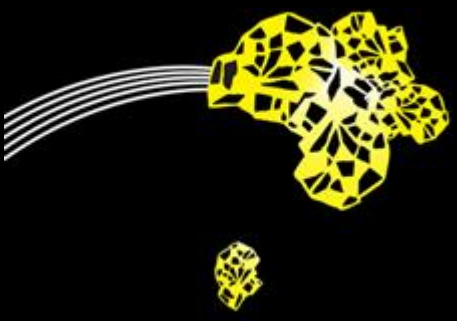
Semi-Automatically Enriching Ontologies: A Case Study in the e-Recruiting Domain

J.F. Wolfswinkel



epiqo.

UNIVERSITY OF TWENTE.

"***The world is everything that is the case.***" *Ludwig Wittgenstein.*

Abstract

This case-study is inspired by a practical problem that was identified by Epiqo. Epiqo is an Austrian company that wants to expand to other countries within Europe and to other domains within Austria with their e-Recruiter system. For the e-Recruiter system to work, it needs domain specific ontologies. These ontologies need to be built from the ground up by domain experts, which is a time-consuming and thus expensive endeavor. This fueled the question from Epiqo whether this could be done (semi-)automatically.

The current research presents a solution for semi-automatically enriching domain specific ontologies. We adapt the general Ontology-Based Information Extraction (OBIE) architecture of Wimalasuriya and Dou (2010), to be more suitable for domain-specific applications by automatically generating a domain-specific semantic lexicon. We then apply this general solution to the case-study of Epiqo. Based on this architecture we develop a proof-of-concept tool and perform some explorative experiments with domain experts from Epiqo. We show that our solution has the potential to provide qualitative "good" enough ontologies to be comparable to standard ontologies.

Preface

It has been a long road for me from my earliest days in the feeble Dutch education system up until this point. Despite all the opportunities that were available to me in life, the many, many obstacles I had to overcome have not made things easy. I am very proud to have achieved this milestone. But as Bill Clinton once said, "*Success is not the measure of a man but a triumph over those who choose to hold him back.*"

First and foremost I would like to thank Klaus Furtmueller of Epiqo for his patience, support and wisdom, that he granted me during my masters project. Further, I would like to thank my supervisors of the University of Twente, Ivan Kurtev and Djoerd Hiemstra, for their valuable comments and guidance. I would also like to thank my family, friends and girlfriend, for their support and believe in my abilities. Lastly, I want to specifically thank Sander Nouta for his understanding and help.

Joost F. Wolfswinkel, 2012, Enschede.

Table of Contents

1. Introduction

This chapter provides the background in section 1.1, the problem statement and research question in section 1.2, objectives and approach of the current research in section 1.3, and concludes with the outline of this report in section 1.4.

1.1. Background

This sub-section gives a short overview of the e-Recruiting domain, ontologies, the Information Extraction (IE) and the Ontology-Based Information Extraction (OBIE) fields, and the company Epiqo (where the case-study was performed).

E-Recruiting

Academically, e-Recruiting is a relatively young research field (Galanaki, 2000), with the first publications dating from 1998 (Bratina, 1998; Hogler, 1998). In the professional field, publications date back as far as 1984 (Gentner, 1984). *"E-Recruiting is the online attraction and identification of jobseekers using corporate or commercial recruiting websites, electronic advertisements on other websites; or an arbitrary combination of these channels including optional methods such as remote interviews and assessments, smart online search agents or interactive communication tools between recruiter and jobseeker/applicant with the goal of effectively selecting the most suitable candidate for a vacancy.*" (Wolfswinkel et al., 2010) To clarify, a jobseeker is a person that is looking for a job. When this jobseeker applies for a job, he/she becomes an applicant. Recruiters are people who actively and passively seek people that somehow, at one point in the future become wanted by the organization(s) they work for. There are two types of e-Recruiting websites: commercial websites and corporate websites. Commercial e-Recruiting websites are portals that bring jobseekers and organizations together, for example monsterboard.nl. Corporate e-Recruiting websites are run by the organizations that seek to hire themselves. These corporate e-Recruiting websites are often part of their main website, for instance the career section of shell.com.

Ontologies

The word "ontology" is used in a wide range of different contexts, all defining ontologies differently. In this research we regard an ontology as a formal and explicit representation of knowledge in a certain domain. This knowledge is represented by concepts that have relationships with each other (Gruber, 1993; Studer et al., 1998). Ontologies can be used to represent some domain for software – this is the way that ontologies will be used for this system – or for communication purposes between human beings, in order for them to have the same conceptualizations of the domain

the ontology represents. This opposed to the Ontology from philosophy, which is about the nature of being (Aristotle, 350 B.C.E.).

Information Extraction and Ontology-Based Information Extraction

Information extraction (IE) is the automatic extraction of information from natural language sources. By processing the natural language texts, IE aims to identify (instances of) certain classes of objects and/or events and their possible relationships (Riloff, 1999; Russell and Norvig, 2003). IE is usually considered to be a subfield of Natural Language Processing (NLP).

NLP is the field regarding the interaction between machines and natural languages. A subfield of IE called Ontology-Based Information Extraction (OBIE) uses ontologies for the information extraction process or even for constructing ontologies (Wimalasuriya and Dou, 2010).

Epiqo

Epiqo is an Austrian company that was founded in 2005, although at that time the company was called Pro.Karriere. Epiqo develops and manages web portals like absolventen.at. It attracts 60,000 visitors per month and has more than 12.000 registered applicants, making it the biggest career platform for graduates in Austria. The development of this portal entailed, among other things, the development of several modules like for instance Rules and Content Taxonomy, which are built as Drupal modules as part of Epiqo's e-Recruiting system "e-Recruiter". Alongside the development of these modules, Epiqo partook in a semantic web research project within the Human Research domain. This project resulted in a web crawler that can crawl job advertisements, an information extraction engine to extract the needed information from these job advertisements, a web service module that communicates between Drupal and the crawler, a taxonomy manager to organize the taxonomy and an indexer that is based on SoIr, for matching between the résumés and the job advertisements.

The vision of Epiqo is to "Develop a powerful, flexible and easy-to-use e-Recruiting solution for enterprises and publishers based on Drupal 7." On top of that, Epiqo wants to expand beyond Austria, starting with the Netherlands. Potential customers would be organizations interested in: running their own recruiting portal, job boards for publishers, niche recruitment sites, customers that want to use it for talent and skills management or recruitment micro-sites. Epiqo seeks to develop a new system that will consist of a distribution of basic features, with the possibility for customers to request additional features. The basic features are job posting and administration, job search abilities, a résumé builder, applicant search abilities, an online application process, and a dashboard. Additional features are job and applicant recommender options, talent

pools, social network integration, a billing system, business intelligence and reporting, and data integration and exchange.

1.2. Problem Statement

In practice, ontologies need to be enriched or even build from scratch in various domains every day. Normally this is done by domain experts, which is a time-consuming and thus expensive activity. Epiqo faces the same problem when entering new markets, whether it is adding either an additional natural language to an existing domain or adding a new domain all together. It costs a considerable amount of time and thus money to develop new ontologies, which hinders Epiqo to expand. Therefore, Epiqo seeks a way to be able build and/or enrich ontologies quicker. The expectations and intentions of semi-automatically enriching and constructing ontologies are that it will considerably reduce the required human effort in the process (Luong et al., 2009). Summarizing, Epiqo wants to save time and it is hypothesized that enriching ontologies semi-automatically for certain domains will be able to deliver this.

> This results in the following research question: *is semi-automatically enriching a given ontology for a certain domain more time-efficient than enriching the same ontology manually?*

When asking such a question is it paramount that the *quality* of both ontologies are "good" enough for the goal they are created for. In this case, the semi-automatically enriched ontology needs to be usable for the use-case of Epiqo. We will use the so-called completeness and exactness of both ontologies to reason about quality. These terms are explained in section 2.5.

1.3. Research Objectives and Approach

The objectives of this research are (1) to develop an approach for semi-automatically enriching domain-specific ontologies, (2) to design a software application that will be able to do this, (3) built a proof-of-concept of this software application for the Epiqo case-study, and (4) performance measurement of time, completeness and correctness of the result ontologies when using this proof-of-concept.

First, we turn to extant literature on topics that deal with similar problems and research directions. Then, we develop an approach to semi-automatically enrich domain-specific ontologies according to the general OBIE architecture. Based on this more or less general solution, we design a software application that adheres to our architecture and addresses Epiqo's requirements. We built a proof-of-concept of this architecture and finally perform experiments to be able to measure time, completeness and exactness.

1.4. Outline

The next chapter delineates related work regarding Information Extraction, Ontology enrichment, Ontology-Based Information Extraction and Performance Measurement. Chapter 3 describes the case-study, the environment, the requirements and the architecture. Next, chapter 4 deals with our proof-of-concept. Chapter 5 describes the experiments that were performed. Finally, chapter 6 is the concluding chapter which describes our contribution, limitations and future work.

2. Related work

This chapter delineates literature on topics that deal with similar problems and research directions as the current research. The information needed to enrich ontologies is mostly available in natural languages, which poses the problem of extracting the information from these natural language resources somehow. As said, in the literature, this is called Information Extraction (IE).

The search started with ontology enrichment and information extraction. In this search we discovered a research field combining these fields called Ontology-Based Information Extraction (OBIE) (Wimalasuriya and Dou, 2010). Below a short report on our literature search: Information Extraction is described in section 2.1, in section 2.2 we describe ontologies, then we describe ontology enrichment in general in section 2.3, section 2.4 deals with OBIE, and finally in section 2.5 relevant performance measurements are discussed.

2.1. Information Extraction

As said, IE is the automatic extraction of information from natural language sources. By processing the natural language texts, IE aims to identify (instances of) certain classes of objects and/or events and their possible relationships (Riloff, 1999; Russell and Norvig, 2003). IE is usually considered to be a subfield of Natural Language Processing (NLP). IE in practice generally works for restricted domains or niches. The most simple IE systems are so-called attribute-based systems, which assume that the entire language source deals with one object of which the system attempts to extract attributes of this object. Regular expressions can be used to handle information collected from the language source. When the natural language source has more than one object, so-called relational-based IE systems can be used. Relational-based IE systems usually contain cascaded finite-state transducers, which are basically concatenations of finite-state automata (FSA) that transform text and pass it on to the next FSA. Often used FSA's include tokenizers, complex word handling, basic group handling, complex phrase handling and structure merging. Tokenizers conform the stream of characters into tokens like words or numbers. The term complex word handling can be confusing, since it deals with combined words or phrases like "software engineer" or "joint venture". Basic group handling divides the identified words into groups. These groups are (a subset of): verb, noun, adjective, adverb, pronoun, preposition, conjunction and interjection. Complex phrase handling combines the basic groups into phrases. Structure merging is the merging of the different structures found in complex phrase handling in order to remove redundant information (Russell and Norvig, 2003). Another widely used technique is the use of a gazetteer list, which is a list of words or phrases that can be recognized in the natural language source. In 2003, Gómez-Pérez et al. presented an overview of different ontology learning projects regarding the extraction of information from natural language sources. The methods needed for obtaining information from the Internet is

surveyed by Yang et al. (2003) and Wimalasuriya and Dou (2010), these include various classification techniques such as Support Vector Machines (SVM), Hidden Markov Models (HMM), Conditional Random Fields (CRF) and Linear Least-Squares Fit. Dumais et al. (1998) concluded that SVM are the most accurate and relatively fast to train. SVM is a model for machine learning, that divides the input into two groups: relevant and non-relevant (Chang and Lin, 2001; Joachims, 1998). Another often used IE method is the construction of partial parse trees, which can be seen as a shallow Natural Language Processing (NLP). When the natural language source is structured, like HTML or XML, the extraction can be easier or additional information can be collected from the structure (tags) themselves. Finally, relatively new is the extraction of information from the results of queries in web-based search-engines. This is often easily obtainable if the search-engines support the use via either REST or SOAP web-services.

2.2. Ontologies

As said, in this research we regard an ontology as a formal and explicit representation of knowledge in a certain domain. This knowledge is represented by concepts that have relationships with each other (Gruber, 1993; Studer et al., 1998).

2.3. Ontology Enrichment

Little approaches have been presented that discuss the use of machine learning for ontology enrichment from Internet sources (Agirre et al., 2000; Omelayenko, 2001). Luong et al. (2009) do present a framework for enriching ontologies using the Internet, with three major steps. First the Internet is searched and crawled for suitable natural language documents based on a relatively small hand-crafted domain ontology. Second, the top ten documents of the result of the first step are filtered using SVM classification based on the relevance for the domain of the ontology. Third, text mining is used to extract information from the result documents of the second step. Text mining is a widely used technique to extract tokens from natural language resources in a certain domain. The actual enriching of the ontology is not described in this particular paper, but suggested as a final step, before starting all over again, with the enriched ontology as input ontology.

2.4. Ontology-Based Information Extraction

General

A subfield of Information Extraction itself called Ontology-Based Information Extraction (OBIE) uses ontologies for the information extraction process and/or constructs an ontology. With the first approach, formal and explicit concepts from existing ontologies are used to guide the information extraction. The latter approach can be performed by building an ontology from the ground up, or enriching an existing ontology. OBIE

systems can be applied to either unstructured or semi-structured natural language texts. An example of an unstructured text is a text file, an example of a semi-structured text is a web-page with particular templates. To be able to use an OBIE system, text corpa are needed. Unfortunately, due to the youth of the field, there are no standardized text corpa available. But even if there would be, when working in a certain domain or niche, people often need to define the text corpus themselves because standards are not available for that particular domain or niche.

Wimalasuriya and Dou (2010) define a general high-level OBIE architecture to which all OBIE systems should comply to. A graphical representation of this architecture can be found in figure 2.1. The text input of OBIE systems is usually first preprocessed before it goes through the IE module, were the actual extraction of information is performed. The ontology itself could be generated internally or by a separate ontology generator, which in turn uses a semantic lexicon. Domain experts could help the system build the ontology by making decisions for the system or changing the ontology afterwards. In a somewhat similar fashion, the domain expert could help the system with the information extraction. The output of an OBIE system is the information that is extracted from the text input, which can be stored in some sort of knowledge base or database. Sometimes, it is even part of some larger query answering system, which a user interacts with.

Despite the youth of the OBIE field, it is full of potential (Kietz et al., 2000; Cimiano et al., 2004; Maynard et al., 2006). First of all, by automatically processing information that is represented in natural language texts, a vast amount of the information on the Internet can be accessed, which would not be possible manually. Second, it creates possibilities for automatic metadata generation, which contributes enormously to the concept of Semantic Web. Third and last, the quality of ontologies can be improved when using OBIE for the evaluation of the quality of ontologies (Wimalasuriya and Dou, 2010).

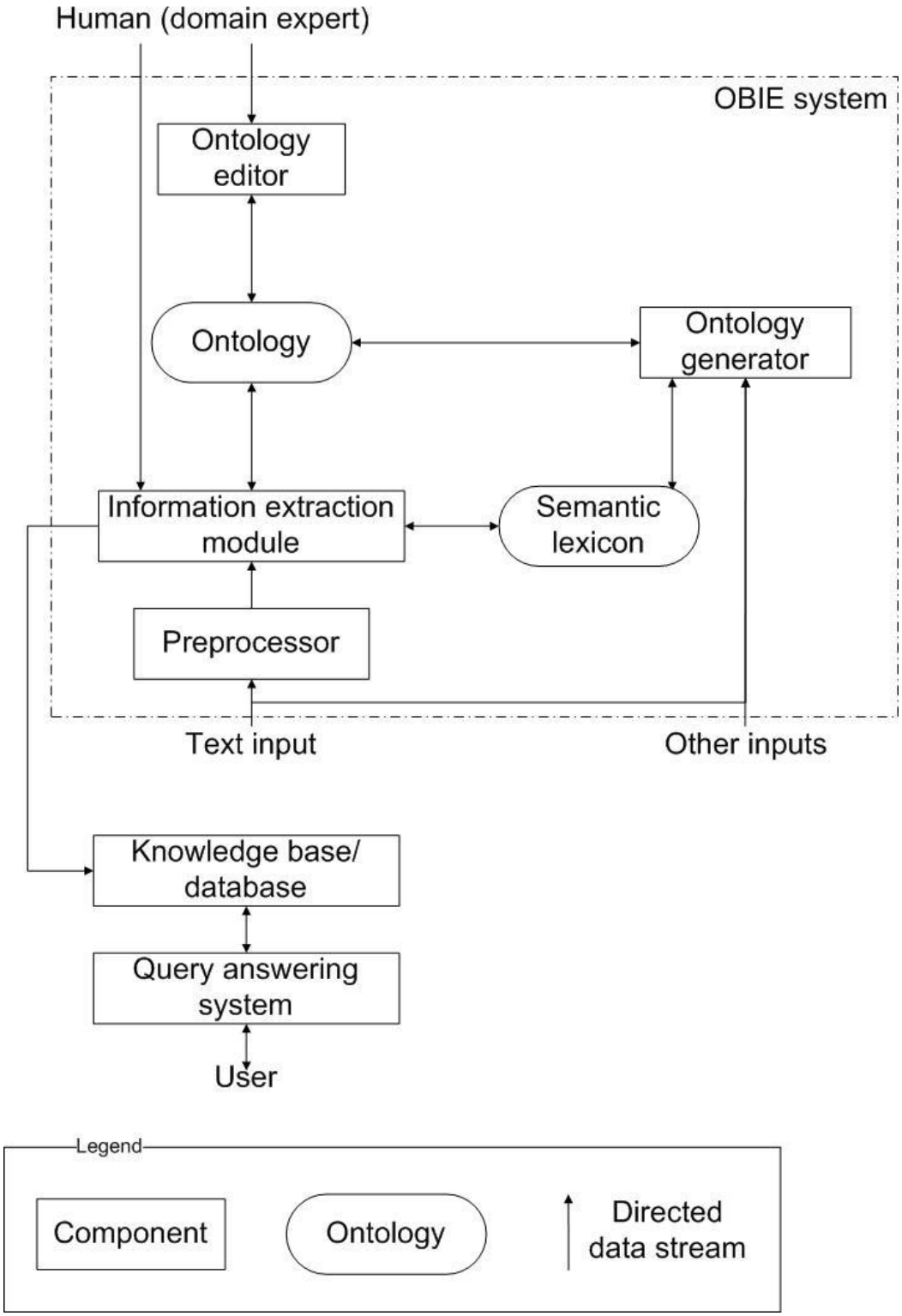


FIGURE 2.1. THE GENERAL ARCHITECTURE OF AN OBIE SYSTEM FROM WIMALASURIYA AND DOU (2010)

IE in the OBIE field

In section 2.1 IE was briefly introduced. A vast amount of the IE techniques that have been developed over the years have been adopted by OBIE systems (Wimalasuriya and Dou, 2010). In this subsection we will take a closer look at IE techniques that are of special interest for the purpose of the current research.

