ERA-NET CHIST-ERA: FWF 1097-N23

uComp: Embedded Human Computation for Knowledge Extration and Evaluation

D4.2 V2
uComp Report on Ontology Evalution and Extension

Editor(s)	Gerhard Wohlgenannt
Responsible Partner	WU
Status-Version:	Final-1.0
Date:	May.15, 2015
Project Number:	FWF 1097-N23
Project Title:	uComp: Embedded Human Computation for Knowledge Extration and Evaluation

Title of Deliverable:	uComp Report on Ontology Evalution and Extension
Date of delivery to the EC:	May 15, 2015

Workpackage responsible for the Deliverable	WP4
Editor(s):	Gerhard Wohlgenannt
Contributor(s):	Gerhard Wohlgenannt
Approved by:	All Partners

Abstract	This deliverable report (Report on Ontology Evaluation and Extension, v2) presents the work done regarding ontology learning and ontology evolution in context with uComp human computation.
Keyword List:	ontology learning, ontology evolution, human computation, spectral association

Document Revision History

Version	Date	Description	Author
First draft	24/04/2014	initial draft	Gerhard Wohlgenannt[1]
0.1	12/05/2015	extended draft	Gerhard Wohlgenannt
0.2	14/05/2015	final version of v2	Gerhard Wohlgenannt

Contents

1 Executive Summary

This deliverable addresses some central points of WP4: (i) applying human computation in ontology learning, both as proof of concept, and especially on a large scale, (ii) comparing the quality of crowd workers and domain experts, and report on other findings made in the process, (iii) applying the processes and tools in different domains, and (iv) describing the technological foundations and details which were necessary for points mentioned.

The first sections of this deliverable introduce the ontology learning system, and the evidence sources used, information which is needed to better understand the remainder of the deliverable.

A main part of the deliverable is the integration and application of human computation in an ontology learning context. In this deliverable on the one hand we introduce some of the technological details when connecting ontology learning to a crowdsourcing service such as provided by the uComp API. More interesting from a scientific perspective are the findings of combining and comparing crowdsourcing and domain experts. Some results have already been presented in D3.1 when creating and evaluating a plugin to the Protege ontology editor. The Protege plugin delegates various ontology verification tasks (verification of concepts, relations and instances) to crowd workers. In this deliverable we compare results of using crowd workers and domain experts to rate the relevance of new concept candidates generated by our ontology learning system. In a nutshell, the quality of crowd worker ratings is generally good, and in line with the observation made by other researchers, domain experts are more strict in judgement than crowd workers. Large-scale experiments using a Protege plugin, and also a big number of evaluations done directly from the ontology learning system, demonstrate the scalability and quality of using crowd workers as an alternative or in combination with domain experts.

We implemented a front end to manage and inspect all parts of the ontology learning system, and especially of the results found. A focus of the frontend is the analysis and visualization of ontology evolution and trend detection aspects, ie. the changes of the domain according to changes in underlying data.

In order to perform experiments in different domains and languages using various system settings, and to repeat them on monthly basis, we had to invest a big effort in increasing the scalability of the ontology learning system. A major breakthrough was the implementation of the spectral association approximation algorithm to improve the performance of spreading activation.

2 Introduction

Ontologies are a cornerstone technology and backbone for the Semantic Web, but the manual creation of ontologies is cumbersome and expensive, therefore there have been many efforts towards (semi-)automatic ontology generation in order to assist ontology engineers. The process of ontology learning (typically from text) in a first step extracts facts and patterns (evidence) from text, and then turns them into shareable high-level constructs, with the goal to fuel everyday applications like Web search and enabling intelligent systems [24].

Our architecture generates lightweight ontologies in various domains in monthly intervals, we use periodic computations to trace the evolution of those domains. As each ontology is

generated from scratch, it is straightforward to measure and compare results obtained by using different settings (regarding the evidence from heterogeneous input sources).

In uComp WP4 we build on the existing ontology learning system and in a first step need to make it ready for the **requirements** in the project. Those requirements are the following:

- Dealing with noisy and heterogeneous data as evidence sources for ontology learning and evolution.
- Computation of confidence values for ontological entities and compare those over time (ontology evolution).
- Impact refinement, adapt the impact of evidence sources over time according to observed quality of suggestions.
- Dealing with multilingual repositories and do multilingual ontology learning. Identify relations across languages.
- Use EHC to evaluate ontological entities, especially concepts.
- Application of the ontology learning algorithms in different domains.

The following **foundational work** needs to be done to make the system ready for meeting the goals above:

- Refine the evidence integration and confidence management model to deal with noisy and heterogeneous data (Section 4).
- We set up a new layout for the database that save the ontologies. This is covered in Section 5.
- Set up a basic version of Embedded Human Computation (EHC) components, starting with a dedicated Game with a Purpose run on Facebook, see Section 6. Furthermore, connect the ontology learning system to the uComp API, which uses services such as CrowdFlower, and monitor the performance of the service.
- Implement a Web interface to i) control and monitor all ontology learning runs, ii) visualize the result and display evolution aspects on various levels of granularity (Section 7).

The **outline** of this deliverable is as follows: Section 3 gives an overview of the ontology learning system necessary the understand the remainder. Section 4 introduces the heterogeneous evidence sources, their integration, and source impact refinement. In Section 5 we present the new database layout, which is followed by the our efforts towards integration of EHC into the ontology learning framework (Section 6). This includes a comparison of the results of human experts vs. crowd workers, and an analysis of large-scale application of crowdsourcing. The new Web interface to the system is described in Section 7, whereas Section 8.1 contains information about the optimization of the system, especially of a new method to speed up spreading activation.

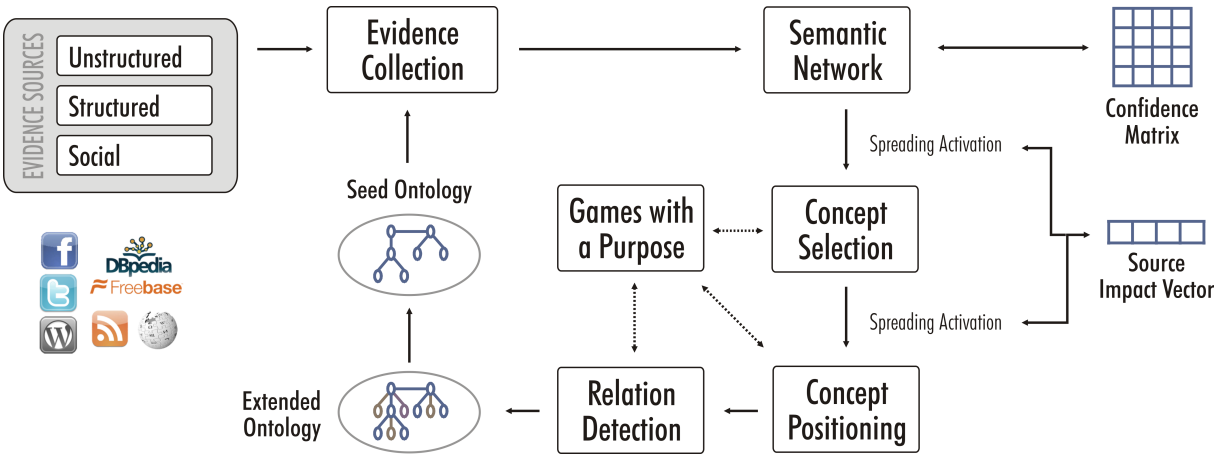

Figure 1: Ontology Learning System Architecture Diagram [23].

3 The Ontology Learning System

The ontology learning system description in v2 of the deliverable is largely the same as in v1, there have been no major updates to the basics of the ontology learning system itself, except for example support for multilingual (German) data sources, which is described in deliverable 4.1. Most updates concern the integration of human computation and ontology learning, and of additional domain experiments, etc., which are described in the sections below.

This section describes the system for learning domain-specific ontologies (T-box) underlying the work presented in this deliverable. The framework evolved since 2005 and has been presented in various publications. The original system [15] learns from domain text only and already includes spreading activation as the major building block to integrate evidence. Weichselbraun et al. evaluate information from social media as additional evidence source [21] and present novel methods for learning non-taxonomic relations [22]. Finally, Wohlgenannt et al. discusses data structures and algorithms for the fine-grained optimization of the system from feedback collected with games with a purpose [23].

The description given here is limited to a basic overview of the system needed to better understand the remainder of the deliverable. Figure 1 gives a graphical illustration of the major building blocks. The initial input to the process is a typically very small seed ontology (a few concepts and relations) in a particular domain. The system starts by collecting evidence for new concepts and relations in heterogeneous sources, i.e. domain text, social sources and structured sources (e.g. WordNet or online ontologies). A semantic network stores the gathered data. This semantic network is then transformed into a spreading activation network, from which new concept candidates are computed. Another round of spreading activation positions each of the new concepts in the seed ontology. Games with a purpose verify the relevance of new concepts and their position, as well as help to optimize the system performance. Finally the extended ontology acts as new seed ontology for the next step of ontology extension. After a pre-configured number of extension steps the system halts.

So what is spreading activation actually used for? The evidence acquisition phase collects a large number of concept candidates and relations to the seed concepts, the number of candidate terms can easily exceed a few thousand. The system includes a big number of

so-called *evidence sources*, for example co-occurring keywords in domain text of US media, UK media, related tags from twitter, disambiguated hyper-/hyponyms from WordNet, and many others. Each of the evidence sources generates a number of new candidate terms (in this context we use *term* and *concept* synonymously) for any seed concept – all this information gets collected in the *semantic network*. A transformation algorithm creates the spreading activation network from the semantic network, the evidence source which suggested the relation influences the weight of the link. Spreading activation detects the most relevant n concepts (where $n = 25$ for example) from all the candidates. Roughly, candidates which are supported by different sources and by sources with a higher link weight ("source impact") are more likely to be selected.

