The uComp Protégé Plugin: Crowdsourcing Enabled Ontology Engineering

Florian Hanika[1], Gerhard Wohlgenannt[1], and Marta Sabou[2]

[1] WU Vienna
{florian.hanika,gerhard.wohlgenannt}@wu.ac.at
[2] MODUL University Vienna
marta.sabou@modul.ac.at

Abstract. Crowdsourcing techniques have been shown to provide effective means for solving a variety of ontology engineering problems. Yet, they are mainly being used as external means to ontology engineering, without being closely integrated into the work of ontology engineers. In this paper we investigate how to closely integrate crowdsourcing into ontology engineering practices. Firstly, we show that a set of basic crowdsourcing tasks are used recurrently to solve a range of ontology engineering problems. Secondly, we present the *uComp Protégé plugin* that facilitates the integration of such typical crowdsourcing tasks into ontology engineering work from within the Protégé ontology editing environment. An evaluation of the plugin in a typical ontology engineering scenario where ontologies are built from automatically learned semantic structures, shows that its use reduces the working times for the ontology engineers 11 times, lowers the overall task costs with 40% to 83% depending on the crowdsourcing settings used and leads to data quality comparable with that of tasks performed by ontology engineers.

Keywords: crowdsourcing, ontology engineering, ontology learning, Protégé plugin

1 Introduction

Ontology engineering consists of a collection of knowledge acquisition and management techniques for creating and maintaining ontologies during their entire life-cycle. Ontology engineering tasks tend to be complex, costly and, above all, time-consuming processes.

Let's consider the task of ontology creation. To reduce its complexity, ontology construction is often bootstrapped by re-using existing or automatically derived ontologies. Ontology learning methods, for example, automatically extract ontologies from (a combination of) unstructured and structured resources. Although the extracted ontologies already provide a good basis for building the ontology, they typically contain questionable or wrong ontological elements and require a phase of verification and redesign (especially pruning) by the ontology engineer. The ontology verification phase involves, among others, checking that the ontology concepts are relevant to the domain of interest and that the extracted subsumption relations are correct.

Crowdsourcing methods provide effective means to solve such ontology verification tasks by outsourcing these to "an undefined, generally large group of people in the form of an open call" [5]. As detailed in Section 2, crowdsourcing has been used effectively to solve a range of ontology engineering tasks. However, crowdsourcing techniques require high upfront investments (understanding the techniques, creating appropriate tasks) and therefore, despite their proven usefulness, these techniques remain outside the reach of most ontology engineers.

In this paper we investigate how to more closely embed crowdsourcing into ontology engineering. In the area of Natural Language Processing (NLP), where the use of crowdsourcing is highly popular [12], there already exists an effort towards supporting easy integration of crowdsourcing methods into linguists' work: the GATE Crowdsourcing Plugin is a new component in the popular GATE NLP platform that allows inserting crowdsourcing tasks into larger NLP workflows, from within GATE's user interface [1]. Noy and colleagues [10] introduce a vision for similar tool support to facilitate the integration of crowdsourcing into ontology engineering. To achieve our goal we seek answer to two research questions:

Which tasks can be crowdsourced? We distill a set of crowdsourcing tasks that are likely to be common to solving a variety of ontology engineering problems and which should be implemented by the desired tool support (Section 2).

How to implement crowdsourcing enabled ontology engineering? We present a tool, the uComp Protégé plugin, which allows ontology engineers to crowdsource tasks directly from within the popular ontology engineering tool and as part of their ontology engineering work (Section 3).

We evaluate some of the functionality of the plugin to estimate the improvements made possible over manually solving a set of tasks in terms of time and cost reductions, while maintaining good data quality (Section 4). Our findings show that, in a scenario where automatically extracted ontologies are verified and pruned, the use of the plugin significantly reduces the time spent by the ontology engineer (11 times) and leads to important cost reductions (40% to 83% depending on the crowdsourcing settings used) without a loss of quality with respect to a manual process.

2 Use of Crowdsourcing for Knowledge Acquisition

Crowdsourcing methods are usually classified in three major genres depending on the motivation of the human contributors (i.e., 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. Popular crowdsourcing marketplaces include Amazon's Mechanical Turk (MTurk) and CrowdFlower (CF). Games with a purpose (GWAPs) enable human contributors to carry out computation tasks as a side effect of playing online games [20]. Finally, in altruistic crowdsourcing a task is carried out by a large number of volunteer contributors. 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 [17] and sum them up in Table 1.

Stage 1: Build and maintain Semantic Web vocabularies Eckert and colleagues [4] relied on MTurk micro-workers to build a concept hierarchy in the philosophy domain. Crowdsourcing complemented the output of an automatic hierarchy learning method in: a) judging the relatedness of concept pairs and b) specifying the level of generality between two terms (more/less specific than). Noy and colleagues [10] focused on verifying the correctness of taxonomic relations. As for GWAPs, the OntoPronto game [17] 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 [16] is a Facebook game where players evaluate whether two concepts are related (e.g. environmental activism, activism), and which label is the most appropriate to describe their relation. The possible relation 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. Guess What?! [9] goes beyond eliciting or verifying relations between concepts to creating complex concept definitions. Players (1) assign a class name to a complex class description (e.g., assign *Banana* to *fruit&yellow&grows on trees*) and (2) verify such class definitions.

