An Infrastructure for Acquiring High Quality Semantic Metadata

Yuangui Lei, Marta Sabou, Vanessa Lopez, Jianhan Zhu,
Victoria Uren, and Enrico Motta

Knowledge Media Institute (KMi), The Open University, Milton Keynes
{y.lei, r.m.sabou, v.lopez, j.zhu, v.s.uren, e.motta}@open.ac.uk

Abstract. Because metadata that underlies semantic web applications is gathered from distributed and heterogeneous data sources, it is important to ensure its quality (i.e., reduce duplicates, spelling errors, ambiguities). However, current infrastructures that acquire and integrate semantic data have only marginally addressed the issue of metadata quality. In this paper we present our metadata acquisition infrastructure, ASDI, which pays special attention to ensuring that high quality metadata is derived. Central to the architecture of ASDI is a verification engine that relies on several semantic web tools to check the quality of the derived data. We tested our prototype in the context of building a semantic web portal for our lab, KMi. An experimental evaluation comparing the automatically extracted data against manual annotations indicates that the verification engine enhances the quality of the extracted semantic metadata.

1 Introduction

The promise of the semantic web is to automate several information gathering tasks on the web by making web data interpretable to software agents [1]. A condition for realizing this technology is the existence of *high quality* semantic metadata that would provide a machine understandable version of the web. By quality we mean that the semantic metadata should accurately capture the meaning of the data that it describes. For example, it should capture the meaning of each entity as intended in the context of its use (describe "jaguar" as a car or as a animal depending on its context). Further, a single semantic identifier should be attached to each entity even if this entity is referred to in the web page using different variants of its name or its name is misspelled. Also, metadata should be up to date when the described web page changes.

However, as previously debated in the literature [16], the characteristics of the web data hamper the acquisition of quality metadata. Besides its large scale, web data is usually distributed over multiple knowledge sources. These sources are heterogeneous in their level of formality, representation format, content and the quality of knowledge they contain. Integrating data from several of these sources often leads to errors that decrease the quality of the metadata. Also, the

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 230–244, 2006.

data on the web is changing continuously so the derived metadata has to be kept up to date.

Our overview of the most relevant infrastructures that acquire and aggregate semantic web metadata reveals that they offer limited or no support for verifying the quality of the derived metadata. In contrast with these, the system we present here, ASDI, provides several means to ensure the quality of the extracted data. First, it aims to reduce ambiguities by taking into account the context in which an entity is mentioned in order to determine its type. Second, it contains a verification engine that checks the validity of any derived metadata against a repository of trusted domain knowledge and against the information available on the web. Finally, since the whole acquisition process is automatic, it can be automatically run whenever new data becomes available, thus ensuring that the semantic metadata is always up to date.

The rest of the paper is organized as follows. We begin by describing the KMi context in which our prototype was designed and tested (section 2). Based on this description we discuss some of the tasks that an infrastructure needs to perform in order to ensure the quality of the derived semantic metadata. We then investigate how current semantic web infrastructures approach quality control for semantic metadata (section 3). In section 4 we present the ASDI infrastructure and detail its components that play a role in the quality control process. Thereafter, we describe an experimental evaluation of ASDI's validation functionality in section 5. Finally, we conclude our paper with a discussion of our results, the limitations of ASDI and future work in section 6.

2 Building the KMi Semantic Web Portal

We have designed and tested our infrastructure in the context of building a semantic web portal for KMi that would provide an integrated access to various aspects of the academic life of our lab[1]. By relying on semantic web technology the content of the portal is usable both by humans and software agents. While the KMi portal is a particular application, we believe that it provides the generic characteristics of a semantic web application scenario. In this section we briefly describe the particularities of the KMi context that are needed to understand the content of this paper (section 2.1). In the second part of this section, section 2.2, based on our experience with the KMi context, we extract and generalize a set of tasks that should be performed by any integration platform in order to ensure the quality of the extracted metadata.

2.1 The KMi Context

In the case of KMi the data relevant for building the semantic portal is spread in several different data sources such as departmental databases, knowledge bases and HTML pages. Information about people, technologies, projects and research areas is maintained in our departmental databases. Bibliographic data is stored

[1] `http://semanticweb.kmi.open.ac.uk`

in an internal knowledge base. In addition, KMi has an electronic newsletter[2], which now contains an archive of several hundreds of news items, describing events of significance to the KMi members.