Spreading activation is further used to position the selected concepts with respect to the seed ontology. For this purpose, the system reverses the activation flow in the network, and for every new concept determines the seed concept with the strongest association.

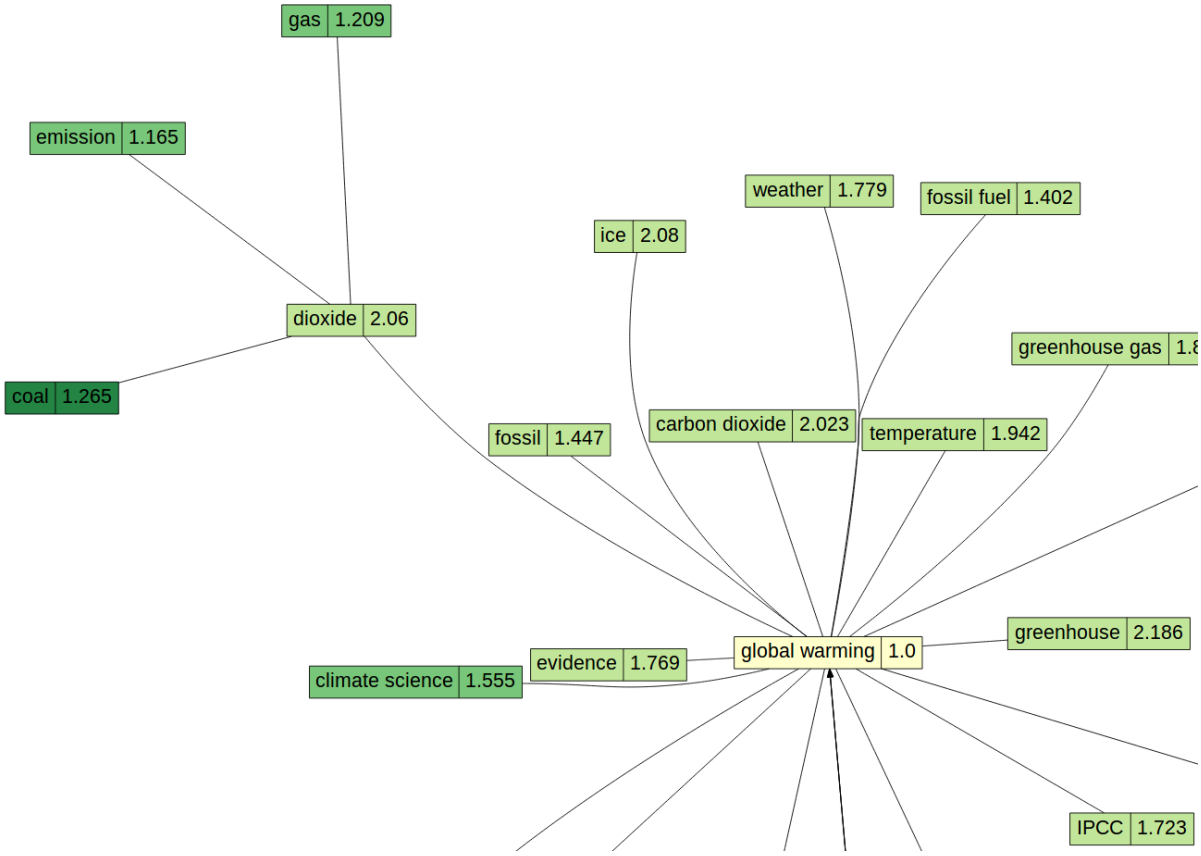

Figure 2: An extended ontology (clipped) – concept labels and relations

Figure 2 shows an extended ontology generated by the ontology learning system. Starting from the seed ontology (yellow), the first round of extension generated and positioned new concepts (light-green, "stage one"). Further rounds, namely stage two (green) and three (dark-green), were used to expand the ontology.

4 The Evidence Sources

A major requirement of WP4 is to allow the system to work with noisy and heterogeneous data as evidence sources. This section describes those sources in Section 4.1, a method to capture and control source impact (Section 4.2), and our approach to adapt source impact ("impact refinement") according to observed quality of suggestions by evidence sources (Section 4.3).

4.1 The Evidence Sources

The first step of the ontology learning process is the collection of evidence data. The input to this process are *seed concepts*. For every seed concept C_s (or rather its label) a suite of algorithms compute related terms. A network data structure (the "semantic network") then represents the relations between seed and candidate terms C_c. Every algorithm that generates related terms for input concepts constitutes an evidence source. The following subsections classify the evidence sources used by the type of underlying dataset (text, social, structured).

4.1.1 Evidence from Text

Ontology learning from text relies on a (domain-specific) text corpus. As our goal is to enable monitoring of ontology evolution, the system ensures that the documents in those domain text corpora stem from the time interval under consideration (in most cases the last week or the last month). The authors have repeatedly and successfully applied the webLyzard suite of Web mining tools (www.weblyzard.com) for generating high-quality domain corpora with the required characteristics. Evidence sources based on temporally segmented domain corpora include [14]:

- *Keywords*: After compiling a target corpus (= set of documents or sentences where the term occurs) for each seed term and source, the system identifies keywords by comparing the term distribution of the target corpus with a reference corpus (all domain documents in the period) by means of co-occurrence statistics.
- *Trigger phrases*: Extract related terms scanning the domain text with Hearst-style patterns.

4.1.2 Evidence from Social Media

We distinguish two basic ways to access online social media for collecting evidence, i.e. typed relations to new term candidates, with seed terms as input:

- Direct access to *related terms* using the TagInfoService interface of the easy Web Retrieval Toolkit (www.semanticlab.net/index.php/eWRT). Currently the system collects evidence from Twitter, Flickr and Del.icio.us. Weichselbraun et al. [21] demonstrate the ontology learning system's benefit from the integration of terminology captured from online social media.
- The webLyzard mirroring services contain components to query Youtube, Facebook, Twitter, etc. to generate domain text corpora. We then apply the extraction methods presented in the previous subsection to these corpora.

unstructured	social		
	delicious	technorati	twitter
global warming	animalcare	agile	aces
building	architects	apple	afghan
coal	atmosphere	architecture	afghanistan
climate change policy	britney	carbon	al_gore
pact	carbonfootprint	carbon-dioxide	alternatefuel
	...	...	...

Table 1: Terms suggested by unstructured and social evidence sources.

4.1.3 Evidence from Structured Data

The existing ontology learning system already uses WordNet as an evidence source and for disambiguation processes. The new system integrates additional structured data sources. The following structured sources serve as a starting point, with further sources being planned to be added over time:

- *DBpedia:* By leveraging components developed in [22], the system queries the DBpedia SPARQL endpoint to gain new terms connected to a seed term. The query uses the seed term as subject, and properties such as *dcterms:subject* or *dbpedia-owl:wikiPageRedirects*. The resulting objects are candidates for the semantic network.

- *WordNet:* WordNet [7] provides hyponyms, hypernyms, and synonyms for terms (Synsets).

Tables 2 and 3 give an overview of evidence sources used in the improved version of the ontology learning system.

	Method		
Data sources			
domain text from:	Keywords/page	Keywords/sentence	Hearst patterns
US news media	1	2	3
UK news media	4	5	6
AU/NZ news media	7	8	9
other news media	10	11	12
Social media: Twitter	13		14
Social media: Youtube	15		16
Social media: Facebook	17		18
Social media: Google+	19		20
NGOs Websites	21	22	23
Fortune 1000 Websites	24	25	26

Table 2: The 26 evidences sources used in the ontology learning process based on domain text.

	Method				
Data source:	hypernyms	hyponyms	synonyms	API	SPARQL
WordNet	27	28	29		
DBpedia					30
Twitter				31	
Flickr				32	

Table 3: The other 6 evidence sources, which are based on WordNet, Social Media APIs, and DBpedia.

4.2 The Source Impact Vector

The source impact vector and impact refinement have already been described in deliverable D4.1, but they are also crucial to the experiments in D4.2, so we have a brief explaination of basic properties in this deliverable, too.

Our approach is novel in its use of an adaptive source impact vector (SIV) to influence spreading activation weights. Previous research used predefined source weights when transforming the semantic network to a spreading activation network.

Equation 1 shows a SIV for a given point in time t_i which contains the impact values I for evidence sources es_j.

$$SIV_{t_i} = \left[\begin{array}{cccc} I_{es_1} & I_{es_2} & \cdots & I_{es_n} \end{array} \right] \tag{1}$$

The source impact values adapt according to user feedback (see below), lowering the impact of sources which tend to yield low-quality evidence data and vice versa. We trace the evolution of the SIV over time. The added temporal dimension transforms the SIV into a source impact matrix. We intialize the SIV (for time t_0) with preset values stemming from the existing system.

4.3 Impact Refinement

Our method gradually adapts the SIV over time according to the observed quality of evidence sources. We assess the quality via the ratio of relevant to non-relevant concept candidates suggested by the respective source. For a smooth adaption of source impact and robustness the SIV depends on devalued data from all past periods within the last 365 days.

