vague definitions. For example, the relation "*hasFilmGenre*", found in LinkedMDB[1] and DBpedia[2], that relates films with the genres they belong to is vague, the reason being that most genres have no clear applicability criteria and, therefore, there will be films for which it is difficult to decide whether or not they belong to a given genre. Other examples of vague classes include "*Famous Person*" and "*Big Building*", in the Cyc Ontology[3], and "*Competitor*", found in the Business Role Ontology[4].

[1] http://linkedmdb.org
[2] http://dbpedia.org
[3] http://www.cyc.com/platform/opencyc
[4] http://www.ip-super.org

Vague ontological definitions can influence in a negative way the comprehension of ontologies by other parties and limit their value as a reusable source of knowledge Alexopoulos et al. (2013). The reason is the subjective interpretation of such definitions that may cause **disagreements** among the people who develop, maintain or use an ontology. Such a situation arose in a real life scenario where we faced significant difficulties in defining concepts like "*Critical System Process*" or "*Strategic Market Participant*" while trying to develop an electricity market ontology. When we asked our domain experts to provide exemplary instances of critical processes, there was dispute among them about whether certain processes qualified. Not only did different domain experts have different criteria of process criticality, but neither could anyone really decide which of those criteria were sufficient for the classification. In other words, the problem was the vagueness of the predicate "*critical*".

Even if such disagreements are overcome by consensus during the ontology development process, they are inevitable as more users alter, extend, or use ontologies. Imagine, for example, an enterprise ontology where the concept "*Strategic Client*" was initially created and populated by the company's executive board, their implicit membership criterion being the amount of revenue the clients generate for the company. Imagine also the new R&D Director querying the instances of this concept while crafting an R&D strategy. If their own applicability criteria for the term "*Strategic*" do not coincide with the board's, using the returned list of clients might lead to poor decisions. Generalizing these examples, some typical use-case scenarios where ontology vagueness may cause problems include:

1. **Instantiating a Vague Ontology:** When domain experts are asked to define instances of vague concepts and relations, then disagreements may occur on whether particular entities constitute instances of them.

2. **Utilizing Vague Facts in Ontology-Based Systems:** When knowledge-based systems reason with vague facts, their output might not be optimal for those users who disagree with these facts.

3. **Integrating Vague Semantic Information:** When semantic data from several sources need to be merged then the merging of particular vague entities can lead to data that will not be valid for all its users.

4. **Evaluating Vague Semantic Datasets for Reuse:** When data practitioners need to decide whether a particular dataset is suitable for their needs, the existence of vague entities can make this decision harder. It can be quite difficult for them to assess *a priori* whether the data related to these entities are valid for their application context.

To reduce the number and intensity of such disagreements we have put forward the notion of **vagueness-aware ontologies** Alexopoulos et al. (2013), informally defined as *"ontologies whose vague entities are accompanied by comprehensive metainformation that describes the nature and characteristics of their vagueness"*. A simple example of such metainformation is whether an ontology entity (e.g., a class) is vague or not; this is important as many ontology users may not immediately realize this. A more sophisticated example is the particular type of the entity's vagueness or the applicability context of its definition. In all cases, the rationale is that having such metainformation, explicitly represented and published along with (vague) ontologies, can improve the latter's comprehensibility and shareability, by narrowing the possible interpretations that its vague entities may assume by human and software agents.

To enable the formal representation of vagueness-aware ontologies we have developed the **Vagueness Ontology (VO)**[5] Alexopoulos et al. (2014), a metaontology that defines the necessary concepts, relations and attributes for creating explicit descriptions of vague ontology entities. VO is meant to be

[5] http://www.essepuntato.it/2013/10/vagueness

used by both producers and consumers of ontologies; the former will utilize it to **annotate** the vague part of their produced ontologies with relevant vagueness metainformation while the latter will **query** this metainformation and use it to make a better use of the vague ontologies.

Annotating ontologies with VO is a primarily manual task, with knowledge engineers and domain experts having to detect the vague entities of a given ontology, determine their vagueness-related characteristics and produce an instance of VO as metadata for the particular ontology. In this deliverable we focus on facilitating the easier and more intuitive execution of this task by means of a vagueness annotation tool, called **Vagueness Annotator for Ontologies (VAO)**[6]. VAO helps vagueness annotators in two ways:

- It helps them detect vague ontology entities, such as classes and properties, in an automatic way by means of a **Vague Entity Classifier** Alexopoulos and Pavlopoulos (2014) that analyzes upon entities' linguistic definitions and decides whether they are vague or not.
- It helps them annotate ontologies with vagueness descriptions according to the Vagueness Ontology, without having to know the latter's implementation details; this is done by means of a structured Q&A process that guides annotators into defining the appropriate information to be included into such a description.

2.1 The Vagueness Ontology

The Vagueness Ontology[7] has been developed following the *SAMOD*[8] (*Simplified Agile Methodology for Ontology Development*) methodology and its relevant documentation is available online[9]. In this section, we focus on describing the requirements the ontology has been designed to satisfy and the main elements it consists of.

2.1.1 Vagueness Ontology Requirements

In an ontology, vagueness may primarily appear in the definitions of classes, object and datatype properties, and datatypes. A class is vague if, in the given domain, context or application scenario, it admits borderline cases, namely if there are (or could be) individuals for which it is indeterminate whether they instantiate the class. Typical vague classes are attributions, namely classes that reflect qualitative states of entities (e.g., *"TallPerson"*, *"ExperiencedResearcher"*, etc.). Similarly, an object property (relation) is vague if there are (or could be) pairs of individuals for which it is indeterminate whether they stand in the relation (e.g., *"hasGenre"*, *"hasIdeology"*, etc.). The same applies for datatype properties and pairs of individuals and literal values. Finally, a vague datatype consists of a set of vague terms. An example is the datatype *"RestaurantPriceRange"* when this comprises the terms *"cheap"*, *"moderate"* and *"expensive"*.

The Vagueness Ontology should enable the annotation of an ontological entity (class, relation or datatype) with a description of the nature and characteristic of its vagueness. In particular, the first thing such a description should explicitly state is whether the entity is actually vague or not. For example, the ontology class *"StrategicClient"* defined as *"A client that has a high value for the company"* is (and should be annotated as) vague while the definition of *"AmericanCompany"* as "A company that has legal status in the Unites States" is not. Moreover, it can often be the case that a seemingly vague

[6] Vagueness Annotator for Ontologies: http://www.essepuntato.it/vao.

[7] Available at http://www.essepuntato.it/2013/10/vagueness

[8] Available at http://www.essepuntato.it/samod

[9] Available at http://www.essepuntato.it/2013/10/vagueness/documentation

element can have a non-vague definition (e.g. *"TallPerson"* when defined as "A person whose height is at least 180cm"). Then this element is not vague in the given ontology and that is something that needs to be explicitly stated.

The second important vagueness characteristic to be explicitly represented is its type. Vagueness can be described according to at least two complementary types: quantitative (or degree) vagueness and qualitative (or combinatory) vagueness Hyde (2008). A predicate has degree-vagueness if the existence of borderline cases stems from the lack of precise boundaries for the predicate along one or more dimensions (e.g. "*bald*" lacks sharp boundaries along the dimension of hair quantity while "*red*" can be vague for both brightness and saturation). A predicate has combinatory vagueness if there are a variety of conditions pertaining to the predicate, but it is not possible to make any crisp identification of those combinations which are sufficient for application. A classical example of this type is "*religion*" as there are certain features that all religions share (e.g. beliefs in supernatural beings, ritual acts) yet it is not clear which are able to classify something as a religion. Based on this typology, we suggest that for a given vague entity it is important to represent and share the following explicitly:

- **The type of the entity's vagueness**: Knowing whether an entity has quantitative or qualitative vagueness is important as elements with an intended (but not explicitly stated) quantitative vagueness can be considered by others as having qualitative vagueness and vice versa. Assume, for example, that a company's CEO does not make explicit that for a client to be classified as strategic, the amount of its R&D budget should be the only factor to be considered. Then, even though according to the CEO the vague class *"StrategicClient"* has quantitative vagueness in the dimension of the R&D budget amount, it will be hard for other company members to share the same view as this term has typically qualitative vagueness.