Stage 2: Align Semantic Web vocabularies The CrowdMap system enlists micro-workers to solve the ontology alignment task [15] by asking them to 1) verify whether a given relation is correct (e.g., "Is conceptA the same as conceptB? yes/no ") and 2) specify how two given terms are related, in particular by choosing between sameAs, isAKindOf and notRelated. SpotTheLink has been instantiated to align the eCl@ss and UNSWPC [17] as well as the DBpedia and PROTON ontologies [18]. The final version of the game solves ontology alignment through two atomic tasks: (1) choosing a related concept – given a DBpedia concept players choose and agree upon a related PROTON concept; (2) specifying the type of relation between two concepts.

Stage 3: Annotate content and maintain annotations In ZenCrowd [3] crowd-workers 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 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? [21] and RISQ! [23] are GWAPs which rely on similar mechanisms: they use LOD facts to generate questions and use the answers to (1) evaluate property rankings (which property of an instance is the most important/relevant); (2) detect inconsistencies; and (3) find doubtful facts. While WhoKnows?! uses a classroom paradigm and aims towards being an educational game, RISQ! is a Jeopardy-style quiz game.

SW Life-cycle Stage	**Approach**	**Genre**	**Solved Task**
Stage 1: Build and maintain Semantic Web vocabularies	InPho [4]	MLab	(T3) Specification of Relation Type (subs)
			(T1) Specification of Term Relatedness
	Noy [10]	MLab	(T2) Verification of Relation Correctness (subs)
	OntoPronto [17]	GWAP	Class vs. instance decisions
			(T3) Specification of Relation Type (subs/instOf)
	Climate Quiz [16]	GWAP	(T3) Specification of Relation Type (8 relations)
	Guess What?! [9]	GWAP	Verify complex class definitions
			Generate class names for complex defs
Stage 2: Align Semantic Web vocabularies	CrowdMap [15]	MLab	(T2) Verification of Relation Correctness (subs/eqv)
			(T3) Specification of Relation Type (subs/eqv)
	SpotTheLink [18]	GWAP	(T1) Specification of Term Relatedness
			(T3) Specification of Relation Type (subs/eqv)
Stage 3: Annotate content, maintain annotations	ZenCrowd [3]	MLab	Text to URL mapping (annotation)
	WhoKnows? [21]	GWAP	Answering quiz questions
	RISQ! [23]	GWAP	Answering quiz questions

Table 1. Overview of approaches addressing problems in various stages of the Semantic Web life-cycle [17], their genres and the type of crowdsourcing tasks that they employ.

2.1 Typical Crowdsourcing Tasks in Ontology Engineering

Based on the analysis above, we distill a set of recurrent basic crowdsourcing task types used to solve a variety of ontology engineering problems, as follows.

T1. Specification of Term Relatedness. Crowd-workers judge whether two terms (typically representing ontology concepts) are related. In some cases they are presented with pairs of terms [4] while in others they might need to choose a most related term from a set of given terms [18]. This type of crowdsourcing task is suitable both in ontology creation [4] and in ontology alignment scenarios [18].

T2. Verification of Relation Correctness. Presented with a pair of terms (typically representing ontology concepts) and a relation between these terms, crowd-workers judge whether the suggested relation holds. Frequently verified relations include generic ontology relations such as equivalence [15] and subsumption [10, 15], which are relevant both in ontology evaluation [10] and ontology alignment scenarios [15].

T3. Specification of Relation Type. In these tasks, crowd-workers are presented with two terms (typically corresponding to ontology concepts) and choose an appropriate relation from a set of given relations. Most efforts focus on the specification of generic ontology relations such as equivalence [16, 15, 18], subsumption [16, 4, 17, 15, 18], disjointness [16] or instanceOf [17, 16]. The verification of domain-specific named relations such as performed by Climate Quiz [16] is less frequent.

T4. Verification of Domain Relevance. For this task, the crowdworkers confirm whether a given term is relevant for a domain of discourse. This task is mostly needed to support scenarios where ontologies are extracted using automatic methods, for example, through ontology learning.

The core crowdsourcing tasks above have been used by several approaches and across diverse stages of ontology engineering, thus being of interest in a wide range of ontology engineering scenarios. As such, they guided the development of our plugin, which currently supports tasks T2, T4, and partially T3.

3 The uComp Protégé Plugin

In order to support ontology engineers to easily and flexibly integrate crowdsourcing tasks within their work, we implemented a plugin in Protégé, one of the most widely used ontology editors. The typical workflow of using the plugin involves the following main stages (as also depicted in Figure 1).

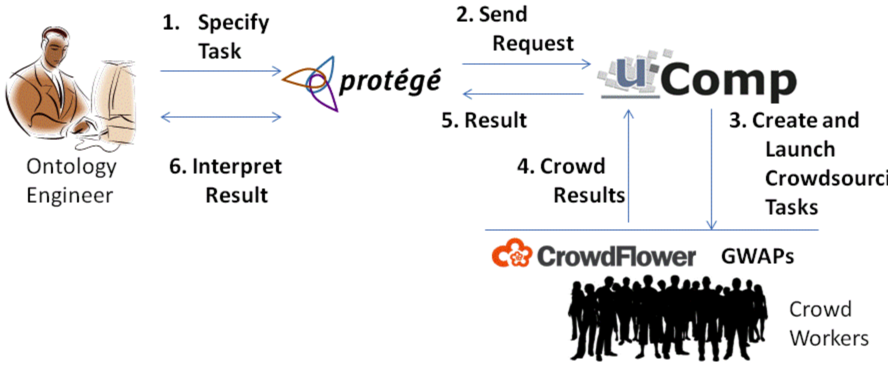


Fig. 1. Main stages when using the uComp plugin.