The procedure is as follows: i) Do a DB lookup to get *concept candidates* suggested by the respective evidence source. ii) Do a lookup of the GWAP ratings for those concept candidates. ii) Use this data as *rel* and *nrel* to compute the quality of the source (Equation 2). iv) Use Equation 3 to devalue past SIVs linearly, ie. the older the SIV entry, the stronger its devaluation.

$$\Delta SIV_{e,t_0} = 1 + \frac{rel - nrel}{rel + nrel} * \frac{1}{5} \tag{2}$$

$$SIV_{e,t_0} = \Delta SIV_{e,t_0} * \frac{\sum_{i=1}^{n} SIV_{e,t_i} * \left(\frac{365 - age(t_i)}{365}\right)}{\sum_{i=1}^{n} \frac{365 - age(t_i)}{365}} \tag{3}$$

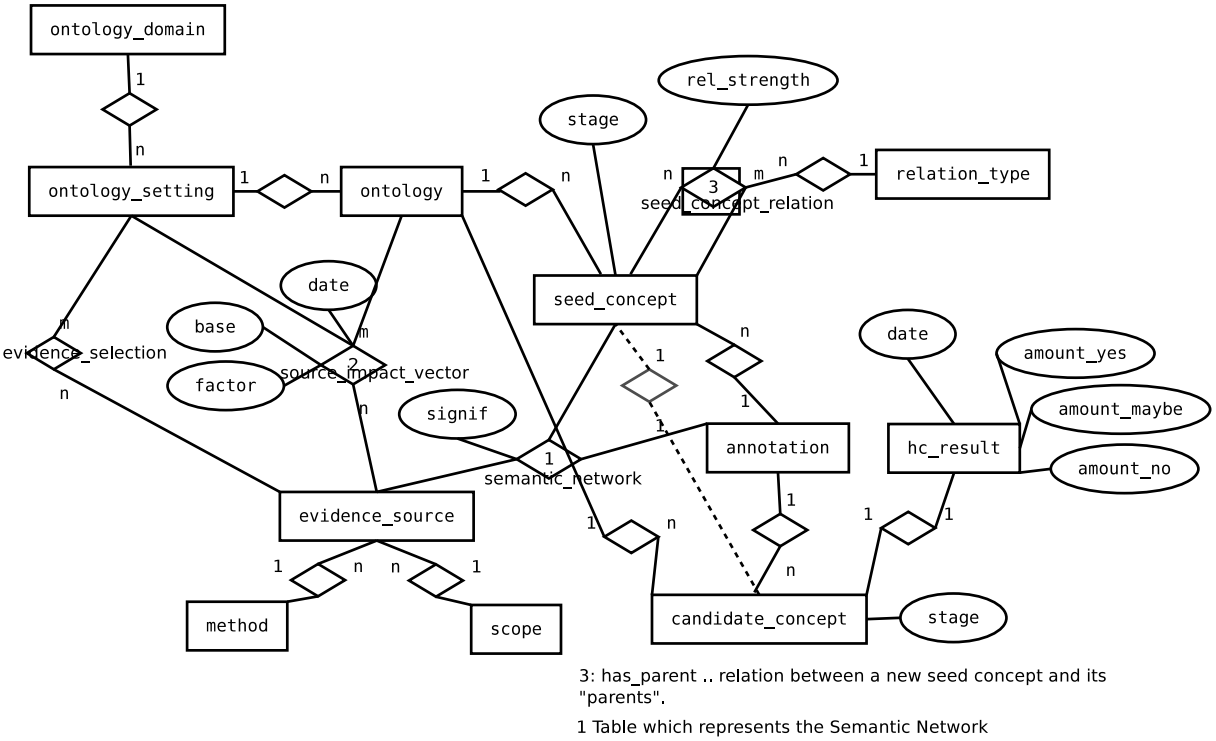

Figure 3: ER schema of the new DB layout

5 Redesign of the Database

The integration of ontology learning and human computation, multilingualiy, and support for new domains made a complete redesign of our ontology learning system database necessary, which is descripted in this section.

This section gives an introduction to the new DB schema used to save our ontologies and all data relevant for evidence integration and source impact refinement. The previous DB format did not have support for flexible selection of evidence sources per ontology domain or the use of multilingual data sources.

Figure 3 shows the ER diagram. We won't go into unnecessary technical detail at this point, but focus on a few key facts:

- Table `ontology_domain` helps to capture different domain, and compute ontologies for that domain with a specific setting.

- Relation table `1` reflects the semantic network, ie. all evidence collected for a single ontology run.

- We can now flexibly select evidence sources per ontology domain / ontology setting – using of the `evidence_source` and `evidence_selection` tables.

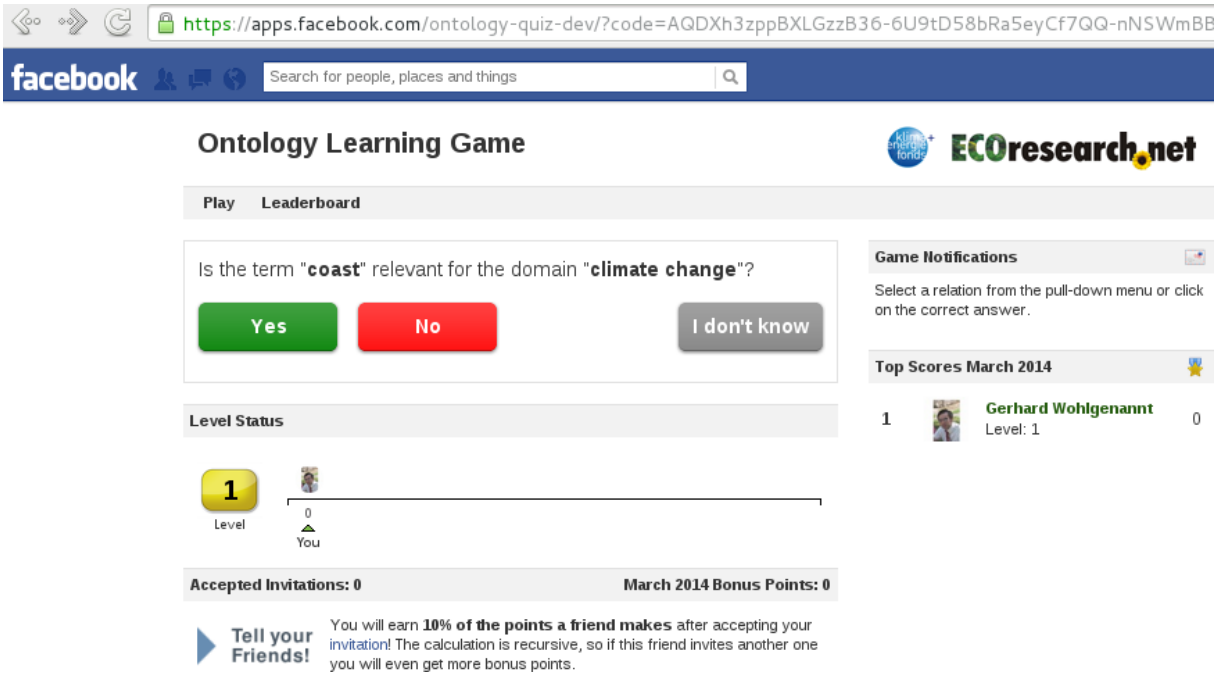

Figure 4: Our Facebook-based game for assessing concept relevant

6 Integration of Ontology Learning and Embedded Human Computation

A rating and validation method is needed to determine the quality of the suggested concepts (and other ontological entities) and therefore the quality of the source which suggested a concept. The rating system itself will return the results of the ratings to the Web service which will start the next computation stage or the impact refinement process if the final stage of an ontology has been reached. The communication between the Web service and the rating system will be explained below.

6.1 Facebook GWAP

A dedicated "game with a purpose" (GWAP) has been set up for this validation process. In this GWAP Facebook users decide whether or not a concept is related to the domain for which the ontology has been computed. The game offers the users the possibility to state that a concept is either related or not related and it is possible to skip questions if the user does not know any answer. The game collects the various answers of the users for each concept and returns them to the ontology learning service. Figure 4 present the interface of the GWAP.

The communication between GWAP and the ontology learning system is done using the JSON format. To send a GWAP task, an object like in the example below is sent to the game. It specifies the date, ontology domain, and the concepts to be evaluated. By convention the game will return the results of a task to the IP address from which the task was sent. Therefore also a port can be provided in the upload to which the results will be returned when combined with the uploader's IP address.

```
{
  10.10.2012 : {
    climate change: [ice,water,...] ,
    finance: [euro,business,...]
  },
  "PORT":5000
}
```

The game tasks are presented to the game players, when results are available. Those are sent back to the ontology learning framework. It is sent in the JSON format similar to the uploaded data. The only difference being that each concept is now at the first position of a new list with the "yes", "i don't know" and "no" votes following it. The result might look like the one below:

```
{
  10.10.2012 : {
    climate change: [ [ice, 1,0,3],[water, 2,3,4] ],
    finance: [ [euro, 10,0,1], [business, 1,0,4] ]
  }
}
```

In a nutshell, the GWAP appends the user votes to the concepts, in the form of (yes-votes, don't know-votes, no-votes). For example, 10 users says that concept *euro* is relevant for the domain of *finance*, 0 users are undecided, and 1 user says the concept is not relevant. We use the majority vote principle to get a final result.

6.2 uCOMP API

At the time the Facebook GWAP (as presented in Subsection 6.1) was implemented there were already plans for a much more extensive application programming interface (API) and a more general GWAP on Facebook that could be customized to the type of task that has to be performed. Therefore the Facebook GWAP in 6.1 can be seen as a very simple prototype for the much more extensive uComp API that followed shortly after.

