Crowdsourcing Enabled Ontology Engineering

Abstract. Recent years have seen an increase in the use of crowdsourcing based methods at various stages of the ontology engineering lifecycle (e.g., verification of subsumption, assigning equivalences between concepts etc) thus laying the foundations of a novel approach to ontology engineering. Take up of this early research by the community at large, especially by practitioners, is however currently hampered by 1) a lack of understanding of which stages of the ontology engineering process can be crowdsourced and 2) tool support in ontology engineering platforms that would allow easy insertion of crowdsourcing into ontology engineering workflows. In this paper we perform an overview of recent works in the area and take a scenario-based approach to identifying those stages of the OE process where crowdsourcing makes sense. Then, we present the uComp Protege plugin, a plugin for the popular Protege ontology engineering platform that facilitates the integration of crowdsourcing stages into the OE process. TBD: clarify novelty (e.g., which new tasks we introduce?), sum up some important evaluation results.

Keywords: human computation, crowdsourcing, ontology engineering, ontology learning, Protege plugin

1 Introduction

[TBD - what is ontology engineering - and how it is always more distributed - WebProtege, GATE TEamWare - to create semantic annotations on text]

Crowdsourcing techniques allow outsourcing a task to "an undefined, generally large group of people in the form of an open call" [4] and are usually classified in three major genres depending on the motivation of the human contributors (e.g., payment vs. fun vs. altruism). Mechanised labour (MLab) is a type of paid-for crowdsourcing, where contributors choose to carry out small tasks (or micro-tasks) and are paid a small amount of money in return (often referred to as micro-payments). The most popular platform for mechanised labour is Amazons Mechanical Turk (MTurk) which allows requesters to post their micro-tasks in the form of Human Intelligence Tasks (or HITs) to a large population of micro-workers (often referred to as turkers). Most projects use crowdsourcing marketplaces such as MTurk and CrowdFlower (CF), where contributors are extrinsically motivated through economic incentives. Games with a purpose (GWAPs) enable human contributors to carry out computation tasks as a side effect of playing online games [12]. Finally, altruistic crowdsourcing refers to cases where a task is carried out by a large number of volunteer contributors, such as in the case of the Galaxy Zoo (www.galaxyzoo.org) project where over 250K volunteers willing to help with scientific research classified Hubble Space Telescope galaxy images (150M galaxy classifications).

RQ1: Which tasks can be crowdsourced? How do they fit in the OE workflow?
RQ2: How to provide tool support for crowdsourcing in OE?
[Current similar tools]

The GATE Crowdsourcing Plugin is a first example of a plugin in a popular toolkit that allows crowdsourcing from within the tool. [1]

Noy and colleagues [7] focus on the task of verifying subclass-superclass relations that make up the ontology hierarchy. Their main interest is in understanding whether crowdsourcing via MTurk is a viable alternative for this task and therefore they perform a series of experiments that compare microworkers against students, evaluate performance differences over different types of ontologies (e.g., upper ontologies vs. generic ontologies) and finally assess crowd-performance in specialized domains. The paper concludes that micro-workers are a viable alternative for verifying subclass-superclass relations and also introduces a vision for a tool support that would facilitate the integration of crowdsourcing into OE workflows. We present such a tool in this paper, that not only allows the ontology engineer to crowdsource subsumption verification, but also a set of other tasks that often appear in ontology engineering scenarios.

2 Use of Crowdsourcing for Knowledge Acquisition

Crowdsourcing methods have been used to support several knowledge acquisition and, more specifically, ontology engineering tasks. To provide an overview of these methods we will group them along the three major stages of the Semantic Life-cycle as identified by Siorpaes in [10], where Stage 1 and 2 cover our notion of ontology engineering [TBD - clarify].

Stage 1: Build and maintain Semantic Web vocabularies . Already in 2010, Eckert and colleagues [3] relied on micro-workers from Amazon Mechanical Turk (MTurk) in the process of building a concept hierarchy in the philosophy domain. Judgements collected from micro-workers complemented the output of an automatic hierarchy learning method and focused on two main tasks: judging the relatedness of concept pairs (5-points scale between unrelated and related) and specifying the level of generality between two terms (more/less specific than). Noy and colleagues [7] focus on the task of verifying subclass-superclass relations that make up the ontology hierarchy as a critical task while building ontologies.

