The uComp Protégé Plugin: Towards Embedded Human Computation for 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. Embedded Human Computation advocates a tight integration of Human Computation (HC) methods into computational workflows. In this paper we discuss two enabling elements for EHC in the domain of ontology engineering. Firstly, we show that a set of basic HC tasks are used recurrently to solve a range of ontology engineering tasks. Secondly, we present the *uComp Protégé plugin* that facilitates the integration of such typical HC tasks into the ontology engineering process from within the 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 significantly reduces the working times for the ontology engineer, reduces the overall budget of the tasks and leads to data quality which is in line with that obtainable with ontology engineers.

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

1 Introduction

Ontology engineering, as the process involving the creation and maintenance of ontologies during their entire life-cycle, is a crucial component of any Semantic Web based system. Traditionally performed by a small group of ontology experts, ontology engineering tends to be a complex, costly but, above all, time-consuming process. In an attempt to solve this situation, the Protégé ontology editor has been therefore redesigned to open up the ontology engineering process to a larger, distributed group of contributors and enable collective knowledge creation through WebProtégé [20]. Similarly, in the area of natural language processing, GATE Teamware extends the GATE linguistic annotation toolkit with distributed knowledge creation capabilities [1]. While these extensions primarily support the collaborative and distributed work of knowledge experts (both ontology engineers and linguists), an increasing trend is allowing large populations of *non-experts* to create knowledge through the use of Human Computation platforms such as games or mechanised labour platforms.

Human Computation (HC) describes systems that combine automated methods with large numbers of human contributors who collaborate to perform a task, typically involving an automated system that serves up task instances and then aggregates the produced results [11]. HC systems 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 Amazon's 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 [21]. Finally, in altruistic crowdsourcing a task is carried out by a large number of volunteer contributors.

Human computation methods have been used to solve a range of knowledge acquisition tasks, in general [14] and ontology engineering tasks in particular as discussed in Section 2. Some of the newer approaches, such as ZenCrowd [3] and CrowdMap [15], are making a move towards a tighter integration of HC methods into computational workflows to solve Semantic Web specific tasks such as entity linking or ontology matching. This is an emerging computational paradigm which goes beyond mere data collection towards embedding the HC paradigm into larger knowledge extraction workflows. We refer to it as *Embedded Human Computation (EHC)*. In the area of Natural Language Processing (NLP), where the use of HC methods is highly popular [12], there already exists an effort towards supporting easy integration of HC methods into linguistic computation workflows: the GATE Crowdsourcing Plugin is a new component in the popular GATE NLP platform that allows inserting HC tasks into larger NLP workflows, from within GATE's user interface [2]. Noy and colleagues [9] introduce a vision for similar tool support that would facilitate the integration of HC into ontology engineering workflows. In this paper we investigate the possibility of such a tool, and make the following contributions:

1. We distill a set of common HC tasks that are likely to be common to solving a variety of ontology engineering tasks and which should be implemented by the desired tool support (Section 2).
2. 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 workflows, thus enabling novel computational workflows in the spirit of EHC (Section 3).
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). The results show that, in a scenario where automatically extracted ontologies are

verified and pruned, the use of HC methods from the ontology editor, significantly reduces the time spent by the ontology engineer, leads to important cost reductions without a loss of quality with respect to a manual process.

2 Use of Human Computation for Knowledge Acquisition

HC 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 Already in 2010, Eckert and colleagues [4] relied on MTurk micro-workers 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 (on a 5-points scale between unrelated and related) and specifying the level of generality between two terms (more/less specific than). Noy and colleagues [9] 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. 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 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 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. Finally, Guess What?! [8] 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 (e.g., assign *Banana* to $fruit\&yellow\&grows\ on\ trees$) and (2) verifying previously generated class definitions.