1. Task Specification. An ontology engineer using Protégé can invoke the functionalities of the plugin from within the ontology editor at any time within his current work. The plugin allows specifying some well defined ontology engineering tasks, such as those discussed in Section 3.2 above. The view of the plugin that is appropriate for the task at hand is added to the editor's user interface via the *Window* → *Views* menu. The ontology engineer then specifies the part of the ontology to verify (eg. a specific class or all classes in the ontology), provides additional information and options in the plugin view and then starts the evaluation. Crowdsourced tasks can be canceled (or paused) anytime during the crowdsourcing process. We further detail the plugin's functionality in Section 3.1.

2. **Task Request.** The plugin uses the uComp API[3] to request the processing of the task by the crowd.
3. **Creation of Crowdsourcing Tasks.** The crowdsourcing process happens through the uComp platform[4], a hybrid-genre crowdsourcing platform which facilitates various knowledge acquisition tasks by flexibly allocating the received tasks to GWAPs and/or mechanised labour platforms alike (in particular, CrowdFlower) [14] depending on user settings.
4. **Collection of Crowd Results.** The uComp platform collects crowd-work harvested by individual genres (GWAPs and micro-task crowdsourcing).
5. **Combination of Crowd Results.** When all crowdsourcing tasks of a job have completed, the platform combines the results and provides them to the plugin.
6. **Result Presentation and Interpretation.** As soon as available, the plugin presents the results to the ontology engineer and saves them in the ontology. All data collected by the plugin is stored in the ontology in `rdfs:comment` fields, for example information about the ontology domain, the crowdsourcing job ID, and the crowd-created results. Depending on the result, the ontology engineer will perform further actions such as deleting parts of the ontology which have been validated as non-relevant.

3.1 Plugin Functionality

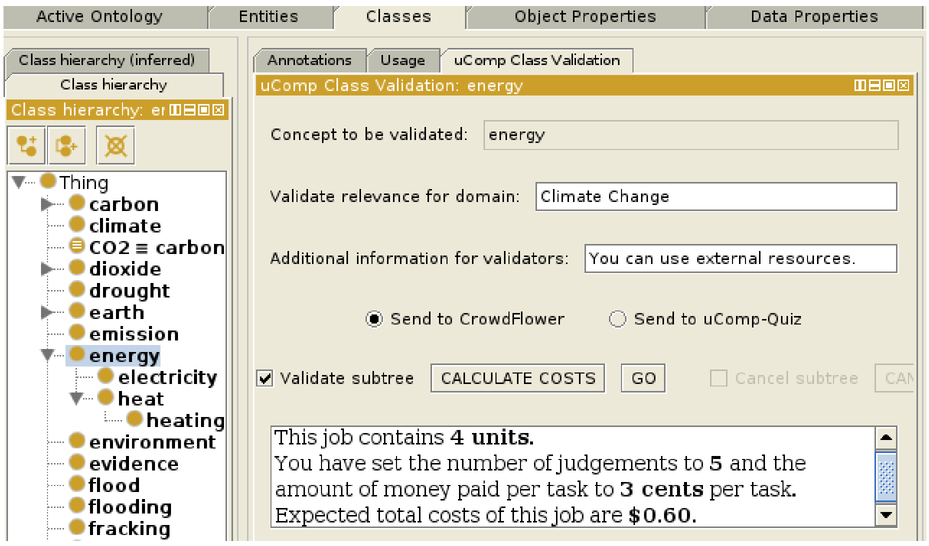
The plugin provides a set of views for crowdsourcing the following tasks:

- Verification of Domain Relevance (T4)
- Verification of Relation Correctness - Subsumption (T2)
- Verification of Relation Correctness - InstanceOf (T2) - the verification of *instanceOf* relations between an individual and a class.
- Specification of Relation Type (T3) is a Protégé view component that collects suggestions for labeling unlabeled relations by assigning to them a relation type from a set of relation types specified by the ontology engineer.
- Verification of Domain and Range where crowd-workers validate whether a property's *domain* and *range* axioms are correct.

The following subsections contain detailed descriptions of all plugin functionalities.

[3] `http://tinyurl.com/uCompAPI`

[4] The platform is being developed in the uComp project (`http://www.ucomp.eu/`)



Concept to be validated: energy

Validate relevance for domain: Climate Change

Additional information for validators: You can use external resources.

(●) Send to CrowdFlower ( ) Send to uComp-Quiz

☑ Validate subtree

This job contains **4 units.**
You have set the number of judgements to **5** and the amount of money paid per task to **3 cents** per task.
Expected total costs of this job are **$0.60.**

Fig. 2. The interface of the uComp Class Validation view used to create a Verification of Domain Relevance (T4) task.

Verification of Domain Relevance (T4) is supported by the "uComp Class Validation" view of the plugin and crowdsources the decision of whether a concept (class) is relevant for a domain. Figure 2 shows the screenshot of this view for the class "energy" before initiating the verification. The plugin view's interface contains the following information:

Task Specific Information such as the concept selected by the user for validation. This part of the view is diverse among different plugin functionalities.