Games with a purpose, have also been used to support the process on ontology creation. OntoPronto game [10] aims to support the creation and extension of Semantic Web vocabularies. Players are presented with a Wikipedia page of an entity and they have to (1) judge whether this entity denotes a concept or an instance; and then (2) relate it to the most specific concept of the PROTON ontology, therefore extending PROTON with new classes and instances. Climate Quiz [9] is a Facebook game where players evaluate whether two concepts presented by the system are related (e.g. environmental activism, activism), and which label is the most appropriate to describe this relation (e.g. is a sub-category of). The possible relations set contains both generic (is a sub-category

of, is identical to, is the opposite of) and domain-specific (opposes, supports, threatens, influences, works on/with) relations. Finally, Guess What?! [6] goes beyond eliciting or verifying relations between concepts and aims to create complex axioms to describe concepts in an ontology. The game explores instance data available as linked open data. Given a seed concept (e.g., banana), the game engine collects relevant instances from DBpedia, Freebase and OpenCyc and extracts the main features of the concept (e.g., fruit, yellowish) which are then verified through the collective process of game playing.The tasks performed by players are: (1) assigning a class name to a complex class description and (2) verifying previously generated class definitions.

Stage 2: Align Semantic Web vocabularies The CrowdMap system enlists micro-workers to solve the overall ontology alignment task [8]. It relies on two types of atomic HITS: the first one, asks crowdworkers to verify whether a given relation is correct ("Is conceptA the same As conceptB? yes/no "); the second task, requests micro-workers to specify how two given terms are related, in particular by choosing between sameAs, isAKindOf and notRelated. CrowdMap is designed to allow sameAs, subsumption or generic mappings between classes, properties and axioms, but currently it only support equivalence and subsumption mappings between classes. SpotTheLink is a game with a purpose that focuses on aligning Semantic Web vocabularies and has been instantiated to align the eCl@ss and UNSWPC [10] as well as the DBPedia and PROTON ontologies [11]. The final version of the game, solves ontology alignment through two atomic tasks (1) choosing a related concept: given a DBPedia concept they need to choose and agree upon a related PROTON concept; (2) specifying the type of relation between two concepts in terms of equivalence or subsumption.

[TBD if time and place] Community focused efforts in Noy2013

Stage 3: Annotate content and maintain annotations ZenCrowd [2] focuses on the entity linking problem, where crowd-workers are used to verify the output of automatic entity linking algorithms. Concretely, given a named entity, e.g., "Berlin", and a set of dbpedia URLs generated automatically, crowd-workers have to choose all the URLs that represent that entity or "None of the above" if no URL is suitable. In essence this is an annotation task. WhoKnows? [13] and RISQ! [16] are two games with a purpose which rely on similar mechanisms: they use LOD facts to generate questions and use the answers to (1) evaluate property ranking (which property of an instance is the most important/relevant); (2) detect inconsistencies; (3) find doubtful facts. The obtained property rankings reflect the wisdom of the crowd and are an alternative to semantic rankings generated algorithmically based on statistical and linguistic techniques. The games differ in the gaming paradigm they adopt. While WhoKnows?! uses a classroom paradigm and aims towards being an educational game, RISQ! is a Jeopardy-style quiz game.

[TBD: More: Celino; Sioarpaes ebay and video annotations]

SW Life-cycle Stage	Approach	Genre	Solved Task
Stage 1: Build and maintain Semantic Web vocabularies	InPho [3]	MLab	specification of subsumption relations
			term relatedness judgemendts
	Noy [7]	MLab	Verification of subsumption relations
	OntoPronto [10]	GWAP	Class vs. instance decisions
			Creation of SubClassOf/InstanceOf relations
	Climate Quiz [9]	GWAP	specify relations between terms
	Guess What?! [6]	GWAP	verify complex class definitions
			Generate class names for complex defs
Satge 2: Align Semantic Web Vocabularies	CrowdMap [8]	MLab	Verification of subsumption/eqv relations
			Specification of subsumption/eqv relations
	SpotTheLink [11]	GWAP	Choosing a related concept
			Creation of eqv/subsupmtion relation
Stage 3: Annotate content and maintain Annotations	ZenCrowd [2]	MLab	text to URL mapping (annotation)
	WhoKnows? [13]	GWAP	
	RISQ! [16]	GWAP	

Table 1. Overview of ontology engineering tasks typically solved with crowdsourcing.