The earlier mentioned gazetteer lists rely on finite-state automata recognizing words or phrases. The words or phrases that the system needs to recognize are somehow available to the system in the form of a list. This list is called a gazetteer list. Such lists are especially useful in identifying members of a certain category, like countries in Europe or former presidents of the United States. For our research, one can imagine that domains have certain categories of their own, possibly making gazetteer lists a useful method for information extraction.

Analyzing the structure of input documents, like HTML or XML, can be used for two purposes. Namely, extracting additional information and pinpointing the location of certain information in a text source. In the OBIE field this is often used to fill knowledge bases with information from the web.

Querying web-based applications is an upcoming method for extracting information from the web (Wimalasuriya and Dou, 2010). With querying web-based applications, the web can function as a big corpus of information. As said, querying such applications is easier if the search-engines support the use via either REST or SOAP web-services. In the OBIE field this is for instance used for collecting additional training samples. One can imagine that it is useful to query certain online databases such as a dictionary.

Most OBIE systems use more than one IE technique and even combine techniques to get the most suited information possible.

2.5. Performance Measurement

Precision and recall

In the IE field, performance measurement is mostly done using the metrics precision and recall. Precision is a measure of exactness, it is the percentage (reflected from 0 to 1) of relevant items among the items that are retrieved. Recall is a measure of completeness, it is the percentage (reflected from 0 to 1) of retrieved relevant items compared to the relevant items overall. Usually, IE systems have to make a trade-off between precision and recall. Precision can be enhanced by only selecting those items that are surely correct, but this obviously reduces recall. Visa versa, enhancing recall can be achieved by extracting as much as possible, hereby reducing precision. Therefore, the so-called F-Measure is used, which is a weighted average of precision

and recall (denoted by β). We will use the following formulas for precision (P), recall (R) and the F-Measure (van Rijsbergen, 1979; Frakes and Baeza-Yates, 1992; Manning and Schütze, 1999; Han and Kamber, 2006):

$$P = \frac{|\{Relevant\}| \cap |\{Retrieved\}|}{|\{Retrieved\}|} \quad (1)$$

$$R = \frac{|\{Relevant\}| \cap |\{Retrieved\}|}{|\{Relevant\}|} \quad (2)$$

$$\text{F-Measure} = \frac{(\beta^2 + 1) * Precision * Recall}{(\beta^2 * Precision) + Recall} \quad (3)$$

When β is set to 1 in the F-Measure (formula 3), precision and recall are regarded to be of equal importance. To weigh precision higher than recall, β needs to be lower than 1. To weigh recall higher than precision, β needs to be higher than 1 (van Rijsbergen, 1979).

Complexity

In order to evaluate a certain solution it is also important to be able to measure the time efficiency. This can be established by calculating the complexity of a certain architecture or algorithm. A mathematical notation called the Big-O notation can be used to characterize efficiency. These characterizations are based on the growth rate of a function. Functions with the same order growth rate will therefore be represented using the same Big-O notation (Fenton and Pfleeger, 1997).

The characterization of a function goes as follows. All constants are ignored and only the dominating term is used to determine the characterization of the growth rate. The dominating term is in this case the fastest growing term.

When applying this to software, we determine the Big-O of an operation and add for sequential operations and multiply for nested operations. For instance, we calculate O(1) for statements and O(n) for every loop. With one nested loop we would get $O(n^2)$.

3. Case study Epiqo

This chapter describes our case-study at Epiqo. First we describe the environment in section 3.1, then we give the requirements of Epiqo regarding their architecture in section 3.2, after which we deal with the expectations of Epiqo of this solution in section 3.3. In section 3.4 we present the architecture for Epiqo. Finally, in section 3.5, we end with conclusions.

3.1. Environment

The system should work in the context of the current e-Recruiting solutions of Epiqo. Their current e-Recruiting solutions are a bundle of flexible and easy-to-use Drupal modules that together make up the previously briefly mentioned system called e-Recruiter. The e-Recruiter system is a Drupal 7 distribution intended for building e-Recruiting platforms.

This section starts with a short introduction of Drupal, followed by an introduction of the e-Recruiter system of Epiqo.

Drupal

Drupal is an open source content management platform, which can be used to build a variety of websites/web-based applications. It was and is being developed in PHP and is distributed with the GNU General Public License. Drupal runs on any operating system that supports PHP, the Apache web server and at least one database management system like MySQL or PostgreSQL. It is an extensible and standard-compliant framework that comes with standard functionality called the Drupal Core. Additional functionality can be used by (installing and) enabling modules, either from Drupal itself or from third-parties. These modules should override functionality in the Drupal Core or add additional functionality, this way nothing in the Drupal Core needs to be altered, which ensures a stable base system.

E-Recruiter System

The e-Recruiter system of Epiqo allows for both recruiters and job seekers to register in their own qualities. After logging on, recruiters can find job seekers who could be interesting as applicants, job seekers can find jobs and/or companies. Features of the e-Recruiter system include job management, registration workflow, taxonomy support, and sophisticated search features. Job management allows recruiters to manage the job advertisements by filling out a template, linking to external job advertisements which are embedded in the website, or job advertisements in a file (for instance a pdf-file). The taxonomy support module ensures easy point and click functionality when

filling out fields like occupational fields, location, or skills. The search features allow recruiters to find the best fit candidate for a job, and job seekers to find the best fitting job to apply for.

The ontology definition language that is used by Epiqo in Drupal is the Simple Knowledge Organization System or SKOS (World Wide Web Consortium, 2012). SKOS can be used to represent knowledge organization systems using the Resource Description Framework (RDF), which is a World Wide Web Consortium (W3C) standard (World Wide Web Consortium, 2012). This standard was initially designed as a meta-model of data, but is currently used as a data format.

The current ontology of Epiqo is used (1) when a user creates his/her résumé, a selection of terms from the ontology can be made from a list or tag cloud, and (2) for information extraction from job advertisements (job location, the necessary skills, languages, field of study). For the information extraction, a look-up is done in the ontology. An annotation is added to both the jobs and résumés using the ontology.

3.2. Requirements

Below the functional, quality, platform, and process requirements for the architecture of the ontology enrichment system are explicated. These requirements were identified based on the wishes of different stakeholders of Epiqo.

Functional

The functional requirements are listed below.

F.1 The system shall semi-automatically enrich a given ontology.
F.2 The system shall enrich a given ontology based on information of the specific domain that can be obtained online.
F.3 The system shall use the amount of appearances of a term in the given sample of job advertisements for selection of candidate terms.
F.4 The system shall only regard nouns and component nouns as candidate terms, names of for instance companies or persons are to be disregarded.
F.5 The system shall check whether a candidate term is a synonym of an existing term in the ontology.
F.6 The system shall check whether a term is a category or is in a category.
F.7 The system shall contain settings, which should at least allow for setting a threshold of the number of appearances of found terms to become a candidate term.
F.8 The system could optionally have a step by step advice component, which allows for an user to accept or reject ontology alteration suggestions.

Requirement F.1, semi-automatically enriching the ontology, is the goal of the system based on the wishes of the management of Epiqo to save time and thus money. The second requirement, F.2, performing the enrichment process based on information from the internet, was formulated to ensure low cost and ease-of-use. For Epiqo, obtaining information from the internet is relatively easy, can be performed (semi)automatically and has low cost because of their current system which has advanced crawler capabilities. The domain experts of Epiqo noticed that the amount of appearances of candidate terms (requirement F.3) and the fact whether the terms are nouns (requirement F.4), indicate importance in most of the cases. Further, for the e-Recruiter system to work properly, it is important to know which terms are synonyms of each other. This is captured in requirement F.5. Requirement F.6 reflects the wishes of the domain experts to be able to update and improve the structure of the ontology. The management of Epiqo and the domain experts, want to be able to set a threshold for the number of appearances of found terms to become a candidate term, this requirements is formalized in F.7. Requirement F.8 is an optional requirement for a step by step advice component, which will allow users (domain experts) to accept or reject alterations that the system suggests.

Quality

The quality requirements are listed below.

Q.1 The enhancement capabilities of the system are important: good documentation shall be provided.
Q.2 The system shall be built in a modular fashion: it shall adhere to the set Drupal standards and conventions of modules.

Since the system is likely to be enhanced in the future, the enhancement capabilities are important. This is reflected in requirement Q.1. To make sure that the system can easily be distributed and used on various Drupal e-Recruiter installations, it is important to design the system with the Drupal standards and conventions in mind. This is captured in requirement Q.2.

Platform

The platform requirements are listed below.

Pl.1 The system shall be designed for Drupal.
Pl.2 The used database management system shall be MySQL.

Since the e-Recruiter system, and all other software systems of Epiqo are developed in Drupal, both the management and the development departments of Epiqo want the system to be implemented in Drupal. Further, for compatibility reasons, the

development department suggests to use a MySQL as database management system. These requirements are captured in Pl.1 and Pl.2 respectively.

Process

The process requirement is listed below.

Pr.1 The system shall be designed in close cooperation with Epiqo.

The management of Epiqo wants the design of this system to be performed in close cooperation with their company Epiqo. This is reflected in requirement Pr.1.

3.3. Expectations of Epiqo

The domain experts of Epiqo would find the system useful when the ontologies that are semi-automatically built have a recall of at least 70 percent when compared to a standard ontology. The precision might be relevant the other way around. If the precision is very low, the quality of the standard ontology might be lower than of the semi-automatically built ontology. This would indicate that the system can provide better quality or at least quality improvements.

3.4. Architecture for Epiqo

The architecture of the system for Epiqo is based on the general OBIE architecture (see figure 2.1) from chapter 2 and the requirements from section 3.2. The architecture for Epiqo is graphically represented in figure 3.1 and explained below.

General System Architecture

To be able to enrich ontologies in basically any natural language within a certain domain, we propose a specific OBIE architecture. Since we are looking for a way to enrich an ontology based on natural text information (from the internet), the OBIE field is a perfect fit. By automatically processing information that is represented in natural language texts, a vast amount of the information can be accessed, which would not be possible manually. Besides this, the OBIE field is specifically applicable for use in particular domains, because the ontologies that are used for the information extraction can be domain specific. However, we believe that this can be taken a step further. Recall that the general OBIE architecture makes use of a semantic lexicon. For the English language, there is one available called WordNet (Princeton University, 2012). For a few other European languages there is a semantic lexicon called EuroWordNet (University of Amsterdam, 2012). Unfortunately, not all languages have freely accessible semantic lexicons, are qualitatively useful, or have semantics lexicons at all.

This poses a problem, since we want our solution to be applicable for any natural language. Besides this, there is another problem with using general semantic lexicons. Semantic lexicons will not know (all) jargon of a domain and the actual semantics can also differ from domain to domain.

As said, we adapt the general OBIE architecture as presented by Wimalasuriya and Dou (2010) to be able to create OBIE systems for any language that also supports the jargon of the domain in question. To achieve this, we suggest to not use a standard provided semantic lexicon, but build a specific one with the system based on textual information from the Internet and the ontology in question. We replace the semantic lexicon with what we call the DomainWordNet. This ontology is built automatically by the system for a specific domain and functions as a semantic lexicon. Further, a DomainWordNet builder is added, which builds the DomainWordNet. Every word will have an entry in the DomainWordNet. Of these so-called terms, their frequency, possible synonyms, basic group (like for instance verb or noun), category, predecessor, and successor will be available.

Naturally, to be able to extract information, one or more sources to extract this information from need to be available. In the case of Epiqo we want to enrich an ontology in the e-Recruiting domain, making it obvious to select sources from that domain. As said, websites with C.V.'s and job advertisements contain the candidate terms to be added to the ontology (like certain skills or professions). Job advertisements describe jobs in certain domains. The terms that can be found in these job advertisements make up the jargon of this domain relevant for e-recruiting purposes. C.V.'s can be much wider than one specific domain, because people tend to have experience in multiple fields and list special skills etc. Therefore a corpus of only job advertisements will be used as an information source.

In order to collect this corpus of job advertisements, the Internet needs to be searched and/or crawled in some way. Epiqo has a crawler, which provides crawled job advertisements in HTML. To be able to use the job advertisements for IE purposes, we want to preprocess the HTML job advertisements to a more convenient structuring and format, analogous to the OBIE preprocessor from the general solution. This is necessary because the HTML that is provided by the crawler is not standardized and might contain other code like JavaScript.

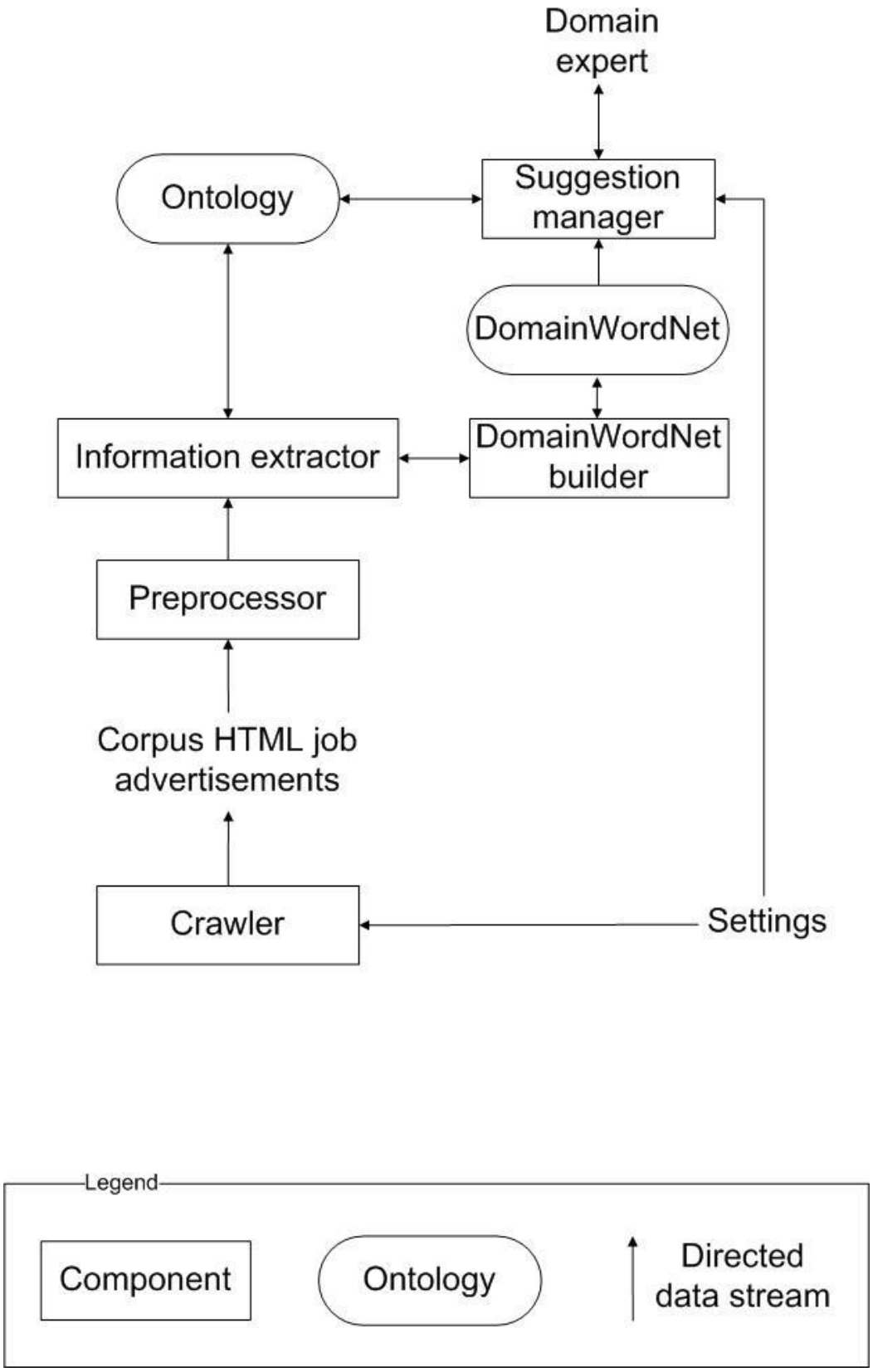


FIGURE 3.1. SUGGESTED ARCHITECTURE FOR EPIQO

As shown in figure 3.1, the architecture for the Epiqo consists of different entities: the DomainWordNet, preprocessor, IE component, DomainWordNet builder and suggestion manager. The crawler and the ontology are entities from the e-Recruiter

system of Epiqo. The rest resides in a custom Drupal module. The DomainWordNet will be created in the database, thus the following components need to be developed within the module: the preprocessor, IE component, DomainWordNet builder and suggestion manager. These components will be described more closely in following sections.