Beside its heterogeneous nature, another important feature of the KMi domain data is that it continuously undergoes changes. The departmental databases change to reflect events such as additions of new projects. KMi-related events are reported in the newsletter and added to the news archive. Therefore, the semantic data that underlies the portal has to be often updated.

We decided to use and extend an existing ontology for representing the semantic metadata on which the portal realizes. We use the AKT reference ontology[3], which has been designed to support the AKT-2 demonstrator.We extended this ontology by adding some domain specific concepts, such as *kmi-planet-news-item*, *kmi-research-staff-member*, *kmi-support-staff-member*, etc.

2.2 Tasks for Insuring Metadata Quality

Based on the characteristics of the KMi semantic web portal context, we identify three generic tasks that are related to ensuring semantic metadata quality and which should be supported by any semantic web infrastructure. Note, however, that while this set of tasks is grounded in our practical experience, it is by no means exclusive. The generic tasks are:

A. Extract information from un-structured or semi-structured data sources in an automatic and adaptive manner. Useful knowledge is often distributed in several data sources which can be heterogeneous from the perspective of their level of structure or the used representation language. Methods have to be developed that extract the required data from each data source. It is important to employ *automatic* methods so that the process can be easily repeated periodically in order to keep the knowledge updated. Another important characteristic of the extraction mechanism is that it should be *adaptable* to the content of the sources that are explored. Being able to distinguish the type of an entity depending on its context is a pre-requisite for ensuring the quality of the semantic metadata. For example, we would expect from an adaptive information extraction tool to identify the different meanings that a given term can have on different web sites. For instance, "Magpie" is the name of a project in many web pages related to KMi, and should be identified as such. However, in other web sites it is more likely that it denotes a bird.

B. Ensure that the derived metadata is free of common errors. Since semantic data is typically gathered from different data sources which were authored by different people, it is often the case that it contains several errors, such as different identifiers that refer to the same entity, or instances whose meaning is not clear and needs to be disambiguated. Errors can be caused by data entry mistakes, by information extraction tools, or by the inconsistency and duplication entries of diverse data sources. We envision two major approaches to avoid these errors.

[2] `http://kmi.open.ac.uk/news`

[3] `http://kmi.open.ac.uk/projects/akt/ref-onto/`

First, one might attempt to tackle these errors before the data has been extracted. In particular, domain specific knowledge can help to avoid some problems, e.g., using lexicons to get rid of some domain specific noisy data. Such knowledge can be either supplied at design time in formats of transformation instructions or be generated automatically and incrementally according to the user's assessment on the performance of the system.

The second way to approach this problem is to clean the semantic data after it has been extracted. This approach requires mechanisms to correctly diagnose the problem at hand and then algorithms to correct each individual problem.

C. Update the semantic metadata as new information becomes available. Since the underlying data sources are likely to change, the whole data acquisition and verification process must be repeated so that the knowledge base is updated in an appropriate fashion.

In our prototype we provide support for all these quality insurance related tasks (as described in section 4). In the following section we overview a set of semantic web applications that rely on data acquisition and integration and describe how they approach the issue of quality control.

3 State of the Art

In this section we describe how existing approaches address semantic metadata quality control. We survey a representative sample of these approaches without performing an exhaustive study of this research direction. In particular, we focus on the approaches which address heterogeneous sources. We therefore leave out the approaches which either support the annotation of textual sources ([17]), the migration of data from structured sources ([2], [14]) and the creation of semantic web data from scratch ([10]).

The On-To-Knowledge project [15] provided one of the first suits of tools to be used for semantics based knowledge management. This tool suite not only supports semantic data acquisition from heterogeneous sources, but also supports ontology sharing, editing, versioning, and visualization. However, it does not provide explicit support to ensure the quality of the acquired semantic data.

The Semantic Content Organization and Retrieval Engine (SCORE) [13] is one of the semantic web based technologies, which has been commercialized. Quality control is addressed by i) enhancement rules which exploit the trusted knowledge to populate empty attribute values and ii) disambiguation mechanisms which make use of domain classification and the underlying trusted knowledge to address ambiguities.

The KIM platform [11] addresses the complete cycle of metadata creation, storage and semantic-based search. It pre-populates its ontology from several publicly available knowledge sources. This populated ontology supports the system to perform named entity recognition (NER) in a wide range of domains. The focus of this work is very much on scaling up and adapting NER to the needs of the semantic web.