The uComp API[1] is a tool to let the crowd evaluate automatically computed results via the uComp quiz or via CrowdFlower. CrowdFlower is the world's leading crowdsourcing service specialized in microtasking with millions of contributors worldwide. The uComp Quiz is a game with a purpose (GWAP) originally developed as a Facebook game and it recently became a standalone application. Its mechanics are similar to the micro tasks of CrowdFlower, however no payment is given to the users completing the tasks (except for some small presents which are given to the most active users each month). The uComp API was developed to unify the API interface of these two similar services. For the Ontology Learning System a component had to be written in Python (the Ontology Learning Systems main programming language) to communicate with the uComp API. It has to be able to make all API functions callable within the Ontology Learning Systems code, to convert and upload our result to the API, to manage

[1] http://soc.ecoresearch.net/facebook/election2008/ucomp-quiz-beta/api/v1/documentation/

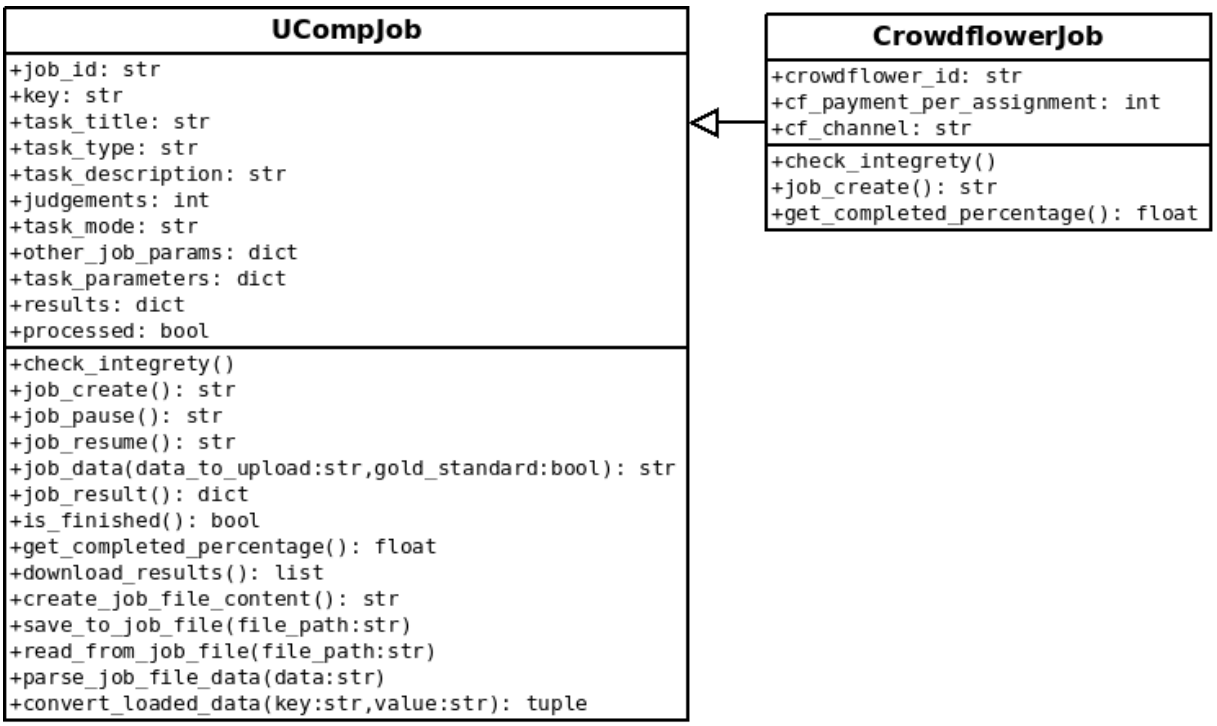

Figure 5: Class Diagram of the uComp and CrowdFlower Modules

the jobs (e.g. check for completion) and to download the results preprocessed by the uComp API.

The documentation of the current uComp API state is available at `http://soc.ecoresearch.net/facebook/election2008/ucomp-quiz-beta/api/v1/documentation/`. For using the uComp API an API key is necessary which can be obtained by contacting the API developers. If jobs should be pushed to CrowdFlower instead of the uComp Quiz then a separate CrowdFlower API key is needed, which is issued by CrowdFlower when creating an account on their web page. The CrowdFlower API key is then used by the uComp API for automatic authentication when pushing data to CrowdFlower.

The base for every task is called a "job". The job has to be created first and contains the basic structure and settings of the task one wants to push to the uComp Quiz or CrowdFlower. The second step is to add the data in form of a UTF-8 encoded CSV file for evaluation to the job. After that, the job can be paused and resumed or the results can be retrieved if the job is finished. All requests to the uComp API have to be performed as HTTP-POST requests and the data sent within the request specific POST variables has to be in the JSON format. The results returned by the API will also be in this format.

For the Ontology Learning System a uComp and a separate CrowdFlower module were created containing the classes UCompJob and CrowdflowerJob as shown in Figure 5. The class CrowdflowerJob is an extension of the UCompJob class. All the basic functionality for communicating with the API is implemented in the UCompJob class as well as some integrity checking on the job information supplied by the user. The CrowdflowerJob just adds a layer of abstraction by automatically handling some CrowdFlower specific settings needed by the uComp API for a job in CrowdFlower. The UCompJob class provides methods for creating,

pausing and resuming jobs, for uploading data and retrieving the results, for getting status information on a job such as the percent of completed data rows, and for serialization and loading of UCompJob instances as JSON files. For the creation of a UCompJob instance the uComp API key and some basic parameters for job such as its name and description need to be given. Alternatively one can specify a file containing a previously serialized job from which all necessary information will be loaded. The usual procedure for a new job is:

1. prepare your data and job configuration parameters
2. create a UCompJob instance with the required parameters
3. call job_create
4. call job_data with your data in CSV format to upload it
5. periodically call is_finished to see if the results ready
6. call download_results to fetch the result CSV from the uComp API

General job parameters defined by the API are *key*, *task_title*, *task_type*, *task_description*, *task_mode* and *judgements*. They are required by the API for every possible type of job. Parameters starting with the prefix "cf_" are CrowdFlower specific parameters, which are not needed when the uComp Quiz is used. They are *cf_payment_per_assignment* and *cf_channel*. Task specific parameters start with the prefix "ts_" and are *ts_default_categories* and *ts_multiple_choice*. When creating an UCompJob or CrowdflowerJob instance all of the general and task specific parameters listed above are available as parameters in both of the constructors of the two classes. The CrowdflowerJob class additionally provides the constructor parameters for the CrowdFlower specific API parameters. However, for future compatibility additional parameters could be supplied via the constructor argument *other_job_params*.

To store the information about a job the uCompJob instances offer serialization (save_to_job_file) and loading (read_from_job_file) methods, which will write the information about the job instance as JSON into a text file. An example of such a serialized job can be seen below, in which the task is to decide domain membership of concepts for the domain "Tennis":


```
{
  "crowdflower_id": 753724,
  "task_type": "classification",
  "job_id": 1093,
  "cf_channel": null,
  "judgements": 5,
  "results": {
    "crowdflower_id": "753724",
    "crowdflower_details": {
      "all_units": 5,
      "completed_non_gold_estimate": 5,
      "needed_judgments": 0,
      "tainted_judgments": 0,
```

```
      "completed_gold_estimate": 0,
      "all_judgments": 25,
      "golden_units": 0,
      "completed_units_estimate": 5,
      "ordered_units": 5
    },
    "crowdflower_result_url": "http://api.crowdflower.com/v1/
                  jobs/753724.csv?full=1&key=33NQODNnoasn",
    "quiz_jobid": 1093,
    "quiz_result_url": "",
    "finished": true,
    "result_url": "https://www.ecoresearch.net/soc/facebook/
                  election2008/ucomp-quiz-beta/api/v1/
                  exportCSV.php?job_id=1093&key=W23rn3nn3n3NDNoa",
    "quiz_details": ""
  },
  "task_description": "Is the given concept relevant for the domain 'Tennis'?",
  "cf_payment_per_assignment": 1,
  "task_title": "Ontology-Domain Concept Membership Rating",
  "other_job_params": {
    "crowdflower_id": 753724,
    "cf_payment_per_assignment": 1
  },
  "task_parameters": {
    "ts_default_categories": {
      "1": "Yes",
      "3": "No",
      "2": "I don't know"
    },
    "ts_multiple_choice": "false"
  },
  "key": "W23rn3nn3n3NDNoa",
  "task_mode": "crowdflower",
  "processed": false
}
```


The modules presented in this section are available for download in the code of the Ontology Learning System. They are located at the paths */src/webservice/hc_evaluation/ucomp.py* and */src/webservice/hc_evaluation/crowdflower.py* inside the project's source code folder and are ready to be used without the rest of the project's code.

6.3 Verfication of different ontological elements: concepts, relations, instances

In deliverable D3.1 we described the development and evaluation of a plugin to the Protege ontology editor aiming to verify ontology elements – concepts, relations, instances – with the help of human computation. We described the results and findings in detail in D3.1, please see there for reference, here we will focus on a short summary of results and findings.

Additionally to the experiments using the Protege plugin editor, we used human computation (via the uComp platform) directly from our ontology learning system. Some of the findings and results are described in sections 6.5 and 6.6.

In D3.1 we investigated the idea of closely embedding crowdsourcing techniques into ontology engineering. Through an analysis of previous work using crowdsourcing for ontology engineering, we concluded that a set of basic crowdsourcing tasks are repeatedly employed 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. This tool includes the verification of concepts, relations and instances from within Protege. More precisely, it verifies concept relevance, domain-range restrictions on relations (properties), instanceOf relation, subClass relations, and also has a module to suggest relation types to unlabeled properties.

Our evaluation focused on assessing the concept of the 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. 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 provided positive feedback about the usability of the plugin.

6.4 Large-scale ontology verfication with human computation

Our large scale ontology verification efforts with human computation were two-fold: on the one side, we did large-scale experiments using the Protege plugin described in deliverable D3.1. Furthermore, we used human computation to verify concept candidates in our ontology learning system. The ontology learning system learns ontologies in multiple domains in monthly intervals, thereby in total leading to a big number of human verification tasks using the uComp API.

Protege plugin As already noted, a detailed description of experiments and results is to be found in deliverable D3.1, please see there for details.

In a nutshell, we did a set of scalability evaluations aimed (1) to asses how well the proposed concept would scale to large ontologies and (2) to investigate whether crowd-workers can replace domain experts in specialized domains. The cost reductions were significant, with

the crowdsourcing payments accounting to only a quarter or less of the estimated ontology engineering costs. Timings are comparable among the two approaches, however, we believe that a self-managed batching of the tasks would have lead to faster completion of tasks as opposed to using the built-in CrowdFlower batch based processing. The obtained quality depends on task difficulty, and ranges from very high (99% accuracy) for a simpler task to an acceptable 89% accuracy for the more difficult task of judging subsumption correctness. As such, our results are in-line with earlier studies that obtained similar quality results from crowds in specialized domains such as philosophy [6] and bio-medicine [17].