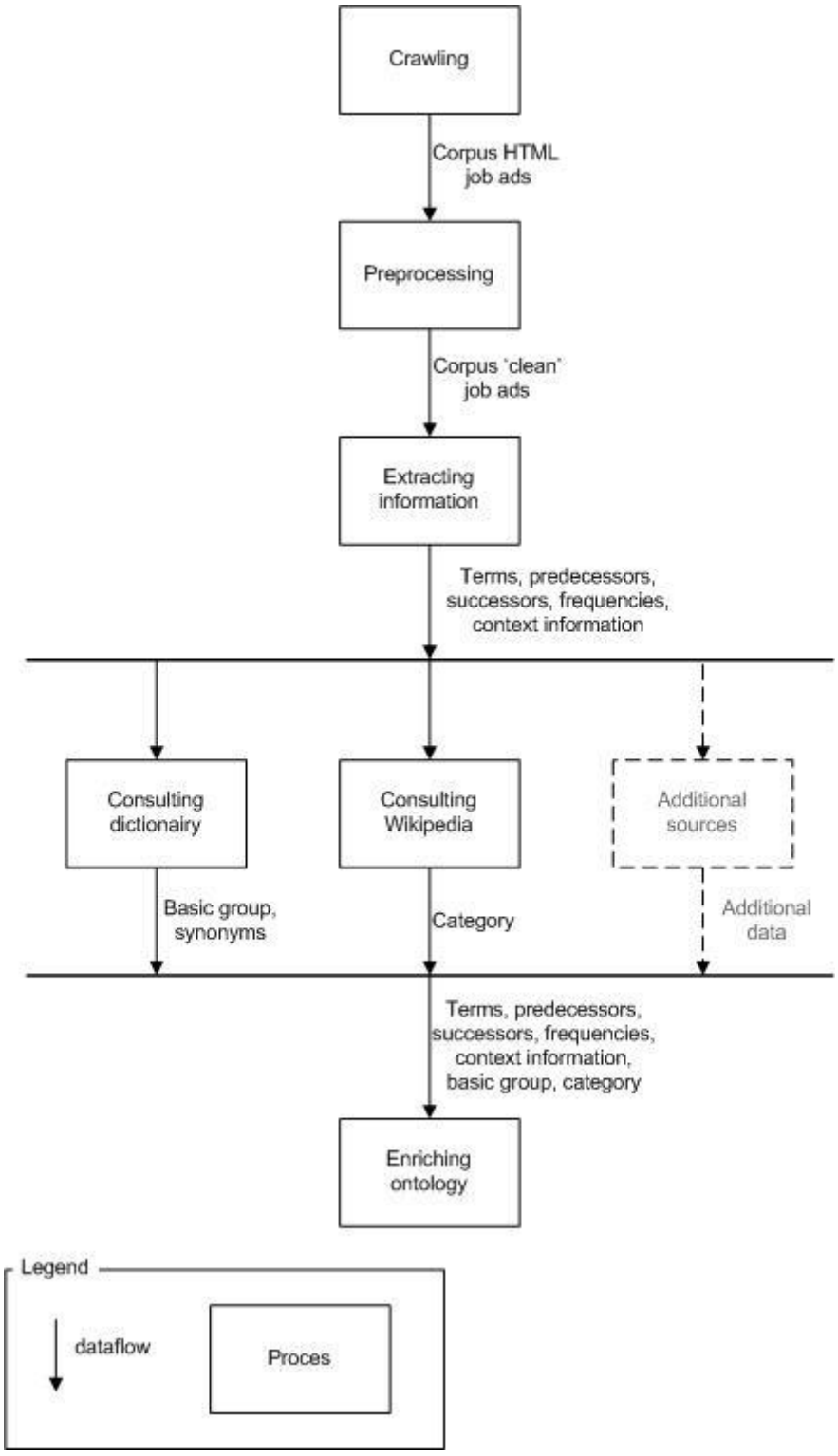


FIGURE 3.2. THE DATAFLOW OF THE SYSTEM

In figure 3.2, the dataflow of the system is represented. The crawler provides a corpus of job advertisements in a certain domain. These job advertisements are preprocessed and ordered into natural text. From this corpus term information is extracted. All this information is stored in the DomainWordNet. The DomainWordNet in turn can be used to enrich the corresponding ontology. As said, the components and their inner workings will be described more closely in following sections.

DomainWordNet

A DomainWordNet is an ontology within the e-Recruiter system. It contains the necessary information as defined in the general architecture: the frequency of a term, possible synonyms of a term, the basic group of a term, the category of a term, and the predecessors and successors of a term.

DomainWordNet Builder Component

This component provides the functionality to update the DomainWordNet, it provides an API to be used by other components. The available functions in the DomainWordNet API are:

- Adding a term
- Removing a term
- Blacklisting a term
- Updating a term

Adding and removing terms are self-explanatory, these functions add and remove terms respectively. The blacklisting function puts the term on a blacklist (which is a gazetteer list), which hides the term from view of the DomainWordNet, but keeps it stored to make sure that it will not be suggested in the future. The last function, updating a term, gives access to change the fields of a term.

Preprocessor Component

This component takes a HTML job advertisement as input and strips every job advertisement from its HTML, Javascript, etc. tags and stores it. When removing the HTML tags, some information could get lost. For instance, a title or header should not be seen as a predecessor of the first word of the following section. To retain as much information as possible, the different sections are stored as new lines. Further, information could even be extracted from the semantics of HTML. Headers and titles not only become a new line, but one could argue that for instance a title could be of more importance than the average word in a certain piece of text. Lists in this particular use case might be skills a jobseeker should have or explicate what the organization is looking for in an employee. Therefore, these special instances are

marked by the preprocessor component, in order for the information extractor component to interpret. The instances to mark:

- Headers: <H1>, <H2>, <H3>, <H4>, <H5>, <H6>.
- Lists: <UL>, <IL>, <LI>, <OL>.
- Title: <TITLE>.

In the e-Recruiter system of Epiqo, there are three custom content types for storing job advertisements. The “Job per file upload”, the “Job per link”, and the “Job per template”. “Job per file upload” is used when a new job is created by uploading a file. “Job per link” is used for referencing to an existing job. “Job per template” is used when a job is created and all the details all directly available. Since the crawler puts all the crawled job in the “Job per link” content type, this is the content type that will be used by the preprocessor component.

The “Job per link” content type has the following fields:

- Title
- Workflow state
- Link
- Organization
- Region
- Location
- Occupational fields
- Fields of study
- Required languages
- Required IT skills
- Required general skills
- Years of experience
- Employment type
- Status
- Crawler

The “crawler” field of the “Job per link” content type is a so-called field-collection. A field-collection is one field to which any number of fields can be attached. The “crawler” field has three fields attached:

- XHTML job
- Full HTML page
- Crawler profile

“XHTML job” contains the job advertisement itself in HTML from the company or job board website. “Full HTML page” is the entire page of the company or job board website on which the job advertisement appears. The XHTML job is an subset of the full HTML page. The “Crawler profile” is a reference to the “Crawler” custom content type, which contains the settings for the to-be-crawled websites. How the crawler works is outside the scope of this research, it simply provides the job advertisements in

the “Job per link” content type. We attached a fourth field to the “crawler” field-collection:

- Clean job

Since the preprocessor component alters the data of the “XHTML job” field, the new field “Clean job” is added to this custom content type to store this data and keep the original data in the “XHTML job” field unchanged.

Information Extraction Component

Taking the “Clean job” text of every job advertisement as input, this component fills the DomainWordNet using the DomainWordNet builder.

The following information is needed in the DomainWordNet:

- Word
- Frequency
- Predecessors
- Successors
- Category of the term
- Word group of the term
- Context information

As explained in the DomainWordNet section, this is stored in an ontology. On a higher level we assume the information to be readily available as values and do not worry about the way it is stored and/or retrieved.

The IE component goes through the “clean jobs” word by word. When the current word is not in the DomainWordNet it is created. The category is looked up in Wikipedia (WikiMedia foundation, 2012) and the word group is looked up in the Google dictionary (Google, 2011). When the current word is in the DomainWordNet, the entry is updated.

As can be seen in the dataflow in figure 3.2, this architecture is suitable for possible additional sources like Wikipedia or the dictionary. Due to the modular nature of the architecture, these components can be added in a straightforward manner.

Suggestion Manager Component

This component is intended for the domain expert to accept or reject the suggestions that this component finds. The accepting and rejecting of suggestions can be done by the user (domain expert) through an graphical user interface.

The suggestion algorithm has two main foci, (1) finding new candidate terms to add to the ontology and (2) suggesting changes in the structure of the ontology. Notice that

the latter could actually also entail selecting new candidate terms, in that case a possible position is identified together with a potential structural change.

The selection of new candidate terms is performed based on three main characteristics: its frequency in the corpus of job advertisements, its word type and its context in the job advertisement. As mentioned in the requirements in section 3.2, the domain experts of Epiqo noticed that the amount of appearances of candidate terms and the fact whether the terms are nouns, indicate importance in most of the cases. In this light the frequency and word type are used to determine importance. If a term has a predecessor or a successor which is in the ontology, the term is marked as a candidate term. Lastly, the marked HTML semantics are used to indicate possible important (related) terms. The algorithm is given below in pseudo code:

```
term:       current term
type:       word type of term
pred:       predecessor of term
succ:       successor of term
freq:       frequency of occurrence of a term
ont:        the ontology that is being enriched
thres:      threshold of tern frequencies
title:      an HTML title tag

FOREACH term
DO    IF    (term.type == 'noun' OR term.type == 'unknown')

      THEN  IF    term.freq >= thres OR term ∈ title OR
            (term.pred ∈ ont OR term.succ ∈ ont)

            THEN  Select term as candidate term.
```

Finding possible structural changes is performed in two different ways based on the category of the term in question. (1) If the category of a certain term A exists in the ontology as a certain term B, suggest term A as a child of B. (2) If a certain term C is the category of a certain term D in the ontology, suggest term C as a parent of term D. Further, HTML semantics are also used to determine structure. The algorithm is given below in pseudo code:

```
list:       an HTML list tag
elem:       elements of list
type:       word type of term
term:       term in first header before list
cand:       array with candidate terms
add:        function to add a term to an array
ont:        the ontology that is being enriched


FOREACH list

DO    FOREACH elem

      DO    IF (elem.type == 'noun' OR elem.type == 'unknown')

            THEN  IF    term ∈ ont

                  THEN  IF elem ∈ ont AND !term.isParentOf(elem)

                        THEN  Suggest term as parent of elem.

                        ELSE  Suggest elem as candidate term as a
                              child of term.

                  ELSEIF      elem ∈ ont

                  THEN        Suggest term as candidate term as a
                              parent of elem.

                  ELSE        Suggest term and elem as candidate
                              terms with term as a parent of
                              elem.
```

Complexity

To be able to determine the complexity of the algorithms we use the Big-O notation as described in section 2.5 of this report.

The first algorithm, for finding candidate terms, starts with a FOREACH statement, which is a loop over all term entities. This has the complexity of O(n). Nested in this FOREACH statement is an IF statement. This IF statement contains an OR statement with two CONTAINS statements. These statements all have the complexity of O(1). Nested in the IF statement one more IF statements with complexity O(1). The body of the IF statement contains basically some simple RETURN statements, which also have complexity O(1). This makes the complexity of the first algorithm O(n), which means that it can be performed in linear time.

The second algorithm, for finding structural suggestions, starts with a FOREACH statement that loops over all found lists. This has the complexity of O(n). Nested in this FOREACH, is another FOREACH statement that loops over all elements of a list. This too has the complexity of O(n). The body of this FOREACH statement contains an IF statement with an OR statement with two CONTAINS statements. These statements all have the complexity of O(1). This IF statement contains two IF statements, both with complexity O(1). Only the first IF statement has another nested IF statement, which also has the complexity of O(1). Then rest of the bodies of the statements are simple RETURN statements with complexity O(1). This makes the complexity if this second algorithm $O(n^2)$, due to the loop nested in a loop. The algorithm can be performed in quadratic time.

3.5. Conclusions

The architecture for Epiqo should be able to adhere to the requirements defined in section 3.2 and function as a solution for the problem of Epiqo that their expansion is hindered by the time needed to create/enrich ontologies. The OBIE field presents an architecture that can be used as a starting point for the architecture for Epiqo. The field is applicable because of the automation possibilities for accessing large amounts of natural language texts and its domain specific nature. The general OBIE architecture presented by Wimalasuriya and Dou (2010) uses a semantic lexicon. Unfortunately, not all natural languages have freely accessible semantic lexicons, have semantic lexicons that are qualitatively useful, or have semantic lexicons at all. Besides this, semantic lexicons will not know (all) jargon of a domain and the actual semantics can also differ between domains.

To overcome these problems with existing semantic lexicons, we suggest to build one for every (sub-)domain. We call this a DomainWordNet, which is also an ontology itself and replaces the standard semantic lexicon. To be able to build such a DomainWordNet we also add a DomainWordNet builder to the architecture.

4. Proof-of-concept Tool for Epiqo

This chapter describes the proof-of-concept tool we developed based on the architecture for Epiqo, to be able to perform some exploratory experiments. First we introduce the tool and explain its setup in section 4.1, then in section 4.2 we explain how we designed the DomainWordNet. In section 4.3 the User Interface is depicted. Finally, in section 4.4, we draw some conclusions.

4.1. The tool

Based on the architecture for Epiqo from section 3.4, we developed a proof-of-concept tool to be able to perform real-life experiments with domain experts. After extensive prototyping on the e-Recruiter system of Epiqo, we discovered that, due to the nature and amount of work it would require, the e-Recruiter system is not suitable to be used for this proof-of-concept. The required work falls outside of the scope of this research, both in time and type of work. It would require adding significant functionality to several Drupal modules of both the Drupal community and Epiqo. The solution that was initially thought to be feasible, did in practice not turn out to deliver enough performance for the information extracting tasks. In concertation with Epiqo it was decided to develop a separate tool. To be able to realize a simple, fast and easy to be built proof-of-concept, we developed the tool in PHP with a MySQL database.

For the proof-of-concept, both the crawler and the ontology from the e-Recruiter system are used. The rest of the components from the architecture for Epiqo are incorporated in the proof-of-concept tool. The tool has its own DomainWordNet and the preprocessing occurs similar to the way it was proposed for the e-Recruiter system. The information extraction in the Information extractor component however, is slightly modified. Since there is no coupling between the tool and the e-Recruiter system, it is not possible to use the ontology for the information extraction tasks. Unfortunately, the automatic insertion of terms into the ontology is also not possible due to the absence of this coupling. In order to still be able to enrich the ontology, the suggestions are given by our own proof-of-concept tool, while the ontologies are being built in the e-Recruiter system. In other words, the domain expert needs to use both systems at the same time and manually mutate the ontology in the e-Recruiter system based on the suggestions of the tool. For the learning process, a certain corpus of HTML job advertisements need to be loaded onto the server and a function needs to be called to start the learning process. The DomainWordNet can be queried and mutated by a set of functions from the DomainWordNet API. The Suggestion manager component has the same functionality as in the original architecture, using the DomainWordNet in a similar fashion.

4.2. DomainWordNet

To make sure that all data that needs to be present in the DomainWordNet is readily available, all paths are saved. Figure 4.1 illustrates the path that is captured for the sentence “Drupal website development.”. Four paths and three terms are stored.

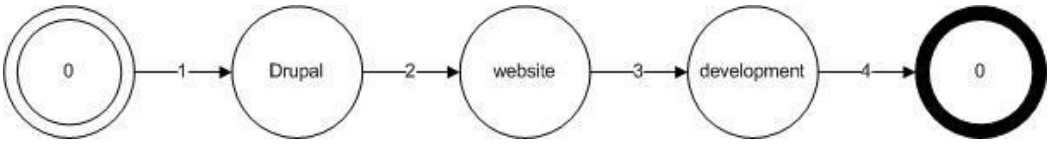


FIGURE 4.1. THE PATH

Storing all the paths ensures that the system is also able to find the predecessor and/or successor of a term, even if the term is a phrase. The predecessors and successors can be queried until the beginning cq. end of a sentence is reached. Due to these paths, the context is always available on demand.

4.3. User Interface

The proof-of-concept tool has a simple web-based User Interface (UI) that enables the user to navigate through the suggestions. The navigation through the suggestions goes as follows. Initially a list is shown with suggested terms, this is shown in figure 4.2. These terms are ordered based on the settings. The terms can be clicked, when a term is clicked all its predecessors and successors are shown. These terms can also be clicked. This way a phrase can be built recursively. For example, we click the word “Drupal” in the initial suggestion list. This gives us two lists, one with all the predecessors of “Drupal” and one with all the successors of “Drupal”, see figure 4.3. We then click on the successors term “developer”, see figure 4.4. This gives us the phrase “Drupal developer”, which is shown on top of the screen and all the predecessor terms and all the successor terms of this phrase. When we now click on either a predecessor or a successor of “Drupal developer”, we extent the phrase on the beginning or the end of the phrase and get all the predecessors and successors of the new, longer phrase. This is possible until no predecessors and successors of a certain phrase can be found.


Task 3

Terms:

Name	Freq
drupal	3288
experience	1859
Web	1458
developer	1229
development	1194

FIGURE 4.2. TERM LIST

Task 3

Term: "drupal "

Predecessors:

Name	Freq
the	364
developer	297
a	209
with	196
are	163

Successors:

Name	Freq
developer	748
php	505
Web	444
developers	429
experience	404

FIGURE 4.3. TERM VIEW DRUPAL

Task 3

Term: "drupal developer "

Predecessors:

Name	Freq
a	94
senior	73
php	67
experienced	57
for	34
contract	28
Sr	24
freelance	19

FIGURE 4.4. TERM VIEW DRUPAL DEVELOPER

4.4. Summary

It was discovered that the e-Recruiter system is not suitable to be used for this proof-of-concept. The solution that was initially thought to be feasible, did in practice not turn out to deliver enough performance for the information extracting tasks. In concertation with Epiqo it was decided to develop a separate proof-of-concept tool in PHP with a MySQL database management system. This proof-of-concept tool adheres to the architecture for Epiqo from section 3.4 of this report, except for the separation between the to-be-enriched ontology and the crawler on one hand (e-Recruiter system) and the rest of the system on the other (proof-of-concept tool).

5. Experiments with Tool

This chapter describes the exploratory experiments that we performed using our proof-of-concept with domain experts from Epiqo. In section 5.1 we introduce the experiments and provide our hypothesis. Next, in section 5.2, we describe our experimental design. In section 5.3 we delineate the execution of our experiments and provide provisional results. In section 5.4 we discuss the results of our experiments.