The CS AKTive Space [12], the winner of the 2003 semantic web Challenge competition, gathers data automatically on a continuous basis. It relies on Armadillo [4] to achieve the task of automatic data acquisition. Quality control related issues such as the problem of duplicate entities are only weakly addressed (for example, by heuristics based methods or using manual input).

MuseumFinland [6] is the first and largest semantic web based portal that aggregates heterogeneous museum collections. In the MuseumFinland application possible errors are logged by the system for correction by a human user.

The Flink system [9], winner of the 2004 semantic web Challenge competition, is an infrastructure for extracting, aggregating and visualizing online social networks. The data is aggregated in an RDF(S) repository and a set of domain-specific inference rules are used to ensure its quality. In particular, identity reasoning (smushing) is performed to determine if different resources refer to the same individual (i.e., co-relation).

All the approaches mentioned above support acquiring semantic web data from heterogeneous sources in an *automatic* fashion. They typically exploit manually (e.g., On-To-Knowledge, SCORE, MuseumFinland) or semi-automatically (e.g., CS AKTive Space) constructed rules to define how metadata is to be extracted from domain specific structured or semi-structured sources.

Although they do provide comprehensive support for the acquisition activity in their specific problem domain, *the support for quality control is relatively weak*. Even though some co-relation (e.g., in CS AKTive Space and Flink) and disambiguation mechanisms (e.g., in SCORE) have been exploited, quality control has not been fully addressed. For example, the problems of duplicate or erroneous entities have not been addressed in any of the approaches mentioned above. Such problems may significantly de-crease the quality of the acquired semantic data.

4 The ASDI Infrastructure

In this section, we describe the infrastructure that we developed to build the semantic portal at KMi. An important characteristic of ASDI is that, in comparison with the approaches we have described in section 3, it addresses the quality control issue. We first present an overview of the ASDI infrastructure. Then we detail two of the most important layers of the infrastructure that ensure adaptive data extraction and the quality check of this extracted data.

4.1 An Overview

Figure 1 shows the four layered architecture of ASDI, which contains:

A Source Data Layer contains the collection of all available data sources such as semi-structured textual documents (e.g. web pages) or structured data in the form of XML feeds, databases, and knowledge bases.

An Extraction Layer is responsible with the generation of semantic data from the source data layer. It comprises an *automatic and adaptive information extraction tool*, which marks-up textual sources, *a semantic transformation*

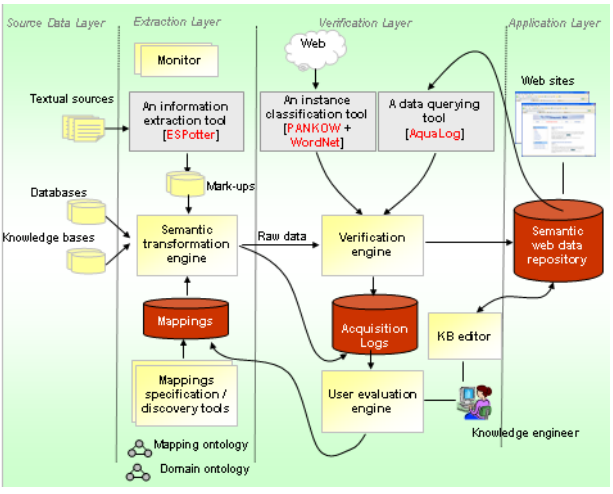

Fig. 1. An overview of the ASDI infrastructure

engine, which converts data from source representations into the specified domain ontology according to the transformation instructions specified in *a mapping ontology*, and a set of *mappings specification/discovery* tools, which support the construction of transformation instructions. The output of this layer consists of i) raw semantic data entries and ii) logs of acquisition operations which describe the provenance of data entries.

A Verification Layer checks the quality of the previously generated semantic data entries. The core component is the *verification engine*, which makes use of a number of semantic web tools to achieve high quality data. While the verification engine is completely automatic, we allow users to inspect the changes made to the semantic data through a *user evaluation engine*. This engine assists users to assess the performance of the system and generates transformation rules according to the feedback given by them. *A KB editor* is also included in this layer to allow users (i.e., knowledge engineers) to inspect the final results and modify them whenever necessary.

An Application Layer sums up all the applications that use the acquired and verified semantic data (stored in the semantic web data repository).

4.2 The Extraction Layer

The role of extraction layer is to acquire data from heterogeneous sources and convert them to semantic web data objects equipped with rich semantic relations.