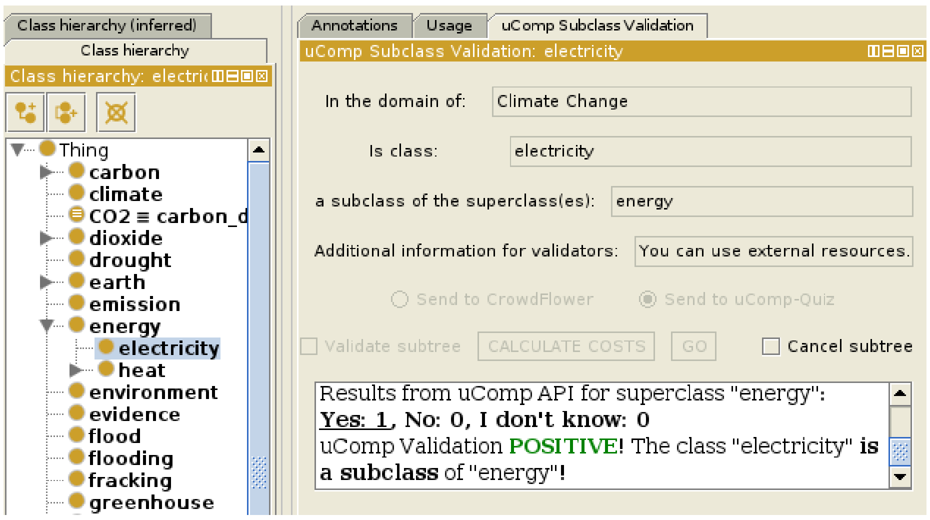
Generic information such as the *domain* of the ontology, i.e., the field of knowledge which the ontology covers, is present in all views of the plugin. If entered once, the domain will be stored in the ontology (as `rdfs:comment`) and be pre-filled subsequently, but it can also be changed at any time.

Additional information For every task, the plugin contains a predefined task description (typically including examples) which is presented to the crowd-worker. If the ontology engineer wants to extend this task description, (s)he can provide more guidelines in the *additional information* field. This functionality is present in all the views of the plugin.

Recursive control allows performing a task (e.g., domain relevance validation) not only for the current class, but for a larger part of or even the entire ontology. If the *Validate subtree* option is selected, the plugin crowdsources the specified task for the current concept and all its subconcepts recursively. To apply the functionality to the entire ontology, the plugin is invoked from the uppermost class, i.e., (*Thing*). For example, in Figure 2, the class "energy" has 3 subconcepts, which adds up to 4 units being verified if *recursive control* is activated.

Calculate costs is a button that computes and displays expected cost of HC verification before actually starting the job. The button is only available in the user interface if CrowdFlower has been selected as crowdsourcing method (and not the GWAP).

`GO` **button** to start the crowdsourcing process.



Thing
- carbon
- climate
- CO2 ≡ carbon_d
- dioxide
- drought
- earth
- emission
- energy
  - electricity
  - heat
- environment
- evidence
- flood
- flooding
- fracking
- greenhouse

In the domain of: Climate Change

Is class: electricity

a subclass of the superclass(es): energy

Additional information for validators: You can use external resources.

( ) Send to CrowdFlower [✓ Send to uComp-Quiz]

```
Results from uComp API for superclass "energy":
Yes: 1, No: 0, I don't know: 0
uComp Validation POSITIVE! The class "electricity" is
a subclass of "energy"!
```

Fig. 3. Screenshot showing the interface for subClassOf relation validation, including the display of results.

Verification of Relation Correctness - Subsumption (T2). is achieved with the *uComp SubClass Validation*. When selecting a class in Protégé, the plugin automatically detects its superclasses (if any) and fills the boxes in the plugin UI. As with any plugin functionality, the elements of the user interface are described in the plugin documentation, and additional information is also given interactively as mouse-over overlays. As soon as results are available these are presented in the UI, as shown in Figure 3. The screenshot gives an example with one evaluator, who rated the *IS-A* relation between "electricity" and "energy" as valid (positive) in the domain of "Climate Change". If the majority of judgements is negative, a button to remove the relation is displayed.

Verification of Relation Correctness - InstanceOf (T2). TBD @ Gerhard

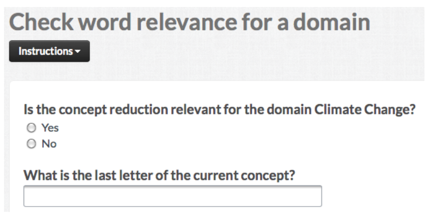
Specification of Relation Type(T3). TBD @ Gerhard

Verification of Domain and Range. TBD @ Gerhard

3.2 Crowdsourcing Task Interfaces

Upon receiving the request from the Protégé plugin, the uComp API selects the appropriate crowdsourcing genre and creates the relevant crowd-jobs. Currently the platform can crowdsource tasks either to GWAPs such as Climate Quiz [16] or to CrowdFlower, with a hybrid-genre strategy currently being developed. In this paper, we test the plugin by crowdsourcing only through CrowdFlower.

Figure 4 depicts the crowdsourcing interfaces created automatically by the uComp platform for the two tasks discussed above, namely the verification of domain relevance (part a) and the validation of subsumption relations (part b) . The uComp platform requires only the task data from the Protégé plugin and it provides relevant instructions as well as gold units to all tasks. Additionally, each crowdsourcing interface is extended with straightforward verification questions (i.e., typing some letters of the input terms). It has been shown experimentally (e.g. [6, 8]), that extending task interfaces with explicitly verifiable questions forces workers to process the content of the task and also signals that their answers are being scrutinized. This seemingly simple technique had a significant positive effect on the quality of the collected data for [6, 8].