Stage 2: Align Semantic Web vocabularies The CrowdMap system enlists micro-workers to solve the ontology alignment task [15]. It relies on two types of atomic HITS: the first one asks crowdworkers to verify whether a given relation is correct (e.g., "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 supports equivalence and subsumption mappings between classes. SpotTheLink is a GWAP that focuses on aligning Semantic Web vocabularies and 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 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.

Stage 3: Annotate content and maintain annotations ZenCrowd [3] 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? [22] and RISQ! [24] are two games with a purpose which rely on similar mechanisms: they use linked open data (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; (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.

2.1 Typical HC Tasks in Ontology Engineering

Based on the analysis above, we observe that a set of recurrent basic HC task types are used to solve a variety of diverse ontology engineering tasks, as follows.

T1. Specification of Term Relatedness. Crowd-workers need to judge whether two terms (typically representing ontology concepts) are related or not. In some cases they are presented with pairs of terms (InPho [4]) while in others they might need to choose a most related term from a set of given terms (SpotTheLink [18]). This type of HC task is suitable in diverse ontology engineering stages, for example, both in ontology creation scenarios (InPho [4]) and in ontology alignment ones (SpotTheLink [18]).

T2. Verification of Relation Correctness. Presented with a pair of terms (typically representing ontology concepts) and a relation between these terms, crowd-workers are required to judge whether the suggested relation holds or not. The majority of work we reviewed report on verifying generic ontology

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 [9]	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?! [8]	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 and maintain Annotations	ZenCrowd [3]	MLab	Text to URL mapping (annotation)
	WhoKnows? [22]	GWAP	Answering quiz questions
	RISQ! [24]	GWAP	Answering quiz questions

Table 1. Overview of HC approaches used to address tasks in various stages of the Semantic Web life-cycle [17], their genres and the type of HC tasks that they employ.

relations such as equivalence [15] and subsumption [9, 15], which are relevant both in ontology evaluation [9] 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 can choose from a set of given relations the relation that best relates the terms. Most efforts focus on the specification of generic ontology relations such as equivalence (Climate Quiz [16], CrowdMap [15], SpotTheLink [18]), subsumption(Climate Quiz [16], InPho [4], OntoPronto [17], CrowdMap [15], SpotTheLink [18]), disjointness (Climate Quiz [16]) or instanceOf (OntoPronto [17], Climate Quiz [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 HC tasks above have been used by several approaches and across diverse stages of ontology engineering, this 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 HC tasks within their ontology engineering workflows, 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).

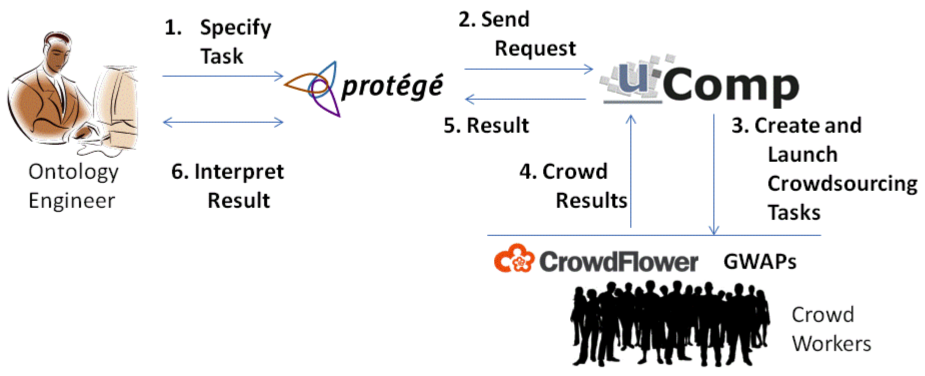


Fig. 1. Main stages when using the uComp plugin.

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 crowdsourcing the received tasks to games with a purpose and mechanised labour platforms alike (in particular, CrowdFlower) [14]. Depending on user settings, the uComp platform delegates the job to a GWAP, to CrowdFlower or to a combination of these two genres. In Section 3.2 we present the crowdsourcing tasks created by the uComp platform.