2.1 Ontology Engineering Tasks for Crowdsourcing - MS

Based on the analysis of the crowdsourcing methods used to support ontology engineering tasks, it emerges that they often converge towards using a range of typical crowdsourcing tasks, which are likely to be of interest in a wide range of ontology engineering scenarios, as follows.

T1. Verification of Term Relatedness. Given two terms (ontology concepts), crowd-workers need to judge whether they are related or not.

T2. Verification of Relation Correctness. Does a certain relation between two ontology entities hold? These could be a set of generic relations (sameAs, subClassOf, instanceOf), but also arbitrary named relations to be specified by the ontology engineer. The crowd here would have to vote (yes/no) for a given triple (Subject - Relation - Object).

T3. Specification of Relation Type. This is also a very difficult task in OL in general - the workers are presented with two terms and can choose between a set of given relations what applies. These relations can be a set of OWL relations that all ontologies have as well as a (restricted) number of domain specific relations specified by the expert (e.g., influences). [BTW, since this is a complex task it could be split up into a sequence of 3 simpler tasks: 1) given two terms workers agree whether these are related or not; 2) those pairs that were judged related are then passed to another task where a correct relation is selected; 3) in the third task, the quality of the relations is checked - practically T2] Available relation labels are some predefined (like subClassOf) and those from the target ontology (all ObjectProperties) – if more than 20 .. spread over multiple windows (depending on window size) Have a limit of 5 relation labels in the Protege interface All pairs with a certain relation (defined by the user in a text field, eg relation) are sent to

the uComp API (or directly to CF). If you want to use just a subset of relations: have a window where you select (checkboxes) the actual pair to be sent. In CrowdFlower – also have the choice to add a free text label output: be able to sort by certainty from CF ..

T4. Verification of Domain Relevance. Is a concept/instance relevant for a domain?

3 Ontology Engineering Scenarios - Ontology Learning - MS

(1 or both of the following) OL - Gerhards work on automatic OL from text & other sources

Eckert has a similar scenario.

OR - using the Watson plugin

TBD: Relate to the "Embedded HC paradigm" =¿ other approaches that enhance algorithms are CrowdMap and ZenCrowd

Ontology construction from scratch is cumbersome and expensive (add citation?). Ontology learning supports the ontology building process by providing an automatically generated starting point. As automatically generated ontologies typically contain questionable or wrong ontological elements, a phase of redesign (especially pruning) is necessary.

In the ontology engineering scenario in this paper we aim to simplify the phase of pruning an ontology of concepts (classes) not relevant to the domain, as well as of *isA* relations that are not valid. The uComp Protege plugin also supports other tasks, such as validating *instanceOf* relations or checking *domain/range* restrictions (see below), but these are not part of the evaluation in this publication.

The remainder of this section gives a brief introduction to the ontology learning system used. The very first version of the system was published in 2005 ([5]), with improvements presented for example in Weichselbraun et al. [14] or Wohlgenannt et al. [15]. The system is geared towards learning lightweight domain ontologies from heterogeneous sources (text, social media, structured data). As we are learning domain ontologies in periodic intervals (monthly) from scratch, there is a focus on ontology evolution experiments, too. In a nutshell, the process is as follows: The ontology learning framework starts from a small seed ontology (typically just a few concepts and relations), and extends this seed ontology with additional concepts and relations. Evidence for new concepts and relations is collected from heterogeneous evidence sources with methods such as co-occurrence analysis or Hearst patterns. The neural network technique of spreading activation is the main algorithm used to determine the most important new concepts from the plethora of evidence. After positioning the new concepts in the ontology, the extended ontology serves as new seed ontology, and another round of extension is initiated. The system currently stops after three extension iterations.

4 The uComp Protege Plugin - GW

This section describes the actual plugin, ie. which tasks have been implemented, and the features and usage of the plugin. Protege was developed in Java, it can easily be extended in the form of *plugins* which are typically Java Archive (.jar) files stored in the Protege `plugin` directory. The most common form of a Protege plugin is a *view plugin*, which implements a single view for a specific area of an ontology (e.g. classes, individuals, object properties, ...).