Check word relevance for a domain

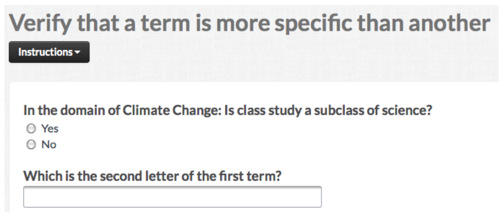
Instructions ▾

Is the concept reduction relevant for the domain Climate Change?

○ Yes
○ No

What is the last letter of the current concept?

(a)



Verify that a term is more specific than another

Instructions ▾

In the domain of Climate Change: Is class study a subclass of science?

○ Yes
○ No

Which is the second letter of the first term?

(b)

Fig. 4. Generated CrowdFlower job interface for (a) the Verification of Domain Relevance (T4) and (b) the Verification of Relation Correctness (T2) tasks.

To ensure a good quality output, by default all created jobs are assigned to Level 3 CrowdFlower contributors which are the contributors delivering, on average, the highest quality work. Also, for the moment we assume that the verified ontologies will be in English and therefore we restrict contributors to the main English speaking countries: Australia, United Kingdom and United States. In each created job we present 5 units per page and for each unit we collect 5 individual judgements. A price per task of \$0.05 was specified for all jobs. A task is complete when all requested judgments have been collected.

The plugin is available from Protégé's central registry as the *uComp Crowdsourcing Validation plugin*. The plugin has been tested with Protégé versions 4.2 and 4.3, as well as the recent version 5.0 (beta). A local configuration file contains the uComp-API key[5] and various adaptable settings (e.g., judgements per unit, price per unit).

[5] Request a key from the uComp team, see `http://tinyurl.com/uCompAPI`

4 Feasibility Evaluation

The first set of evaluations focuses on assessing the feasibility of using the uComp plugin as part of various tasks performed during ontology engineering. The primary goal is understanding the time and cost savings made possible by the use of the plugin in comparison to scenarios where an ontology engineer performs the tasks. Another set of evaluations focusing on the scalability of the approach is presented in Section 5.

We evaluate the Plugin in the context of an ontology learning scenario as described in the introduction because i) bootstrapping ontology engineering by extracting an initial ontology automatically is a feasible and frequent ontology engineering approach and ii) automatically generated ontologies present errors that are best solved through human intervention. After the verification step with the uComp plugin, the resulting ontologies are used as part of a media monitoring tool[6] with the purpose of visualising information extracted from text corpora in a way that is meaningful to the web (i.e., non-specialised) audience. Therefore, agreement on the elements of these ontologies by the general public is, in this particular use case scenario, very important.

The goal of the evaluation is to assess the improvements that the uComp Plugin could enable in an ontology engineering scenario in terms of typical project completion aspects such as time, cost and quality of output. Concretely, the 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 the total task time (T_{tt}) 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 (T_{oe}). In a crowdsourced scenario, $T_{oe} < T_{tt}$, because the ontology engineer is only actively working during the outsourcing of the task. In contrast, in a traditional scenario $T_{oe} = T_{tt}$.

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 (C_{oe}) and payments to crowd-workers (C_{cs}). Ontology engineer costs are computed by multiplying the time they spend on the task (T_{oe}) with an average monthly wage. To allow comparison to other studies [11], 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 studies have shown that the quality of various knowledge acquisition tasks performed by crowd-workers is, in general, similar to (or even better than) the quality of tasks performed by ontology engineers [19, 10, 13]. While the quality of the obtained data is not the core focus of our evaluation, we expect to obtain similar results to previous studies.

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

[6] `http://www.ecoresearch.net/climate/`

4.1 Evaluation Setup

The setup involves a group of 8 ontology engineers which perform the same tasks over the same datasets but using two different approaches. In the first setting (S_Manual), all ontology engineers used the traditional (that is manual) approach to perform the ontology engineering tasks. In the second setting (S_Crowd), four of the eight ontology engineers used the Plugin to crowdsource (that is, create and launch) the same ontology engineering tasks, after being given a brief tutorial about the plugin (30 minutes). The two settings were then compared along the time, cost and quality dimensions. Time was measured as number of minutes to complete the task. Regarding the evaluators, four were experienced Protégé users, the other four work in the Semantic Web area but have limited knowledge of Protégé and were shortly trained in Protégé. None of the ontology engineers involved had any strong expertise in a particular domain.

Nr. of	**Climate Change Ontology**	**Finance Ontology**	**Wine Ontology**	**Human Ontology**
Classes	101	77	138	3304
Relations	61	50	-	-
IsA Relations	43	20	228	3761
Unnamed Relations	18	30	-	-
Instances	0	0	206	0

Table 2. Overview of the ontologies used in the feasibility and scalability evaluations.

Evaluation Data The input to all evaluation tasks are ontologies generated by the ontology learning algorithm described in [22] (primarily) from textual sources. We evaluate the plugin over two ontologies covering two diverse domains (climate change and finance). We chose a general knowledge domain (finance) and a domain which requires domain familiarity or interest (climate change). More specialised domains will be evaluated as future research, but earlier work has already [10] investigated crowd-worker performance across ontologies of different domains/generality. The ontologies tested as part of this evaluation experiments are of small to medium size (see Table 2). The Climate Change ontology has 101 classes and 61 relations (out of which 43 are taxonomic relations) while the Finance ontology has 77 classes and 50 relations (20 of which are taxonomic relations). The ontologies were used as generated. The ontologies used in the evaluation process, the instructions given to the manual evaluators, and the results, are found online[7]. [TBD-check] Additionally we have made use of the Wine.owl ontology[8] when evaluating the instanceOf verification tasks.