Ontology Learning In sections 6.5 and 6.6 we describe the results and comparisons between the use of human computation of and domain experts to evaluate concept candidates generated by the ontology learning system.

6.5 Comparison of evaluation results – Domain experts versus human computation

Table 4 shows a comparison between CrowdFlower based evaluation and the evaluation of suggested concepts by domain experts. It contains the absolute number of relevant concepts and the percentage of relevant concepts in relation to the total amount. The numbers shown are represent an aggregated value over ontologies created between January 2015 and August 2015 for the domain "Tennis". For each month an ontology rated by domain experts as well as one rated by CrowdFlower workers was calculated. These ontologies were calculated in three stages with a limit of 50 annotations per evidence source and stage. The top 25 suggested concepts were sent to CrowdFlower or the domain experts to be evaluated. Concepts rated to be relevant for the given domain were then included into the ontology for the next stage. As

-	Domain Experts	CrowdFlower Workers
Avg Relevant Stage 1	16.7	18.5
Avg Relevant Stage 2	11	16.83
Avg Relevant Stage 3	7.14	12
Avg Percent Stage 1	63.32%	74.47%
Avg Percent Stage 2	43.62%	68.19%
Avg Percent Stage 3	27.33%	49.22%
Avg Relevant Total	33.86	47.33
Avg Percent Total	44.54%	64.06%

Table 4: Relevant concepts for domain "Tennis"

shown in Table 4 domain experts are more restrictive in their choice of which concepts count as relevant for the domain. CrowdFlower workers in general accepted 64% - roughly two thirds - of the suggested concepts as relevant for the domain while at the same time domain experts on average rated only 44% as relevant.

Table 5 shows the list of candidate concepts for the domain *Tennis* from January 2015, for which domain experts and Crowdflower workers had different opinions on weather to include them into the monthly ontology or not. The column "Crowdflower Decision Support" displays

the percentage of Crowdflower users that voted for or against the suggested concept. Concepts

Candidate Concept	Domain Experts	Crowdflower	Crowdflower Decision Support
defender	Yes	No	94.44%
australia	Yes	No	60.00%
title	Yes	No	60.00%
pitch	Yes	No	54.17%
attack	No	Yes	100.00%
draw	No	Yes	80.00%
league	No	Yes	80.00%
manager	No	Yes	76.92%
half	No	Yes	60.00%
world	No	Yes	60.00%
side	No	Yes	58.33%
pass	No	Yes	40.00%
run	No	Yes	40.00%
test	No	Yes	40.00%

Table 5: Disputed candidate concepts

were rejected by the domain experts mostly because they were too broad and not specific enough for a single domain. The candidates *half*, *world*, *side*, *pass*, *run* and *test* were rejected by the domain experts but accepted by the CrowdFlower workers. However all of them were only accepted by a slight majority of positive votes over the negative ones, or in some cases (the ones that only scored 40%) most users choose to answer that they are not sure. After reviewing these concepts one can clearly say that they are either too broad or do not fit at all in the domain *Tennis*, which is reflected by the decision support. On the other hand some decisions of the domain experts may also be incorrect. In particular the candidate *defender*. It can be argued that this concept is also too broad to be part of the *Tennis* domain. In the case of the candidate *australia* the domain experts were correct to accept it as the Australian Open takes is held in January. It seems that part of the CrowdFlower workers were aware of this fact, but 60% were not.

6.6 Comparison of Human Computation in different domains

For the evaluation of the evolution, ontologies from different domains were selected and re-computed once every month to trace their evolution. All of the settings, that are included in the following comparison, were computed with the same basic configuration. They only differ in the domain and in some cases language or evaluation settings as visible in Table 6. For all of the included settings ontologies were created once per month from January 2015 until July 2015.

The evolution of a domain is evaluated by tracing added and removed concepts in ontologies of consecutive months and comparing it to the total amount of concepts in the ontology. The absolute values for each ontology can be seen in Table 7 and 8.

Table 9 shows the relative change of ontologies in those intervals. The values presented in this table are calculated as $1 - (|concepts_{nton+1}|/|ontology_{n+1}|)$. The term $|concepts_{nton+1}|$

Setting	Domain	Language	Evaluation
spectral_brute_limit_50	Climate Change	English	Domain Experts
tennis_spectral_brute_limit_50	Tennis	English	Domain Experts
tennis_spectral_brute_limit_50_crowdflower	Tennis	English	CrowdFlower
nahost_spectral_brute_limit_50_crowdflower	Nahost-Krise	German	CrowdFlower

Table 6: Overview of included settings

Domain	Jan		Feb		Mar		Apr	
Climate Change (E)	+14	-16	+17	-17	+12	-12	+17	-13
Tennis (E)	+15	-37	+16	-15	+12	-18	+10	-7
Tennis (CF)	+7	-23	+26	-13	+17	-28	+24	-12
Nahost-Krise (CF)	+37	-24	+17	-38	+39	-26	+34	-42

Table 7: Amount of added and removed concepts (Jan-Apr 2015).

represents the amount of concepts that exist in the ontology at calculation date n as well as in n+1. $|ontology_{n+1}|$ is the size (=amount of concepts) of the ontology at calculation date n+1.

The domains *Climate Change* and *Tennis* experience only little change over time - on average between 26 and 35%. On the other hand, the German domain *Nahost-Krise* is subject to dramatic changes over time - on average around 81%. From the data, we can see that the evolution of a domain is mostly depending on the subject of the domain and not much on the evaluation method used (CrowdFlower or domain experts in this case). Both *Tennis* domains - the one evaluated by experts as well as the one by CrowdFlower workers - show almost the same amount of volatility. Therefore, outsourcing the task of evaluating candidate concepts to crowd workers does not negatively impact the evolution of ontologies in terms of higher or lower concept volatility compared to domain experts.

Between January 2015 and July 2015 the ontology learning system suggested 217 different candidate concepts for ontologies calculated with the setting *tennis_spectral_brute_limit_50_crowdflower* and domain *Tennis*. Table 10 shows that around two thirds of the candidate concepts in the domain *Tennis* CrowdFlower workers make the same decision as domain experts. After reviewing the cases where CrowdFlower and domain experts judgements differ, the majority of the judgements of the domain experts are the correct ones. CrowdFlower workers tend to also include concepts into ontologies that would be rejected by a

Domain	May		Jun		Jul	
Climate Change (E)	+11	-16	+8	-17	+14	-8
Tennis (E)	+8	-8	+5	-15	+7	-4
Tennis (CF)	+12	-20	+10	-12	+18	-11
Nahost-Krise (CF)	+32	-31	+45	-35	+44	-44

Table 8: Amount of added and removed concepts (May-July 2015).

Domain	Jan	Feb	Mar	Apr	May	Jun	Jul	AVG	STD
CC (E)	27.4%	33.3%	23.5%	30.9%	22.0%	19.5%	29.8%	26.6%	5.1%
T (E)	18.4%	50.9%	42.5%	46.2%	27.3%	23.8%	36.7%	35.1%	12.2%
T (CF)	39.5%	41.0%	36.4%	27.8%	22.2%	19.2%	24.1%	30.0%	8.8%
N-K (CF)	69.9%	53.1%	86.7%	91.9%	84.2%	93.8%	91.7%	81.6%	14.9%

Table 9: Percentage of ontology change in between months

Domain	Candidates	Matching	Different
Tennis	217	149	68

Table 10: CrowdFlower vs. domain experts: judgements of concept candidates

domain expert as been to broad or ambiguous, e.g. "run", "play", "shoes", "grass". However there were also a number of concepts accepted by the CorwdFlower workers, that clearly do not belong to the domain *Tennis*, e.g. "dog", "boxing", "black", etc. When comparing the amount of CrowdFlower workers accepting a concept versus the amount rejecting it, one can observe that the higher if the percentage of workers accepting a concept is the more likely it is to belong to the domain in question. From our observations we can see that if more that 70% voted in favour of a concept, then this concept was most likely part of the domain or at least vaguely connected to it.

7 Interface for Tracing Evolution and System Management

This section gives an introduction of the Web interface which has been set up in uComp to management and monitor ontology learning runs, as well as to visualize ontology evolution.

Ontology evolution is concerned with the adaptation of the ontology to changes in the domain (data-driven change), changed user requirements (user-driven change) or to correct flaws in the original design. Ontology evolution requires frequent updates or rebuilding of the ontology, esp. if investigating emerging trends and patterns in highly dynamic domains. In such a context, a greatly automated ontology learning process is very beneficial. We show a prototype that aims to keep manual input in ontology learning and evolution to a minimum by automating the workflow in the ontology learning cycle. It delegates demand for human input to sources that are cheaper and much more scalable then conventional evaluation by domain experts. So, one of the goals is to minimize manual (domain expert and engineer) effort in repeated ontology learning cycles.

7.1 Related Work in this Area

The evaluation of newly acquired concept candidates with Games with a Purpose (GWAPs) or human labor markets such as CrowdFlower is a central factor to make our system scalable. Noy et al. [17] demonstrate the suitability of Crowdsourcing with Amazon Mechanical Turk for evaluating hierarchical relations in ontologies. GWAPs have already been used for example

for mapping Wikipedia articles to specific classes in the Proton ontology in the OntoPronto game [19] or for relation detection between concepts [18]. Existing tools typically do not offer a tight integration of evaluation results into the learning algorithms, however.

Ontology evolution can be defined as the "timely adaptation of an ontology to the arising changes and the consistent management of these changes" [8]. It helps to keep ontologies up-to-date and useful. The presented prototype integrates heterogeneous input sources in the evolution process, which to our knowledge is a novel approach except for initial efforts in the RELExO framework [16]. In contrast to the Probabilistic Ontology Model (POM) in Text2Onto [2], which aims at change management aspects of ontology evolution, our automated approach targets the detection of trends and patterns in the data structures underlying and reflecting the ontology.

7.2 The Web Service & Administration Interface

Here we includes technical information about the Web service and the corresponding administrative interface. The main function of the Web service is to guide the workflow, ie. calling the involved components with the right parameters and handling the communication between internal and external services.