5.1. Rationale experiments

To be able to determine whether our solution can save a domain expert time in enriching an ontology, both the time that it takes enrich an ontology without the tool and the time it takes a domain expert with use of the tool need to be measured.

> Hypothesis:
> *Using the proof-of-concept tool, that has learned from x job advertisements, enriching a given ontology will take a domain expert less time than when a domain expert who enriches the same ontology by going through x job advertisements by hand.*

However, to be able to compare these times, the result ontologies need to somehow be comparable. That is, the quality needs to be acceptably similar. We test the comparability using the performance measurement that was discussed in section 2.5. The by Epiqo suggested 70 percent recall between both ontologies will be used as an indicator for quality. As said, the precision actually works the other way around. If the precision is very low, the quality of the standard ontology might be lower than of the semi-automatically built ontology. Since all terms are chosen by domain experts, this would indicate that the system can provide quality improvements which were missed manually.

5.2. Experimental design

To be able to answer the hypothesis, we performed three experiments with each three tasks using three domain experts from Epiqo. Each domain expert performed one experiment. Two of the three tasks entailed using the proof-of-concept tool and the e-Recruiter system. The third task entailed using just the e-Recruiter system. This last task required the domain experts to go through a sample of job advertisements by hand. This is comparable to how Epiqo currently creates ontologies of domains. Since the domain experts built the ontologies in the editor of the e-Recruiter system, the domain experts worked with an interface that they are familiar with. However, a short explanation of the proof-of-concept tool is necessary. This explanation together with the task sheet that was provided to the domain experts can be found in Appendix A.

The domain that will be used in the experiments is the Drupal domain. This domain was chosen because of the good knowledge the domain experts of Epiqo have of this domain. Since we seek to enrich ontologies, and not build them from scratch, we start with the following set of classifications (roots of the Drupal recruiting ontology based on experience from Epiqo) that make up the Drupal recruiting domain. We call these classifications the “main terms” of the ontology.

- Drupal skills
- Fields of study
- General skills
- Industry fields
- IT skills
- Occupational fields

The domain experts had to perform the following tasks:

- Without use of the proof-of-concept tool, the domain experts had to enrich an ontology based on sample of 100 testing job advertisements. The time it takes a domain expert to perform this task is measured. This task will from now on be referred to as manual-100.
- With use of the tool, the domain experts had to enrich an ontology based on the by the tool suggested terms. The proof-of-concept tool has learned from 100 job advertisements. The time it takes a domain expert to perform this task is measured. This task will from now on be referred to as tool-100.
- With use of the proof-of-concept tool, the domain experts had to enrich an ontology based on the by the tool suggested terms. The tool has learned from 1.000 job advertisements. The time it takes a domain expert to perform this task is measured. This task will from now on be referred to as tool-1.000.

The first domain expert, let’s call this expert DE1 had to perform the tasks in the following order:

- Task 1: manual-100
- Task 2: tool-100
- Task 3: tool-1.000

The second domain expert, let’s call this expert DE2, had to perform the same tasks in a different order.

- Task 1: tool-100
- Task 2: manual-100
- Task 3: tool-1.000

The third domain expert, let’s call this expert DE3, had to perform the same tasks in the same order as DE2.

- Task 1: tool-100
- Task 2: manual-100
- Task 3: tool-1.000

Unfortunately only three domain experts where available at Epiqo, which makes the sample size of domain experts rather small. To be able to conclude anything statistically, one would require a significant larger sample of domain experts. Besides this, the sample size of job advertisements in the manual task is also relatively small. The choice for 100 job advertisements was made with the amount of manual labor for the domain experts in mind. It has to be a large enough sample to be able to generalize the outcome, on the other hand, it needed to be workable for the domain experts in a reasonable amount of time. Apart from this, the amount was selected without specific reasons. It could very well have been for instance 90 or 110. The third task of the domain experts is a factor 10 of the initial sample, this provides us with information about the scalability on one hand and accuracy with bigger sample sizes on the other. Since learning is inevitable to occur between the different tasks, the order of the tasks is slightly different for DE1 and DE2. This enabled us to reason about the learning effects. DE3 had to perform the experiment in the same order as DE2, because the results of DE2 suggested learning between the first and the second task. This will be substantiated in the next section.

During the experiments, the domain experts were allowed to manipulate the ontologies however they saw fit based on the information they derive from the corpus of job advertisements/suggested terms. The domain experts were not however, allowed to change the main terms. This to ensure the comparability of the ontologies afterwards. Since these main terms are derived from the by Epiqo defined Drupal ontology, we did not expect this to pose a problem for the domain experts during the execution of the experiments. During each experiment the domain experts were asked to write down all the terms he/she uses from the corpus of job advertisements that did not directly found their way into the ontology.

As said, the performance was measured by measuring the time it took the domain experts to perform a task and using the metrics precision and recall. These metrics are used to check the quality of the result ontologies to ensure comparability. When two ontologies are comparable, we can reason over the time needed to perform the tasks. For this comparability we strived for recall of 70 percent or higher, based on the opinions of the domain experts of Epiqo. Since the precision needs to be interpreted a little differently than normal, as explained in section 2.3, we do not use the F-measure.

We performed three different sets of measurements:

Let: M100 – terms of ontology created with task manual-100
T100 – terms of ontology created with task tool-100
T1000 – terms of ontology created with task tool-1.000
S100 – suggested terms with task tool-100
S1.000 – suggested terms with task tool-1.000

1. For every domain expert we compared M100 with T100 and with T1.000.
2. For every domain expert we compared a golden standard with M100 and several golden standards with each other.
3. For every domain expert we compared M100 with terms of ontologies that can be created when we adjust them based on lessons learned from the experiments.

The next section will further elaborate on these sets of measurements. Before we do so, we would like to mention the limitations of this experimental design.

The experiments were performed using domain experts from just one company: Epiqo. Further, the experiments just dealt with one domain: the Drupal domain.

The sample size of domain experts is far from ideal. Generalizing over just three subjects is not really possible. Unfortunately, this is where the practical side came in. Epiqo did not have more domain experts available for performing the experiments. Apart from the sample size of domain experts, also the sample size of the samples of job advertisements that were used in the different tasks are relatively small and disjoint. Here we had to keep in mind that the tasks should remain do-able in a reasonable time-frame for the domain experts at Epiqo. The samples were deliberately disjoint to prevent too much learning.

With the research question in mind, the time measurements of the tasks the domain experts performed in each experiment were of vital importance. Unfortunately, comparing the times of performing the task of enriching an ontology manually and enriching an ontology with use of the tool is only useful when the quality of both the ontology is comparable. Further, the rigid pre-set structure of the ontologies is necessary to be able to make more clear comparisons between the different ontologies. When besides the terms, the structure of the different ontologies also differ, it is much harder to compare quality.

With these kinds of experiments, when the same domain expert performs three similar tasks in a sequential order, learning is inevitable to occur. Further, it is virtually impossible to determine the knowledge of the domain experts relative to each other. This makes it hard to determine whether certain choices are based on expert opinion or lack thereof.

5.3. Execution

The goal of the current research and thus these experiments is to find out whether semi-automatically enriching a given ontology is more time-efficient than enriching the same ontology manually. But as said, before we can reason about this based on the experiments, we need to know whether the quality of the semi-automatically enriched ontologies are in comparison with the manual enriched ontologies. We will now discuss the results. We start with the direct results of the experiments, followed by the sets of measures we defined in section 5.2, to conclude with a synthesis of the whole.

Tables 5.1, 5.2 and 5.3, show the distribution over the main terms in the ontology and the time per domain expert for the manual task, the task based on the suggestions of the tool that learned from 100 job advertisements and the task based on the suggestions of the tool that learned from 1.000 job advertisements respectively.

The tool used for the tool-100 task provided 400 initial suggestions and 1.946 suggestions for the tool-1.000 task. Each suggestion being a term that is suggested to be relevant for incorporation into the ontology and functions as a starting point for phrase inspection simultaneously.

	DE1	**DE2**	**DE3**
Drupal skills	17	11	8
Fields of study	4	0	8
General skills	10	5	3
Industry fields	1	1	1
IT skills	137	83	65
Occupational fields	31	12	20
Total	*200*	*112*	*105*
Time	2:45	1:00	2:05

TABLE 5.1. AMOUNT OF IDENTIFIED TERMS IN TASK MANUAL-100 AND TIME

	DE1	**DE2**	**DE3**
Drupal skills	1	6	3
Fields of study	3	1	7
General skills	6	3	1
Industry fields	0	0	5
IT skills	16	22	13
Occupational fields	14	20	15
Total	*40*	*52*	*44*
Time	0:45	1:30	0:55

TABLE 5.2. AMOUNT OF IDENTIFIED TERMS IN TASK TOOL-100 AND TIME

	DE1	DE2	DE3
Drupal skills	6	8	3
Fields of study	3	1	8
General skills	11	4	1
Industry fields	1	1	2
IT skills	55	57	33
Occupational fields	20	9	17
Total	*96*	*80*	*64*
Time	1:30	1:20	1:05

TABLE 5.3. AMOUNT OF IDENTIFIED TERMS IN TASK TOOL-1.000 AND TIME

From the results of the first two experiments with DE1 and DE2 we noticed the following (as can be seen in tables 5.1 and 5.2). DE2 performed the manual-100 task 2.75 times faster than DE1. DE2 spend twice the amount of time on task tool-100 than DE1. This made us suspect some sort of learning about creating such an ontology. Therefore we performed a third experiment with DE3, giving this domain expert the same tasks as DE1 and DE2, but in the exact same order as DE2. DE3 performed the tasks in comparable times to DE1. When we look at the amount of selected terms, it seems that DE2 performs average for manual-100 and tool-1.000 and above average for task tool-100. So we assume that no significant learning has taken place between the manual-100 and the tool-100 task, which caused the extreme time difference between the tool-100 task and the manual-100 task of DE2.

Measurement 1

The first set of measurements compares M100 with T100 and M100 with T1.000 per domain expert. These comparisons were intended to check the quality of the ontologies of a domain expert when he/she uses the tool. This comparison is visually represented in figure 5.1. Ideally, the two circles overlap fully. Since the three samples of job advertisements that were used in the experiments are mutually exclusive and the sample sizes are relatively small, it is likely that some terms do not exist in both the sample that is used for the task manual-100 and the task tool-100 or tool-1.000 respectively. Beware that the metric precision needs to be interpreted different than one might be used to. The precision measures how many terms are in the ontology of the task using the tool that are also in M100. But the terms that are not in M100 are not false positives per se. Those terms are also chosen by the domain expert, but were probably just not in the sample of 100 job advertisements of the manual-100 task.

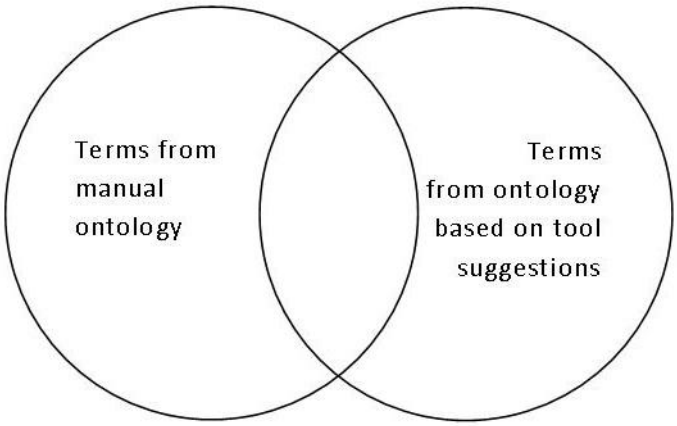


FIGURE 5.1. MEASUREMENT 1

Task manual-100 versus Task tool-100

Here, M100 is regarded as the standard. The metrics precision and recall from section 2.5 where determined per main term in the ontologies. The calculations and specifics can be found in Appendix B. Below the overall precision and recall of M100 compared to T100 are given per domain expert.

DE1
Precision = 24 / 24 + 16 ≈ 0.60
Recall = 24 / 200 ≈ 0.12

DE2
Precision = 25 / 25 + 27 ≈ 0.48
Recall = 25 / 112 ≈ 0.22

DE3
Precision = 34 / 34 + 18 ≈ 0.65
Recall = 34 / 105 ≈ 0.32

The low recall of all three the domain experts, could be ascribed to the two disjunctive and small samples of 100 job advertisements M100 and T100. The low recall of DE1 is understandable given the relatively large difference between the number of identified terms in task manual-100 and those in task tool-100; 200 versus 40. For DE2 and DE3 the same holds in a more moderate form; 112 versus 52 and 105 versus 44 respectively.

We listed the terms that are in M100, but not in T100 and determined the reason why these terms could not be selected by the domain expert. This list can be found in Appendix C. We defined six categories for terms that where selected for the ontology

in task manual-100 and not for the ontology in task tool-100. The categories with belonging frequencies of task tool-100 are given below in table 5.4.

Cause of absence classifications	**DE1**	**DE2**	**DE3**
Term has no type	19	4	8
Unknown term	70	31	26
Not selected word type	8	7	9
Frequency too low	49	22	22
Phrase does not exist	15	16	9
No apparent reason, was available	15	7	5
Total	*176*	*87*	*79*

TABLE 5.4. CAUSE OF ABSENCE CLASSIFICATIONS OF TASK TOOL-100

The first category that we identified is the group of terms that actually were in the DomainWordNet, but did not have a type. This would probably have occurred when the webservice that was used to obtain the word type did not provide a word type available in the enumeration or when the webservice might have timed out. In this case a NULL was found and the term was not selected as a suggestion because of this. Table 5.4 shows that this was the case for nineteen, three and eight terms for DE1, DE2 and DE3 respectively. The second category are the terms that were not in the DomainWordNet based on the learning sample T100. Seventy-three, three and eight terms were absent in the DomainWordNet for DE1, DE2 and DE3 respectively. The third category of terms had a word type that was not selected in the settings to be incorporated into the suggestions, such as word type “verb”. DE1, DE2 and DE3 had eight, seven and nine of such terms in M100 and not in T100 respectively. Category four are the terms that had a too low category to be suggested. As can be seen in table 5.4, DE1 had 47 of such terms, DE2 had 22 of such terms and DE3 had 22 of such terms. The fifth category consists of phrases that were not suggested by the tool, this was the case for fifteen, seven and five terms of DE1, DE2 and DE3 respectively. The sixth and last category is the group of terms that actually was suggested to the domain experts and were in M100, but were not in T100. For some unknown reason the domain experts esteemed these terms to be valuable for the ontology during task manual-100, but disregarded them when they were suggested during task tool-100.

This list of cause of absence classifications could be valuable data for improving the algorithm and settings. The third set of measures tests possible improvements based on these findings.

Task manual-100 versus Task tool-1.000

Here, M100 is regarded as the standard. The calculations and specifics of the precision and recall metrics of M100 compared to T1.000 can be found in Appendix D. Below the overall precision and recall of M100 compared to T1.000 are given for each domain expert.

DE1
Precision = 65 / 65 + 30 ≈ 0.68
Recall = 65 / 200 ≈ 0.33

DE2
Precision = 45 / 45 + 35 ≈ 0.56
Recall = 45 / 112 ≈ 0.40

DE3
Precision = 37 / 37 + 28 ≈ 0.57
Recall = 37 / 105 ≈ 0.35

When we compare the precision and recall of M100 versus T100 with M100 versus T1.000 of DE1 and DE2, both the precision and recall improve for the domain experts individually when the sample size of job advertisements is increased. For DE1 the precision increased with 0.08 and the recall with 0.21, which is a thirteen percent and a 75 percent increase respectively. For DE2 the precision increased with 0.08 and the recall with 0.18, which is a seventeen percent and a 82 percent increase respectively. With DE3, precision decreased with 0.08, which is twelve percent, but did recall increased with 0.10, which is 40 percent. An overview of this comparison is given in table 5.5.

	DE1		**DE2**		**DE3**	
Precision	+ 0.08	+ 13%	+ 0.08	+ 17%	- 0.08	- 12%
Recall	+ 0.21	+ 175%	+ 0.18	+ 82%	+ 0.10	+ 40%

TABLE 5.5. PRECISION AND RECALL OF COMPARISON M100 VS. T100 WITH M100 VS. T1.000

Because the learning sample used with the task tool-1.000 is ten times bigger than the sample used for task tool-100, the suggestions are expected to be more accurate because M100 is disjoint with both T100 and T1.000. This would result in a higher precision. This is the case for DE1 and DE2, with an increase of fifteen and seventeen percent respectively. Yet, the precision of DE3 decreased with twelve percent. The direct results from tables 5.1, 5.2 and 5.3 do not give any clear indication as to what might have caused this decrease of precision for DE3.

Although the samples are disjoint, the larger learning sample of job advertisements used in task tool-1.000 caused more terms from M100 to be present in T1.000 than in T100 for DE1, DE2 and DE3, with a 175 percent, 82 percent and 40 percent increase respectively.

As for the previous comparison between M100 and T100, we listed the terms that are in M100, but not in T1.000 and determined the reason why these terms could not be selected. This list can be found in Appendix D. Here we also used the six identified

classifications and listed them in table 5.6 with the belonging frequencies per domain expert.

Cause of absence classifications	**DE1**	**DE2**	**DE3**
Term has no type	32	5	11
Unknown term	8	13	8
Not selected word type	8	1	13
Frequency too low	48	12	17
Phrase does not exist	2	12	0
No apparent reason, was available	37	24	19
Total	*135*	*67*	*68*

TABLE 5.6. CAUSE OF ABSENCE CLASSIFICATIONS OF TASK TOOL-1.000

The amount of terms that were absent from T1.000 but were in M100 is lower for all domain experts. The increase in sample size does, as expected and shown with our recall metric, improve the ontology. Since the tool suggested more terms during task tool-1.000 than during task tool-100, it is to be expected that the webservice that was used to obtain the word type did not provide a word type available in the enumeration or when the webservice might have timed out a number of times more. Terms that were in M100 but not suggested, the unknown terms category, was reduced considerably with the increase of sample size. From 70 to eight, from 31 to thirteen and from 26 to eight for DE1, DE2 and DE3 respectively. The terms that were not suggested because of their word type or because their frequency was too low stayed about the same. The frequency of some terms got high enough to be suggested and other new terms surfaced which did not get a frequency higher than the threshold. Most of the phrases were available with the bigger sample. The terms that actually were suggested to the domain experts and were in M100, but were not in T1.000, is much higher. Thirty-seven, 24 and 19 as opposed to fifteen, seven and five of DE1, DE2 and DE3 respectively. As with the previous comparison, for some unknown reason the domain experts esteemed these terms to be valuable for the ontology during task manual-100, but disregarded them when they were suggested during task tool-1.000.