[7] http://tinyurl.com/ucomp

[8] `http://www.w3.org/TR/owl-guide/wine.rdf`

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; and 2) obtain evaluation results over a range of tasks. These tasks are:

T_DomRel:Verification of Domain Relevance (T4). For each concept of the ontology decide whether it is relevant for the domain in question (in our case, climate change and finance). In S_Manual, evaluators were asked to perform this task by assigning True/False values to a class level annotation property that we created for the purposes of our experiments (named *uComp_class_relevance*).

T_SubsCorr: Verification of Relation Correctness – Subsumption (T2). For all subsumption relations in the ontology evaluators verified whether they were correct. In S_Manual, evaluators recorded their judgements in an annotation property at the relation level created for the purpose of the experiments (*uComp_subclassof_check*).

T_InstOfCorr: Verification of Relation Correctness – InstanceOf (T2). TBD@MS

T_RelTypeSpect: Specification of Relation Type(T3). TBD@MS

T_DomRangeVerif: Verification of Domain and Range. TBD@MS - will we do this?

4.2 Evaluation Results

Task Duration. Table 3 lists the task duration for the two ontologies and the two settings, detailed in terms of the average time intervals spent by the ontology engineer (T_{oe}), by using crowdsourcing (T_{cs}) and the total time of the task ($T_{tt} = T_{oe} + T_{cs}$). In the case of S_Crowd, the time needed for the ontology engineers to create and launch the crowdsourcing task was on average between 1 and 2 minutes. To simplify calculations, we chose to take the average time as 2 minutes across all tasks. We notice that the time reduction ratio for the ontology engineer across the two settings (computed as the ratio of the ontology engineering time in Setting 1 and Setting 2) is significant and ranges from a 13.7 fold reduction to a 7.5 fold reduction, with an overall average of 11: thus ontology engineers need to spend 11 times less time on the task when using the Plugin than in the manual scenario. The duration of the overall task increases and varies between 2.4 and 4.7 hours. Note however, that the current evaluation setup maximizes quality rather than speed. Faster completion rates (possibly at the expense of data quality) could have been obtained by not restricting the geographical location and previous achievements of the crowd-workers.

Costs. For the cost analysis, we compute average costs for the total task (C_{tt}) as the sum of the average cost of the ontology engineer (C_{oe}) and the average cost of the crowd-sourced tasks (C_{cs}) as detailed in Table 4. Considering an annual salary of \$54,000 and a corresponding \$26 hourly wage[9], average ontology

[9] In practice, considering benefits, overhead and vacation, the actual costs for a productive hour are likely to be higher than \$26. Nevertheless, we decided to keep \$26 in order to be able to compare our findings to similar studies.

	Climate Change Ontology						Finance Ontology					
	T_DomRel			**T_SubsCorr**			**T_DomRel**			**T_SubsCorr**		
	T_{oe}	T_{cs}	T_{tt}	T_{oe}	T_{cs}	T_{tt}	T_{oe}	T_{cs}	T_{tt}	T_{oe}	T_{cs}	T_{tt}
S_Manual (Avg)	27.4	0	27.4	23.0	0	23.0	21.3	0	21.3	15.0	0	15.0
S_Manual (StdDev)	5	0	5	6.2	0	6.2	7.1	0	7.1	5.6	0	5.6
S_Crowd (Avg)	2	240	242	2	280	282	2	140	142	2	200	202
S_Manual/S_Crowd	13.7	-	0.11	12.5	-	0.08	10.65	-	0.15	7.5	-	0.07

Table 3. Task duration in minutes per ontology, evaluation task and setting.

engineering costs were computed based on the average times shown in Table 3. Cost savings were then computed for each cost category.

	Climate Change Ontology						Finance Ontology					
	T_DomRel			**T_SubsCorr**			**T_DomRel**			**T_SubsCorr**		
	C_{oe}	C_{cs}	C_{tt}	C_{oe}	C_{cs}	C_{tt}	C_{oe}	C_{cs}	C_{tt}	C_{oe}	C_{cs}	C_{tt}
S_Manual (Avg)	11.9	0	11.9	9.9	0	9.9	9.2	0	9.2	6.5	0	6.5
S_Crowd (Avg)	0.9	8.48	9.38	0.9	3.58	4.48	0.9	6.49	7.39	0.9	1.67	2.57
Cost Savings (%)	92.4	-	21.2	90.1	-	54.7	90.2	-	19.7	86.15	-	60.5
S_CrowdCheap (Avg)	0.9	1.02	2.1	0.9	0.43	1.33	0.9	0.78	1.68	0.9	0.2	1.1
Cost Savings (%)	92.4	-	82.3	90.1	-	86.5	90.2	-	81.7	86.15	-	83

Table 4. Average costs (in \$) for the ontology engineer (C_{oe}), crowd-workers (C_{cs}) and the entire task (C_{tt}) across ontologies and settings.

Ontology engineer cost savings are high and range from 92.4% to 86.15%, averaged at 89.9%. For the entire task, cost savings are moderate (19.7% - 60.5%, Avg = 39%), with Setting 2 reducing S_Manual costs with 40%. Note, however, that task level cost savings will ultimately depend on the cost that ontology engineers decide to pay to crowd-workers. For example, choosing a cheaper task setting than currently (i.e., 3 judgements, with \$0.01 per task vs. the current 5 judgements and \$0.05 per task) will lead to average cost savings of 83.3% for the total task (S_CrowdCheap in Table 4). From the plugin's perspective, the major goal is reducing ontology engineering costs, as crowdsourcing costs will depend on the constraints of the ontology engineer and are hard to generalise.