Installation and setup. The plugin is available at `TODO--addlinkonceuploadedtoProtege`, includes detailed documentation about the tasks and the usage of the plugin. TODO (if necessary) documentation is accessible at .. To use the plugin you need to your uComp-API key[1] in a file named `ucomp_api_key.txt` in folder `.Protege`.

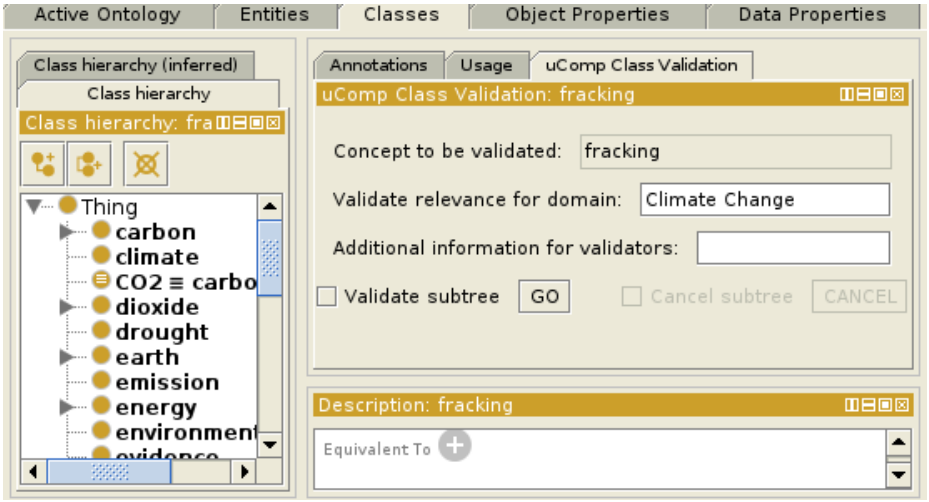

Fig. 1. Screenshot showing the interface for validating a concept.

Figure 1 shows the basic interface for the domain relevance task, the other tasks have similar interface. Initially, the user adds the new view in the Protege window. The plugin interface contains task specific information (eg. the concept selected by the user for validation), generic information such as the *domain* of the ontology and *additional information* (see below), and a `GO` button start the validation process.

Usage. The process itself is similar for all types of evaluations:

1. Add the corresponding view to the user interface via the *Window* → *Views* menu. The ontology engineer selects the part of the ontology to verify (eg.

[1] Request a key from the uComp team, see `http://soc.ecoresearch.net/facebook/election2008/ucomp-quiz-beta/api/v1/documentation/`

the *classes*), and adds the corresponding uComp HC validation frame to the user interface.
2. The user selects which part of the ontology should be verified (eg. a specific class or all classes in the ontology).
3. The user can provide additional information and options in the plugin view and then starts the evaluation.
4. A requested is sent to the uComp API.
5. Depending on user settings, the uComp API delegates the job to a GWAP or to CrowdFlower.
6. As soon as available, the plugin presents the results to the user and saves them in the ontology.
7. The user can always cancel (pause) the validation process.
8. Depending on the result, the user will perform further actions such as deleting parts of the ontology which have been validated as non-relevant.

All data collected by the plugin which should be persistent is stored in the ontology in `rdfs:comment` fields, for example information about the domain, the job ID, and results from the game.

The User Interface What is common among all tasks handled by the plugin, is the selection of a *domain* and that the user can provide additional information about the task. The domain is simply the field of knowledge which the ontology covers. If entered once, the domain will be stored in the ontology (as `rdfs:comment`) and be pre-filled subsequently, but it can also be changed at any time. For every task, the plugin contains a predefined task description (typically including examples) which is presented to the HC user. If the ontology engineer wants to extend this task description, he or she can provide more guidelines in the field *additional information*. In many cases the user of the plugin wants to verify not only a single class, but a part of the ontology, or even the whole ontology. This can be achieved with the *Validate subtree* option. If the option is selected, the current concept and all its subconcepts are verified (recursively). The verify the entire ontology, the user just starts from the uppermost class (*Thing*). In the remainder of the section we give some details about the parts of the plugin used in the evaluation (see Section 5), ie. verification of domain relevance and of relation correctness.

Task 1 - Verification of Domain Relevance. Verification of domain relevance of a class (called "uComp Class Validation" in the plugin) helps to decide if a concept (class) is relevant for a domain. First, the ontology engineer adds the corresponding view (*Window* → *Views* → *Class Views* → *uComp Class Validation*) to the UI. This is the simplest type of validation task, so the process is very much in line with the *usage* described above. Figure 1 shows the UI for the class "fracking" before initiating the verification.

