ERA-NET CHIST-ERA: FWF 1097-N23

uComp: Embedded Human Computation for Knowledge Extration and Evaluation

D4.2 V1
uComp Report on Ontology Evalution and Extension

Editor(s)	Gerhard Wohlgenannt
Responsible Partner	WU
Status-Version:	Final-1.0
Date:	May.15, 2014
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, 2014

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, v1) presents the improvements regarding ontology learning and ontology evolution until month 18.
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 v1	Gerhard Wohlgenannt

Contents

1 Executive Summary **3**

2 Introduction **3**

3 The Ontology Learning System **4**

4 The Evidence Sources **5**
4.1 The Evidence Sources 5
4.1.1 Evidence from Text 6
4.1.2 Evidence from Social Media 7
4.1.3 Evidence from Structured Data 7
4.2 The Source Impact Vector 8
4.3 Impact Refinement 8

5 Redesign of the Database **9**

6 Integration of Ontology Learning and Embedded Human Computation **9**

7 Interface for Tracing Evolution and System Management **11**
7.1 Related Work in this Area 12
7.2 The Web Service & Administration Interface 12
7.3 Automation 14

8 Optimization and Scalablity **14**
8.1 Introduction to Performance Optimization 14
8.2 Related Work on Performance Optimization 15
8.3 Methods 16
8.4 Evaluation 18
8.5 CONCLUSIONS ON PERFORMANCE OPTIMIZATION 21

9 Conclusions **21**

1 Executive Summary

This document describes the intitial version of improvements and integration efforts for our ontology learning and ontology evolution framework. We did a lot of work and making the system more efficient and scalable, worked on ontology evolution aspects and started efforts towards integration with the uComp API. As the uComp API is in a very preliminary state yet, integration is only starting. Furthermore we resigned the underlying ontology learning database to make it compatible with requirements in uComp. All foundational work on scalability of the ontology learning system was necessary to achieve uComps' goal of learning ontologies in periodic intervals in multiple domains and with multiple languages.

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 [23].

At the moment, our architecture generates lightweight ontologies in the domain of "climate change" in monthly intervals, we use periodic computations to trace the evolution of the domain. 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:

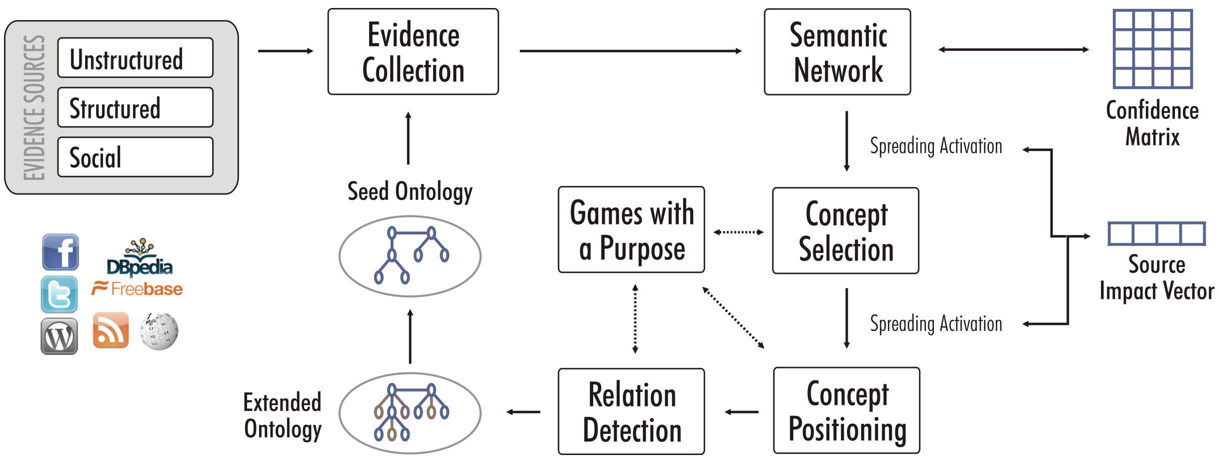


Figure 1: Ontology Learning System Architecture Diagram [22].