Data Quality. Lower completion times and costs should not have a negative effect on the quality of the crowdsourced data. Since we do not possess a baseline for either of the two tasks, we will perform a comparative evaluation and contrast inter-rater agreement levels between ontology engineers with those of crowdworkers. We have measured inter-rater agreement with Fleiss' Kappa which is used to assess reliability of agreement with a fixed number of raters and categorical ratings assigned to a number of items.

	Climate Change		**Finance**	
	Ontology		**Ontology**	
	T_DomRel	**T_SubsCorr**	**T_DomRel**	**T_SubsCorr**
S_Manual	0.338 (8)	0.502 (8)	0.496 (8)	0.419 (8)
S_Crowd	0.633 (4)	0.841 (4)	0.520 (4)	0.826 (4)
S_ManualCrowd	0.392 (12)	0.582 (12)	0.505 (12)	0.508 (12)

Table 5. Fleiss' Kappa values of inter-rater agreement per setting and when combining the data of the two settings.

Table 5 presents inter-rater agreement per task and per setting, with the number of raters per task given in parentheses. According to the interpretation of Landis and Koch [7] the inter-rater agreement among manual expert evaluators (S_Manual) is moderate. Agreement among the four groups of CrowdFlower workers is substantial (S_Crowd). The combined agreement (manual expert and crowdworkers) is always higher than for manual evaluators alone. A detailed inspection of results reveals that judgement is difficult on some questions, for example relevance of given concepts for the climate change domain often depends on the point of view and granularity of the domain model. But in general, crowdworkers have a higher inter-rater agreement, which often corresponds with the majority opinion of manual experts, thereby raising Fleiss' kappa (S_ManualCrowd). Also, the agreement between the crowd and experts is higher than among experts, possibly because crowdsourcing data is the majority view derived from 5 judgements as compared to a single expert judgement.

Plugin Usability was assessed by means of the System Usability Scale (SUS), the most used questionnaire for measuring perceptions of usability [2]. Based on data collected from the entire test population (i.e., all 8 ontology engineers), we obtained a SUS score of 85, which corresponds to the 90th percentile rank and positions the plugin in the class of "A" type system, that is systems with maximal usability. All evaluators agreed that (a) they would prefer using the plugin instead of performing the tasks manually and that (b) the use of the plugin saved a lot of their time. They considered the recursive task verification particularly useful when focusing on large (parts of the) ontologies. One suggested making the plugin data and results more visually appealing, and showing the anticipated cost before crowdsourcing – both of which have been implemented in the meantime. Given the small scale of the usability evaluation, we consider it only as indicative that the Plugin has a good usability.

5 Scalability Evaluations

While the experiments above confirm the cost savings made possible by the plugin as well as the plugin's usability, it is important to also investigate the scalability of the proposed approach and its applicability when working with highly domain-specific ontologies. To answer these questions we ran a second set of experiments focusing on large and domain specific ontologies.

5.1 Experimental data and tasks

As experimental data we chose the human.owl ontology which represents the human anatomy part of the NCI thesaurus and was made available as part of the Anatomy track of the Ontology Matching Initiative[10]. As shown in Table 2, this ontology is several degrees of multitude larger than the ontologies used for the feasibility evaluation. Additionally, it is a domain specific ontology and therefore allows evaluating the usefulness of the plugin for medium-sized to large domain specific ontologies, concretely for the anatomy domain.

We focus on evaluating two functionalities of the plugin, namely the domain relevance check and checking the correctness of subsumption relations. Given the large-scale of these experiments it is not feasible to performed them with domain experts as well. Therefore the focus is not on understanding the time/cost savings with respect to manual work, but rather to measure the time and cost involved when running such large-scale tasks.

Unlike in the previous experiments, the data is an approved ontology (not a learned ontology) and therefore we can assume it as correct. This allows us to measure the quality of the crowd work even without having a baseline created by domain experts. Our strategy is that of modifying the human.owl ontology by adding incorrect data to it and assess how effective the crowd is in filtering out the incorrect cases. Accordingly, we have performed the following changes to human.owl.

For experiments focusing on measuring the domain relevance of ontology concepts, human.owl already provides 3304 concepts. We have extended the ontology with 1000 additional classes with labels extracted using ontology learning techniques from corpora related to the domains of climate change and tennis. The 1000 labels have been manually verified to exclude any labels that might refer to the human body. These 1000 classes have been added as random leaf classes to the ontology resulting in a total of 4304 concept labels to be verified for domain relevance. More precisely, for the 500 terms each from the domains of *climate change* and *tennis*, the insertion algorithm randomly selects a concept from the original ontology and then inserts a new sub-concept which has the term as concept label. This strategy guarantees that non-relevant concepts are evenly distributed in the ontology.

For experiments involving the verification of the correctness of subsumption relations, human.owl provides 3761 correct subsumption relations. To introduce incorrect relations, we have identified 800 pairs of leaf concepts and swapped their places in the ontology, therefore creating 1600 incorrect subsumption relations. Again, concepts to be swapped are randomly selected, and marked with an *rdfs:comment* tag to easily find incorrect concepts in subsequent analysis.