- **The dimensions of the term's quantitative vagueness:** When the entity has quantitative vagueness it is important to state explicitly its intended dimensions. E.g., if a CEO does not make explicit that for a client to be classified as strategic, its R&D budget should be the only pertinent factor, it will be rare for other company members to share the same view as the vagueness of the term "*strategic*" is multi-dimensional.

Furthermore, vagueness is **subjective** and **context dependent**. The first has to do with the same vague entity being interpreted differently by different users. For example, two company executives might have different criteria for the entity "*StrategicClient*", the one the amount of revenue this client has generated and the other the market in which it operates. Similarly, context dependence has to do with the same vague entity being interpreted or applied differently in different contexts even by the same user; hiring a researcher in industry is different to hiring one in academia when it comes to judging his/her expertise and experience.

Therefore we additionally suggest that one should explicitly represent the **creator** of a vagueness annotation of a certain entity as well as the **applicability context** for which the entity is defined or in which it is used in a vague way. In particular, context-dependent can be i) the description of vagueness of an entity (i.e. the same entity can be vague in one context and non-vague in another) and ii) the dimensions related to a description of vagueness having quantitative type (i.e. the same entity can be vague in dimension A in one context and in dimension B in another). Please note that here we adopt the context-as-a-box metaphor Benerecetti et al. (2000) according to which a context is a "box" that contains knowledge in form of logical statements and whose boundaries are determined by specific contextual attributes (e.g. location, time, purpose etc). When a vague term is related to a particular context, then this context has the jurisdiction to interpret the term's meaning and assess its validity in given statements Bao et al. (2010).

Summarising the above, the Vagueness Ontology should enable users to ask the following competency questions about the entities of an ontology:

- *What entities have been explicitly defined either as vague or non-vague?*
- *What entities that have been defined both as vague and non-vague at the same time and why?*
- *What entities of a specific type (e.g., classes) have been defined either as vague or non-vague?*
- *What entities are characterised by a specific vagueness type?*
- *What entities have been recognised as vague, by whom and according to which vagueness type (if any)?*
- *What entities have quantitative vagueness and in what dimensions?*
- *What entities have quantitative vagueness, in what dimensions and what is the context of their dimensions (if any)?*
- *What entities are vague, in what contexts and according to whom?*

2.1.2 Ontology Anatomy

An overall view of the Vagueness Ontology (VO) is depicted in Figure 1 via a Graffoo diagram **?** that describes its main classes and properties. VO uses several entities defined in external ontologies, i.e., the PROV-O Lebo et al. (2013) (prefix *prov*), OADM Sanderson et al. (2013) (prefix *oa*), and the Situation ontology design pattern[10] (prefix *sit*). To show how to use the various entities of the ontology to describe vagueness/non-vagueness annotations, we introduce the following natural language scenario:

> The object property *ex:isExpertInResearchArea* is considered vague by John Doe in the context of researcher hiring. Moreover, he describes it as quantitatively vague since, for him, expertise is relevant to the number of her publications and projects; two different dimensions that he thinks relates to the contexts of Academia (i.e., number of relevant publications) and Industry (i.e., number of relevant projects).

The main class of the ontology is *VaguenessAnnotation*, which describes any annotation (i.e., *oa:Annotation*) of an ontological entity with information about its vagueness. A vagueness annotation is a particular act done by someone (i.e., an agent, identified by an individual of the class *prov:Agent*) who associates a description of vagueness/non-vagueness (called the body of the annotation, and defined through the property *oa:hasBody*) to the entity in consideration (called the target of the annotation, and defined through the property *oa:hasTarget*). This is formalised as follows[11]:

```
Class: VaguenessAnnotation
  SubClassOf: prov:Entity that prov:wasAttributedTo some prov:Agent,
              oa:Annotation that (oa:hasTarget exactly 1) and
                (oa:hasBody min 1
                  (DescriptionOfNonVagueness or DescriptionOfVagueness))
```

[10] Available at http://www.ontologydesignpatterns.org/cp/owl/situation.owl.

[11] All the entities of the Vagueness Ontology are introduced in Manchester Syntax Horridge and Patel-Schneider (2012), while the examples of use of the ontology are presented in Turtle Prud'hommeaux and Carothers (2013).

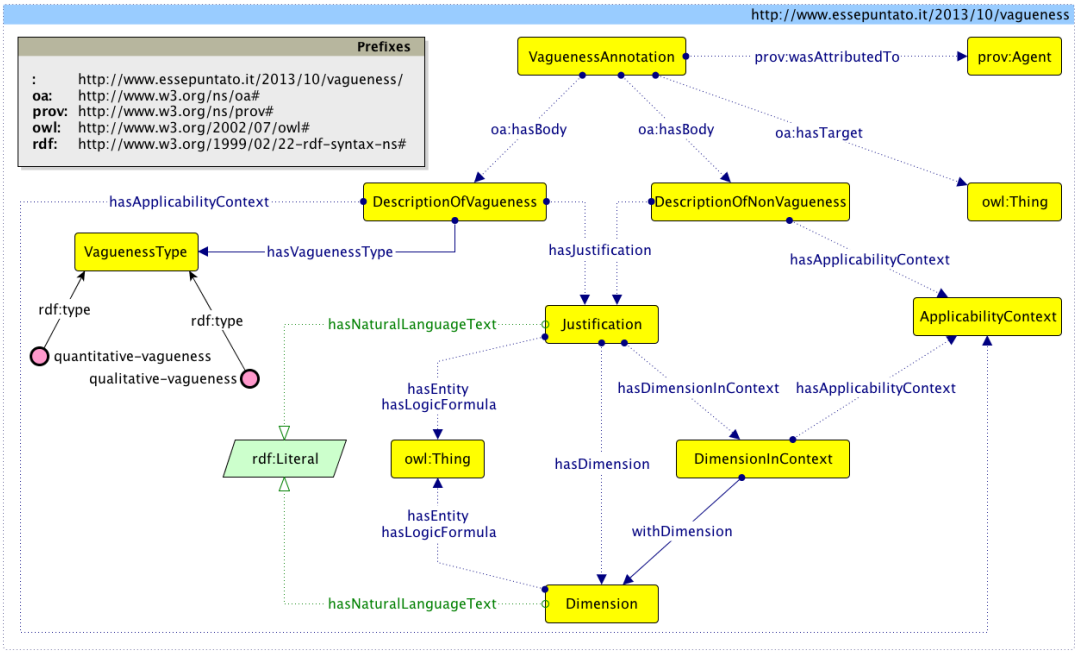

Figure 1: The Graffoo diagram of the overall structure of the Vagueness Ontology.