4&5 Collection of Crowd Results. The uComp platform collects and combines crowd-work harvested through various genres and provides the data to the plugin.

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

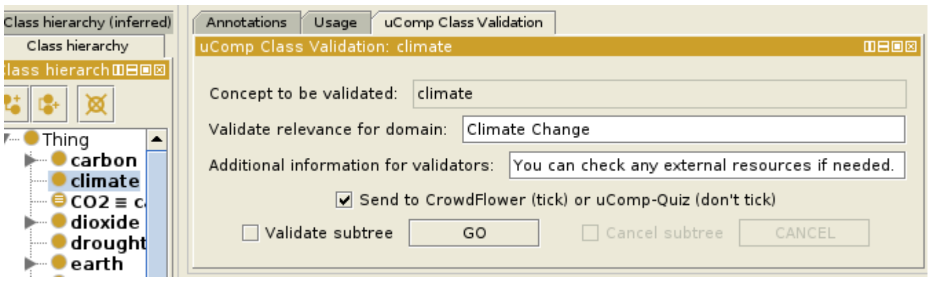
6. Result Presentation and Interpretation. As soon as available, the plugin presents the results to the ontology engineer and saves them in the ontology. All data collected by the plugin is stored in the ontology in `rdfs:comment` fields, for example information about the ontology domain, the HC job 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* restrictions are correct.

In this paper we focus on the first two functionalities, which we now describe in more detail.



Concept to be validated: climate
Validate relevance for domain: Climate Change
Additional information for validators: You can check any external resources if needed.
☑ Send to CrowdFlower (tick) or uComp-Quiz (don't tick)

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. First, the ontology engineer adds the corresponding view (*Window* → *Views* → *Class Views* → *uComp Class Validation*) to the editor's UI. Figure 2 shows the screenshot of this view for the class "climate" before initiating the verification. The plugin view's interface contains the following information:

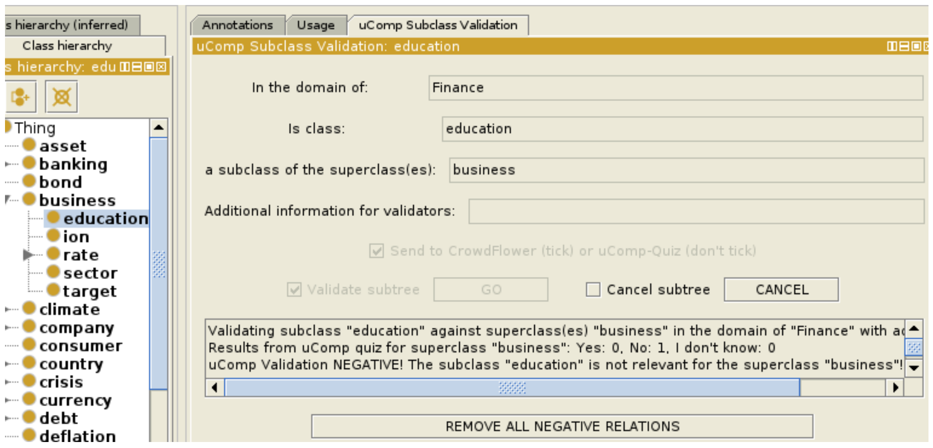
Task Specific Information such as the concept selected by the user for validation. This part of the view is diverse among different plugin functionalities.

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 crowdworker. 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 In many cases the ontology engineer wants to perform 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*).