We expect the recall to increase even further when the sample is increased to for instance 10.000 job advertisements.

Measurement 2

We derived a “golden standard” GS based on the ontologies of the M100’s of the three domain experts and compared this ontology to the individual M100’s. We also derived a “golden standard” based on the ontologies of tasks tool-100 called GST100 and task tool-1.000 called GST1.000. In the second set of measurements we compare GS with the M100 of each domain expert and with GST100 and GST1.000 successively.

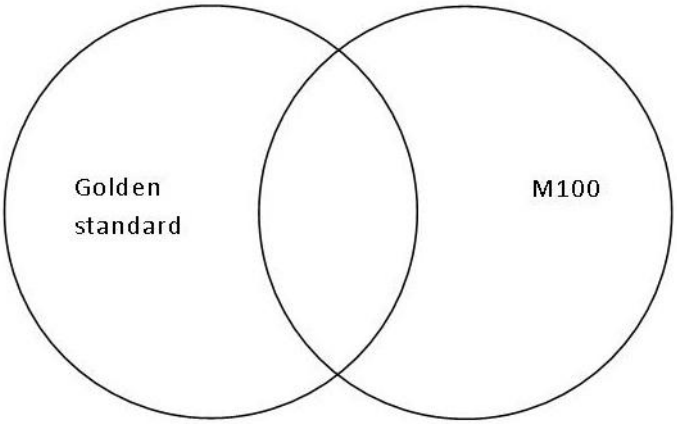


FIGURE 5.2. MEASUREMENT 2

For a term to be selected for a golden standard ontology, terms had to adhere to two conditions.

1. The term had be present in at least two of the three ontologies.
2. The term had to be an exact, though case-insensitive, match.

The golden standard ontology can be found in Appendix E. Table 5.7 shows the distribution over the main terms in the manual-100 ontologies of the domain experts and the Golden Standard. Table 5.8 shows the origin of the terms in the Golden Standard ontology, how many terms came from the manual-100 ontology of DE1 and DE2, DE1 and DE3, DE2 and DE3, and DE1, DE2 and DE3 respectively.

	DE1	**DE2**	**DE3**	**GS**
Drupal skills	17	11	8	**8**
Fields of study	4	0	8	**2**
General skills	10	5	3	**3**
Industry fields	1	1	1	**1**
IT skills	137	83	65	**82**
Occupational fields	31	12	20	**11**
Total	*200*	*112*	*105*	***107***

TABLE 5.7. DISTRIBUTION MAIN TERMS PER DOMAIN EXPERT AND GOLDEN STANDARD

	{DE1,DE2}	{DE1,DE3}	{DE2,DE3}	{DE1,DE2, DE3}	{DE1} {DE2} {DE3}
Drupal skills		1	1	6	14
Fields of study		2			8
General skills		2		1	11
Industry fields	1				1
IT skills	22	21	1	38	83
Occupational fields		6	1	4	37
Total	*23*	*32*	*3*	*49*	*154*

TABLE 5.8. SOURCE OF TERMS FOR GOLDEN STANDARD

The calculations and specifics of the precision and recall metrics of the golden standard GS compared to M100 can be found in Appendix F. Below the overall precision and recall of the golden standard compared to M100 per domain expert are given.

DE1
Precision = 105 / 105 + 95 ≈ 0.53
Recall = 105 / 107 ≈ 0.98

DE2
Precision = 75 / 75 + 37 ≈ 0.67
Recall = 75 / 107 ≈ 0.70

DE3
Precision = 84 / 84 + 21 ≈ 0.80
Recall = 84 / 107 ≈ 0.79

Since DE1 has the most terms in M100, its precision compared to the golden standard is lower and the recall quite high. We expected DE3 to have the lowest recall and highest precision due to the fact that DE3 identified the least amount of terms. DE2 however, scored lower on recall. DE2 has very little overlap with DE3 and less overlap with DE1 than DE3 has (as shown in table 5.8). This in combination with the fact that DE2 has selected more terms makes for a lower recall due to the impact of DE2 on GS.

The golden standard ontology GST100 and the golden standard ontology GST1.000 can be found in Appendix G. Tables 5.9 and 5.11 show the distribution over the main terms in the tool-100 and tool-1.000 ontologies of the domain experts and the Golden Standard. Table 5.10 and 5.12 show the origin of the terms in the golden standard ontologies.

	DE1	DE2	DE3	GST100
Drupal skills	1	6	3	**1**
Fields of study	3	1	7	**3**
General skills	6	3	1	**1**
Industry fields	0	0	5	**0**
IT skills	16	22	13	**12**
Occupational fields	14	20	15	**11**
Total	*40*	*52*	*44*	***28***

TABLE 5.9. DISTRIBUTION MAIN TERMS PER DOMAIN EXPERT AND GOLDEN STANDARD

	{DE1,DE2}	{DE1,DE3}	{DE2,DE3}	{DE1,DE2, DE3}	{DE1} {DE2} {DE3}
Drupal skills		1			8
Fields of study		2		1	4
General skills	1				8
Industry fields					5
IT skills		2		10	19
Occupational fields		6	1	4	23
Total	*1*	*11*	*1*	*15*	*67*

TABLE 5.10. SOURCE OF TERMS FOR GOLDEN STANDARD

	DE1	DE2	DE3	GST1.000
Drupal skills	6	8	3	**2**
Fields of study	3	1	8	**3**
General skills	11	4	1	**1**
Industry fields	1	1	2	**1**
IT skills	55	57	33	**30**
Occupational fields	20	9	17	**12**
Total	*96*	*80*	*64*	***49***

TABLE 5.11. DISTRIBUTION MAIN TERMS PER DOMAIN EXPERT AND GOLDEN STANDARD

	{DE1,DE2}	{DE1,DE3}	{DE2,DE3}	{DE1,DE2, DE3}	{DE1} {DE2} {DE3}
Drupal skills			1	1	12
Fields of study		2		1	5
General skills				1	13
Industry fields	1				2
IT skills	6	7		17	68
Occupational fields	2	7		3	19
Total	*9*	*16*	*1*	*23*	*119*

TABLE 5.12. SOURCE OF TERMS FOR GOLDEN STANDARD

The calculations and specifics of the precision and recall metrics of the GS golden standard compared to GST100 and the metrics of GS compared to GST1.000 can be found in Appendix H. Below the overall precision and recall of the golden standard GS compared to GST100 and GST1.000 are given respectively.

GS compared to GST100
Precision = 18 / 18 + 10 ≈ 0.64
Recall = 18 / 107 ≈ 0.17

GS compared to GST1.000
Precision = 32 / 32 + 17 ≈ 0.65
Recall = 32 / 107 ≈ 0.30

The disagreements of the domain experts in the result ontologies of task T-100 and task T-1.000 are too great to be able to get useful precision and recall measurements. The tables 5.9 and 5.11 show that the amount of terms in the golden standards is much lower than the average of the domain experts. The tables 5.10 and 5.12 show that most terms are only in one of the three ontologies. Therefore the recall can never get higher than 28 / 107 ≈ 0.26 and 49 / 107 ≈ 0.46 respectively.

Measurement 3

The third set of measurements quantify certain ideal circumstances. First we looked at category six of the classifications that were identified in the first measure, that for some unknown reason the domain experts esteemed these terms to be valuable for the ontology during task manual-100, but disregarded them when they were suggested during one of the tasks using the tool. When the domain experts had hypothetically chosen those terms both during task manual-100 and during task tool-100 the following comparison can be made: M100 versus (M100 ∩ S100) ∪ T100. When the domain experts had chosen those terms both during task manual-100 and during task

tool-1.000 the following comparison can be made: M100 versus (M100 ∩ S1.000) ∪ T1.000. After this measure we look at the more ideal circumstances.

M100 versus (M100 ∩ S100) ∪ T100

Here, M100 is regarded as the standard. The calculations and specifics of the precision and recall metrics of M100 compared to (M100 ∩ S100) ∪ T100 can be found in Appendix I. Below the overall precision and recall of M100 compared to (M100 ∩ S100) ∪ T100 are given per domain expert.

DE1
Precision = 39 / 39 + 16 ≈ 0.71
Recall = 39 / 200 ≈ 0.20

DE2
Precision = 32 / 32 + 27 ≈ 0.54
Recall = 32 / 112 ≈ 0.29

DE3
Precision = 39 / 39 + 18 ≈ 0.68
Recall = 39 / 105 ≈ 0.37

For DE1, the precision and recall of M100 compared to T100 from measurement 1, were 0.60 and 0.12 respectively. So if DE1 would have selected the terms from the sixth category, the precision would have been eighteen percent higher and the recall would have been 40 percent higher. For DE2, the precision and recall of M100 compared to T100 from measurement 1, were 0.48 and 0.22 respectively. In this case if DE2 would have selected the terms from the sixth category, the precision would have been thirteen percent higher and the recall would have been 32 percent higher. For DE3, the precision and recall of M100 compared to T100 from measurement 1, were 0.65 and 0.25 respectively. So if DE3 would have selected the terms from the sixth category, the precision would have been twelve percent higher and the recall would have been twenty percent higher.

M100 versus (M100 ∩ S1.000) ∪ T1.000

Here, M100 is regarded as the standard. The calculations and specifics of the precision and recall metrics of M100 compared to (M100 ∩ S1.000) ∪ T1.000 can be found in Appendix J. Below the overall precision and recall of M100 compared to (M100 ∩ S1.000) ∪ T1.000 are given.

DE1
Precision = 102 / 102 + 30 ≈ 0.77
Recall = 102 / 200 ≈ 0.51

DE2
Precision = 69 / 69 + 35 ≈ 0.66
Recall = 69 / 112 ≈ 0.62

DE3
Precision = 56 / 56 + 28 ≈ 0.67
Recall = 56 / 105 ≈ 0.53

For DE1, the precision and recall of M100 compared to T1.000 from measurement 1, are 0.68 and 0.33 respectively. When DE1 would have selected the terms from the sixth category, the precision would have been thirteen percent higher and the recall fifty-eight percent higher. For DE2, the precision and recall of M100 compared to T1.000 from measurement 1, are 0.56 and 0.40 respectively. When DE2 would have selected the terms from the sixth category, the precision would have been eighteen percent higher and the recall fifty-five percent higher. For DE3, the precision and recall of M100 compared to T1.000 from measurement 1, are 0.57 and 0.35 respectively. When DE3 would have selected the terms from the sixth category, the precision would have been eighteen percent higher and the recall fifty-one percent higher.

With the bigger learning sample of task tool-1.000 in combination with the correction of the sixth cause of absence, the recall of three domain experts reaches a level of more than 50 percent.

M100 versus More ideal

Here, M100 is regarded as the standard. The first cause of absence classification, when a term did not have a type, is due to the use of a unreliable webservice. This could be prevented. The second cause of absence classification, unknown term, could be due to the fact that the term was not in the sample. This cannot be prevented without experiments with bigger samples of job advertisements. The third cause of classification, not selected word type, is debatable. It appears that not all of the terms that the domain experts select are nouns, but it does reduce the amount of suggestion considerably and only a few are left out for this reason. The next cause of classification, the fourth one regarding the frequency of a term, is also debatable. Reducing the threshold will increase the amount of suggestions considerably, but the amount of missed terms due to the threshold is relatively big. Creating a dynamic threshold that can be adjusted by the domain expert while using the tool could solve this. For the sake of the creation of the ideal T100 we assumed this would provide us with 50 percent of the previously missed terms. The fifth cause of absence, the phrase does not exist, cannot be prevented. This is like the unknown terms, the phrases are most likely not present in the sample. The sixth and last cause of absence, as explained earlier could be solved.

For this measure we assumed that the system always find the word type of a term, that the system does suggest half of the terms that were not suggested due to a too low frequency and domain experts are consistent. This gives us the following possible quality of the ontologies for M100 versus T100:

DE1
Precision = 82 / 82 + 16 ≈ 0.84
Recall = 82 / 200 ≈ 0.41

DE2
Precision = 47 / 47 + 27 ≈ 0.64
Recall = 47 / 112 ≈ 0.42

DE3
Precision = 58 / 58 + 18 ≈ 0.76
Recall = 58 / 105 ≈ 0.55

This gives us the following possible quality of the ontologies for M100 versus T1.000:

DE1
Precision = 158 / 158 + 30 ≈ 0.84
Recall = 158 / 200 ≈ 0.79

DE2
Precision = 80 / 80 + 35 ≈ 0.70
Recall = 80 / 112 ≈ 0.71

DE3
Precision = 75 / 75 + 28 ≈ 0.73
Recall = 75 / 105 ≈ 0.71

By applying the lessons learned in the first two measurements, we achieved a recall over 70 percent for all domain experts. This showed that the quality needed for comparison can be achieved using our solution.

5.4. Results and discussion

Apart from the fact that generalizing over a sample of three domain experts is statistically not possible, the measured times did not unanimously show a time improvement when using the tool.

Measurement one showed us that two samples of 100 job advertisements are not comparable enough for our purpose, we ascribe this to the fact that they are too small and disjoint. The sample of 1.000 job advertisements however, showed a great improvement. (Keeping in mind that the samples might still be relatively small and off course disjoint.) Based on these experiments we were able to identify six classifications of cause of absence of the terms that were in M100, but not in the belonging ontology that was being enriched using the tool. These are:

1. Term has no type
2. Term does not exist in DomainWordNet
3. Terms with this word type are filtered out
4. Terms with too low frequency are filtered out
5. Phrase does not exist in DomainWordNet
6. No apparent reason, was available in DomainWordNet

Measurement two showed that the domain experts have a fair amount of agreement among each other when performing the manual-100 task. With the tool-100 and tool-1.000 tasks however, the domain experts were mainly in disagreement.

Measurement three was intended to show the effects that possible improvements on our proof-of-concept could have on the quality of the ontologies. These improvements were based on the six classifications of cause of absence that we identified in measurement one. From measurement three we can conclude that with the proper adjustments we can get the quality “good” enough to be able to compare the ontologies that are created manually and those with use of the tool.

Due to the disagreements of the domain experts in the tasks tool-100 and tool-1.000 and the inconsistent times between the domain experts, we were not able to conclude anything regarding the time-efficiency of the proof-of-concept. Unfortunately, this prevents us from being able to accept or reject our hypothesis. We did however, identify ways to improve the quality of the suggestions and thereby the quality of the ontologies that can be enriched using our solution. These are summed up in the first column of table 5.13, the second column shows the numbers of the causes of absence that are expected to be positively affected by the suggested improvements.

Suggested improvements	Affects
A. Eliminate dependencies	1
B. Use bigger samples of job advertisements	2, 5
C. Improve the filter rules	3, 4
D. Use stemming	2, 3, 4, 5
E. Use full algorithm	2, 5

TABLE 5.13. SUGGESTIONS FOR QUALITY IMPROVEMENT ENRICHED ONTOLOGIES

The first suggested improvement A is based on the first cause of absence, terms that did not have a type in the DomainWordNet. This was due to the dependence on third party webservices, which did not always provide the requested data. Eliminating all dependencies on third parties, by for instance running a dictionary within the system, will make sure that all necessary data will be available. The second suggested improvement B is based on the absence of some terms and phrases in the DomainWordNet. The result ontologies from different tasks, based on different samples, were compared. Since the samples that were used for the experiments were relatively small and disjoint, not all terms were in both samples. Comparing the results from the tasks tool-100 and tool-1.000, shows that the amount of terms that are identified for both ontologies increases. Therefore we suggest to use bigger samples of job advertisements. The third suggested improvement C is based on the filter rules that were used in the experiments. Both the selected word types and the frequency surfaced as being too strict. From the information from the domain experts of Epiqo we decided to regard only nouns as word type, the experiments show that the domain experts also select words with other word types. Since there was nothing known about frequencies we set the threshold practically at random on 10. We suggest to make both these filter options, word type and frequency threshold, a dynamic setting which can be altered real-time. This way the domain expert can adjust the word type and frequency at will by looking at the changes it produces in the suggested terms instantly. The fourth suggested improvement D is based on all but the first and last cause of absence. By applying stemming, more terms will be regarded as the same term. This will result in higher frequencies and more positive hits. The fifth and last suggested improvement E is based on the fact that we did not use the full algorithm as described in section 3.4, because in our proof-of-concept tool there is no coupling between the ontology and the intelligence of the system. We expect this to provide additional terms and phrases in the suggestions and thereby lowering the missing terms and phrases.

6. Conclusion

This is the concluding chapter of this report and provides insight in the contribution that the current research provides in section 6.1, its limitations in section 6.2 and possible future research directions in section 6.3.

6.1. Contribution

Our research objectives were (1) developing an approach for semi-automatically enriching domain-specific ontologies, (2) designing a software application that will be able to do this, (3) building a proof-of-concept of this software application for the Epiqo case-study, and (4) performance measurement of time, completeness and correctness of the result ontologies when using this proof-of-concept.

All four research objectives were met. We developed an approach for semi-automatically enriching domain-specific ontologies by adapting the general OBIE architecture of Wimalasuriya and Dou (2010). We added the DomainWordNet, a semantic lexicon that is built automatically based on relevant input. Apart from not needing a semantic lexicon in the desired natural language, which is the case with traditional methods, our approach will also ensure that the semantic lexicon is truly specific for the particular domain. The architecture that we designed for Epiqo makes use of this approach and adheres to the requirements set by Epiqo. A proof-of-concept was developed that was used for experiments with domain experts from Epiqo. The results of these experiments were used for performance measurement.