To address the issue of *adaptive information extraction*, we use ESpotter [18], a named entity recognition (NER) system that provides an adaptive service. ESpotter accepts the URL of a textual document as input and produces a list of the named entities mentioned in that text. The adaptability is realized by

means of domain ontologies and a repository of lexicon entries. For example, in the context of the KMi domain, ESpotter is able to mark the term "Magpie" as a project, while in other domains it marks it as a bird.

For the purpose of converting the extracted data to the specified domain ontology (i.e., the ontology that should be used by the final applications), an instance mapping ontology has been developed, which supports i) representation independent semantic transformations, ii) the generation of rich semantic relations along with the generation of semantic data entries, and iii) the specification of domain specific knowledge (i.e. lexicons). This lexicons are later used by the verification process. Using this ontology (see details in [7]) one can define a set of mappings between the schema of the original data sources and the final domain ontology. A semantic transformation engine is prototyped, which accepts structured sources and transformation instructions as input and produces semantic data entries. Since writing these mapping rules manually is a considerable effort, we are currently focusing on semi-automating this process.

To ensure that the acquired data stays *up to date*, a set of monitoring services detect and capture changes made in the underlying data sources and initiate the whole extraction process again. This ensures a sustainable and maintenance-free operation of the overall architecture.

4.3 The Verification Layer

The role of verification layer is to identify problems of the extracted data entries and to resolve them properly. This layer relies on two components. First, an automatic verification engine employs several tools to verify the extracted data. Second, a user evaluation tool allows a knowledge engineer to evaluate and fine-tune the verification process of the engine. We describe both components.

The Verification Engine. The goal of the verification engine is to check that each entity has been extracted correctly by the extraction layer. For example, it checks that each entity has been correctly associated with a concept and in the cases when the type of the entity is dubious it performs disambiguation. This engine also makes sure that a newly derived entity is not a duplicate for an already existing entity. The verification process consists of three increasingly complex steps as depicted in figure 2. These steps employ several semantic web tools and a set of resources to complete their tasks.

Step1: Checking the internal lexicon library. In the first step, *a lexicon library*, which maintains domain specific lexicons, is checked. If no match is found there, the verification process continues.

Step2: Querying the semantic web data repository. The second step uses an ontology-based QA tool, AquaLog [8], to query the already acquired semantic web data (which is assumed to be correct, i.e. trusted) and to solve obvious typos and minor errors in the data. This step contains a disambiguation mechanism, whose role is to de-reference ambiguous entities (e.g., whether the term "star wars" refers to the Lucas' movie or President Reagan's military programme).

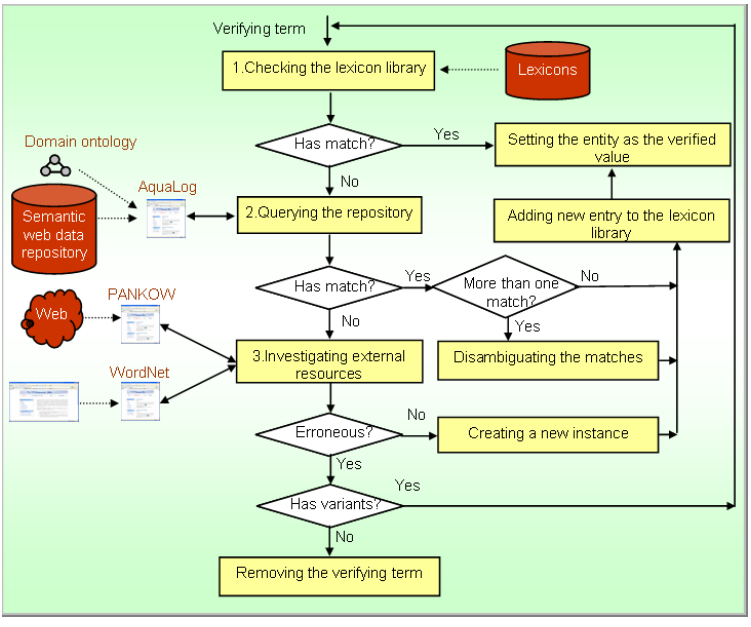

Fig. 2. The overall algorithm of the data verification engine

Step3: Investigating external resources. If the second step fails, the third step relies on investigating external resources such as the web. An instance classification tool is developed, which makes use of PANKOW [3] and WordNet [5], to determine the appropriate classification of the verified entity.