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



In the domain of: Finance

Is class: education

a subclass of the superclass(es): business

Additional information for validators:

[✓] Send to CrowdFlower (tick) or uComp-Quiz (don't tick)

[✓] Validate subtree

[ ] Cancel subtree

```
Validating subclass "education" against superclass(es) "business" in the domain of "Finance" with a
Results from uComp quiz for superclass "business": Yes: 0, No: 1, I don't know: 0
uComp Validation NEGATIVE! The subclass "education" is not relevant for the superclass "business"!
```

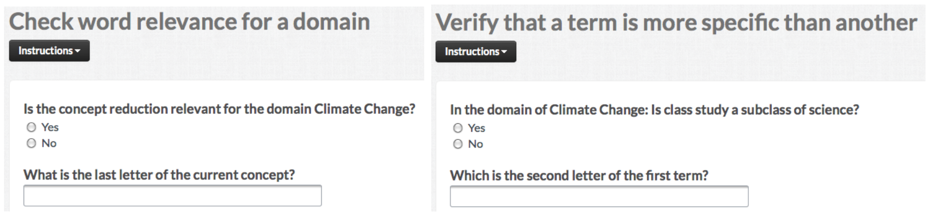
Fig. 3. Screenshot showing the interface for subClassOf relation validation, including the display of results

Verification of Relation Correctness - Subsumption (T2). The second task is the verification of relation correctness, more precisely the verification of IS-A (subClassOf) relations between classes. The corresponding view is named *Class Views* → *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 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 “education” and “business” as invalid. As the majority of ratings is negative, a button to remove the relation is displayed.

3.2 HC Task Interfaces

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

Figure 4 depicts the crowdsourcing interfaces created automatically by the uComp platform for the two tasks discussed above, namely the verification of domain relevance and the validation of subsumption relations. 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. [5, 7]), that extending task interfaces with explicitly verifiable questions forces workers to process the content of the task and also signals to them that their answers are being scrutinized. This seemingly simple technique had a significant positive effect on the quality of the collected data [5, 7].


Check word relevance for a domain

Is the concept reduction relevant for the domain Climate Change?
○ Yes
○ No

What is the last letter of the current concept?

Verify that a term is more specific than another

In the domain of Climate Change: Is class study a subclass of science?
○ Yes
○ No

Which is the second letter of the first term?

(a) (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 Sates. 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. Future versions of the plugin will provide higher control over task settings from within Protégé.

3.3 Implementation and Installation Details

Protégé can easily be extended in the form of *plugins* which are typically Java Archive (.jar) files stored in the Protégé `plugin` directory. The most common form of a Protégé 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 has been included into Protégé's central registry and can be installed from within Protégé with the *File* → *Check for Updates* menu item. A window entitled *Automatic Update* will pop up, where the *uComp Crowdsourcing Validation plugin* can be selected from the list of downloads. The download files contain the plugin itself, and a detailed documentation. To use the plugin a uComp-API key[5] must be stored in the `ucomp_api_settings.txt` file in the `.Protege` folder. More details are found in the plugin's documentation.

4 Evaluation

Ontology construction from scratch is a time-consuming and complex process and it is often bootstrapped by re-using existing ontologies or ontologies derived by automatic methods from non-ontological resources (e.g., text corpora, folksonomies, databases etc). Ontology learning methods, for example, automatically extract ontologies from a variety of unstructured and structured resources or a combination thereof. The extracted ontologies 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 typically involves, among others, checking that the ontology concepts are relevant to the domain of interest and that the extracted subsumption relations are correct. Since the uComp plugin supports these tasks, we will evaluate them in an ontology engineering scenario by reusing automatically learned ontologies.

The ontology learning system used to generate the input ontologies [23] is geared towards learning lightweight domain ontologies from heterogeneous sources (text, social media, structured data). The learning process starts from a small seed ontology (typically just a few concepts and relations), and extends it with additional concepts and relations based on evidence 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.

4.1 Evaluation Goal

The goal of the evaluation is to assess the improvements that the uComp Plugin could enable in a ontology engineering scenario using ontology learning in terms

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

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 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_{hc}). 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 [10], 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, 9, 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.