Task 2 - Verification of Relation Correctness. The second task is the verification of relation correctness, more precisely the verfication of IS-A (subClassOf) relations between classes. The corresponding view is named *Class Views* → *uComp SubClass Validation*. When selecting a class in Protege, the plugin automatically detects its superclasses (if any) and fills the boxes in the

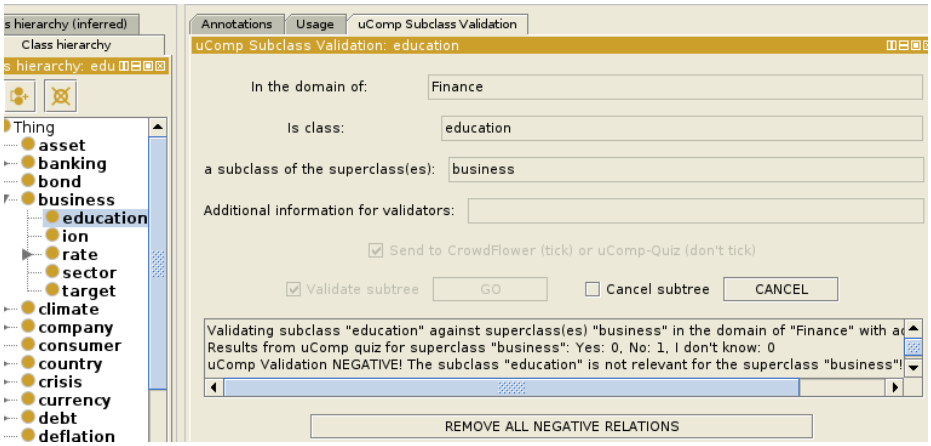

Fig. 2. Screenshot showing the interface for subClassOf relation validation, including the display of results

.

plugin UI. As soon as results are available these are presented in the UI, as shown in Figure 2. The screenshot gives an example wit one evaluator, who rated the *IS-A* relation between "education" and "business" as invalid. As the majority of ratings is negative, a button to remove the relation is displayed.

Task 3 - 5. The uComp Protege plugin also supports 3 other tasks, which we will cover only briefly, as they are not part of the user evaluation in Section 5. Task 3 is the verfication of *instanceOf* relations between an individual and a class, i.e. the crowd helps to verify if a given *instanceOf* is valid. Task 4 is concerned with validating *domain* and *range* of properties, which results in two separate sub-tasks (domain, range). And finally, we implemented a Protege view component that collects suggestions for labelling unlabeled relations from a set of given properties (relations).

5 Evaluation

5.1 Evaluation Goal

The goal of the evaluation is to assess the improvements that the uComp Plugin could enable in a typical ontology engineering scenario in terms of typical project completion aspects such as time, cost and quality of output. The usability of the plugin is an additional criteria that should be evaluated. Concretely, our evaluation goals can be summarised into the following questions:

Time *How does the use of the plugin affect the time needed to perform ontology engineering tasks?* - We distinguish here the total task time (Ttt) as the time taken from the start of the ontology engineering task until its finalisation; and the time of the ontology engineer spent actively in the task (Toe). In

a crowdsourced scenario, Toe ¡ Ttt, because the ontology engineer is only actively working during the outsourcing of the task and the review of the result. In contrast, in a traditional scenario Toe = Ttt. What is of interest to us is the time reduction ratio, that is (Ttt-Toe)/Ttt. (this will be computed as an average over multiple measurements, and over various ontology engineering tasks).

Cost *Are there cost benefits associated with the use of the plugin?* We compute costs related to payments for the involved work-force, that is payments to ontology experts and payments to crowd-workers. Costs of ontology experts are computed by multiplying the time they spend on the task (Toe) with an average salary. In order to allow comparison to other similar cost-focused studies [1], the wage of a research scientist was assumed to be $54,000 per annum.

Quality *What are the implications on the quality of the resulting output when using the Plugin?* Several earlier studies [e.g., Thalhammer13, Sabou13, Noy?, Sabou13] have shown that the quality of the semantic tasks performed by crowd-workers is in general similar to (or even better than) the quality of tasks performed by ontology engineers. While the quality of the obtained data is not the core focus of our evaluation, we expect to obtain results similar to those already published [TBD: note however, that unlike earlier studies which focus on simple, atomic tasks, we focus on more complex engineering tasks - is this really true?]