Considering the aforementioned example, the annotation made by John Doe can be expressed as follows:

```
@prefix : <http://www.essepuntato.it/2013/10/vagueness/> . @prefix
ex: <http://www.essepuntato.it/resource/> . ex:annotation a
:VaguenessAnnotation ; prov:wasAttributedTo ex:john-doe ;
  oa:hasBody ex:description-of-vagueness ;
  oa:hasTarget ex:isExpertInResearchArea .
ex:isExpertInResearchArea a owl:ObjectProperty . ex:john-doe a
prov:Agent .
```

A vagueness annotation must specify a description of vagueness or non-vagueness for the annotated entity, in the form of an instance of the class *DescriptionOfVagueness* or *DescriptionOfNonVagueness* respectively. Vagueness descriptions must specify a vagueness type (one of the individuals of the class *VaguenessType*, i.e., *quantitative-vagueness* and *qualitative-vagueness*), and must provide at least one justification (i.e., an individual of the class *Justification*) for considering the target ontological entity vague. The individuals of the class *DescriptionOfNonVagueness*, instead, require only the specification of at least one justification. This class is meant to be used for entities that would typically be considered vague but which, for some reason, in the particular ontology are not (e.g. the *"TallPerson"* example in section 2.1.1). Formalisation here is as follows:

```
Class: DescriptionOfNonVagueness SubClassOf: hasJustification min 1
Class: DescriptionOfVagueness
  SubClassOf: hasJustification min 1 , hasVaguenessType exactly 1
ObjectProperty: hasJustification
```

```
  Domain: DescriptionOfNonVagueness or DescriptionOfVagueness
  Range: Justification
ObjectProperty: hasVaguenessType
  Domain: DescriptionOfVagueness Range: VaguenessType
```

Considering again the previous example, the John Doe's description of vagueness can be defined as follows:

```
ex:description-of-vagueness a :DescriptionOfVagueness ;
  :hasJustification ex:justification ;
  :hasVaguenessType :quantitative-vagueness .
```

The justifications of descriptions of vagueness/non-vagueness (i.e., individuals of the class *Justification*) aim at explaining the possible reasons behind such descriptions. Vagueness dimensions, in turn, (i.e., individuals of the class *Dimension* referred by the object property *hasDimension* and being always part of a justification) always refer to descriptions of quantitative vagueness and indicate some measurable characteristic of the annotated entity in which it is vague. Both justifications and dimensions can be defined as natural language text (i.e., the data property *hasNaturalLanguageText*), an entity (i.e. the object property *hasEntity*), a more complex logic formula (i.e., the object property *hasLogicFormula*) or any combination of them. The relevant formalisation is as follows:

```
Class: Justification
  SubClassOf: hasNaturalLanguageText some rdfs:Literal or
                hasEntity some owl:Thing or hasLogicFormula some owl:Thing
ObjectProperty: hasDimension
  Domain: Justification that inverse hasJustification only
            (DescriptionOfVagueness that
              (hasVaguenessType value quantitative-vagueness))
  Range: Dimension
Class: Dimension
  SubClassOf: hasNaturalLanguageText some rdfs:Literal or
                hasEntity some owl:Thing or hasLogicFormula some owl:Thing
```

Please note that while the properties *hasEntity* and *hasLogicFormula* share the same range class, i.e., *owl:Thing*, their intended meaning is different. The former property can be used to specify a certain resource (e.g., *dbpedia:H-index*) as (part of) a justification of a certain description. The latter property, instead, is used to link to a resource, which provides a justification, that actually "puns" a particular restriction or constraint on certain entities, e.g., *ex:hasNumberOfPublication some integer[>0]*.

Continuing the previous example, the justification and the related dimensions can be described as follows:

```
ex:justification a :Justification ;
  :hasNaturalLanguageText "It is not possible to define the exact minimum
  number of relevant publications and projects that make a researcher
  expert in a given area." ;
  :hasDimension ex:dimension-publications , ex:dimension-projects .
ex:dimension-publications a :Dimension ;
  :hasNaturalLanguageText "The number of relevant publications." .
ex:dimension-projects a :Dimension ;
  :hasNaturalLanguageText "The number of relevant projects." .
```

As introduced before, descriptions of vagueness/non-vagueness and related dimensions can be characterised by particular contexts of application (i.e., individuals of the class *ApplicabilityContext*), which

means that they can be applied within the boundaries of such particular contexts (i.e., the same entity can be vague in one context and non-vague in another). The contextualisation of descriptions is facilitated by an assertion between the description in consideration and the related context through the object property *hasApplicabilityContext*. In the case of dimensions, on the other hand, the context-dependent object is the *relation* between justifications and dimensions. Thus, to represent this, we reify the relation linking a justification to a dimension using an instance of the class *DimensionInContext*, that allows one to specify and the applicability context of such relation. VO formalises this as follows:

```
ObjectProperty: hasApplicabilityContext
  Domain: DescriptionOfNonVagueness or DescriptionOfVagueness or
          DimensionInContext
  Range: ApplicabilityContext
ObjectProperty: hasDimensionInContext
  Domain: Justification that inverse hasJustification only
          (DescriptionOfVagueness that
            (hasVaguenessType value quantitative-vagueness))
  Range: DimensionInContext
Class: DimensionInContext
  SubClassOf: sit:Situation , withDimension exactly 1 ,
              hasApplicabilityContext exactly 1
ObjectProperty: withDimension
  Characteristics: Functional SubPropertyOf: sit:isSettingFor
  Domain: DimensionInContext Range: Dimension
```

According to the above definitions, it is possible to complete the description of the aforementioned example as follows:

```
ex:description-of-vagueness
  :hasApplicabilityContext ex:researcher-hiring-context .
ex:researcher-hiring-context a :ApplicabilityContext .
ex:justification :hasDimensionInContext
  ex:dimension-publications-in-context , ex:dimension-projects-in-context .
ex:dimension-publications-in-context a :DimensionInContext ;
  :withDimension ex:dimension-publications ;
  :hasApplicabilityContext ex:academia-context .
ex:dimension-projects-in-context a :DimensionInContext ;
  :withDimension ex:dimension-projects ;
  :hasApplicabilityContext ex:industry-context .
ex:academia-context a :ApplicabilityContext . ex:industry-context a
:ApplicabilityContext .
```

This approach allows the reuse of the same dimension in different contexts and reasoners to infer automatically all the *hasDimension* assertions starting from the individuals of the class *DimensionInContext* by means of the sub-property chain *hasDimensionInContext o withDimension* defined in the object property *hasDimension*.

2.2 The Vague Entity Classifier

The Vague Entity Classifier[12] Alexopoulos and Pavlopoulos (2014) is a system that is able to distinguish between vague and non-vague term word senses and, consequently, to determine whether a given

[12] Available as a POST REST Web Service at http://glocal.isoco.net/kplatform/api/VaguenessDetection/determineDefinitionVagueness

Vague Adjectives	Non Vague Adjectives
Abnormal: not normal, not typical or usual or regular or conforming to a norm	Compound: composed of more than one part
Impenitent: impervious to moral persuasion	Biweekly: occurring every two weeks
Notorious: known widely and usually unfavorably	Irregular: falling below the manufacturer's standard
Aroused: emotionally aroused	Outermost: situated at the farthest possible point from a center
Yellowish: of the color intermediate between green and orange in the color spectrum, of something resembling the color of an egg yolk	Unfeathered: having no feathers

Table 1: Sample Vague and Non-Vague Adjective Senses

ontology entity linguistic definition is vague or not. For example, the definition of the ontology class *"StrategicClient"* as *"A client that has a high value for the company"* is (and should be) characterized as vague while the definition of *"AmericanCompany"* as *"A company that has legal status in the Unites States"* is not. As we will explain in subsequent sections, the classifier's role within the framework is to provide an initial suggestion to the users as to which of the ontology's entities are vague and which not.

From a technical point of view, the classifier uses the Multinomial Naive Bayes algorithm and it is trained with vague and non-vague sense examples, carefully constructed from $WordNet$. In particular, the training dataset we used comprised 2,000 adjective senses, collected from $WordNet$, such that 1,000 of them had a vague definition and the rest a non-vague one. A sample of these senses is shown in Table 1 while the whole dataset is available online[13]. The dataset was initially constructed by an ontology expert but, since the task of classifying a text as vague or not can be quite subjective, we also asked from two other human judges to annotate a subset of the dataset's definitions (100), and we measured inter-annotator agreement between all three. We found mean pairwise JPA (Joint Probability of Agreement) equal to 0.81 and mean pairwise K (Cohen's Kappa) equal to 0.64, both of which indicate a reasonable agreement.