Our research question was: *is semi-automatically enriching a given ontology for a certain domain more time-efficient than enriching the same ontology manually?* We were not able to obtain conclusive data to be able to answer the research question, we did generate data regarding quality. We showed that our solution has the potential to provide qualitative “good” enough ontologies to be comparable to standard ontologies. We thrust that this provides future researchers the opportunity to enhance and expand our exploratory experiments to be able to answer our research question in the future. To do so, one should take into account the lessons learned from our experiments:

- The tool was too dependent on third party services
- The samples of job advertisements were too small
- Too little sound data was available to create adequate filter rules

6.2. Limitations

Due to the fact that this is a case-study, most of our findings are only based on data obtained from Epiqo. Since Epiqo operates in the e-Recruiting domain, our data is almost fully restricted to this domain.

The experiments are bound by the following limitations:

- The samples came from only one sub-domain: Drupal jobs within the e-recruiting domain.
- The sample size of domain experts is small.
- The samples of job advertisements are small.
- The samples of job advertisements are disjoint.
- There is a rigid pre-set structure to the ontologies.
- Possible learning effects between tasks.
- Unknown knowledge of the domain experts.

6.3. Future work

Updating proof-of-concept

Based on the lessons learned from the exploratory experiments testing our proof-of-concept, we expect that the proof-of-concept can be adjusted in such a way, that we can get the quality of the ontologies enriched based on the suggestions of our solution "good" enough to be comparable to manually enriched ontologies. The necessary adjustments to our proof-of-concept are:

- Running a local dictionary on the machine that runs the tool, this way, if a certain term exists, the word type can be stored.
- Creating a real-time setting for the user (domain expert) to constantly be able change the frequency threshold on run-time. This should minimize the terms that get lost due to a too low frequency of the term in the DomainWordNet.
- Add a coupling with the ontology that is being enriched. This way the full functionality of our solution can be tested.
- Stemming should be added, the proof-of-concept tool does not perform any stemming at all. It is even useful to perform extreme stemming and regard "theming" and "theme development" as the same terms.

Additional experiments

One of the lessons we learned from the experiments of the current research was that domain experts esteemed some terms to be valuable for the ontology during task manual-100, but disregarded them when they were suggested during the tasks using the tool. One of the possible reasons for this could be that the UI of the tool is not clear

enough. To solve this, or at least to be sure that this is not an issue, we suggest future research to perform the following experiment:

- Performing experiments to obtain an optimal UI, it could be that domain experts miss terms due to the interface.

To be able to conclude whether our solution will or will not safe time when enriching ontologies, additional experiments need to be conducted:

- New experiments using the updated version of our proof-of-concept tool, with a much bigger sample of domain experts. Time would again need to be measured. It would be interesting if future research would also measure the saturation of the ontologies in relation to the time. This way it would also be known how much time it takes domain experts to reach certain levels of saturation, making the experiments more comparable among the domain experts.

In the case that it seems fruitful to proceed investigating this solution, we would suggest future research to broaden the scope by performing the following experiments:

- Additional new experiments in different domains.
- Additional new experiments in different settings, like different (types of) organizations.

Thoughts on improving/altering solution

The general solution for Epiqo could be improved by updating the Suggestion manager in such a way that multiple domain experts can provide their opinion on the suggestions. Letting the majority of the domain experts decide whether a suggestion should be accepted, could provide an instant golden standard ontology. With some more functionality, like comments or suggestions from one domain expert to the others, it could potentially have an even greater positive impact on the quality of semi-automatically generated ontologies.

8. References

Agirre, E., Ausa, O., Havy, E., and Martinez, D., “Enriching Very Large Ontologies Using the WWW”, *European Conference on Artificial Intelligence, 1st Ontology Learning Workshop*, Berlin, Germany, 2000.

Aristotle, “Categories,” *The Internet Classic Archive*, 350 B.C.E.

Bratina, T. G. and Bratina, T. A., "Electronic career search," *Journal of Employment Counseling*, vol. 35, no. 1, pp. 17-24, 1998.

Dumais, S.T., Platt, T., Heckerman, D., and Sahami, M., “Inductive learning algorithms and representations for text categorization,” *Proceedings of Conference on Information and Knowledge Management*, pp. 148–155, 1998.

Chang, C., and Lin, C., “LIBSVM : a library for support vector machines,” http://www.csie.ntu.edu.tw/~cjlin/libsvm/ visited at 2011-03-04.

Cimiano, P., Handschuh, S., and Staab, S., “Towards the self-annotating web,” *Proceedings of the 13th International Conference on World Wide Web,* ACM, New York, 2004.

Fenton, N. E. and Pfleeger, S. L., “Software Metrics; A Rigorous and Practical Approach,” International Thompson Publishing, Boston, 1997.

Frakes, W., and Baeza-Yates, R., “Information retrieval, data structures and algorithms ,” Prentice Hall, New York, 1992.

Galanaki, E., "The decision to recruit online: a descriptive study," *Career Development International*, vol. 7, no. 4, pp. 243-251, 2002.

Gentner, C.A. “The computerized job seeker,” *The Personnel Administrator*, vol. 29, no. 8, pp. 65-67, 1984.

Gómez-Pérez, A., and Manzano-Macho, D., “A survey of ontology learning methods and techniques”. Deliverable 1.5, IST Project IST-2000-29243 - OntoWeb, 2003.

Google, “Google Dictionary,” http://www.google.com/dictionary, retrieved on 2011-06-06.

Gruber, T.R., “A translation approach to portable ontology specifications,” *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.

Han, J., and Kamber, M., “Data Mining: Concepts and Techniques,” 2nd edition, Morgan Kaufmann, San Francisco, pp. 616–617, 2006.

Hogler, R. L., Henle, C., and Bemus, C., "Internet recruiting and employment discrimination: a legal perspective," *Human Resource Management Review*, vol. 8, no. 2, pp. 149-164, 1998.

Joachims, T., “Text categorization with support vector machines: learning with many relevant features”, *Proceedings of the 10th European Conference of Machine learning*, pp. 137–142, 1998.

Kietz, J., Maedche, A., and Volz, R., “A method for semi-automatic ontology acquisition from a corporate intranet,” *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns, Workshop on Ontologies and Text*, Berlin, Germany, 2000.

Luong, H.P, Gauch, S., and Wang, Q., “Ontology Learning Trough Focused Crawling and Information Extraction,” *Proceedings of the Internation Conference on Knowledge and Systems Engineering*, Hanoi, Vietnam, 2009.

Manning, C., and Schütze, H., “Foundations of Statistical Natural Language Processing,” MIT press, Cambridge, England, 1999.

Maynard, D., Peters, W., and Li, Y., “Metrics for evaluation of ontology-based information extraction,” *Proceedings of the WWW 2006 Workshop on Evaluation of Ontologies for the Web,* ACM, New York, 2006.

Maynard, D., Li, Y., and Peters, W., “NLP Techniques for Term Extraction and Ontology Population,” *Proceedings of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pp. 107-127, 2008.

Moens, M., “Information Extraction: Algorithms and Prospects in a Retrieval Context,” Springer-Verlag, Secaucus, NJ, 2003.

Omelayenko, B., “Learning of ontologies for the Web: the analysis of existent approaches”, *Proceedings of the international workshop on Web dynamics*, London, England, 2001.

Princeton University, “WordNet; A lexical database for English,” http://wordnet.princeton.edu, retrieved on 2012-05-25.

Riloff, E., “Information extraction as a stepping stone toward story understanding.” In: Ram, A., and Moorman, K., “Understanding language understanding: Computational models of reading,” MIT Press, Cambridge, MA, 1999.

Russell, S.J., Norvig, P., “Artificial Intelligence: A Modern Approach” (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, 2003.

Staab, S., “Handbook on ontologies,” Berlin, Heidelberg: Springer, 2009.

Studer, R., Benjamins, V.R., and Fensel, D., “Knowledge engineering: principles and methods,” *Data Knowledge Engineering*, vol. 25, no. 1, pp. 161-197, 1998.

Turing, A.M. (1950). Computing machinery and intelligence. Mind, 59, 433-460.

University of Amsterdam, “EuroWordNet; Building a multilingual database with wordnets for several European languages,” http://www.illc.uva.nl/EuroWordNet, retrieved on 2012-05-25.

Yang, Y., Zhang, J., and Kisiel, B., “A scalability analysis of classifiers in text categorization”, *Proceedings of 26th ACM SIGIR Conference*, pp. 96-103, 2003.

Van Rijsbergen, C. “Information Retrieval,” Butterworths, London, 1979.

Wikimedia foundation, “Wikipedia,” http://www.wikipedia.org/, retrieved on 2011-06-06.

Wimalasuriya, D.C., and Dou, D., “Ontology-based Information Extraction: An Introduction and a Survey of Current Approaches”, *Journal of Information Science*, vol. 36, no. 306, 2010.

Wolfswinkel, J.F., Furtmueller, E., and Wilderom C.P.M., “Reflecting on e-Recruiting Research Using Grounded Theory”, *Proceedings of the European Conference on Information Systems*, Pretoria, South Africa, 2010.

Word Wide Web Consortium, “W3C,” http://www.w3.org/2004/02/skos/, retrieved on 2012-06-03.

Appendices

A. Task Sheet Domain Expert Experiment

Experiment DE

In this experiment you will be asked to enrich an ontology three times. Please use the editor in the e-Recruiter system to do so and only use the plugin as an information source. Please use the synonym functionality of the e-Recruiter system for synonyms. Please fulfill each task completely before starting the next and do not go back. Perform the tasks in an ascending order, first task 1, then task 2 and finally task 3. Please record time per task.

Reaching the e-Recruiter system
Please use your favorite browser to go to the following address:

http://recruiter4.joost.dev1.zites.net

User: "********"
Password: "*******"

and

User: "********"
Password: "********"

In the grey shortcut bar (second bar from the top), three links are visible. Task 1, Task 2 and Task 3 respectively. These links will guide you to the taxonomy that you will enrich for each task.

Reaching the plugin
Please use your favorite browser to go to the following address:

http://jfwolfswinkel.nl

User: "********"
Password: "********"

Three links will appear. Task 1, Task 2 and Task 3 respectively. These links will guide you to the resources needed to perform each task.

Task 1
Based on the terms/phrases provided by the plugin, enrich the ontology in the system. Perform this task however you see fit, but keep track of all the suggested terms that you use to enrich the ontology that do not go into the ontology one-to-one. This can be done in a text-file called “Task 1”.

Task 2
Manually go through all the provided 100 job advertisements and enrich the ontology “Task 2” in the system. Perform this task however you see fit, but keep track of all the terms/phrases from the job advertisements that you use to enrich the ontology that do not go into the ontology one-to-one. This can be done in a text-file called “Task 2”.

Task 3
Based on the terms/phrases provided by the plugin, enrich the ontology in the system. Perform this task however you see fit, but keep track of all the suggested terms that you use to enrich the ontology that do not go into the ontology one-to-one. This can be done in a text-file called “Task 3”.

B. Metrics Measurement 1-100

DE1
Precision = 24 / 24 + 16 ≈ 0,60
Recall = 24 / 200 ≈ 0,12

Drupal skills (17 items vs. 1 item)
Precision = 0 / 0 + 1 ≈ 0
Recall = 0 / 17 ≈ 0

Fields of study (4 items vs. 3 items)
Precision = 1 / 1 + 2 ≈ 0,33
Recall = 1 / 4 ≈ 0,25

General skills (10 items vs. 6 items)
Precision = 4 / 4 + 2 ≈ 0,66
Recall = 4 / 10 ≈ 0,40

Industry fields (1 item vs. 0 items)
Precision = 0 / 0 + 0 ≈ 0
Recall = 0 / 1 ≈ 0

IT skills (137 items vs. 16 items)
Precision = 12 / 12 + 4 ≈ 0,75
Recall = 12 / 137 ≈ 0,09

Occupational fields (31 items vs. 14 items)
Precision = 7 / 7 + 7 ≈ 0,50
Recall = 7 / 31 ≈ 0,23

DE2
Precision = 25 / 25 + 27 ≈ 0,48
Recall = 25 / 112 ≈ 0,22

Drupal skills (11 items vs. 6 items)
Precision = 1 / 1 + 5 ≈ 0,17
Recall = 1 / 11 ≈ 0,09

Fields of study (0 items vs. 1 item)
Precision = 0 / 0 + 1 ≈ 0
Recall = 0 / 0 ≈ 0

General skills (5 items vs. 3 items)
Precision = 0 / 0 + 3 ≈ 0
Recall = 0 / 5 ≈ 0

Industry fields (1 item vs. 0 items)
Precision = 0 / 0 + 0 ≈ 0
Recall = 0 / 1 ≈ 0

IT skills (83 items vs. 22 items)
Precision = 19 / 19 + 3 ≈ 0,86
Recall = 19 / 83 ≈ 0,23

Occupational fields (12 items vs. 20 items)
Precision = 5 / 5 + 15 ≈ 0,25
Recall = 5 / 12 ≈ 0,42

DE3
Precision = 34 / 34 + 18 ≈ 0,65
Recall = 26 / 105 ≈ 0,25

Drupal skills (8 items vs. 3 items)
Precision = 0 / 0 + 3 ≈ 0
Recall = 0 / 8 ≈ 0

Fields of study (8 items vs. 7 items)
Precision = 3 / 3 + 4 ≈ 0,43
Recall = 3 / 8 ≈ 0,38

General skills (3 items vs. 1 items)
Precision = 0 / 0 + 1 ≈ 0
Recall = 0 / 3 ≈ 0

Industry fields (1 item vs. 5 items)
Precision = 1 / 1 + 4 ≈ 0,20
Recall = 1 / 1 ≈ 1,00

IT skills (65 items vs. 13 items)
Precision = 12 / 12 + 1 ≈ 0,92
Recall = 12 / 65 ≈ 0,18

Occupational fields (20 items vs. 15 items)
Precision = 10 / 10 + 5 ≈ 0,67
Recall = 10 / 20 ≈ 0,50

C. Absent term listings measurement 1-100

DE1

Drupal skills

Term	Type	Freq (path)	Available
CCK	NULL		N
Custom module development	Noun noun noun	8 20	Y
Display Suite	Verb noun	0 1	N
Drush	Unknown	2	N
Image Cache	Noun -	0 2	N
Node Queue	Noun -	0 1	N
Organic Groups	- noun	0 101	N
Panels	noun	4	N
Press Flow	Verb verb	3 0	N
Theming	verb	19	N
Ubercart	unknown	17	Y
Ubercart Bulk Discount	Unknown - -	17 0 0	N
Ubercart Coupon	Unknown -	17 0	N
Ubercart Free Order	Unknown - -	17 0 0	N
Ubercart Product Keys	Unknown - -	17 0 0	N
Views	noun	9	N
Zen Theme	- noun	0 19	N

General skills

Term	Type	Freq (path)	Available
Decision making skills	Noun noun noun	0 0 93	N
Presentation skills	noun noun	0 93	N
Relationship skills	Noun noun	0 93	N
Team collaboration skills	Noun noun noun	0 0 93	N
Team player	Noun noun	95 2	Y
Writing skills	Noun noun	1 93	Y

Fields of study

Term	Type	Freq (path)	Available
Engineering	noun	14	Y
Life Sciences	Noun noun	6 1	N
Math	-	0	N

Industry fields

Term	Type	Freq (path)	Available
eCommerce	unknown	4	N

IT skills

Term	Type	Freq (path)	Available
*NIX	Unknown	-	N
.NET	unknown	-	N
Adobe Acrobat Pro	- - -	-	N
Akamai	-	-	N
Android	noun	4	N
Application Server	Noun noun	16 0 / 0 14	N
ASP	-	-	N
C	NULL	1	N
C#	-	-	N
C++	-	-	N
CakePHP	-	-	N
CGI	-	-	N
CodeIgnitor	-	-	N
ColdFusion	-	-	N
COTS	noun	1	N
CURL	-	-	N
CVENT	-	-	N
CVS	NULL	-	N
DAS	-	-	N
DB2	-	-	N
DHTML	NULL	-	N
DNS	-	-	N
Dojo	Noun	1	N
DOM	-	-	N
Dreamweaver	Unknown	1	N
Eclipse	-	-	N
EXT	-	-	N
Firefox	Unknown	2	N
Flash	Verb	12	N
FTP	NULL	-	N
GIMP	-	-	N
Git	Noun	4	N
Gomez	-	-	N
Google Analytics	Verb unknown	5 1 / 1 5	N
Googlemap	-	-	N
Hibernate	Verb	2	N
HTTP	NULL	-	N
HttpUnit	unknown	2	N
Illustrator	-	-	N
Internet Explorer	Unknown -	6 -	N
Java	noun	7	N
JIRA	-	-	N
Joomla	Unknown	3	N
JPA design	- noun	0 64	N
jQuery Mobile	Unknown adjective	21 0	N
jSON	unknown	2	N

JSP	Unknown	2	N
JUnit	-	-	N
LDAP	Unknown	1	N
Magento	Unknown	1	N
Memcache	NULL	-	N
Mercurial	-	-	N
Microsoft Dynamics	Unknown -	2 -	N
Microsoft IIS	Unknown unknown	2 1 / 1 2	N
Microsoft Office	Unknown noun	2 0 / 0 22	N
MochiKit	-	-	N
Moodle	-	-	N
MS Access	NULL noun	- 9	N
MS Excel	NULL -	- -	N
MS Outlook	NULL -	- -	N
MS PowerPoint	NULL -	- -	N
MS Project	NULL noun	- 71	N
MS SQL	NULL NULL	- -	N
MS Word	NULL noun	- 7	N
MVC	Unknown	2	N
Objective-C	-	-	N
Omniture	-	-	N
OOAD	Unknown	2	N
OOP	Unknown	1	N
OpenId	Unknown	2	N
Perl	Noun	3	N
Phonegap	-	-	N
Photoshop	Verb	2	N
PhPUnit	Unknown	2	N
Plone	-	-	N
Prototype	Noun	1	N
Prototyping	-	-	N
Python	Noun	9	N
Querqus	-	-	N
RDBMS	-	-	N
REST	Verb	1	N
RHEL	-	-	N
Rspec	-	-	N
RSS	NULL	-	N
Ruby	Noun	4	N
Ruby on Rails	Noun preposition -	4 - -	N
SaaS	Unknown	3	N
Safari	Noun	2	N
SAML	-	-	N
SAN	Noun	7	N
SCAP	-	-	N
SDLC	Unknown	1	N
Section 508 compliance	Noun - -	5 - -	N
SEO	Unknown	6	N
Shell scripting	- verb	- 3	N