Usability *Is the plugin usable?* As any end-user tools, the plugin should be easy to understand and use by the average ontology engineer already familiar with the Protege environment.

5.2 Evaluation Setup

The setup involves working with two groups of ontology engineers, which will perform the same OE tasks.

Evaluators Victims: Adrian, Jrgen (? knows the tool), 2 guys from TU Vienna (Olger, Fajar) Non-Protege victims: Stefan, Matyas, Michael, Philipp, Heinz, Max Gbel

Group 1 This group will make use of traditional (that is manual) methods to perform the three OE tasks. The group consists of X ontology engineers.

Group 2 This group, consisting of Y ontology engineers, will use the Plugin to perform the three OE tasks. Group 2 will be provided a short tutorial about the plugin (30 minutes) and will have to perform a usability questionnaire about the plugin.

Evaluation Data The input to all evaluation tasks are ontologies generated by an ontology learning algorithm (primarily) from textual sources.

For every ontology snapshot, ie. every result of an ontology learning stage, as well as the resulting ontology the system exports an OWL file (using the Turtle seralization format, which can be converted to RDF/XML easily). The conversion of our internal format for lightweight ontologies is straightforward for concepts (which resemble to OWL classes). WordNet hyper- and hyponym relations are mapped to the OWL subClassOf property. The only challenging aspect is the representation of unlabeled relations in OWL. Our system creates ObjectProperties named "relation_n", where "n" is an auto-incrementing number, as it is not possible to use the same ObjectProperty more than once. The label for those relations is plainly "relation". The periodically created versions of the domain ontology can be uploaded to a triple store and are thereby accessible via the SPARQL.

We evaluate the plugin over two ontologies covering two diverse domains. Some details of the input ontologies are:

	Climate Change Ontology	**Finance Ontology**
Nr. of Classes	101	77
Nr. of Relations	62	50
Nr. of IsA Relations	43	20
Nr. of Un-named Relations	18	30

Table 2. Overview of the ontologies used in the experiments.

Evaluation Tasks We perform the evaluation of the plugin over two different ontology engineering tasks in order to 1) test different functionalities of the plugin; 2) obtain evaluation result over a range of tasks.

Both user groups will perform the following tasks:

Task 1: Check concept relevance for a domain For each concept of the ontology decide whether it is relevant for the domain in question (in our case, climate change and finance). Input: concept (class) and domain name; Output: true/false ratings – we use a uComp_class_relevance annotation in Protege to save the results – domain experts use true/false

Task 2: Check the correctness of isA relations For all isA relations in the ontology verify whether they are correct. Manual experts set the value for a certain annotation uComp_subclassof_check

Evaluation Metrics For time: For each participant measure Toe, and Ttt, for each task. Compute averages per participant Compute group averages and differences

For costs: Compute avg costs per participant and per group and differences

For quality: Since we do not have a baseline, we will proceed as follows: Task 1: compute pair-wise inter-expert agreement for both groups; and between groups

this will measure how different the output is between the plugin and non-plugin group. If it is small then the results are similar; Compute also Cohens Kappa Task 2: same approach as with Task 1 Task 3: Here I think we should evaluate the quality of labels ourselves (em and Gerhard) for each produced ontology; and then compute a precision value

	Climate Change Ontology		**Finance Ontology**	
	Task 1	Task 2	Task 1	Task 2
Manual validation - AVG time	27.375 (8)	23.0 (8)	21.0 (7)	13.833 (6)
Manual validation - STDDEV	5.023 (8)	6.225 (8)	7.638 (7)	5.210 (6)
CF validation - time				
Time reduction / ratio				

Table 3. Time measures .. TODO.

	Climate Change Ontology		**Finance Ontology**	
	Task 1	Task 2	Task 1	Task 2
Manual validation	0.338 (8)	0.500 (8)	0.502 (7)	0.509 (6)
Percentage valid – Manual	0.71	0.5	0.72	0.15
TODO				

Table 4. Quality measures .. TODO.

We have measures inter-rater agreement using the statistical measure of Fleiss' Kappa. Fleiss' Kappa assesses reliability of agreement with a fixed number of raters and categorical ratings assigned to a number of items.