Finally, to assess the accuracy of the classifier, we performed experiments with both $Wordnet$ word senses and real ontology definitions, derived from the CiTo[14] ontology; the achieved accuracy in both cases was around **82%**. As we will explain in the next section, the role of the Vagueness Classifier within our framework is to provide an initial suggestion to the users as to whether an ontology entity is vague or not.

2.3 The Vagueness Annotator for Ontologies

The Vagueness Annotator for Ontologies (VAO) facilitates the annotation of ontologies according to the Vagueness Ontology. The typical process an ontology developer follows to perform this annotation consists of the following steps:

1. **Ontology Specification**: In this step, the user specifies to the system the ontology to be annotated.

[13] http://glocal.isoco.net/datasets/VagueSynsets.zip
[14] http://purl.org/spar/cito/

Currently this is done by means of the ontology's URL, but additional ways will be implemented in future versions.

2. **Ontology Parsing and Vague Entity Identification**: In this step, the system parses the provided ontology, generates descriptions for the ontology's main entities (classes, object properties and datatypes) that can be vague and provides for each entity a form to facilitate its vagueness-related annotation (see figure 2. At the same time it applies the Vagueness Entity Classifier to the linguistic definition of each entity (usually available as an `rdfs:comment`) in order to provide the user with an initial suggestion as to whether the entity is vague or not (provided as the default selected value in the relevant form).

3. **Entity Vagueness Annotation**: In this step, the user utilizes the provided form under the description of each ontology entity in order to specify its vagueness-related characteristics, as specified in the Vagueness Ontology (see section 2.1). The form "guides" the user in providing the appropriate combinations of of relevant meta-information by means of the Q&A process of figure 5 (explained in detail below).

4. **Vagueness Ontology Instance Generation**: In this final step, the system automatically generates a vagueness annotation for the provided ontology in the form of an RDF Vagueness Ontology instance. This instance can then be incorporated to the initial ontology so that the latter's users can execute the vagueness-related competency questions described in section 2.1.

Figure 2: Ontology Parsing and Vague Entity Identification

The annotation of a given ontology entity, as shown in figure 5, starts with the user determining whether the entity is actually vague or not. If the entity is non-vague then the user needs to determine whether this is the case only for the particular ontology/domain or not. For example, the entity *"Film"* is generally non-vague. whereas the entity *"TallPerson"* when defined as *"A Person with height greater than 185cm"* is not vague even though the term *"Tall"* is generally vague. To make this clear, the user

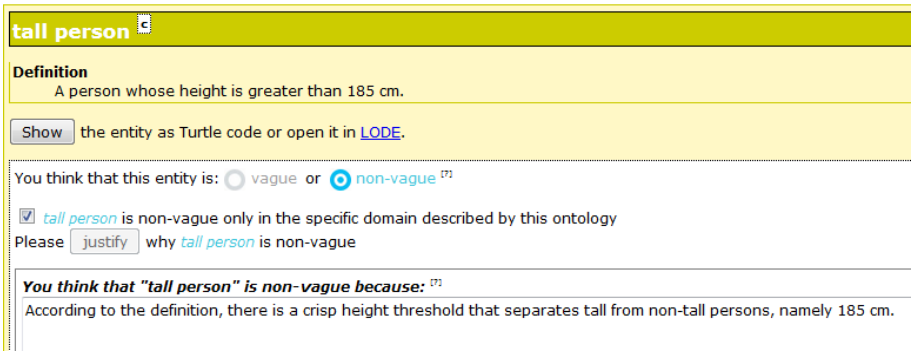

Figure 3: Annotation of a Non-Vague Entity

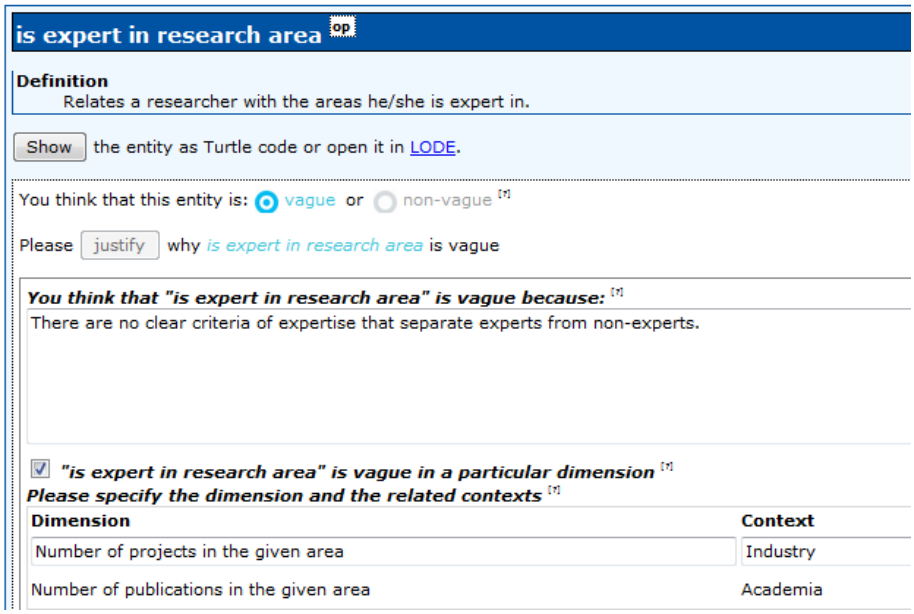

Figure 4: Annotation of a Vague Entity with Context-Dependent Dimensions

should in the latter case provide an explicit justification for the entity's non-vagueness, as shown in figure 3.

On the other hand, if the entity is vague then the user can provide a justification for its vagueness and, in addition, he/she can (optionally) determine any particular dimensions in which the entity is vague. For example, it might be that the relation *"isExpertInResearchArea"* is vague in the dimension *"NumberOfPublications"* in the context of *"Academia"* and in the dimension *"NumberOfProjects"* in the context of *"Industry"*. Figure 4 shows how this information is defined via the Vagueness Annotator interface.

If vagueness dimensions are defined, then the entity's vagueness is taken to be quantitative, otherwise it is considered qualitative. In all cases, it should be clarified that, apart from the vague/non-vague

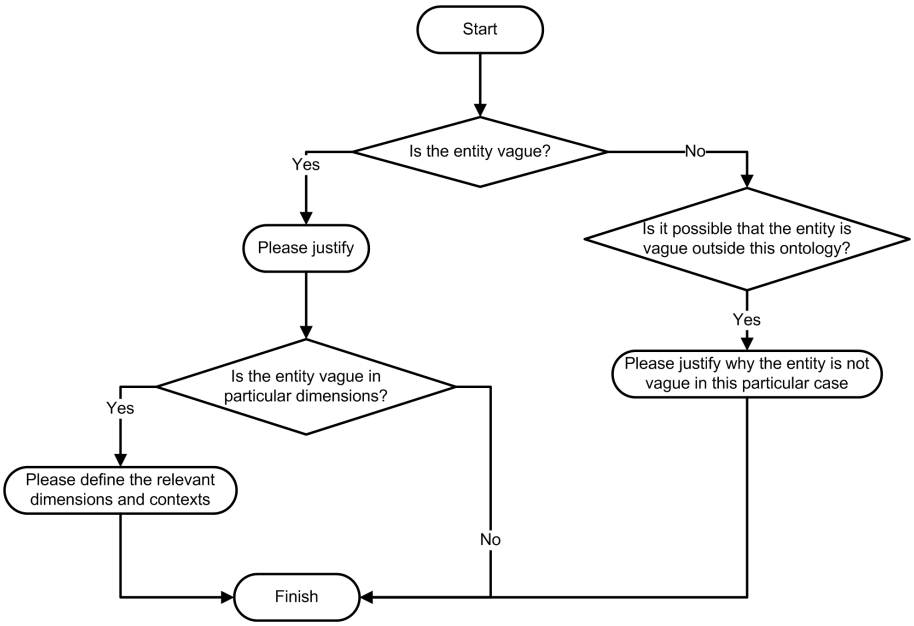

Figure 5: Entity Vagueness Annotation Process

suggestion, the Vagueness Annotator is not currently capable of automatically generating any of the rest of the above information, like justifications, dimensions or contexts. This is something that needs to be done by knowledge engineers and domain experts who will determine this information based on their expertise, experience and/or the particular domain or application context of the given ontology.