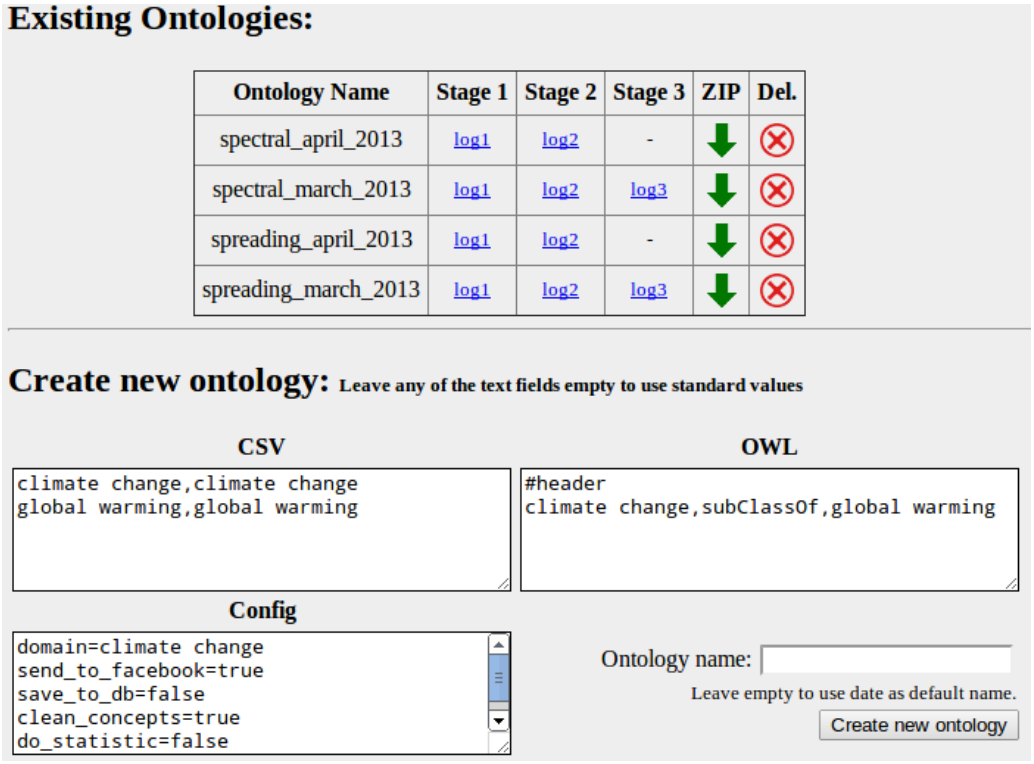

Figure 6: The Administration Interface (clipped)

In our environment, a cron job initiates the generation of new ontologies for all predefined configurations at the end of each month via the REST API of the Web service. A monthly interval is appropriate for our purposes, any other interval is conceivable.

The Web service handles the following jobs which help to minimize manual intervention:

- Check for the existence and correct installation of the required Linux and Python components and the availability of the keyword computation service; notify the user if anything is missing.

- Create the folder structure for new ontologies in the file system

- Handle and save log, config and JSON files for each ontology

- Create graphical representations of the created ontologies for each stage

- Compute new source impact values based on the results of the evaluation.

Figure 6 shows parts of the administration interface. The interface is divided into four parts. At the top (not shown in the screenshot) it displays information about the current status of the system and provides a link to the Web service's global log file. Below there is a list of ontologies existing in the system. For any ontology the user can view the logs for the three stages, download all data or delete it. The logs also contain the resulting ontology graph.

The user has a wide variety of parameter settings to choose from, these can be grouped into the following classes:

- *Algorithms and evidence sources:* Set the algorithms to be used to create the new ontology (eg. *spreading activation* or *spectral association*), or set the period of time to be used.

- *Testing:* Just compute the ontologies, but do not save the results into the database (`save_to_db`), save the results into another database to better separate results for production and testing environments (`db_name`), do (not) update the source impact values after the completed run (`do_statistics`).

- *Evaluation:* Disable evaluating and filtering terms via the evaluation service but just keeping all concept candidates automatically (`send_to_facebook`), or not filtering the concepts even if GWAP evaluation has been done (`clean_concepts`).

The text areas *CSV* and *OWL* are for entering the seed ontology for a new ontology learning process. The OWL text area receives the seed concepts and their relations as triples of subject, predicate and object. These concepts are consistent with the CSV area where a regular expression can be set for each concept; the text based evidence sources (eg. keyword detection) use the regular expression as a lexical representation of the concept.

Finally the user can give the new ontology a name, if omitted, a name including creation date and time will be generated.

The last part of the interface (not shown in the screenshot) displays information about ontology computations currently running, including their names, starting time, parameter settings, etc., and gives the option to terminate running computations.

7.3 Automation

A lot of effort has been made to automate the system as far as possible. A Web service (see next section) controls the workflow, evaluation (GWAPs/CrowdFlower) is the only task in the learning cycle where human input cannot be avoided. Furthermore, to speed up computations we use caching strategies in various processes:

- The evidence collection phase covers processes that are computationally complex (such as the computation of keywords via co-occurrence statistics) or call third party APIs. With the help of the eWRT toolkit [2] the framework applies fine-grained caching strategies to only call the respective evidence collection service for a seed when the necessary data cannot be derived from previous computations already existing in the system.
- The evaluation service (Facebook GWAP) stores the results of past concept validation processes, and lets users only evaluate entirely new concepts. To allow for changes in the domain, concepts have to be re-evaluated after a period of six months.
- To improve the run-time performance of the spreading activation algorithms we experiment with an approximation technique called spectral association [10].
- When manually calling the ontology extension process, eg. for experimenting with parameter settings, new domains or revised code, various steps in the process can be deactivated easily and thereby forced to re-use existing data.

8 Optimization and Scalablity

8.1 Introduction to Performance Optimization

The addition of new evidence sources, new domains and optimization experiments massively increased the computational demands to our original ontology learning system. We improved scalability by applying advanced caching mechanisms, etc., but one major bottleneck remained, which is the spreading activation algorithm. This part of deliverable D4.2 presents our work on implementing and evaluating the so-called *spectral association* approximation techniques to the spreading activation algorithm.

Spreading activation is a method to search semantic networks, which can be used in ontology learning for example for finding semantically related concept candidates for existing concepts [15]. In the ontology learning system used as the foundation and test bed spreading activation is an essential tool used for the selection of domain-relevant concept candidates from a big semantic network as well as for positioning the new concepts in the ontology [21]. Spreading activation helps us to pick the most relevant concepts and associations from a vast number of evidence (candidate concepts and relations) generated from heterogeneous input sources.

In highly dynamic domains, and especially if the evolution of ontologies is of interest, new versions or updates of ontologies need to be generated in regular intervals. This makes the

[2] www.weblyzard.com/ewrt

run-time performance for the ontology learning algorithms an important factor. Spreading activation is an iterative process and can be time-consuming to calculate, therefore Havasi et al. suggest a method called *spectral association* to compute the resulting activation levels after a number of steps of spreading activation in one step [11]. Spectral association approximates spreading activation using spectral decomposition of the concept matrix C, for details see Section 8.3.

This section compares the two methods, i.e. the iterative spreading activation method and spectral association, in a specific environment for learning lightweight domain ontologies. We evaluate various aspects, including the run-time performance of both techniques and the impact on relevance of the resulting ontologies, discuss the implementation, and provide a general reflection of pros and cons of the methods observed as well as hints on when to apply which technique. The datasets used or domain is not important for the runtime of the evaluated algorithms – only the size of the semantic network.

Section 8.2 provides an overview of related work. The methods of spreading activation and spectral association are described in detail in Section 8.3. Section 3 introduces the ontology learning framework which applies the methods. Section 8.4 evaluates the run-time performance and other characteristics of the methods described earlier, and finally Section 8.5 summarizes the findings and gives an outlook on future work.

8.2 Related Work on Performance Optimization

Spreading activation was first introduced by Collins et al. as a theory of human semantic processing [3]. Spreading activation is frequently used in information retrieval, Crestani provides a survey of spreading activation techniques on semantic networks in associative information retrieval [4]. The conclusions of the survey are positive, spreading activation is capable of providing good results.

Hasan proposes a system to apply spreading activation for information access within organizations [9]. The system integrates (i) documents, (ii) statistically derived information such as terms and entities extracted from those documents and (iii) a precise knowledge in form of an organizational ontology into a spreading activation network. User feedback on relevance of query results adapts the weights in the spreading activation network (learning).

Katifori et al. outline a framework which applies spreading activation over personal ontologies in the context of a personal interaction management system [13]. The goal of spreading activation is context inference depending on the users' recent actions and a populated personal information ontology to support user actions, for example generating suggestions when filling a Web form. The method is extended by augmenting the personal ontology with cached data from external repositories [5]. The selection of external data relates to the (spreading) activation level of entities already in the personal ontology or cached data.

Spectral association is an approximation technique for spreading activation networks, first presented in [11]. The paper describes the Colorizer application, which hypothesizes color values that represent given words and sentences. The common sense reasoning application determines the colors depending on physical descriptions of objects and emotional connotations. Spectral association interpolates colors for unknown concepts based on semantic relatedness to (in terms of color) known concepts. To ease computational complexity, Colorizer uses the spectral association approximation as a measure of semantic relatedness.

Spectral association is a variant of the AnalogySpace representation [20]. AnalogySpace addresses the problem of reasoning over large common sense knowledge bases with the characteristics of noisy and subjective data. The goal is to find rough conclusions based on similarities and tendencies, as traditional proof procedures are not feasible in such an environment. AnalogySpace forms analogical closures of a semantic network through dimensionality reduction.

A number of tools already utilize spectral association. Sentic Corner [1] is an application that dynamically collects audio, video, images and text related to and suitable for a user's current mood and displays this content on a multi-faceted classification Website. The system detects user mood via the analysis of semantics and sentics on the user's microblog (e.g. Twitter) postings. Spectral association is one of the fundamental methods used in the system, it generates semantically related concepts from the descriptions of music, films, images and text itself.