Table 4 presents quality measures .. bla TODO. Inter-rater agreement rates measured with Fleiss' Kappa are consistent and rather high, except for the class relevance verification task for the *Climate Change* ontology. The number of raters per task is given in parantheses. TODO refine *Percentage valid* reflects the ratio of unit where the majority of raters confirm the validity.

```
Usability -- OLGA:
Although I didn't succeed to run the plug-in, i think, i could already answer the questi

1)The documentation was easy to understand.
-> 4, th part about installing plug-in was a bit of mixture with general info about prot
2) The plugin functionality was easy to use.
-> 5 , it is especially good that one can choose to make the check for all ontology at o
3) I would prefer to use the plugin as opposed to doing these tasks manually.
-> 5
4) The use of the plugin saves a lot of time to the ontology engineer.
-> 5
```

References

1. K. Bontcheva, I. Roberts, L. Derczynski, and D. Rout. The GATE Crowdsourcing Plugin: Crowdsourcing Annotated Corpora Made Easy. In *Proc. of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. ACL, 2014.
2. G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking. In *Proceedings of the 21st International Conference on World Wide Web*, pages 469–478. ACM, 2012.
3. Kai Eckert, Mathias Niepert, Christof Niemann, Cameron Buckner, Colin Allen, and Heiner Stuckenschmidt. Crowdsourcing the Assembly of Concept Hierarchies. In *Proc. of the 10th Annual Joint Conference on Digital Libraries*, JCDL '10, pages 139–148. ACM, 2010.
4. J. Howe. Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business, 2009. http://crowdsourcing.typepad.com/.
5. Wei Liu, Albert Weichselbraun, Arno Scharl, and Elizabeth Chang. Semi-automatic ontology extension using spreading activation. *Journal of Universal Knowledge Management*, 0(1):50–58, 2005.
6. T. Markotschi and J. Völker. Guess What?! Human Intelligence for Mining Linked Data. In *Proc. of the Workshop on Knowledge Injection into and Extraction from Linked Data at the International Conference on Knowledge Engineering and Knowledge Management (EKAW-2010)*, 2010.
7. N. F. Noy, J. Mortensen, M. A. Musen, and P. R. Alexander. Mechanical Turk As an Ontology Engineer?: Using Microtasks As a Component of an Ontology-engineering Workflow. In *Proceedings of the 5th Annual ACM Web Science Conference*, WebSci '13, pages 262–271. ACM, 2013.
8. C. Sarasua, E. Simperl, and N. F. Noy. CrowdMap: Crowdsourcing Ontology Alignment with Microtasks. In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, ISWC'12, pages 525–541. Springer-Verlag, 2012.
9. Arno Scharl, Marta Sabou, and Michael Föls. Climate Quiz: a Web Application for Eliciting and Validating Knowledge from Social Networks. In *Proceedings of the 18th Brazilian symposium on Multimedia and the web*, WebMedia '12, pages 189–192. ACM, 2012.
10. K. Siorpaes and M. Hepp. Games with a Purpose for the Semantic Web. *Intelligent Systems, IEEE*, 23(3):50 –60, 2008.
11. S. Thaler, E. Simperl, and K. Siorpaes. SpotTheLink: Playful Alignment of Ontologies. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1711–1712. ACM, 2011.
12. L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, 2008.
13. J. Waitelonis, N. Ludwig, M. Knuth, and H. Sack. WhoKnows? Evaluating Linked Data Heuristics with a Quiz that Cleans Up DBpedia. *Interact. Techn. Smart Edu.*, 8(4):236–248, 2011.
14. Albert Weichselbraun, Gerhard Wohlgenannt, and Arno Scharl. Refining non-taxonomic relation labels with external structured data to support ontology learning. *Data & Knowledge Engineering*, 69(8):763–778, 2010.
15. Gerhard Wohlgenannt, Albert Weichselbraun, Arno Scharl, and Marta Sabou. Dynamic Integration of Multiple Evidence Sources for Ontology Learning. *Journal of Information and Data Management*, 3(3):243–254, 2012.

16. L. Wolf, M. Knuth, J. Osterhoff, and H. Sack. RISQ! Renowned Individuals Semantic Quiz - a Jeopardy like Quiz Game for Ranking Facts. In *Proc. of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 71–78. ACM, 2011.