2.4 Related Work

The practice of developing dedicated tools to cope with specialized aspects and dimensions of the ontology development process has been exemplified by several relevant works. For example, OntOWLClean Welty (2006) is an OWL-based tool that facilitates the easier and more intuitive application of OntoClean Guarino and Welty (2009), a well-known methodology for the evaluation of the ontological adequacy and logical consistency of taxonomic relationships. Another related tool is OntoParts Keet et al. (2012) that helps ontology authors decide which particular types of part-whole relations are appropriate for their ontology. An evolution of OntoParts is FORZA Keet et al. (2013) that supports also the task of linking a domain ontology to categories of the DOLCE foundational ontology Masolo et al. (2003).

With respect to vagueness, the most relevant existing framework is Fuzzy OWL 2 [15], a Protege plugin that allows ontology developers to encode fuzzy ontologies by means of OWL 2 annotation properties Bobillo and Straccia (2011). Nevertheless, as we have argued in Alexopoulos et al. (2013), the focus of this approach (and of fuzzy ontologies in general) is on enabling the definition and automated processing of fuzzy degrees of vague ontology entities and not so much on clarifying their intended interpretation (e.g., the dimensions of a given vague concept). In that sense, our work is complementary to fuzzy ontologies, covering a related but different need.

[15] http://nemis.isti.cnr.it/ straccia/software/FuzzyOWL/

3 Reusing Linked Open Vocabularies

So far, Linked Data principles and practices are being adopted by an increasing number of data providers, getting as result a global data space on the Web containing hundreds of LOD datasets Heath and Bizer (2011). There are already several guidelines for generating, publishing, interlinking, and consuming Linked Data Heath and Bizer (2011). An important task, within the generation process, is to build the vocabulary to be used for modelling the domain of the data sources, and the common recommendation is to reuse as much as possible available vocabularies Heath and Bizer (2011); Hyland et al. (2014). This reuse approach speeds up the vocabulary development, and therefore, publishers will save time, efforts, and resources.

There are research efforts, like the NeOn Methodology Suarez-Figueroa et al. (2012), the Best Practices for Publishing Linked Data - W3C Working Group Note Hyland et al. (2014), and the work proposed by Lonsdale et al. Lonsdale et al. (2010). However, at the time of writing we have not found specific and detailed guidelines that describe how to reuse available vocabularies at fine granularity level,i.e., reusing specific classes and properties. Our claim is that this difficulty in how to reuse vocabularies at fine grained level is one of major barriers to the reuse of vocabularies on the Web and in consequence to deployment of Linked Data.

Moreover, the recent success of Linked Open Vocabularies (LOV[16]) as a central point for curated catalog of ontologies is helping to convey on best practices to publish vocabularies on the Web, as well as to help in the Data publication activity on the Web. LOV comes with many features, such as an API, a search function and a SPARQL endpoint.

In this section we propose a set of guidelines for this task, and provide technological support by means of a plugin for Protégé, which is one of the popular frameworks for developing ontologies in a variety of formats including OWL, and RDF(S). It is backed by a strong community of developers and users in many domains. One success on Protégé also depends on the availability to extend the core framework adding new functionalities by means of plug-ins. In addition, we propose to explore, design and implement a plug-in of LOV in Protégé for easing the development of ontologies by reusing existing vocabularies at fine grained level. The tool helps to improve the modeling and reuse of ontologies used in the LOD cloud.

3.1 Reusing vocabulary terms when building ontologies

In this section, we describe the workflow for reusing available vocabulary terms when building ontologies based on an existing catalogue of vocabularies. Recall that we do not specify which will be the expressivity of the resulting ontology. We foresee that the resulting ontology is likely to be a lightweight ontology. However the data publisher is responsible for adding more semantics when reusing existing vocabulary terms. Figure 6 depicts the proposed workflow.

In a nutshell, the activity of building vocabularies by reusing available vocabulary terms consists of the following tasks:

- Search for suitable vocabulary terms to reuse from any vocabulary repository, such as BioPortal[17], LOV[18], Bartoc.org[19], etc. The search should be conducted by using the terms of the application domain.

[16] http://lov.okfn.org/dataset/lov/
[17] http://bioportal.bioontology.org/
[18] http://lov.okfn.org/
[19] http://bartoc.org

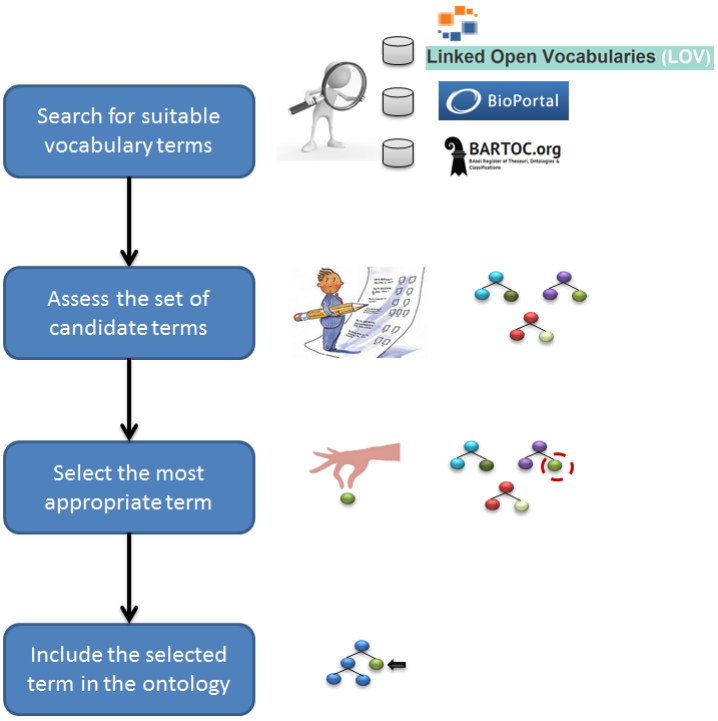

Figure 6: Workflow for reusing available terms when building ontologies

- Assess the set of candidate terms from the vocabulary repository. The goal of this task is to find out if the candidate terms are useful and relevant for the ontology being built. For this purpose we can rely on metrics provided by the vocabulary repository. In the particular case of LOV, the results include a score related to their "importance" in the corpus for each term retrieved. For every vocabulary in the LOV, terms (classes, properties, datatypes, instances) are indexed and a full text search feature is offered[20]. The Linked Open Vocabularies search engine ranking algorithm takes into account the popularity of the term within the LOV ecosystem and most importantly assigned a different score depending on which label property a searched term matched Vandenbussche et al. (2015).

- Select the most appropriate term taking into account the assessment task. To distinguish between those candidate terms which are the most suitable, we propose to use the following criteria (i) the stability of the URI namespace, (ii) the trustworthiness of the publisher of the vocabulary and (iii) the presence or absence of a community using the vocabulary. With respect to LOV repository we can also rely on the score of terms.

- Include the selected term in the ontology that has being developed. There are at least three alternatives in this case

  - Include the selected term and use it directly, i.e., as it is, in the local ontology by defining local axioms to or from that term in the local ontology.

[20] `http://lov.okfn.org/dataset/lov/terms`

- Include the selected term, create a local term, and define the `rdfs:subClassOf` or `rdfs:subPropertyOf` axiom to relate both terms.
- Include the selected term, create a local term, and define the `owl:equivalentClass` or `owl:equivalentProperty` axiom to relate both terms.