We will now detail all these three steps.

Step1: Checking the lexicon library. The lexicon library maintains domain specific lexicons (e.g., abbreviations) and records the mappings between strings and instance names. One lexicon mapping example in the KMi semantic web portal is that the string "ou" corresponds to the *the-open-university* entity. The verification engine will consider any appearances of this abbreviation as referring to the corresponding entity.

The lexicon library is initialized by lexicons specified through the mapping instruction and expands as the verification process goes on. By using the lexicon library, the verification engine is able to i) exploit domain specific lexicons to avoid domain specific noisy data and ii) avoid repeating the verification of the same entity thus making the process more efficient. However, there is a risk of mis-identifying different entities in different contexts which share the same name. For example, in one context the name *Victoria* may refer to the entity *Victoria-Uren* and in other contexts it may not.

If no match can be found in this step, the engine proceeds to the next step. Otherwise, the verification process ends.

Step2: Querying the semantic web data repository. The semantic web data repository stores both the semantic entries extracted from trusted knowledge sources and the final results of the verification engine. In this second step, the verification engine relies on AquaLog to find possible matches for the verified entity in the semantic web data repository.

AquaLog is a fully implemented ontology-driven QA system, which takes an ontology and a natural language query as an input and returns answers drawn from semantic data compliant with the input ontology. In the context of the ASDI infrastructure, we exploit AquaLog's string matching algorithms to deal with obvious typos and minor errors in the data. For example, in a news story a student called *Dnyanesh Rajapathak* is mentioned. The student name is however misspelled as it should be *Dnyanesh Rajpathak*. While the student name is successfully marked up and integrated, the misspelling problem is carried into the generated data as well. With support from AquaLog, this problem is corrected by the verification engine. AquaLog queries the knowledge base for all entities of type *Student* and discovers that the difference between the name of the verified instance (i.e., *Dnyanesh Rajapathak*) and that of one of the students (i.e, *Dnyanesh Rajpathak*) is minimal (they only differ by one letter). Therefore, AquaLog returns the correct name of the student as a result of the verification. Note that this mechanism has its downfall when similarly named entities denote different real life objects.

If there is a single match, the verification process ends. However, when more matches exist, contextual information is exploited to address the ambiguities.

In the disambiguation step, the verification engine exploits i) other entities appearing in the same news story and ii) the semantic relations contained in the semantic web data repository as the contextual information. To illustrate the mechanism by means of a concrete example, suppose that in the context of the KMi semantic web portal, ESpotter marks *Victoria* as a person entity in a news story. When using AquaLog to find matches for the entity, the verification engine gets two matches: *Victoria-Uren* and *Victoria-Wilson*. To decide which one is the appropriate match, the verification engine looks up other entities referenced in the same story and checks whether they have any relation with any of the matches in the knowledge base. In this example, the *AKT* project is mentioned in the same story, and the match *Victoria-Uren* has a relation (i.e., *has-project-member*) with the project. Hence, the appropriate match is more likely to be *Victoria-Uren* than *Victoria-Wilson*.

Step3: Investigating external resources. When a match cannot be found in the internal resources, the entity can be:

1) partially correct, e.g., the entity *IEEE-conference* is classified as an *Organization*,

2) correct but new to the system, e.g., the entity *IBM*, is correctly classified as an *Organization*, but the local knowledge sources do not contain this information so it cannot be validated,

3) miss-classified, e.g., *sun-microsystems* is classified as a *Person*, and

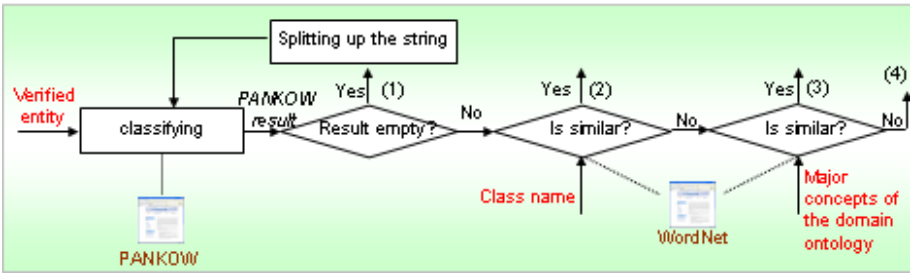

Fig. 3. The algorithm of the instance classification tool

4) erroneous, which does not make any sense and should be removed, e.g. the entity *today* classified as a *Person*.