The Glass Infrastructure [10] allows the discovery of latent connections between people, projects, and ideas in an organisation. The system uses spectral association to generate a "semantic space" from project information, where closeness in the space signifies similarity of people, projects and ideas.

8.3 Methods

As here we focus on comparing spreading activation and spectral association, this section will describe them in some detail.

Spreading activation is a technique for searching associative, neural and semantic networks. The method requires a network structure with numeric or discrete relations between the network nodes. In the beginning the activation level of all nodes in the network is set to zero. The process starts by labeling source nodes, that is setting the activation level of one or more nodes to a value higher than the firing threshold (F). Every unfired node with an activation level $> F$ fires to all its connected nodes, the energy propagated results from a multiplication of the energy of the source, the weight of the connection, and the decay factor D. In further iterations unfired nodes that received activation in the previous step will fire to their neighbors. The lower the decay factor D, the less activation is spread to nodes further away from the original source nodes. The process terminates when no more unfired nodes (with activation above the firing threshold F) exist in the network. As a result of the process, the activation levels of all nodes in the network give measures of association to the source nodes, i.e. the higher the activation level of a node in the network at the end, the stronger the association to the source node(s).

Equation 4 outlines the activation level adaption of a single node A_j when receiving energy from node A_i via a connection with weight $W_{i,j}$, reduced by the static decay factor D. Obviously, this energy accumulation is applied for every incoming (and firing) node of A_j.

$$A_j = A_j + (A_i \cdot W_{i,j} \cdot D) \tag{4}$$

As already mentioned, spreading activation can be time-consuming to calculate [11]. Spectral association approximates many steps of spreading activation. The spectral association method starts with the transformation of the spreading activation network into a square symmetric matrix C of concepts. The rows and columns of C are labeled with the concepts from the network, and the values in the matrix represent the relation strength between the concepts.

The relation strength of a concept to itself is 1. If there was no connection in the spreading activation network between two concepts the value 0 results. Step two is to scale rows and columns to unit vectors. Applying C to a vector of activations (of one or more concepts) yields the result of one round of spreading activation. The operator e^C to simulate any number of spreading activation rounds (with diminishing returns) is calculated by Equation 5, see [11]:

$$e^C = 1 + C + \frac{C^2}{2!} + \frac{C^3}{3!} + \ldots \tag{5}$$

As C is square symmetric, it can be decomposed to $C = V\Lambda V^t$. In this eigendecomposition, V is the orthogonal real matrix of eigenvectors, whereas Λ is the diagonal matrix of eigenvalues. We can raise this expression to any power, and cancel anything but the power of Λ. What follows is that

$$e^C \approx V e^{\Lambda} V^t \tag{6}$$

To further ease computation, one can apply *dimensionality reduction* [11], i.e. keeping only the largest eigenvalues and their corresponding eigenvectors.

In our implementation of the spreading activation algorithm we use real valued weights. The weights are normalized in the range [0.0, 1.0]. We experimented with a number of decay factor values, and decided on $D = 0.3$. As we are looking for concepts closely related to and in close vicinity of the seed concept, a small decay factor is appropriate. No firing threshold is used, i.e. $F = 0.0$.

The implementation of the spectral association approximation of spreading activation starts with building the square symmetric concept matrix. Next a normalization step takes place. Havasi et al. propose to use unit vectors for rows and columns [11], but this led to increasing returns in Equation 5 when increasing n in $\frac{C^n}{n!}$ – which is not the expected behavior. Using the *determinant* of the matrix instead for normalization gave results as expected.

We calculate the operator e^C in two ways. The first variant uses spectral decomposition (eigendecomposition) to ease the approximation of e^C, in line with the description in Section 8.3 of the seminal paper [11]. This variant is referenced as *spectral association* or *SPECTRAL* in the evaluation section. For the computation of eigenvectors and eigenvalues we use NumPy's `linalg.eigh` module [12], which is geared towards the generation of eigenvectors for symmetric matrices. NumPy[3] is a package for scientific computing in Python. The second variant, called *brute force* or *BRUTE* just calculates e^C according to Equation 5 – without the use of eigendecomposition. This makes the matrix multiplications more complicated, but saves the computation of eigenvectors. We use *brute force* (i) to validate the correct implementation of the *spectral association* method, and (ii) as a baseline when evaluating the run-time characteristics of spectral association.

Finally, as soon as e^C is computed, it can be used to simulate the activation of nodes simply by multiplying (using the dot-product) e^C with a concept vector where the values for the concepts to activate are set to 1, else 0.

The current version of the spectral association method can be found as an open source package on the Web[4] – free for use and modification. It is written in the Python programming

[3] http://www.numpy.org

[4] `http://wwwai.wu.ac.at/~wohlg/spectral_association`

language and uses `numpy` for all matrix operations. The most important parameter for run-time performance tuning is `APPROX_DEPTH`. In Equation 5, the implementation calculates C as far as the power of `APPROX_DEPTH`. We currently use the value of 10 – which proved to be more than sufficient in our experiments, as the factor gets very small (close to zero) with higher powers. The same parameter is used in the approximation of e^{Λ}, too.

8.4 Evaluation

This section provides an extensive evaluation of the methods described. As the purpose of spectral association is to approximate and speed up the spreading activation process, the evaluation focuses on a comparison of run-time characteristics. Furthermore we check if there are significant differences in the resulting ontologies regarding quality of concepts or quality of concept positioning.

Table 11 presents timings for the ontology extension phase, i.e. the total runtime of the step of classic spreading activation (`SPREADING`), and the two variants of spectral association: with spectral decomposition (`SPECTRAL`) and brute force (`BRUTE`). `concepts` refers to the number of nodes in the spreading activation network, which also determines the size of the concept matrix C in spectral association. The `connections` are the links between concepts in the network. The table includes *average* data over multiple ontology generation runs for each of the three ontology extension stages (`Avg stage-1/2/3`). Furthermore, it depicts the single stage with the most concepts and connections (`biggest`) and a spectral association run which uses dimensionality reduction (`With DR`).

Run	concepts	connections	SPREADING	SPECTRAL	BRUTE
Avg stage-1	2495	3303	00:00:10 (1)	00:05:56	00:00:11
Avg stage-2	3843	9054	00:40:56 (91)	00:30:05	00:00:57
Avg stage-3	6101	18655	00:38:47 (86)	01:22:40	00:01:54
Biggest	6842	22342	00:44:35 (109)	01:43:46	00:02:35
With DR	6842	22342	–	00:35:33	–

Table 11: Run-times of the ontology extension step depending on the number of `concepts` and the number of `connections` for spreading activation and the two variants of spectral association.

It was very surprising to see that the *brute force* method performed best by far overall, for networks with > 5000 connections 20-40 times faster than the other two. Classic spreading activation sometimes outperformed spectral association, especially for small networks, and if no dimensionality reduction was applied.

We performed a thorough analysis to investigate *where time is spent*:

- *SPREADING activation*: As expected, almost all of the computing time is consumed by applying the activation algorithm to the network in concept detection and concept positioning.
- *BRUTE force* spends most of its time (about 70%) generating the operator e^{C} according to Equation 5.

- *SPECTRAL association* uses almost all of its time in the generation of e^C. In doing so, NumPy consumes about 33% for generating the eigenvectors and -values. The matrix multiplication in Equation 6 consumes the remaining computation time. This single matrix multiplication is surprisingly costly, keeping in mind that we have around 10 matrix multiplications (with matrices of same size) in Equation 5 when computing e^C *brute force* – depending on the `APPROX_DEPTH`. Additional investigation showed that the matrices in Equation 6 are very dense, and that the concept matrix C in Equation 5 is typically sparse – an important point further covered below.

Another surprising observation is the huge difference between *Avg stage-1* (2672 concepts, 11 seconds) and *Avg stage-2* (3154 concepts, 26 minutes) for the method *spreading activation*. This is caused by two factors: (i) In stage-1 the network depth is very low (most nodes are connected to the seed concepts directly), whereas in stage-2 the network depth increases – which results in more steps of activation propagation. (ii) The number of activation processes differs. In stage-1 the system just needs to activate the seed concepts, while in stage-2 it positions the new concepts from stage-1 and activates the network. This results in about 60-110 activations, the exact number is given in parentheses in Table 11. A single activation in stage-2 and stage-3 has a runtime of about 20-30 seconds.

The eigenvector variant of spectral association can be approximated further (and thereby sped up) by *dimensionality reduction* (DR, see Section 8.3). Our implementation has an (optional) parameter to set the requested number of dimensions, we used a value of 50 in the experiments. As can be seen in Table 11, line `With DR`, applying DR reduces runtime to about 35%. The 35% largely result from calling NumPy for the calculation of eigenvectors (see above), so this variant is very effective to speed up the process. The use of DR had no impact on the resulting ontologies, the approximation works as expected.

Table 12 depicts memory usage for the three methods depending on the network size. The presented data reflects the maximum amount of memory allocated during runtime of the process.

Run	concepts	connections	SPREADING	SPECTRAL	BRUTE
stage-1	2672	3531	0.5 GB	0.4 GB	0.4 GB
stage-2	3154	12243	1.0 GB	2.0 GB	1.7 GB
stage-3	6842	22342	1.2 GB	3.3 GB	2.7 GB

Table 12: Maximum memory usage of the ontology extension step depending on the stage and algorithm used.

We did not focus on optimizing memory usage at all, the goal of Table 12 is to give rough comparison of the methods, and to observe scaling regarding memory consumption. As expected, memory usage tends to scale linearly with the number of concepts and connections. In variant *SPECTRAL*, memory usage is highest during NumPy's computation of eigenvectors and -values. If required, there will be a number of options to decrease memory usage, for example experimenting with Numpy's sparse matrix types or specific optimizations regarding memory in the Python code. The machine we ran the experiments on had enough memory to prevent any swapping to disk.