- Refine the evidence integration and confidence management model to deal with noisy and heterogeneous data (Section 4).
- A new layout for the database that saves 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.
- 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 initial efforts towards integration of EHC into the ontology learning framework (Section 6). 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. Finally, Section 8.5 concludes the deliverable.

3 The Ontology Learning System

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 [14] 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 [20] and present novel methods for learning non-taxonomic relations [21]. 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 [22].

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 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

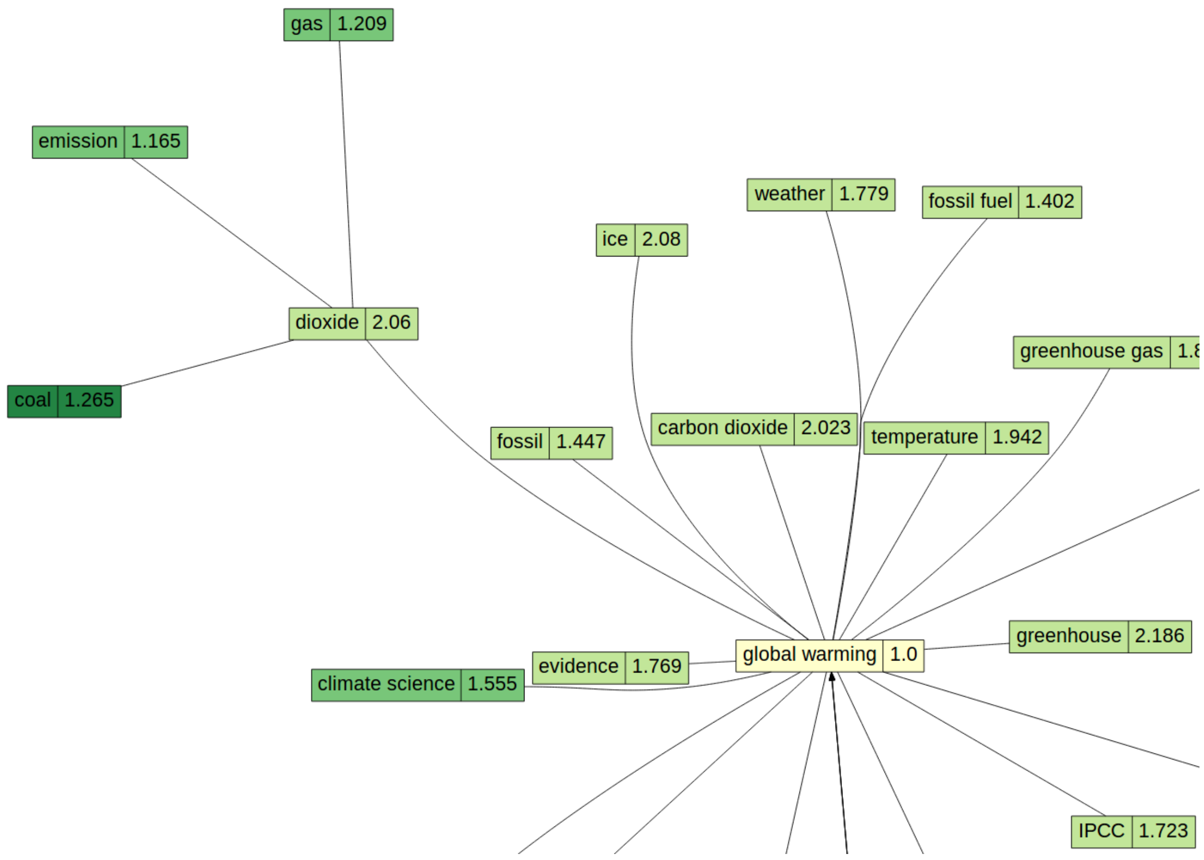


Figure 2: An extended ontology (clipped) – concept labels and relations

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 [13]:

- *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. [20] 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 [21], 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 [6] 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

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.

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

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

5 Redesign of the Database

This section gives a very brief 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.

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 only return the results of the ratings to the Web service which will start the impact refinement process. The communication between the Web service and the rating system will be explained below.