Silverlight	Unknown	2	N
Smarty	-	-	N
SOAP	-	-	N
SQL	NULL	-	N
SSH	-	-	N
SSO	-	-	N
Subversion	Unknown	1	N
Sysadmin	-	-	N
Telnet	-	-	N
Test Driven Development	Noun adjective noun	8 0 0 / 0 0 178	N
Titanium	Noun	1	N
Unfuddle	-	-	N
Unit Testing	Noun verb	5 3 / 3 21	N
Unix	Noun	7	N
Varnish	Noun	2	N
VBScript	-	-	N
Virtualmin	-	-	N
Visual Studio	Adjective noun	0 2	N
VLAN	-	-	N
VPN	-	-	N
W3C standards	- noun	- 8	N
WAMP	Unknown	1	N
Webmin	-	-	N
Website Baker	noun -	38 -	N
Wireframes	Noun	1	N
Wordpress	Unknown	4	N
XHTML	NULL	-	N
XML	NULL	-	N
YUI	-	-	N
Zend	Noun	1	N

Occupational fields

Term	**Type**	**Freq (path)**	**Available**
.NET Developer	- noun	- 156	N
Consultant	Noun	7	N
Contractor	noun	9	N
Designer	Noun	23	Y
Developer	Noun	156	Y
Drupal 7 Developer	Unknown - noun	399 - -	N
Drupal Back End Developer	Unknown noun noun noun	399 0 - -	N
Drupal Configurator	Noun -	399 -	N
Drupal Module Developer	Noun noun noun	399 5 0	N
Drupal Programmer	Noun noun	399 53	Y
Drupal Software Developer	Noun noun noun	399 0 0	N
Freelance	Adjective noun	0 9	N

Contractor			
Freelancer	Noun	2	N
Integrator	Noun	1	N
Programmer	Noun	12	Y
Project manager	Noun noun	71 6 / 6 29	Y
Quality engineer	Noun noun	10 0 / 0 9	N
Requirements engineer	Noun noun	50 2 / 2 9	Y
SEM Specialist	NULL noun	- 5	N
Senior Drupal Developer	Adjective unknown noun	- - - / 2 399 2	Y
Team Lead	Noun verb	95 2 / 2 21	Y
Tech Lead	Noun verb	7 1 / 1 21	N
Web Application Development	Noun noun noun	246 6 2 / 6 16 2 / 2 5 178	Y
Web Development	Noun noun	23 178 / 247 23	Y
Web Software Developer	Noun noun noun	247 0 -	N

DE2

Drupal skills

Term	**Type**	**Freq (path)**	**Available**
Block creation	Noun noun	1 8	N
Core API	Noun -	8 -	N
Display Suite	Verb noun	- 1	N
Drupal template theming	Unknown noun verb	399 51 -	N
Drupal UI theming	Unknown unknown verb	399 3 -	N
Drush	Unknown	2	N
Image Cache	Noun -	1	N
Node Queue	Noun -	1	N
Panels	Noun	4	N
Views	Noun	9	N

Fields of study

Term	**Type**	**Freq (path)**	**Available**

General skills

Term	Type	Freq (path)	Available
communication skills	Noun noun	21 18	Y
interpersonal skills	Adjective noun	- 93	N
project management skills	Noun noun noun	71 11 5	Y
verbal skills	Adjective noun	- 93	N
written skills	Verb noun	- 93	N

Industry fields

Term	Type	Freq (path)	Available
e-commerce	Noun	4	N

IT skills

Term	Type	Freq (path)	Available
.NET	-	-	N
Akamai CDN	- -	- -	N
Android development	Noun noun	4 -	N
ASP	-	-	N
Basic subversion usage	Adjective unknown noun	- - -	N
C	-	-	N
C#	-	-	N
C++	-	-	N
CakePHP	-	-	N
CGI	-	-	N
Coldfusion	-	-	N
CURL	-	-	N
CVS Knowledge	-	- -	N
Design patterns	Noun noun	64 -	N
DNS	-	-	N
Document Object Model	Noun noun -	5 - -	N
Dojo	Noun	1	N
Drupal	Unknown	399	Y
Eclipse	-	-	N
Ext JS	- -	- -	N
frameworks	Noun	8	N
FTP	-	-	N
game development	Noun noun	4 2	N
GIMP	-	-	N
Git usage	Noun noun	4 -	N
HTML5	-	-	N
iphone	Unknown noun	8 -	N

development			
Java	Noun	7	N
JOOMLA	Unknown	3	N
jQuery Mobile	Unknown adjective	21 -	N
jSON	Unknown	2	N
JSP	Unknown	2	N
Junit	-	-	N
LDAP	Unknown	1	N
Microsoft IIS	Unknown unknown	2 1	N
Microsoft SQL	Unknown -	2 -	N
mobile development frameworks	Adjective noun noun	- - -	N
MochiKit	-	-	N
Moodle	-	-	N
Objective-C	-	-	N
OOP	Unknown	1	N
OpenId	Unknown	2	N
Perl	Noun	3	N
PHP5	-	-	N
phpUnit	Unknown	2	N
REST	verb	-	N
Ruby	noun	4	N
Shell scripting	- -	- -	N
Silverlight	Unknown	2	N
SOAP	-	-	N
software design	Noun noun	34 -	N
software implementation	Noun noun	34 -	N
SQL	-	-	N
SSH	-	-	N
SSO techniques	- noun	- -	N
Telnet	-	-	N
Unix	Noun	7	N
VBScript	-	-	N
VLAN	-	-	N
VPN	-	-	N
WEB 2.0	-	-	N
web services	Noun noun	247 2	Y
Wordpress	Unknown	4	N
YUI	-	-	N

Occupational fields

Term	Type	Freq (path)	Available
backup systems administrator	Noun noun -	1 - -	N
Developer Drupal	Noun unknown	156 90	Y
Drupal Back End Developer	Unknown noun noun noun	399 - - -	N
Drupal Configurator	Unknown -	399	N
Drupal Themer	Unknown unknown	399 55	Y
PHP Architect	Unknown noun	98 -	N
PHP expert	Unknown noun	98 1	Y

DE3

Drupal skills

Term	Type	Freq (path)	Available
CCK	NULL	-	N
Display Suite	Verb noun	- -	N
Image Cache	Noun -	1 -	N
Node Queue	Noun -	1 -	N
Organic group	- noun	- -	N
Panels	noun	4	N
Theming	verb	-	N
Views	noun	9	N

Fields of study

Term	Type	Freq (path)	Available
Computer Engineering	Noun noun	19 2	Y
Information Systems	Noun noun	41 2	Y
Math	-	-	N
Software design	Noun noun	34 -	N
Systems Analysis	Noun noun	22 -	N

General skills

Term	Type	Freq (path)	Available
Analytical skills	Adjective noun	- -	N
Communication skills	Noun noun	21 18	Y
Team collaboration skills	Noun noun noun	95 - -	N

Industry fields

Term	Type	Freq (path)	Available

IT skills

Term	Type	Freq (path)	Available
ASP	-	-	N
C	NULL	-	N
C#	-	-	N
C++	-	-	N
ColdFusion	-	-	N
COTS	Noun	1	N
css3	-	-	N
CURL	-	-	N
DB2	-	-	N
DHTML	NULL	-	N
Dreamweaver	Unknown	1	N
Eclipse IDE	- -	- -	N
EXT	-	-	N
Flash	-	-	N
GIMP	-	-	N
Git	Noun	4	N
Gomez	-	-	N
Hibernate	Verb	-	N
HTML5	-	-	N
HTTP	NULL	-	N
Illustrator	-	-	N
Java	Noun	7	N
JIRA	-	-	N
Joomla	Unknown	3	N
jSON	Unknown	2	N
JSP	Unknown	2	N
Junit	-	-	N
memcache	NULL	-	N
Microsoft Office	Unknown noun	2 -	N
Microsoft SQL	Unknown NULL	2 -	N
MVC	Unknown	2	N
OOAD	Unknown	2	N
OOP	Unknown	1	N
Photoshop	Verb	-	N
phpUnit	Unknown	2	N
Python	Noun	9	N
Rspec	-	-	N
Ruby	Noun	4	N
SEO	Unknown	6	N
Silverlight	Unknown	2	N
Smarty	-	-	N
SQL	NULL	-	N
Subversion	Unknown	1	N
Ubercart	Unknown	17	Y

UNIX	Noun	7	N
Varnish	Noun	2	N
VBScript	-	-	N
Visual Studio	Adjective noun	- -	N
WAMP	Unknown	1	N
Wordpress	Unknown	4	N
xhtml	NULL	-	N
xml	NULL	-	N
YUI	-	-	N

Occupational fields

Term	**Type**	**Freq (path)**	**Available**
Drupal 7 Developer	Unknown - noun	399 - -	N
Drupal Back End Developer	Unknown noun noun noun	399 - - -	N
Drupal Configurator	Unknown -	399 -	N
Drupal Consultant	Unknown noun	399 -	N
Junior Drupal Developer	Adjective unknown noun	- - -	N
LAMP Engineer	Noun noun	20 -	N
PHP Architect	Unknown noun	98 -	N
Requirements engineer	Noun noun	50 2	Y
Senior Drupal Developer	Adjective unknown noun	- - -	N
Senior Web Software Developer	Adjective noun noun noun	- - - -	N

D. Metrics Measurement 1-1.000

DE1
Precision = 65 / 65 + 30 ≈ 0,68
Recall = 65 / 200 ≈ 0,33

Drupal skills (17 items vs. 6 items)
Precision = 5 / 5 + 1 ≈ 0,83
Recall = 5 / 17 ≈ 0,29

Fields of study (4 items vs. 3 items)
Precision = 1 / 1 + 2 ≈ 0,33
Recall = 1 / 4 ≈ 0,25

General skills (10 items vs. 11 items)
Precision = 6 / 6 + 5 ≈ 0,55
Recall = 6 / 10 ≈ 0,60

Industry fields (1 item vs. 1 item)
Precision = 1 / 1 + 0 ≈ 1
Recall = 1 / 1 ≈ 1

IT skills (137 items vs. 55 items)
Precision = 43 / 43 + 12 ≈ 0,78
Recall = 43 / 137 ≈ 0,31

Occupational fields (31 items vs. 20 items)
Precision = 10 / 10 + 10 ≈ 0,50
Recall = 10 / 31 ≈ 0,32

DE2
Precision = 45 / 45 + 35 ≈ 0,56
Recall = 45 / 112 ≈ 0,40

Drupal skills (11 items vs. 8 items)
Precision = 2 / 2 + 6 ≈ 0,25
Recall = 2 / 11 ≈ 0,18

Fields of study (0 items vs. 1 items)
Precision = 0 / 0 + 1 ≈ 0
Recall = 0 / 0 ≈ 0

General skills (5 items vs. 4 items)
Precision = 2 / 2 + 2 ≈ 0,50
Recall = 2 / 5 ≈ 0,40

Industry fields (1 items vs. 1 items)
Precision = 1 / 1 + 0 ≈ 1,00
Recall = 1 / 1 ≈ 1,00

IT skills (83 items vs. 57 items)
Precision = 38 / 38 + 19 ≈ 0,67
Recall = 38 / 83 ≈ 0,46

Occupational fields (12 items vs. 9 items)
Precision = 2 / 2 + 7 ≈ 0,22
Recall = 2 / 12 ≈ 0,17

DE3
Precision = 37 / 37 + 28 ≈ 0,57
Recall = 37 / 105 ≈ 0,35

Drupal skills (8 items vs. 3 items)
Precision = 1 / 1 + 3 ≈ 0,25
Recall = 1 / 8 ≈ 0,13

Fields of study (8 items vs. 8 items)
Precision = 6 / 6 + 2 ≈ 0,75
Recall = 6 / 8 ≈ 0,75

General skills (3 items vs. 1 item)
Precision = 0 / 0 + 1 ≈ 0
Recall = 0 / 3 ≈ 0

Industry fields (1 item vs. 2 items)
Precision = 1 / 1 + 1 ≈ 0,50
Recall = 1 / 1 ≈ 1,00

IT skills (65 items vs. 33 items)
Precision = 19 / 19 + 14 ≈ 0,58
Recall = 19 / 65 ≈ 0,29

Occupational fields (20 items vs. 17 items)
Precision = 10 / 10 + 7 ≈ 0,59
Recall = 10 / 20 ≈ 0,50

E. Absent term listings Measurement 1-1.000

DE1

Drupal skills

Term	Type	Freq (path)	Available
CCK	NULL	-	N
Display suite	Verb NULL	- -	N
Drush	Unknown	8	N
Image cache	Noun noun	24 2	Y
Node queue	Noun noun	37 2	Y
Press flow	Verb verb	9 1	N
Theming	verb	118	N
Ubercart	Unknown	105	Y
Ubercart Bulk Discount	Unknown noun noun	105 2 2	Y
Ubercart Coupon	Unknown noun	105 2	Y
Ubercart Free Order	Unknown adjective noun	105 0 -	N
Ubercart Product Keys	Unknown noun noun	105 3 2	Y
Zen theme	Noun noun	11 5	Y

Fields of study

Term	Type	Freq (path)	Available
Engineering	noun	67	Y
Life Sciences	Noun noun	50 6	Y
Math	noun	2	N

General skills

Term	Type	Freq (path)	Available
Decision making skills	Noun NULL noun	6 - -	N
Presentation skills	Noun noun	21 5	Y
Relationship skills	Noun noun	28 1	Y
Team collaboration skills	Noun noun noun	658 2 2	Y

IT skills

Term	Type	Freq (path)	Available
*NIX	Unknown	3	N
Adobe Acrobat Pro	Noun noun noun	27 1 1	Y
Akamai	Unknown	4	N
Apache	NULL	-	N
Application server	Noun noun	197 4	Y

C	NULL	-	N
C#	-	-	N
C++	-	-	N
CGI	NULL	-	N
CodeIgnitor	Unknown	5	N
ColdFusion	Unknown	7	N
COTS	Noun	2	N
CURL	Verb	2	N
CVENT	Unknown	1	N
CVS	NULL	-	N
DB2	-	-	N
DHTML	NULL	-	N
DNS	NULL	-	N
DOM	Noun	8	N
Dreamweaver	Unknown	6	N
Eclipse	Noun	8	N
EXT	Unknown	6	N
Flash	verb	63	N
FTP	NULL	-	N
GIMP	Noun	1	N
Gomez	Unknown	1	N
Googlemap	Unknown	4	N
Hibernate	verb	7	N
HTTP	NULL	-	N
HttpUnit	Unknown	6	N
Illustrator	Noun	9	N
Internet Explorer	Unknown noun	77 6	Y
JIRA	Unknown	2	N
JPA design	Unknown noun	1 1	N
JQuery mobile	Unknown adjective	216 4	Y
JUnit	-	-	N
LDAP	Unknown	7	N
Mercurial	NULL	-	N
Microsoft Dynamics	NULL NULL	- -	N
Microsoft IIS	NULL unknown	- 8	N
MochiKit	Unknown	1	N
Moodle	Unknown	3	N
MS Excel	NULL NULL	- -	N
MS Outlook	NULL noun	- 3	N
MS Project	NULL noun	- 466	N
MS SQL	NULL NULL	- -	N
MS Word	NULL NULL	- -	N
Objective-C	Unknown	3	N
Omniture	Unknown	1	N
OOAD	Unknown	6	N
OpenId	Unknown	6	N
Phonegap	Unknown	1	N
Photoshop	Verb	59	N
PhPUnit	Unknown	7	N

Plone	Unknown	2	N
Prototyping	Verb	3	N
Querqus	Unknown	1	N
RDBMS	NULL	-	N
REST	Verb	19	N
RHEL	Unknown	1	N
Rspec	-	-	N
RSS	NULL	-	N
Ruby on Rails	Noun NULL noun	22 - -	N
SAML	Unknown	1	N
SAN	Noun	72	Y
SCAP	Unknown	1	N
SDLC	Unknown	11	Y
Section 508 compliance	Noun - -	5 - -	N
Shell scripting	NULL verb	- 40	N
Silverlight	Unknown	5	N
Smarty	Noun	8	N
SQL	NULL	-	N
SSH	Unknown	9	N
SSO	Unknown	2	N
Subversion	NULL	-	N
Sysadmin	Unknown	1	N
Telnet	Noun	1	N
Test Driven Development	Noun adjective noun	58 1 1	Y
Titanium	Noun	5	N
Unfuddle	Unknown	6	N
Unit Testing	Noun NULL	32 11	Y
Varnish	Noun	7	N
VBScript	Unknown	1	N
Virtualmin	Unknown	2	N
VLAN	Unknown	1	N
VPN	NULL	-	N
W3C standards	- NULL	-	N
WAMP	Unknown	6	N
Webmin	Unknown	2	N
Website Baker	Noun noun	374 1	Y
Wireframes	Noun	15	Y
XHTML	NULL	-	N
XML	NULL	-	N

Occupational fields

Term	**Type**	**Freq (path)**	**Available**
.NET Developer	.Noun noun	1 1229	Y
Consultant	Noun	47	Y
Contractor	NULL	-	N
Designer	Noun	119	Y
Developer	Noun	1229	

Drupal 7 Developer	Unknown – noun	3288 - -	N
Drupal Back End Developer	Noun noun noun noun	3288 2 2 2	Y
Drupal Configurator	Noun unknown	3288 355	Y
Drupal Programmer	Noun noun	3288 395	
Drupal Software Developer	Noun noun noun	3288 356 353	Y
Freelance Contractor	NULL NULL	- -	N
Freelancer	Noun	22	Y
Integrator	Noun	6	N
Programmer	Noun	107	Y
Quality engineer	Noun noun	84 0	N
Requirements engineer	Noun noun	332 6	Y
SEM Specialist	NULL noun	- 10	N
Senior Drupal Developer	Adjective noun noun	78 3288 73	Y
Tech Lead	Noun verb	18 2	Y
Web Application Development	Noun noun noun	1458 163 29	Y
Web Development	Noun noun	1458 240	Y
Web Software Developer	Noun noun noun	1458 8 7	Y