Complementing the evaluation of run-time performance, we also compared the quality of results. Spectral association is intended as approximation of spreading activation, so results should roughly be the same, although there are various parameter settings available. We did three ontology extension runs for March, April and May 2013 which extended the seed ontology with 25 candidate concepts in each of the three extension iterations, summing up to 225 candidate concepts. Manual evaluation by domain experts showed that there was no significant difference in the relevance of candidate concepts. The relevance was about 50% for all three methods (50% BRUTE, 50% SPECTRAL, 49% SPREADING). We also investigated the effect of the application of the three methods on the quality of concept positioning; no significant differences were found.

The experiments conducted suggest the following *pros and cons*, as well as *areas of application* for the tree methods:

- *Spreading activation*: Among the pros of spreading activation is the interpretability of the activation process, which can be traced easily, whereas spectral association is rather a black box model. Spreading activation provides parameters for tuning the activation process to fit ones specific needs (mainly with the firing threshold F and decay D), while we do not see a simple way how to apply these tuning parameters in the spectral association process. As stated above, the *depth* of the network, i.e. the number of propagation steps necessary, strongly affects runtime performance. The more propagation steps, the slower the process. Naturally, the number of propagation steps also depends on decay factor D and firing threshold F.

  The application of the method is advisable for small networks and situations in which runtime performance is not an issue, in the (unlikely) case of only one or a few activations needed, or if parameter tuning is required.

- *Variants of spectral association*: In contrast to spreading activation, in both spectral association variants the main factor is the generation of e^C, once this is done activation is very fast - it's just a simple $matrix \circ vector$ multiplication. This is a crucial point, activation processes can be done almost instantly with this method. If the activation process has to be further quickened, one can apply dimensionality reduction by using only the largest eigenvalues and their corresponding eigenvectors [11]. Dimensionality reduction also speeds up the the generation of e^C, in our experiments by a factor of approx. 3. However, it is not applicable in the *brute force* variant (as we do not compute eigenvectors).

  The sparser the concept matrix, i.e. the less connections between concepts, the faster the computation of e^C with the *brute force* method. We also tested with a dense matrix (almost all values $\neq 0$), in that case *brute force* is consistently slower (around factor 2) than *SPECTRAL* overall. But in a typical scenario with a sparse concept matrix the computations in Equation 5 are very efficient.

  - *Spectral association (with eigendecomposition)*: The variant allows the application of dimensionality reduction to further reduce and approximate e^C, and thereby enables even faster activation processes. It is well suited for very dense networks and if the runtime of activation is critical, and not so much the generation of e^C.

– *Brute force*: In our run-time analyses which have to be further confirmed in other environments and programming languages, the described *brute force* approach is surprisingly efficient. *Brute force* is preferable when the number of activations is limited and the time spent in both the generation of e^C as well as activations is critical – as in our ontology learning framework, where the method was performing best by far.

8.5 Conclusions on Performance Optimization

This section compares a classic method for searching networks, i.e. spreading activation, with spectral association. Spectral association approximates the computationally intensive activation process using a matrix operator called e^C. The generation of this matrix via spectral decomposition is complex for large spreading activation nets, therefore spectral association is particularly well suited where e^C is used for many activation processes. In situations where instant results for an activation (e.g. in interactive applications) are needed, spectral association is the only option. We also tested a simpler way to generate the operator e^C, named *brute force*, which – under the circumstances we investigated – provides far superior runtime performance in generating e^C.

The main contributions of this section are (i) extensively evaluating spreading activation vs. spectral association in the domain for ontology learning regarding various sizes of concept networks, (ii) showing that the *brute force* method is very efficient (20–40 times faster than spreading activation) in our experiments, (iii) the provision of an (open-source) implementation of spectral association in Python, (iv) giving hints under what circumstances to apply which method.

Future work will include experimenting with a wider range of network sizes and parameter settings, using other libraries for example to compute eigenvectors, and the application of other programming languages to verify the results.

9 Conclusions

This deliverable (D4.2 v2) provides a detailed view on results in the WP4 workpackage. Firstly, the results include improvements regarding runtime and scalability, the setup of an infrastructure for doing ontology evolution experiments, the preparations to handle multiple domains, and work on impact refinement and integration of ontology learning the Embedded Human Computation. Using the infrastructure and the ontology learning frontend, we present the results from a large body of experiments. These experiments were done in multilingual and heterogeneous domains, over the cause of about 2 years.

References

[1] Cambria, E., Hussain, A., Eckl, C.: Taking refuge in your personal sentic corner. In: Bandyopadhyay, S., Okumurra, M. (eds.) Proceedings of IJCNLP, Workshop on Sentiment Analysis where AI meets Psychology. pp. 35–43. Chiang Mai, Thailand (2011)

[2] Cimiano, P., Maedche, A., Staab, S., Voelker, J.: Ontology learning. In: Staab, S., Rudi Studer, D. (eds.) Handbook on Ontologies, pp. 245–267. International Handbooks on Information Systems, Springer Berlin Heidelberg (2009), http://dx.doi.org/10.1007/978-3-540-92673-3_11

[3] Collins, A.M., Loftus, E.F.: A spreading-activation theory of semantic processing. Psychological Review 82(6), 407–428 (1975)

[4] Crestani, F.: Application of spreading activation techniques in information retrieval. Artificial Intelligence Review 11(6), 453–482 (1997)

[5] Dix, A.J., Katifori, A., Lepouras, G., Vassilakis, C., Shabir, N.: Spreading activation over ontology-based resources: from personal context to web scale reasoning. International Journal of Semantic Computing 4(1), 59–102 (2010)

[6] Eckert, K., Niepert, M., Niemann, C., Buckner, C., Allen, C., Stuckenschmidt, H.: Crowdsourcing the Assembly of Concept Hierarchies. In: Proc. of the 10th Annual Joint Conf. on Digital Libraries. pp. 139–148. JCDL '10, ACM (2010)

[7] Fellbaum, C.: Wordnet an electronic lexical database. Computational Linguistics 25(2), 292–296 (1998)

[8] Haase, P., Stojanovic, L.: Consistent evolution of owl ontologies. In: Proceedings of the Second European Semantic Web Conference, Heraklion, Greece. pp. 182–197 (2005), http://citeseer.ist.psu.edu/haase05consistent.html

[9] Hasan, M.M.: A spreading activation framework for ontology-enhanced adaptive information access within organisations. In: van Elst, L., Dignum, V., Abecker, A. (eds.) AMKM. Lecture Notes in Computer Science, vol. 2926, pp. 288–296. Springer (2003)

[10] Havasi, C., Borovoy, R., Kizelshteyn, B., Ypodimatopoulos, P., Ferguson, J., Holtzman, H., Lippman, A., Schultz, D., Blackshaw, M., Elliott, G.T.: The glass infrastructure: Using common sense to create a dynamic, place-based social information system. AI Magazine 33(2), 91–102 (2012)

[11] Havasi, C., Speer, R., Holmgren, J.: Automated color selection using semantic knowledge. In: AAAI Fall Symposium Series. Arlington, Texas (2010)

[12] Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001–), http://www.scipy.org/

[13] Katifori, A., Vassilakis, C., Dix, A.J.: Ontologies and the brain: Using spreading activation through ontologies to support personal interaction. Cognitive Systems Research 11(1), 25–41 (2010)

[14] Liu, W., Weichselbraun, A., Scharl, A., Chang, E.: Semi-automatic ontology extension using spreading activation. Journal of Universal Knowledge Management 0(1), 50–58 (2005), http://www.jukm.org/jukm\textunderscore0\textunderscore1/semi\textunderscoreautomatic\textunderscoreontology\textunderscoreextension

[15] Liu, W., Weichselbraun, A., Scharl, A., Chang, E.: Semi-automatic ontology extension using spreading activation. Journal of Universal Knowledge Management 0(1), 50–58 (2005)

[16] Maynard, D., Aswani, N.: Bottom-up Evolution of Networked Ontologies from Metadata (NeOn Deliverable D1.5.4) (2010)

[17] Natasha F. Noy, Jonathan Mortensen, P.A., Musen, M.: Mechanical turk as an ontology engineer? In: Proceedings of the ACM Web Science 2013 (WebSci'13). Paris, Forthcoming (02-04 May 2013)

[18] Scharl, A., Sabou, M., Föls, M.: Climate quiz: a web application for eliciting and validating knowledge from social networks. In: WebMedia. pp. 189–192 (2012)

[19] Siorpaes, K., Hepp, M.: OntoGame: Weaving the semantic web by online games. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) 5th European Semantic Web Conference (ESWC). vol. 5021, pp. 751–766. Springer (2008)

[20] Speer, R., Havasi, C., Lieberman, H.: Analogyspace: Reducing the dimensionality of common sense knowledge. In: Fox, D., Gomes, C.P. (eds.) AAAI. pp. 548–553. AAAI Press (2008)

[21] Weichselbraun, A., Wohlgenannt, G., Scharl, A.: Augmenting lightweight domain ontologies with social evidence sources. In: Tjoa, A.M., Wagner, R.R. (eds.) 9th International Workshop on Web Semantics, 21st International Conference on Database and Expert Systems Applications (DEXA 2010). pp. 193–197. IEEE Computer Society Press, Bilbao, Spain (August 2010)

[22] Weichselbraun, A., Wohlgenannt, G., Scharl, A.: Refining non-taxonomic relation labels with external structured data to support ontology learning. Data & Knowledge Engineering 69(8), 763–778 (2010)

[23] Wohlgenannt, G., Weichselbraun, A., Scharl, A., Sabou, M.: Dynamic integration of multiple evidence sources for ontology learning. Journal of Information and Data Management (JIDM) 3(3), 243–254 (2012)

[24] Wong, W., Liu, W., Bennamoun, M.: Ontology learning from text: A look back and into the future. ACM Computing Surveys 44(4), 20:1–20:36 (Sep 2012)