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

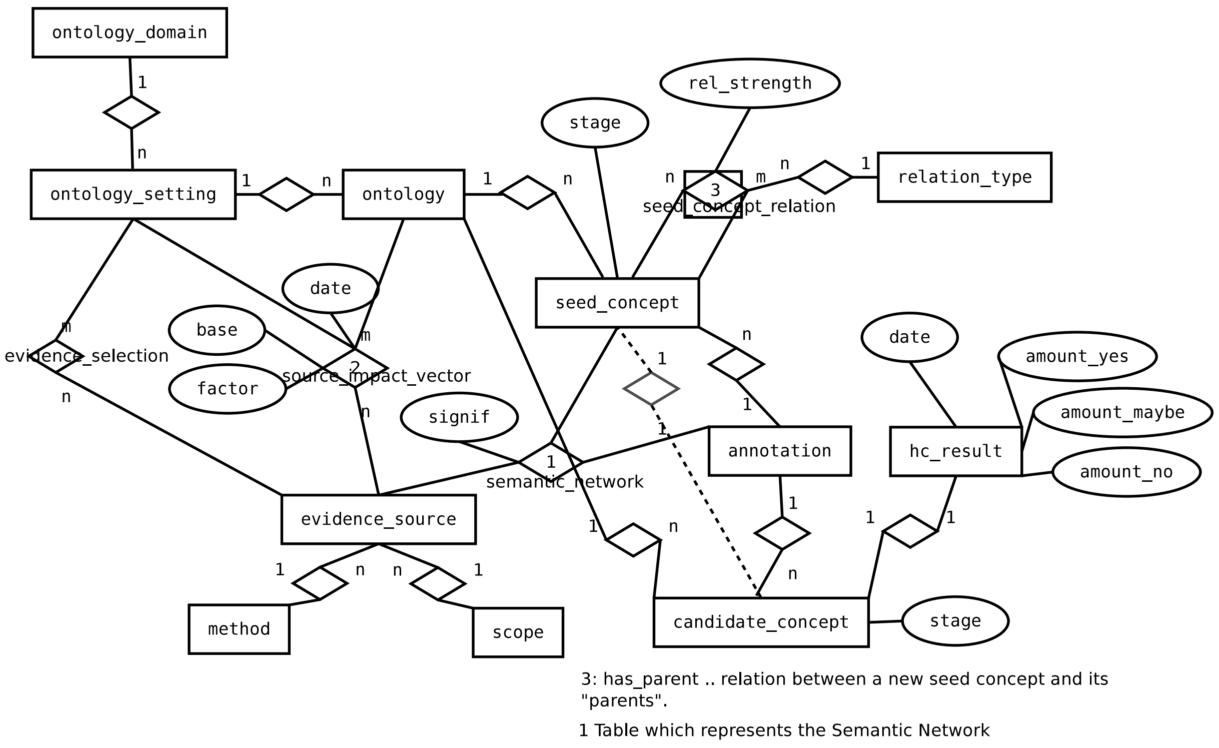


Figure 3: ER schema of the new DB layout

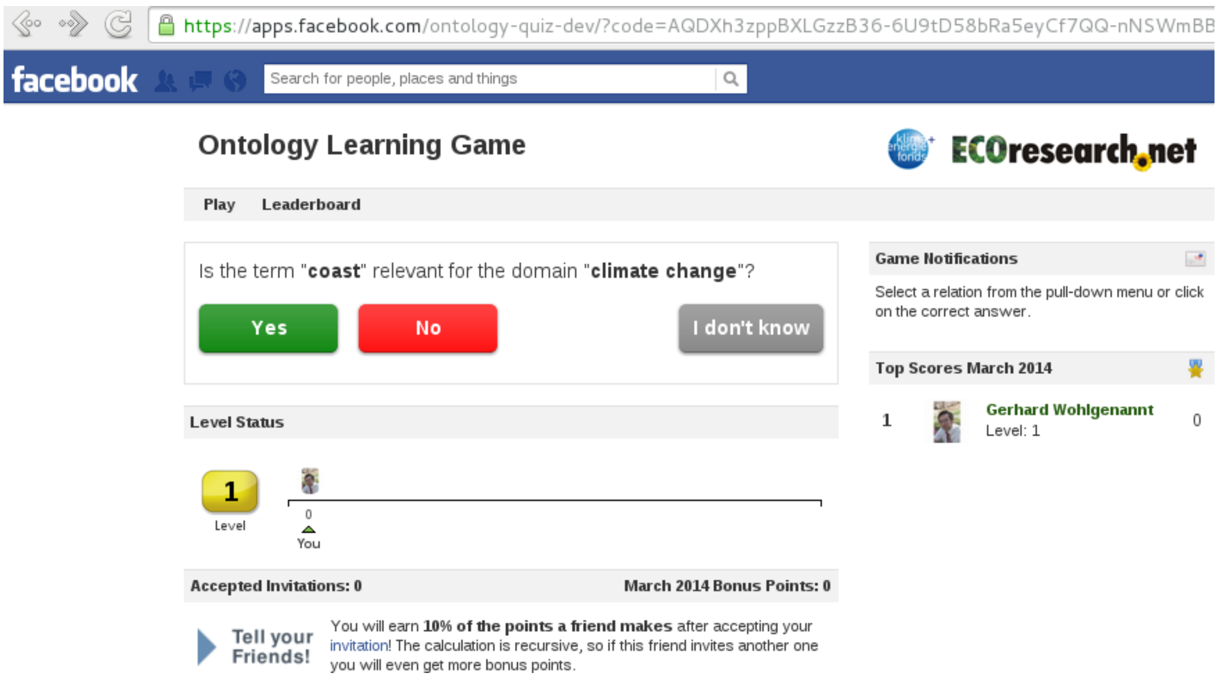
know any answer. The game collects the various answers of the users for each concept and returns them to the ontology learning service. Figure 4 present the interface of the GWAP.