The task of this step is to find out in which category the verified entity falls into. For this purpose, a classification tool is developed, which uses PANKOW and WordNet to support the classification of unknown terms. Figure 3 shows the algorithm of the instance classification tool. We describe each step of this process and provide as an example the process of verifying the *IBM* entity which was classified by ESpotter as an *Organization*.

Step 3.1. The PANKOW service is used to classify the string *IBM*. PANKOW employs an unsupervised, pattern-based approach on web data to categorize the string and produces a set of possible classifications along with ranking values. As shown in figure 3, if PANKOW cannot get any result, the term is treated as erroneous but still can be partially correct. Thus, its variants are investigated one by one until classifications can be drawn. If PANKOW returns any results, the classifications with the highest ranking are picked up. In this example, the term "company" has the highest ranking.

Step 3.2. Next the algorithm uses WordNet to compare the similarity between the type of the verified entity as proposed by the extraction layer (i.e., "organization") and an alternative type for the entity as returned by PANKOW (i.e.,"company"). If they are similar (which is the case of the example), it is concluded that the verified entity is classified correctly (its derived type is similar to that which is most frequently used on the web) but it was not added yet into the trusted knowledge base. Thus, a new instance (*IBM* of type *Organization*) needs to be created and added to the repository.

If the two compared terms are not similar, other major concepts of the domain ontology are compared to the web-endorsed type (i.e.,"company") in an effort to find a classification for the entity. If one concept is found similar to the web-endorsed type, it is concluded that the verified entity was wrongly classified by the extraction layer. The verification engine then associates the verified entity with the correct concept and places it back to the step 2 (which is to seek matches from the semantic web data repository). Otherwise, it can be safely concluded that the verified entity is erroneous.

The User Evaluation Engine. While the verification engine is completely automatic, it is often the case that the knowledge engineers wish to verify the correctness of this process and to adjust it. For this we are currently developing an user evaluation engine. The user evaluation engine accepts acquisition logs as input and produces semantic transformation rules for improving the performance of the system. As shown in figure 1, the transformation rules are represented in terms of the instance mapping ontology. The user evaluation engine comprises a tool which assists users to browse the provenance of semantic data entries and the verification operations carried out and allows users to give feedback. Another tool generates transformation rules according to user's feedback.

5 Evaluation

The KMi semantic web portal has been running for several months generating and maintaining semantic data from the underlying sources of the KMi web site in an automated way. In this section, we describe an experimental evaluation of the quality control mechanism provided by ASDI. We first describe the evaluation setup and the metrics we use (section 5.1). We then discuss the results of the evaluation (section 5.2).

5.1 Evaluation Setup

We randomly chose 36 news stories from the KMi news archive and then asked several KMi researchers to manually mark them up in terms of *person, organization* and *projects*. Because the annotators have a good knowledge of the KMi domain we consider the result of the annotation as a Gold Standard. We used ASDI to produce semantic data from these news stories and compared the automatically extracted data to the manual annotations. To illustrate the important role of the verification engine, the performance of ESpotter is introduced in the comparison, which shows the quality of the extracted data before and after the verification process.

We assess the results in terms of *recall, precision* and *f-measure*, where recall is the proportion of all possible correct annotations that were found by the system with respect to the ones that can be in principle extracted from the source text, precision is the proportion of the extracted annotations that were found to be correct, and f-measure evaluates the overall performance by treating recall and precision equally.

5.2 Evaluation Results

Table 1 shows the recall rates of ESpotter and ASDI. The manual annotation of the news stories identified 92 people, 74 organizations, and 21 projects. Compared to these manual annotations, ESpotter failed to identify 17 people, 16 organizations, and 5 projects, thus reaching an overall recall of 0.798. ASDI reached a slightly lower recall (0.775) as it missed 21 people, 16 organizations, and 5 projects in comparison with the manual annotation. The major reason for

Table 1. Recall of ESpotter and ASDI

Type	People	Organizations	Projects	Total
Manual annotations	92	74	21	187
ESpotter failures	17	16	5	38
ESpotter Recall	**0.815**	**0.783**	**0.761**	**0.798**
ASDI failures	21	16	5	42
ASDI Recall	**0.771**	**0.783**	**0.761**	**0.775**

this lower recall is that the instance classification tool sometimes has problems in providing the appropriate classification. In some cases, PANKOW cannot find enough evidence to produce a satisfactory classification. For example, the classification of the person named "Marco Ramoni" returns an empty result. As a consequence, the verification engine loses one correct entry.