DE2

Drupal skills

Term	**Type**	**Freq (path)**	**Available**
Block creation	Noun noun	10 1	Y
core API	Noun -	- -	N
Display Suite	Verb noun	- -	N
Drupal template theming	Unknown noun verb	3288 356 -	N
Drupal UI theming	Unknown unknown verb	3288 2 -	N
Drush	unknown	8	N
Image Cache	Noun -	24 2	Y
Node Queue	Noun -	37 2	Y
Panels	Noun	26	Y

Fields of study

Term	**Type**	**Freq (path)**	**Available**

General skills

Term	Type	Freq (path)	Available
communication skills	Noun noun	185 140	Y
interpersonal skills	Adjective noun	- -	N
project management skills	Noun noun noun	466 45 16	Y

Industry fields

Term	Type	Freq (path)	Available

IT skills

Term	Type	Freq (path)	Available
.NET	-	-	N
Akamai CDN	- -	-	N
android development	Noun noun	24 1	Y
Apache2	-	-	N
Basic subversion usage	Adjective unknown noun	- - -	N
C#	-	-	N
C++	-	-	N
CGI	-	-	N
ColdFusion	-	-	N
CURL	-	-	N
CVS knowledge	- noun	- -	N
design patterns	Noun noun	558 6	Y
DNS	-	-	N
Document Object Model	Noun noun -	- - -	N
Dojo	Noun	13	Y
Drupal	Unknown	3288	Y
frameworks	Noun	-	N
FTP	-	-	N
GIMP	-	-	N
Git usage	Noun noun	50 -	N
HTML5	-	-	N
iphone development	Unknown noun	43 -	N
jQuery Mobile	Unknown adjective	216 4	Y
JSP	Unknown	23	Y
JUnit	-	-	N
Microsoft IIS	Unknown unknown	- -	N
Microsoft SQL	Unknown -	- -	N
mobile development frameworks	Adjective noun noun	- - -	N
Moodle	-	-	N
PHP	unknown	933	Y
PHP5	-	-	N

phpUnit	unknown	7	N
Shell scripting	- -	-	N
Silverlight	unknown	5	N
software design	Noun noun	284 11	Y
software implementation	Noun noun	284 -	N
SSH	-	-	N
SSO techniques	- noun	- -	N
Telnet	-	-	N
VBScript	-	-	N
VLAN	-	-	N
VPN	-	-	N
WAMP	Unknown	6	N
WEB 2.0	-	-	N
web services	Noun noun	1458 167	Y

Occupational fields

Term	**Type**	**Freq (path)**	**Available**
backup systems administrator	Noun noun -	12 4 4	Y
Developer Drupal	Noun unknown	1229 297	Y
Drupal Back end Developer	Unknown noun noun noun	3288 2 2 2	Y
Drupal Configurator	Unknown -	3288 -	N
drupal programmer	Unknown noun	3288 1	Y
PHP Architect	Unknown noun	933 47	Y
php developers	Unknown noun	933 5	Y
PHP Expert	Unknown noun	933 8	Y
Themer	unknown	76	Y
Web Designer	Noun noun	1458 3	Y

DE3

Drupal skills

Term	**Type**	**Freq (path)**	**Available**
CCK	NULL	-	N
Display Suite	Verb NULL	- -	N
Image Cache	Noun noun	24 2	Y
Node Queue	Noun noun	37 2	Y
Organic group	Adjective noun	- -	N
Panels	Noun	26	Y
Theming	verb	-	N

Fields of study

Term	**Type**	**Freq (path)**	**Available**
Math	noun	2	N
Systems Analysis	Noun noun	232 3	Y

General skills

Term	Type	Freq (path)	Available
Analytical skills	Adjective noun	- -	N
Communication skills	Noun noun	185 140	Y
Team collaboration skills	Noun noun noun	658 2 2	Y

Industry fields

Term	Type	Freq (path)	Available

IT skills

Term	Type	Freq (path)	Available
Apache	NULL	-	N
C	NULL	-	N
C#	-	-	N
C++	-	-	N
ColdFusion	Unknown	7	N
COTS	Noun	2	N
css3	-	-	N
CURL	Verb	-	N
DB2	-	-	N
DHTML	NULL	-	N
Dreamweaver	Unknown	6	N
Eclipse IDE	Noun NULL	8 2	N
EXT	Unknown	6	N
Flash	Verb	-	N
GIMP	Noun	1	N
Gomez	Unknown	1	N
Hibernate	Verb	-	N
HTML5	-	-	N
HTTP	NULL	-	N
Illustrator	Noun	9	N
JIRA	Unknown	2	N
jSON	Unknown	20	Y
JSP	Unknown	23	Y
Junit	-	-	N
memcache	Unknown	10	Y
Microsoft Office	NULL noun	- -	N
Microsoft SQL	NULL NULL	- -	N
MVC	Unknown	19	Y
OOAD	Unknown	6	N
OOP	Unknown	12	Y
Photoshop	Verb	-	N
PHP	Unknown	933	Y
phpUnit	Unknown	7	N
Rspec	-	-	N
Silverlight	Unknown	5	N

Smarty	Noun	8	N
SQL	NULL	-	N
Subversion	NULL	-	N
Varnish	Noun	7	N
VBScript	Unknown	1	N
Visual Studio	Adjective noun	- -	N
WAMP	Unknown	6	N
Web services	Noun noun	1458 167	Y
xhtml	NULL	-	N
xml	NULL	-	N
YUI	Unknown	13	Y

Occupational fields

Term	**Type**	**Freq (path)**	**Available**
Drupal 7 Developer	Unknown - noun	3288 - -	N
Drupal Back End Developer	Unknown noun noun noun	3288 2 2 2	N
Drupal Configurator	Unknown unknown	3288 6	Y
Drupal Consultant	Unknown noun	3288 393	Y
Junior Drupal Developer	Adjective unknown noun	- - -	N
LAMP Engineer	Noun noun	171 2	Y
PHP Architect	Unknown NULL	933 47	Y
Senior Developer	Adjective noun	- -	N
Senior Drupal Developer	Adjective unknown noun	- - -	N
Senior Web Software Developer	Adjective noun noun noun	- - - -	N

F. Golden Standard Ontology GS

Drupal skills	
CCK	1,2,3
Display Suite	1,2,3
Drush	2,3
Image Cache	1,2,3
Node Queue	1,2,3
Panels	1,2,3
Theming	1,3
Views	1,2,3
Fields of study	
Computer Science	1,3
Math	1,3
General skills	
Analytical skills	1,3
Communication skills	1,2,3
Team collaboration skills	1,3
Industry fields	
eCommerce	1,2
IT skills	
.NET	1,2
AJAX	1,2,3
Apache	1,2,3
ASP	1,2,3
C	1,2,3
C#	1,2,3
C++	1,2,3
CGI	1,2
ColdFusion	1,2,3
COTS	1,3
CSS	1,2,3
CURL	1,2,3
DB2	1,3
DHTML	1,2,3
DNS	1,2
Dojo	1,2
Dreamweaver	1,3
Drupal	1,2,3
Eclipse	1,2
EXT	1,3
Flash	1,3
FTP	1,2
GIMP	1,2,3

Git	1,3
Gomez	1,3
HTML	1,3
HTML5	2,3
HTTP	1,3
Illustrator	1,3
Java	1,2,3
JavaScript	1,2,3
JIRA	1,3
Joomla	1,2,3
Jquery	1,2,3
jQuery Mobile	1,2
jSON	1,2,3
JSP	1,2,3
JUnit	1,2,3
LAMP	1,2,3
LDAP	1,2
Linux	1,2,3
Memcache	1,3
Microsoft IIS	1,2
Microsoft Office	1,3
MochiKit	1,2
Moodle	1,2
MVC	1,3
MySQL	1,2,3
Objective-C	1,2
OOAD	1,2,3
OOP	1,2,3
OpenId	1,2
Perl	1,2
PHP	1,2,3
Photoshop	1,3
phpUnit	1,2,3
Python	1,2,3
REST	1,2
Rspec	1,3
Ruby	1,2,3
SEO	1,3
Shell scripting	1,2
Silverlight	1,2,3
Smarty	1,3
SOAP	1,2
SQL	1,2,3
SSH	1,2
Subversion	1,3
Telnet	1,2

Unix	1,2,3
Varnish	1,3
VBScript	1,2,3
Visual Studio	1,3
VLAN	1,2
VPN	1,2
WAMP	1,2,3
Web services	1,2,3
Wordpress	1,2,3
XHTML	1,2,3
XML	1,2,3
YUI	1,2,3
Zend	1,2

Occupational fields

Drupal 7 Developer	1,3
Drupal Back End Developer	1,2,3
Drupal Configurator	1,2,3
Drupal Programmer	1,2,3
Drupal Themer	1,2,3
PHP Architect	2,3
Project manager	1,3
Requirements engineer	1,3
Senior Drupal Developer	1,3
Web Designer	1,3
Web Developer	1,3

G. Measurements GS

DE1

Precision = 105 / 105 + 95 ≈ 0.53

Recall = 105 / 107 ≈ 0.98

DE2

Precision = 75 / 75 + 37 ≈ 0.67

Recall = 75 / 107 ≈ 0.70

DE3

Precision = 84 / 84 + 21 ≈ 0.80

Recall = 84 / 107 ≈ 0.79

H. Golden Standard Ontologies GST100 and GST1.000

GST100

Drupal skills	
Module development	1,3
Fields of study	
computer science	1,2,3
information technology	1,3
software engineering	1,3
General skills	
management skills	1,2
Industry fields	
IT Skills	
AJAX	1,2,3
Apache	1,2,3
CSS	1,2,3
Drupal	1,3
HTML	1,2,3
Javascript	1,2,3
Jquery	1,2,3
LAMP	1,2,3
Linux	1,2,3
MySQL	1,2,3
PHP	1,2,3
Web services	1,3
Occupational fields	
drupal developer	1,3
drupal programmer	2,3
drupal themer	1,3
marketing manager	1,3
php developer	1,2,3
product manager	1,3
project manager	1,3
senior developer	1,3
software developer	1,2,3
web designer	1,2,3
web developer	1,2,3

GST1.000

Drupal skills	
rules	2,3
views	1,2,3
Fields of study	
computer science	1,3
information technology	1,2,3
software engineering	1,3
General skills	
technical skills	1,2,3
Industry fields	
e-commerce	1,2
IT skills	
AJAX	1,2,3
ASP	1,2,3
CSS	1,2,3
Drupal	1,3
Firefox	1,3
Git	1,2,3
HTML	1,3
Java	1,2,3
Javascript	1,2,3
Joomla	1,2,3
Jquery	1,2,3
json	1,2
LAMP	1,2,3
Linux	1,2,3
MVC	1,2
MySQL	1,2,3
OOP	1,2
Perl	1,2,3
PHP	1,3
python	1,2,3
Ruby	1,2,3
SEO	1,3
SOAP	1,2,3
Solr	1,2
SVN	1,3
Ubercart	1,3
Unix	1,2,3
XSLT	1,2,3
YUI	1,2

Zend	1,2
Occupational fields	
Business analist	1,3
drupal architect	1,2
drupal developer	1,2,3
drupal engineer	1,2
drupal themer	1,2,3
marketing manager	1,3
php developer	1,3
product manager	1,3
project manager	1,3
software engineer	1,3
web designer	1,3
web developer	1,2,3

I. Measurements GST100 and GST1.000

GS compared to GST100

Precision = 18 / 18 + 10 ≈ 0.64

Recall = 18 / 107 ≈ 0.17

GS compared to GST1.000

Precision = 32 / 32 + 17 ≈ 0.65

Recall = 32 / 107 ≈ 0.30

J. Metrics Measurement 3-100

DE1

Precision = 39 / 39 + 16 ≈ 0,71
Recall = 39 / 200 ≈ 0,20

Drupal skills (17 items vs. 3 item)
Precision = 2 / 2 + 1 ≈ 0,66
Recall = 2 / 17 ≈ 0,12

Fields of study (4 items vs. 4 items)
Precision = 2 / 2 + 2 ≈ 0,50
Recall = 2 / 4 ≈ 0,50

General skills (10 items vs. 8 items)
Precision = 6 / 6 + 2 ≈ 0,75
Recall = 6 / 10 ≈ 0,60

Industry fields (1 item vs. 0 items)
Precision = 0 / 0 + 0 ≈ 0
Recall = 0 / 1 ≈ 0

IT skills (137 items vs. 16 items)
Precision = 12 / 12 + 4 ≈ 0,75
Recall = 12 / 137 ≈ 0,09

Occupational fields (31 items vs. 24 items)
Precision = 17 / 17 + 7 ≈ 0,71
Recall = 17 / 31 ≈ 0,55

DE2

Precision = 32 / 32 + 27 ≈ 0,54
Recall = 32 / 112 ≈ 0,29

Drupal skills (11 items vs. 6 items)
Precision = 1 / 1 + 5 ≈ 0,17
Recall = 1 / 11 ≈ 0,09

Fields of study (0 items vs. 1 item)
Precision = 0 / 0 + 1 ≈ 0
Recall = 0 / 0 ≈ 0

General skills (5 items vs. 5 items)
Precision = 2 / 2 + 3 ≈ 0,40
Recall = 2 / 5 ≈ 0,40

Industry fields (1 item vs. 0 items)
Precision = 0 / 0 + 0 ≈ 0
Recall = 0 / 1 ≈ 0

IT skills (83 items vs. 24 items)
Precision = 21 / 21 + 3 ≈ 0,88
Recall = 21 / 83 ≈ 0,25

Occupational fields (12 items vs. 23 items)
Precision = 8 / 8 + 15 ≈ 0,35
Recall = 8 / 12 ≈ 0,67

DE3
Precision = 39 / 39 + 18 ≈ 0,68
Recall = 31 / 105 ≈ 0,30

Drupal skills (8 items vs. 3 items)
Precision = 0 / 0 + 3 ≈ 0
Recall = 0 / 8 ≈ 0

Fields of study (8 items vs. 9 items)
Precision = 5 / 5 + 4 ≈ 0,56
Recall = 5 / 8 ≈ 0,63

General skills (3 items vs. 2 items)
Precision = 1 / 1 + 1 ≈ 0,50
Recall = 1 / 3 ≈ 0,33

Industry fields (1 item vs. 5 items)
Precision = 1 / 1 + 4 ≈ 0,20
Recall = 1 / 1 ≈ 1,00

IT skills (65 items vs. 14 items)
Precision = 13 / 13 + 1 ≈ 0,93
Recall = 13 / 65 ≈ 0,20

Occupational fields (20 items vs. 16 items)
Precision = 11 / 11 + 5 ≈ 0,69
Recall = 11 / 20 ≈ 0,55

K. Metrics Measurement 3-1.000

DE1

Precision = 102 / 102 + 30 ≈ 0,77
Recall = 102 / 200 ≈ 0,51

Drupal skills (17 items vs. 13 items)
Precision = 12 / 12 + 1 ≈ 0,92
Recall = 12 / 17 ≈ 0,71

Fields of study (4 items vs. 5 items)
Precision = 3 / 3 + 2 ≈ 0,60
Recall = 3 / 4 ≈ 0,75

General skills (10 items vs. 14 items)
Precision = 9 / 9 + 5 ≈ 0,64
Recall = 9 / 10 ≈ 0,90

Industry fields (1 item vs. 1 item)
Precision = 1 / 1 + 0 ≈ 1
Recall = 1 / 1 ≈ 1

IT skills (137 items vs. 65 items)
Precision = 52 / 52 + 12 ≈ 0,81
Recall = 52 / 137 ≈ 0,38

Occupational fields (31 items vs. 36 items)
Precision = 26 / 26 + 10 ≈ 0,72
Recall = 26 / 31 ≈ 0,84

DE2

Precision = 69 / 69 + 35 ≈ 0,66
Recall = 69 / 112 ≈ 0,62

Drupal skills (11 items vs. 12 items)
Precision = 6 / 6 + 6 ≈ 0,50
Recall = 6 / 11 ≈ 0,55

Fields of study (0 items vs. 1 items)
Precision = 0 / 0 + 1 ≈ 0
Recall = 0 / 0 ≈ 0

General skills (5 items vs. 6 items)
Precision = 4 / 4 + 2 ≈ 0,67
Recall = 4 / 5 ≈ 0,80

Industry fields (1 items vs. 1 items)
Precision = 1 / 1 + 0 ≈ 1,00
Recall = 1 / 1 ≈ 1,00

IT skills (83 items vs. 66 items)
Precision = 47 / 47 + 19 ≈ 0,71
Recall = 47 / 83 ≈ 0,57

Occupational fields (12 items vs. 18 items)
Precision = 11 / 11 + 7 ≈ 0,61
Recall = 11 / 12 ≈ 0,92

DE3
Precision = 56 / 56 + 28 ≈ 0,67
Recall = 56 / 105 ≈ 0,53

Drupal skills (8 items vs. 6 items)
Precision = 4 / 4 + 3 ≈ 0,57
Recall = 4 / 8 ≈ 0,50

Fields of study (8 items vs. 9 items)
Precision = 7 / 7 + 2 ≈ 0,78
Recall = 7 / 8 ≈ 0,88

General skills (3 items vs. 3 item)
Precision = 2 / 2 + 1 ≈ 0,67
Recall = 2 / 3 ≈ 0,67

Industry fields (1 item vs. 2 items)
Precision = 1 / 1 + 1 ≈ 0,50
Recall = 1 / 1 ≈ 1,00

IT skills (65 items vs. 41 items)
Precision = 27 / 27 + 14 ≈ 0,74
Recall = 27 / 65 ≈ 0,42

Occupational fields (20 items vs. 22 items)
Precision = 15 / 15 + 7 ≈ 0,68
Recall = 15 / 20 ≈ 0,75