It is worth mentioning the following considerations

- We want to promote a "responsible" reutilization, i.e., if the user already found something interesting in a particular vocabulary, the user would probably like to dig a little bit more on it, suggesting other terms in it, before jumping somewhere else.
- We want to keep consistency of reused terms . Therefore, it would be necessary to (1) check and update the domain/range of the selected terms; (2) change cardinalities; and (3) add some restrictions.
- We want to bring as many knowledge as possible from the reused term, e.g., for a particular property it would be necessary to check if a inverse property exists and import it as well.

3.2 Linked Open Vocabularies (LOV)

The intended purpose of the LOV Vandenbussche et al. (2015) is to help users to find and reuse terms of vocabularies in Linked Open Data. For achieving that purpose, the LOV gives access to vocabularies metadata and terms using programmatic access with APIs. LOV[21] catalogue is a hub of curated vocabularies used in the Linked Open Data Cloud, as well as other vocabularies suggested by users for their reuse. Some of the three main features of LOV are

1. Search ontologies: The main LOV's feature is the search of vocabulary terms. These vocabularies are categorized within LOV according to the domain they address. In this way, LOV contributes to ontology search by means of (a) keyword search and (b) domain browsing.
2. Assess terms: LOV provides a score for each term retrieved by a keyword search. This score can be used during the assessment stage.
3. Interconnect ontologies: In LOV, vocabularies rely on each other in seven different ways. These relationships are explicitly stated using VOAF vocabulary[22].

Futhermore, the LOV APIs give a remote access to the many functions of LOV through a set of RESTful services[23]. The basic design requirements for these APIs is that they should allow applications to get access to the very same information humans do via the User Interfaces. More precisely the

1. vocabulary terms (classes, properties, datatypes and instances) providing functions to query the LOV search engine, with autocompletion features;
2. vocabulary browsing, in which a client can get access to the current list of vocabularies contained in the LOV catalogue and search for vocabularies for further purpose;
3. agents, or the ontology's creators, contributors or organizations. It also contains the search with autocompletion of an agent.

[21] `http://lov.okfn.org/dataset/lov/`
[22] `http://lov.okfn.org/vocab/voaf`
[23] `http://lov.okfn.org/dataset/lov/apidoc/`

3.3 ProtégéLOV

ProtégéLOV is an open source tool that provides support to the methodological guidelines described in section 3.1. It is written in Java programming language as a plugin for the Protégé ontology editor. It can be easily installed by just copying the jar file provided at the ProtégéLOV website[24] into the plugins directory of an existing Protégé installation (see Figure 7). We have tested with ProtÃ©gÃ© 4.3 and latest.

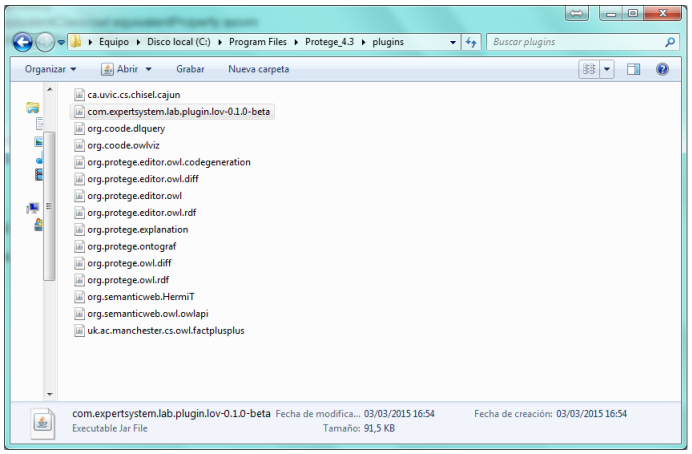

Figure 7: Plugins folder

Then, upon a new start, the user should select *Linked Open Vocabularies* item, within the *Ontology views* menu item, as it shown in Figure 8.

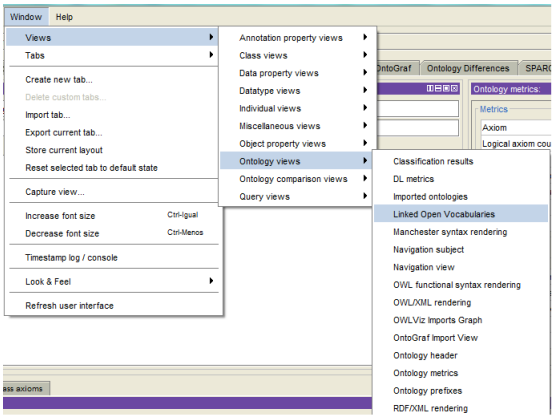

Figure 8: Menu item

Currently, ProtégéLOV provides the following five functionalities:

1. **Search for a particular term (class or property) in LOV repository**. The user selects a particular term from the *Class*, *Object property* or *Data property* navigator. Next, the user switches

[24] `http://labs.mondeca.com/protolov/`

to the *Linked Open Vocabularies View* and performs a search on the LOV repository. The system takes as input the selected term, and calls the LOV REST API to get the LOV terms that match the criteria. The plugin provides the following information (if it is available) for each term (1) URI, (2) score, (3) label, (4) local name, and (5) usage note.

2. **Reuse directly a particular term from LOV repository** The user selects the *reuse directly* option. Next, the system replaces the selected term from the *Class*, *Object property* or *Data property* navigator, and replaces by the selected term from LOV.

3. **Add the particular term and define the** `owl:equivalentClass` **or** `owl:equivalentProperty` **axiom**. The user selects the *add entity and equivalent axiom* option. Next, the system includes the new term on the local ontology, and defines `owl:equivalentClass` or `owl:equivalentProperty` axiom to relate both terms.

4. **Add the particular term and define the** `rdf:subClassOf` **or** `rdfs:subPropertyOf` **axiom**. The user selects the *add entity and sub-entity axiom* option. Next, the system includes the new term on the local ontology, and defines `rdfs:subClassOf` or `rdfs:subPropertyOf` axiom to related both terms.

Figure 9 depicts the three main actions the user can perform.

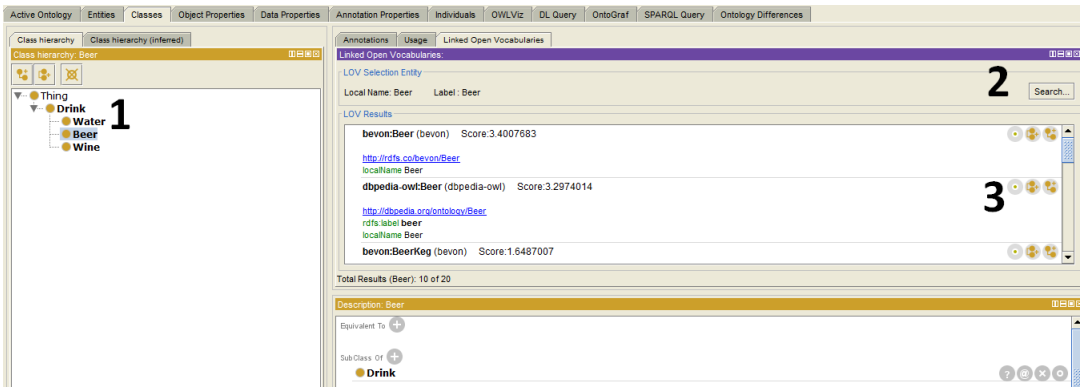

Figure 9: Three actions currently available in the plugin after looking up a term in LOV catalogue: (a) reuse it directly, (b) add equivalent axiom and (c) add subEntity axiom

4 Recommending Semantic Dataset Summaries

4.1 Background

From the beginning of the K-Drive project, one key question we've been trying to enable data consumers to answer is the following: *"Given an application scenario where semantic data is required, how suitable is a given existing dataset for the purposes of this scenario?"*. With that goal in mind, work packages WP1 and WP4 focused on specifying, designing and developing a semantic data summarization system that allows users to define and execute their own custom dataset summarization tasks in order to decide whether and to what extent given datasets are suitable for their goals.