The communication between GWAP and the ontology learning system is done in JSON. To send a GWAP task, a JSON object like in the example below is sent. It specifies the date, ontology domain, and the concepts to be evaluated.

```
{ 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. The result might look like this:

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


https://apps.facebook.com/ontology-quiz-dev/?code=AQDXh3zppBXLGzzB36-6U9tD58bRa5eyCf7QQ-nNSWmBB

Ontology Learning Game

Play Leaderboard

Is the term "**coast**" relevant for the domain "**climate change**"?

Yes No I don't know

Game Notifications

Select a relation from the pull-down menu or click on the correct answer.

Top Scores March 2014

1 Gerhard Wohlgenannt Level: 1 0

Level Status

1 Level 0 You

Accepted Invitations: 0 March 2014 Bonus Points: 0

Tell your Friends! You will earn **10% of the points a friend makes** after accepting your invitation! The calculation is recursive, so if this friend invites another one you will even get more bonus points.

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

In a nutshell, the GWAP API 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.

As the uComp API is just getting ready, there is no prototype yet that uses the features of the uComp API. Future work will focus on using the uComp API for validating ontological entities, and on studying the quality of crowd rating and their impact on the ontology learning cycles.

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. [16] 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 [18] or for relation detection between concepts [17]. 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" [7]. 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 [15]. 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

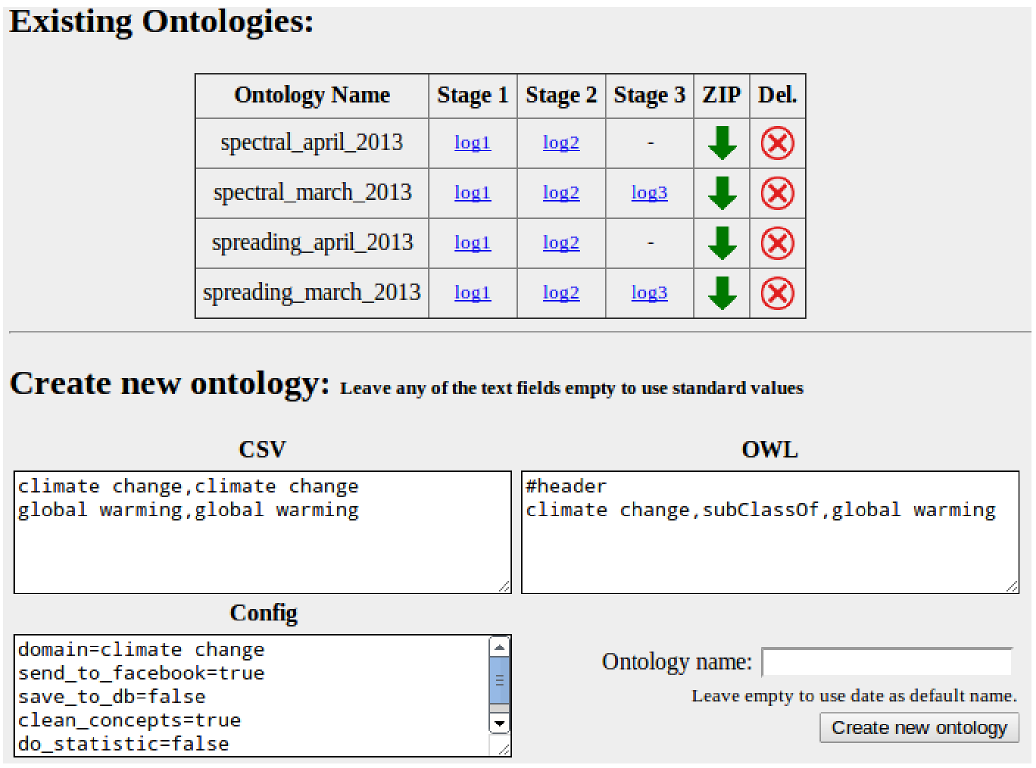
Here we includes technical information about the Web service and the corresponding administrative interface. The main function of the Web service is to guide the workflow, ie. calling the involved components with the right parameters and handling the communication between internal and external services.

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 5 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



Existing Ontologies:

Ontology Name	Stage 1	Stage 2	Stage 3	ZIP	Del.
spectral_april_2013	log1	log2	-	⬇	⊗
spectral_march_2013	log1	log2	log3	⬇	⊗
spreading_april_2013	log1	log2	-	⬇	⊗
spreading_march_2013	log1	log2	log3	⬇	⊗

Create new ontology: Leave any of the text fields empty to use standard values

CSV

```
climate change,climate change
global warming,global warming
```

OWL

```
#header
climate change,subClassOf,global warming
```

Config

```
domain=climate change
send_to_facebook=true
save_to_db=false
clean_concepts=true
do_statistic=false
```

Ontology name:

Leave empty to use date as default name.

Figure 5: The Administration Interface (clipped)

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 [1] 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 [9].
- 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

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 [14]. 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 [20]. 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.

[1]www.weblyzard.com/ewrt

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 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 [10]. 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 [8]. 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 [12]. 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 [10]. 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 [19]. 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 [9] 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 [10]. 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 [10]:

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

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* [10], 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 [10], 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 [10]. 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 [11], which is geared towards the generation of eigenvectors for symmetric matrices. NumPy[2] 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.

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

The current version of the spectral association method can be found as an open source package on the Web[3] – free for use and modification. It is written in the Python programming 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 4 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 4: 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.

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

- *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 4. 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 4, 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 5 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 5: 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 5 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 [10]. 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

In this deliverable (D4.2 v1) a detailed view on many achievements in WP4 has been reported. The achievements include improvements regarding runtime and scalability, the setup of an infrastructure for doing ontology evolution experiments, the preparations to handle multiple domains, and an initial version of impact refinement and integration of ontology learning the Embedded Human Computation.

With this foundational work, we plan to implement all of the requirements for *v2* of D4.2, in a nutshell that is mulitlingual ontology learning in multiple domains combined with ontology evolution and trend detection experiments,

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

[7] 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

[8] 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)

[9] 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)

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

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

[12] 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)

[13] 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

[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)

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

[16] 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)

[17] 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)

[18] 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)

[19] 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)

[20] 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)

[21] 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)

[22] 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)

[23] 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)