[10] `http://oaei.ontologymatching.org/2014/anatomy/index.html`

5.2 Experimental Results

6 Summary and Future Work

In this paper we investigated the idea of closely embedding crowdsourcing techniques into ontology engineering. Through an analysis of previous works using crowdsourcing for ontology engineering, we concluded that a set of basic crowdsourcing tasks are repeatedly used to achieve a range of ontology engineering processes across various stages of the ontology life-cycle. We then presented a novel tool, a Protégé plugin, that allows ontology engineers to use these basic crowdsourcing tasks from within their ontology engineering working context in Protégé. An evaluation of the plugin in an ontology engineering scenario where automatically learned ontologies in two different domains are assessed for domain relevance and subsumption correctness, revealed that the use of the plugin reduced overall project costs, lowered the time spent by the ontology engineer (without extending the time of the overall tasks to over 4 hours) and returned good quality data that was in high agreement with ontology engineers. Finally, our evaluators have provided positive feedback about the usability of the plugin.

Our evaluation focused on assessing the concept of a crowdsourcing plugin. Although we plan to make use of the uComp platform, this particular evaluation forwarded all tasks directly to CrowdFlower and therefore is not influenced by the particularities of the uComp framework. As a first evaluation of the plugin, we focused on small-scale ontologies and therefore cannot offer insights, at this stage, about how the plugin scales to large-scale ontologies. Another important question to address is whether crowd-workers can replace domain experts for application scenarios where domain specific meaning must be conveyed by the domain models. Both of these issues are important for our future work.

We consider our work as a first step towards the wide adoption of crowdsourcing by the ontology engineering community, and therefore, we see ample opportunities for future work. Firstly, since the use of crowdsourcing has matured enough, it is a good time to move on from isolated approaches towards a methodology of where and how crowdsourcing can efficiently support ontology engineers. Such methodological guidelines should inform tools such as our own plugin while our plugin could offer a means to build and test these guidelines. Secondly, in terms of the plugin development, we plan further extending its functionality (1) to support additional HC tasks; (2) to allow greater control over job settings as well as (3) to permit monitoring of the results as they become available from within Protégé. Thirdly, the scalability of the proposed approach remains to be investigated - while the current evaluation did not focus on this aspect, we expect that the automatic distribution of tasks for parallel processing will make this approach feasible for dealing with large ontologies as well. We also plan to evaluate the plugin in other ontology engineering scenarios (e.g., ontology matching) and to conduct larger scale usability studies. Future work will also reveal best use cases of the plugin identifying those cases when it can be used to collect generic knowledge as opposed to application areas where it should be used to support the work of a distributed group of domain experts.

Acknowledgments. The work presented in this paper was developed within project uComp, which receives the funding support of EPSRC EP/K017896/1, FWF 1097-N23, and ANR-12-CHRI-0003-03, in the framework of the CHIST-ERA ERA-NET.

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. J. Brooke. *SUS: a quick and dirty usability scale.* Taylor and Francis, London, UK, 1996.
3. 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.
4. K. Eckert, M. Niepert, C. Niemann, C. Buckner, C. Allen, and H. 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.
5. J. Howe. Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business, 2009. http://crowdsourcing.typepad.com/.
6. A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing User Studies with Mechanical Turk. In *Proc. of the 26th Conference on Human Factors in Computing Systems*, pages 453–456, 2008.
7. JR. Landis and GG. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.
8. F. Laws, C. Scheible, and H. Schütze. Active Learning with Amazon Mechanical Turk. In *Proc. of the Conf. on Empirical Methods in NLP*, pages 1546–1556, 2011.
9. 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.
10. 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.
11. M. Poesio, U. Kruschwitz, J. Chamberlain, L. Robaldo, and L. Ducceschi. Phrase Detectives: Utilizing Collective Intelligence for Internet-Scale Language Resource Creation. *Transactions on Interactive Intelligent Systems*, 3(1):1–44, April 2013.
12. M. Sabou, K. Bontcheva, and A. Scharl. Crowdsourcing Research Opportunities: Lessons from Natural Language Processing. In *Proc. of the 12th International Conference on Knowledge Management and Knowledge Technologies (iKNOW), Special Track on Research 2.0*, 2012.
13. M. Sabou, K. Bontcheva, A. Scharl, and M. Föls. Games with a Purpose or Mechanised Labour?: A Comparative Study. In *Proc. of the 13th International Conference on Knowledge Management and Knowledge Technologies*, i-Know '13, pages 1–8. ACM, 2013.

14. Marta Sabou, Arno Scharl, and Michael Föls. Crowdsourced Knowledge Acquisition: Towards Hybrid-genre Workflows. *International Journal of Semantic Web and Information Systems*, 9(3):14–41, 2013.
15. 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.
16. A. Scharl, M. Sabou, and M. 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.
17. K. Siorpaes and M. Hepp. Games with a Purpose for the Semantic Web. *Intelligent Systems, IEEE*, 23(3):50 –60, 2008.
18. 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.
19. S. Thaler, E. Simperl, and S. Wölger. An Experiment in Comparing Human-Computation Techniques. *IEEE Internet Computing*, 16(5):52–58, 2012.
20. L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, 2008.
21. 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.
22. G. Wohlgenannt, A. Weichselbraun, A. Scharl, and M. Sabou. Dynamic Integration of Multiple Evidence Sources for Ontology Learning. *Journal of Information and Data Management*, 3(3):243–254, 2012.
23. 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.