In WP7, we have shifted our focus on how the above system can be personalized, i.e., how it can be adapted to the particular preferences, goals and characteristics of each user. The specific task that we have identified as being in need of personalization is the recommendation by the system, of summarization tasks to (new and existing) users. This is a useful feature of the system in cases when i) the user's intended summarization task has already been defined and applied by other users or ii) when the user does not know, maybe from lack of expertise or experience, the complete range of dataset aspects that he/she needs to assess for a given task/goal.

In this context, in D7.1 we defined a User Model that captures the preferences of users with respect to the summarization tasks they use/prefer for particular goals. Given these preferences, collaboration filtering mechanisms can then be used to recommend summarizations tasks to (either new or existing) users, based on user and task similarity. In this deliverable, we focus on how this recommendation can be best facilitated in the UI level by describing a corresponding novel interactive user interface based on a novel interaction paradigm called Collaboration Spheres.

4.2 The Collaboration Spheres Interaction Paradigm

The Collaboration Spheres aim at providing a mechanism to explore, share and reuse Summarization Tasks (STs) and User Expertise based on the exploitation of semantic descriptions, relations and similarities between STs and users in order to provide advanced search and recommendation functionalities. This type of exploratory search is especially appropriate in domains where social aspects like collaboration and the notion of a community play a relevant role in order to incrementally expand the knowledge assets of such community.

In general, there are different ways in which users can express an information need (or query) in order to retrieve content from a repository. Well-known approaches include, among others, faceted search, where the user selects the most relevant features from a predefined set in order to constrain the search space, and free text query interfaces, which rely on natural language processing technologies to parse and match the query against the overall content. However, it is usually the case that users lack the precise knowledge about the exact features of the information to be retrieved or the skills required to express them in specific query formalisms.

The difficulty is considerably lower when, instead of formulating a query, users are enabled to search by example, exploiting the potential similarities between such examples and the results. We follow the same principle behind the use of examples e.g. in education, in order to facilitate the assimilation of complex concepts by students. By selecting a number of representative exemplars users can express the properties that must be observed in the expected results without explicitly or formally describing such properties. This query-by-example method relieves users from the task of formulating potentially complex queries, delegating such complexity to the underlying system.

Additionally, this approach allows users to explore the search space through the **context of interest** described by the aggregated properties of the selected exemplars.

For example, by putting together different STs that are related to the goal of Semantic Annotation and analyzing the relatedness of this context of interest against the Summarization Tasks contained in the repository, it is possible to establish a connection between this goal and other data-related goals. This kind of exploratory search is specially aimed towards gaining new insights on existing information and discovering relations between them that were not previously explicit. Of course, recommendation mechanisms are the key to establish such similarities between a context of interest and the search space; however, in order for such mechanisms to be effective, they need to be provided to users in a way that simplifies interaction, especially when it comes to creating the context of interest and visualizing the results of the exploratory search.

The Collaboration Spheres visualization metaphor and the prototype interface implementing it make use of concentric spheres that represent different types of similarity metrics between the context of interest, expressed by the user as a collection of goals, STs and other users that the user finds relevant for a particular purpose, and the results obtained by the recommendation system.

In particular, the user interface associated to the Collaboration Spheres visual metaphor contains three main spheres:

- **The User.** It displays the active user.
- **The Inner Sphere (context of interest).** It represents the context of interest. This circle contains the users, goals and STs that are selected or pre-defined by the active user. In order to create a context of interest, the inner sphere is populated by drag-and-dropping relevant items and users from existing lists.
- **The Outer Sphere (Recommended STs).** It contains the Summarization Tasks that are recommended to the active user based on his/her profile and the context of interest defined in the previous sphere. The recommendation obtained by this layer uses a collaborative filtering algorithm that works under the assumption that "similar users utilize similar summarization tasks for similar goals".

The distance between the center, i.e., the user and the context of interest, and the outer sphere, where recommendation results are displayed, provides a notion of confidence about the recommendations. The closer to the center, the more specific the recommendation result will be with respect to the user and the current context of interest. Moreover, the collaboration spheres interface is agnostic from the different similarities calculated by the underlying recommender system.

The implementation of the Collaboration Spheres for Summarization Task Recommendation is still undergoing; nevertheless the user interface is expected to look like the one in figure 10. The figure depicts an implementation of the CF paradigm for the American Psychology Association (APA), the major US publisher in the Psychology domain. The main interest of APA was to provide their users with intuitive means to identify relevant authors and articles not only in terms of the topics contained but mainly through their potential relations with other articles and authors.

The prototype interface for APA consists of 4 spheres. The first two are the User and Context spheres while the other two have to do with recommendations. In particular, the third sphere contains other authors that are recommended based on the context of interest while the fourth sphere contains recommended articles. Beyond the spheres, the interface also includes:

- A Tag Cloud: This shows, in a summarized and visual way, the set of tags and concepts associated to the aggregated users and articles contained in the Collaboration Spheres for a particular context of interest.
- Contextual Item Information: After clicking on any of the items (an author or an article), a brief summary of information about such item, e.g. other articles by the same author and the abstract of the article, appear in this box.

5 Conclusions

With respect to section 2 we presented the design of the Vagueness Annotator for Ontologies (VAO), a user interface that facilitates the annotation of vague ontological entities with explicit formal descriptions of the nature and characteristics of their vagueness. Our immediate future work is to perform a