4.2 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 (Setting 1), all ontology engineers used the traditional (that is manual) approach to perform the ontology engineering tasks. In Setting 2, three ontology engineers used the Plugin to crowdsource (that is, create and launch) the same ontology engineering tasks. They were given a brief tutorial about the plugin (30 minutes) and filled in a short usability questionnaire about using the plugin. The two settings is then compared along the time, cost and quality dimensions.

Evaluation Data The input to all evaluation tasks are ontologies generated by the ontology learning algorithm described above and in [23] (primarily) from textual sources. We evaluate the plugin over two ontologies covering two diverse domains (climate change and finance). Table 2 lists some statistics about the sizes of these ontologies.

Nr.	**Climate Change Ontology**	**Finance Ontology**
Classes	101	77
Relations	61	50
IsA Relations	43	20
Unnamed Relations	18	30

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

4.3 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:

Task 1: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 Stage 1, 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 ($uComp\_class\_relevance$).

Task 2: Verification of Relation Correctness – Subsumption (T2). For all subsumption relations in the ontology evaluators were asked to verify whether they are correct. In Stage 1, evaluators recorded their judgements in an annotation property at the relation level created for the purpose of the experiments ($uComp\_subclassof\_check$).

4.4 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 human computation (T_{hc}) and the total time of the task ($T_{tt} = T_{oe} + T_{hc}$). In the case of the second setting, 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 ontology engineering time in Setting 1 divided by the same time in Setting 2) is significant and ranges from a 13.7 fold reduction to a 7.5 fold reduction, on average.

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_{hc}) as detailed in Table 4. Considering an annual salary of \$54,000 and a corresponding \$26 hourly wage, average ontology engineering costs were computed based on the average times shown in Table 3. Cost savings were then computed for each cost category.

On average, cost savings for ontology engineer costs are high and range from 92.4% to 86.15%, averaged at 89.9%. For the task as a whole, cost savings are

	Climate Change Ontology						**Finance Ontology**					
	Task 1			**Task 2**			**Task 1**			**Task 2**		
	T_{oe}	T_{hc}	T_{tt}	T_{oe}	T_{hc}	T_{tt}	T_{oe}	T_{hc}	T_{tt}	T_{oe}	T_{hc}	T_{tt}
Setting 1 (Avg)	27.4	0	27.4	23.0	0	23.0	21.3	0	21.3	15.0	0	15.0
Setting 1 (StdDev)	5	0	5	6.2	0	6.2	7.1	0	7.1	5.6	0	5.6
Setting 2 (Avg)	2	240	242	2	280	282	2	140	142	2	200	202
Setting 1/Setting2 ratio	13.7	-	0.11	12.5	-	0.08	10.65	-	0.15	7.5	-	0.07

Table 3. Task duration in minutes per ontology and setting.

.

	Climate Change Ontology						**Finance Ontology**					
	Task 1			**Task 2**			**Task 1**			**Task 2**		
	C_{oe}	C_{hc}	C_{tt}	C_{oe}	C_{hc}	C_{tt}	C_{oe}	C_{hc}	C_{tt}	C_{oe}	C_{hc}	C_{tt}
Setting 1 (Avg)	11.9	0	11.9	9.9	0	9.9	9.2	0	9.2	6.5	0	6.5
Setting 2 (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%
Setting 3 (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

Table 4. Average costs (in \$) for the ontology engineer (C_{oe}), crowd-workers (C_{hc}) and the entire task (C_{tt}) across ontologies and settings. Setting 3 uses a less expensive HC strategy by requesting 3 judgements per unit and paying \$0.01 per task.

moderate and range from 19.7% to 60.5% and average over all tasks to 39%, thus meaning that in Setting 2 the costs of the total tasks are about 60% of the tasks solved manually. It is important to note, however, that the task level cost savings will ultimately depend on the cost that ontology engineers are willing to pay to crowd-workers. In our settings, we payed \$0.05 per task and requested 5 judgements for each unit, but a high number of micro-tasks are payed much lower, mostly at \$0.01 per task and request a lower number of judgments, typically 3 (last row in Table 4). From the plugin development's perspective, the major goal is reducing ontology engineering costs, as crowdsourcing costs will depend on the time, financial and quality constraints of the ontology engineer and are therefore 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 using the statistical measure of Fleiss' Kappa used to assess reliability of agreement with a fixed number of raters and categorical ratings assigned to a number of items.