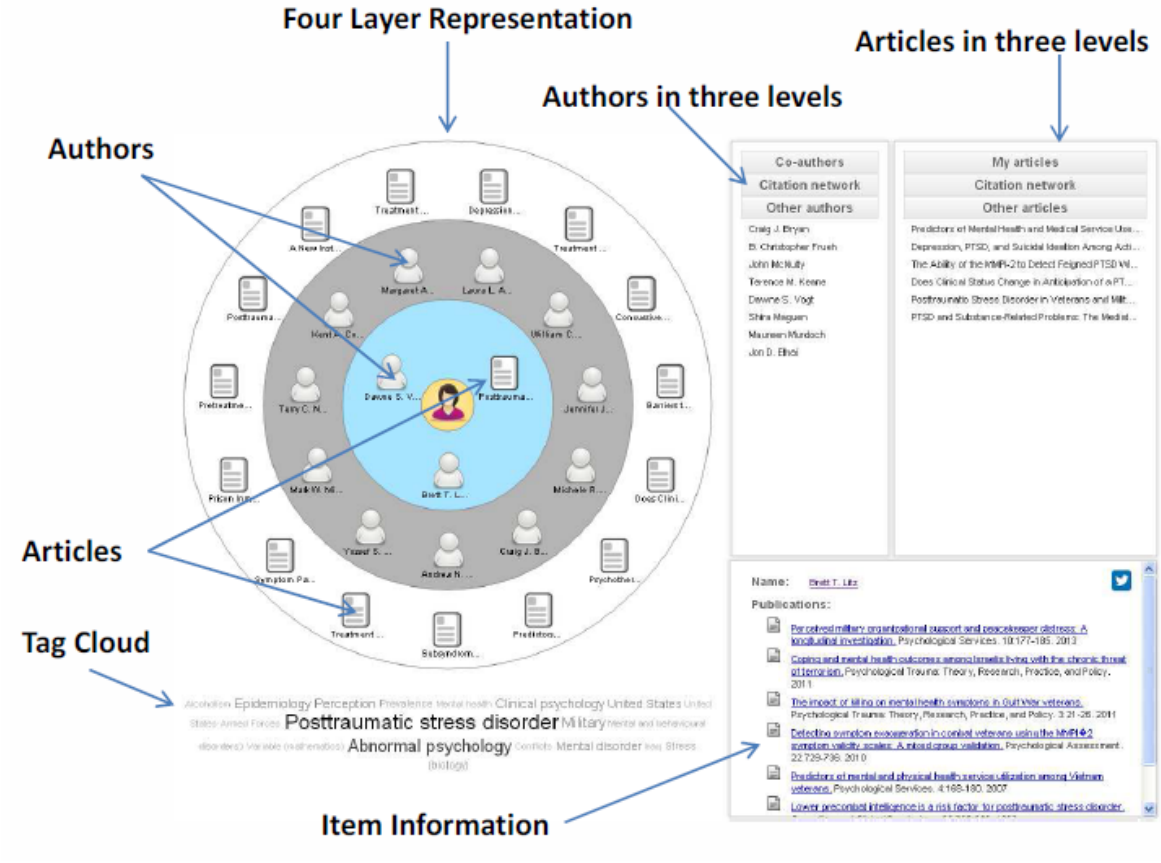

Figure 10: Entity Vagueness Annotation Process

user-based evaluation of VAO that will assess the level of usability and learnability of the tool,the perceived accuracy of the system's capability in automatically detecting vague entities as well as the overall importance, in terms of added value and feasibility, of the whole approach of annotating ontologies with vagueness metadata, were rated quite positively by the evaluators.

With respect to section 3 we have presented (1) initial guidelines that describe how to reuse available vocabularies at fine-grained level, i.e., by reusing specific classes and properties, and (2) a tool that provides technological support by means of a plugin for Protégé that access LOV API. As future work we plan to evaluate the guidelines and tool with a large set of users. Also, we plan to include access to other repositories, e.g., Bioportal and Bartoc. Moreover, we plan to develop a plugin with the same functionalities for other tools, such as Web-Protégé[25] and TopBraid composer[26].

Finally, with respect to section 4 we presented a user interface for facilitating the personalized recommendation of dataset summarization tasks, based on the Collaboration Spheres interaction paradigm. The implementation and evaluation of the interface are key parts of of our future work.

[25] http://webprotege.stanford.edu/
[26] http://www.topquadrant.com/tools/ide-topbraid-composer-maestro-edition/

Acknowledgement

This research has been funded by the European Commission within the 7th Framework Programme/-Marie Curie Industry-Academia Partnerships and Pathways schema/PEOPLE Work Programme 2011 project K-Drive number 286348 (cf. `http://www.kdrive-project.eu`).

References

Alexopoulos, P. and Pavlopoulos, J. (2014). A Vague Sense Classifier for Detecting Vague Definitions in Ontologies. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 33–37, Gothenburg, Sweden. Association for Computational Linguistics.

Alexopoulos, P., Peroni, S., Villazon-Terrazas, B., and Pan, J. Z. (2014). Annotating Ontologies with Descriptions of Vagueness. In *ESWC 2014 Satellite Events - Revised Selected Papers*, Lecture Notes in Computer Science, Berlin, Germany. Springer.

Alexopoulos, P., Villazon-Terrazas, B., and Pan, J. (2013). Towards Vagueness-Aware Semantic Data. In Bobillo, F., Carvalho, R. N., da Costa, P. C. G., d'Amato, C., Fanizzi, N., Laskey, K. B., Laskey, K. J., Lukasiewicz, T., Martin, T., Nickles, M., and Pool, M., editors, *URSW*, volume 1073 of *CEUR Workshop Proceedings*, pages 40–45. CEUR-WS.org.

Bao, J., Tao, J., McGuinness, D. L., and Smart, P. (2010). Context representation for the semantic web. In *Web Science Conference*. Event Dates: 26th April - 27th April 2010.

Benerecetti, M., Bouquet, P., and Ghidini, C. (2000). Contextual Reasoning Distilled. *Journal of Experimental & Theoretical Artificial Intelligence*, 12(3):279 – 305.

Bobillo, F. and Straccia, U. (2011). Fuzzy Ontology Representation using OWL 2. *International Journal of Approximate Reasoning*, 52(7):1073–1094.

Francescomarino, C. D., Ghidini, C., and Rospocher, M. (2012). Evaluating Wiki-Enhanced Ontology Authoring. In ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Aquin, M., Nikolov, A., Aussenac-Gilles, N., and Hernandez, N., editors, *EKAW*, volume 7603 of *Lecture Notes in Computer Science*, pages 292–301. Springer.

Guarino, N. and Welty, C. (2009). An Overview of OntoClean. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 201–220. Springer Berlin Heidelberg.

Heath, T. and Bizer, C. (2011). *Linked Data: Evolving the Web into a Global Data Space: Theory and Technology*, volume 1. Morgan & Claypool Publishers.

Horridge, M. and Patel-Schneider, P. F. (2012). OWL 2 Web Ontology Language: Manchester Syntax (Second Edition). W3C working group note, World Wide Web Consortium.

Hyde, D. (2008). *Vagueness, Logic and Ontology*. Ashgate New Critical Thinking in Philosophy.

Hyland, B., Atemezing, G., and Villazon-Terrazas, B. (2014). Best Practices for Publishing Linked Data. W3C Working Group Note. `http://www.w3.org/TR/ld-bp/`.

Keet, C. M., Fernández-Reyes, F. C., and Morales-González, A. (2012). Representing mereotopological relations in OWL ontologies with ONTOPARTS. In Simperl, E. et al., editors, *Proceedings of the 9th Extended Semantic Web Conference (ESWC'12)*, volume in print of *LNCS*. Springer.

Keet, C. M., Khan, M. T., and Ghidini, C. (2013). Ontology authoring with forza. In He, Q., Iyengar, A., Nejdl, W., Pei, J., and Rastogi, R., editors, *CIKM*, pages 569–578. ACM.

Lebo, T., Sahoo, S., and McGuinness, D. (2013). PROV-O: The PROV Ontology. W3C recommendation, World Wide Web Consortium.

Lonsdale, D., Embley, D. W., Ding, Y., Xu, L., and Hepp, M. (2010). Reusing ontologies and language components for ontology generation . *Data & Knowledge Engineering*, 69(4):318 – 330. Including Special Section: 12th International Conference on Applications of Natural Language to Information Systems (NLDB07) - Three selected and extended papers.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). WonderWeb Deliverable D18 Ontology Library (final). Technical report, IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web.

Prud'hommeaux, E. and Carothers, G. (2013). Turtle - Terse RDF Triple Language. W3C candidate recommendation, World Wide Web Consortium.

Ren, Y., Parvizi, A., Mellish, C., Pan, J., van Deemter, K., and Stevens, R. (2014). Towards Competency Question-Driven Ontology Authoring. In Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., and Tordai, A., editors, *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 752–767. Springer International Publishing.

Sanderson, R., Ciccarese, P., and Van de Sompel, H. (2013). Designing the W3C Open Annotation Data Model. In *Proceedings of the 5th Annual ACM Web Science Conference (WebSci13)*, pages 366–375. ACM Press.

Shapiro, S. (2006). *Vagueness in Context*. Oxford University Press.

Simperl, E., Mochol, M., Burger, T., and Popov, I. (2009). *Achieving maturity: The state of practice in ontology engineering in 2009*, pages 983–991. LNCS 5871. Springer Berlin Heidelberg.

Suarez-Figueroa, M.-C., Gomez-Perez, A., Motta, E., and Gangemi, A. (2012). *Ontology Engineering in a Networked World*. Springer, Berlin.

Vandenbussche, P.-Y., Atemezing, G. A., Poveda-Villalónc, M., and Vatant, B. (2015). LOV: a gateway to reusable semantic vocabularies on the Web. *Semantic Web Journal (under review)*. `http://www.semantic-web-journal.net/system/files/swj974.pdf`.

Welty, C. A. (2006). OntOWLClean: Cleaning OWL ontologies with OWL. In Bennett, B. and Fellbaum, C., editors, *FOIS*, volume 150 of *Frontiers in Artificial Intelligence and Applications*, pages 347–359. IOS Press.