Table 5 presents inter-rater agreement per task and per setting. The number of raters per task is given in parentheses. According to the interpretation of Landis and Koch [6] the inter-rater agreement among manual expert evalua-

	Climate Change Ontology		**Finance Ontology**	
	Task 1	**Task 2**	**Task 1**	**Task 2**
Setting 1	0.338 (8)	0.502 (8)	0.500 (7)	0.388 (7)
Setting 2	0.633 (4)	0.841 (4)	0.520 (4)	0.826 (4)
Setting combined	0.392 (12)	0.582 (12)	0.510 (11)	0.494 (11)

Table 5. Fleiss' Kappa values of inter-rater agreement in each setting and in the combined setting.

tors (Setting 1) is moderate. Agreement among the four groups of CrowdFlower workers is substantial. 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 tendency to higher inter-rater agreement, which often corresponds with the majority opinion of manual experts, thereby raising Fleiss' kappa (Setting combined).

Plugin Usability was assessed by asking three ontology engineers to answer to three questions on a scale from 0 (completely disagree) to 5 (completely agree), as follows:

– The plugin functionality was easy to use. [5, 4, 4].
– I would prefer to use the plugin as opposed to doing these tasks manually. [5, 5, 5]
– The use of the plugin saves a lot of time to the ontology engineer. [5, 5, 5]

From free-form comments, the evaluators considered the recursive task verification as very useful in their work as it would allow them to easily verify large parts of the ontology. Given the small scale of the usability evaluation, we consider it as only indicative at this stage of the fact that the Plugin has a good usability. One user suggested to make the plugin data and results more visually appealing, as well as showing the anticipated sum of cost before sending the task to CrowdFlower – both of which is under development.

5 Summary and Future Work

In this paper we introduced the notion of Embedded Human Computation, as a novel computational paradigm relying on a tight integration of HC tasks within ontology engineering workflows. Through an analysis of previous works using HC for ontology engineering, we concluded that a set of basic HC tasks emerge across various stages of the ontology life-cycle. We then presented a novel tool, a Protégé plugin, that allows ontology engineers to easily crowdsource some typical HC tasks from within their ontology engineering working context in Protégé in the spirit of EHC. 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, reduced the time of the ontology engineer (without extending the time of the overall tasks to over 4 hours), returned a good quality data that was in high agreement with ontology engineers. Finally, our evaluators have provided positive feedback about the usability of the plugin.

In terms of future work, we believe the use of HC in our community has matured enough to move on from isolated approaches towards a methodology of where and how HC can efficiently support ontology engineers. Such methodological guidelines should inform tools such as our own plugin. 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. We also plan further and larger scale usability studies and evaluations in other ontology engineering scenarios as well.

References

1. K. Bontcheva, H. Cunningham, I. Roberts, A. Roberts, V. Tablan, N. Aswani, and G.. Gorrell. GATE Teamware: A Web-based, Collaborative Text Annotation Framework. *Lang. Resour. Eval.*, 47(4):1007–1029, December 2013.
2. 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.
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. 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.
6. JR. Landis and GG. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.
7. 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.
8. 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.
9. 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.

10. 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.
11. A. J. Quinn and B. B. Bederson. Human Computation: A Survey and Taxonomy of a Growing Field. In *Proc. of Human Factors in Computing Systems*, pages 1403–1412, 2011.
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. Tania Tudorache, Csongor Nyulas, Natalya F. Noy, and Mark A. Musen. WebProtege: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. *Semant. Web J.*, 4(1):89–99, January 2013.
21. L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, 2008.
22. 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.
23. 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.
24. 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.