Table 2 shows the precision of ESpotter and ASDI. ESpotter discovered 87 people, 96 organizations, and 19 projects when working on the sample stories. Among them, 11 people and 32 organizations are not correct. This results in an overall precision of 0.787. On the other hand, ASDI obtained 86 person entities, 74 organization entities, and 19 project entities. Among them, 12 person entities and 4 organization entities are wrong. Hence, the overall precision of ASDI is 0.911. Note that the ASDI application improves the precision rate significantly. One major problem of ESpotter is the significant amount of redundant entries. For example, values like "open-university" and "ou" are often treated as the same entity. The verification engine gets rid of this problem by defining lexicons and relying on AquaLog to spot similar entities.

ESpotter derives several inaccurate classifications, which lead to a number of erroneous values, such as considering "IBM global education", or "the 2004 IEEE" *Organization* type entities. These values are successfully corrected during the verification process by looking up their variants. Finally, some erroneous values produced by ESpotter are kept out of the target knowledge base, as they do not make any sense. Examples are"workshop chair", "center", etc.

To give an overall insight in the performance of ASDI versus that of ESpotter we computed the F-measure of these systems by giving equal importance to both Recall and Precision. The values (listed in table 3) show that ASDI performs better than ESpotter. This means that the quality of the extracted data is improved by our verification engine.

Table 2. Precision of ESpotter and ASDI

Type	People	Organizations	Projects	Total
ESpotter discovered	87	96	19	202
ESpotter spurious	11	32	0	43
ESpotter Precision	**0.873**	**0.667**	**1**	**0.787**
ASDI discovered	86	74	19	179
ASDI spurious	12	4	0	16
ASDI Precision	**0.860**	**0.946**	**1**	**0.911**

Table 3. F-Measure of ESpotter and ASDI

F-measure	People	Organizations	Projects	Total
ESpotter	**0.843**	**0.72**	**0.864**	**0.792**
ASDI	**0.813**	**0.856**	**0.864**	**0.837**

6 Discussion

The core observation that underlies this paper is that, in the case of semantic web applications that rely on acquiring and combining semantic web data from several data sources, it is crucial to ensure that this semantic data has a high quality. By quality here we mean that the semantic data contains no duplicates, no errors and that the semantic descriptions correctly reflect the nature of the described entities. Our survey of a set of semantic web applications that gather data from several sources shows that little or no attention is paid to ensure the quality of the extracted data. In most cases heuristics based algorithms are used to ensure referential integrity. In contrast with these efforts, our semantic web infrastructure, ASDI, focuses on ensuring the quality of the extracted metadata.

Our evaluation of the quality verification module shows that it improved the performance of the bare extraction layer. ASDI outperforms ESpotter by achieving 91% precision and 77% recall. In the context of the KMi portal precision is more important than recall - erroneous results annoy user more than missing information. We plan to improve the recall rate by introducing additional information extraction engines to work in parallel with ESpotter. Such a redundancy is expected to substantially improve recall. Another future work we consider is to evaluate the added value of each component of the verification engine, i.e., determine the improvements brought by each individual component.

An interesting feature of ASDI is that it relies on a set of tools that were developed in the context of the semantic web. These are: the ESpotter adaptive NER system, the PANKOW annotation service and an ontology based question answering tool AquaLog. This is a novelty because many similar tools often adapt existing techniques. For example, the KIM platform adapts off the shelf NER techniques to the needs of the semantic web. By using these tools we show that the semantic web reached a development stage where different tools can be safely combined to produce new, complex functionalities. Another benefit we derived by using these domain independent tools is that our verification engine is highly portable. We are currently making it available as a web service.

Once set up, ASDI can run without any human intervention. This is thanks to the monitors that identify any updates in the underlying data structures and re-initiate the semantic data creation process for the new data.

We are, however, aware of *a number of limitations* associated with ASDI. For example, the manual specification of mappings in the process of setting up the ASDI application makes the approach heavy to launch. We currently address this issue by investing the use of automatic or semi-automatic mapping algorithms. A semi-automatic mapping would allow our tool to be portable across several different application domains.

Another limitation is related to AquaLog's lack of providing a degree of similarity between an entity and its match. For example, when querying for the entity *university-of-London*, AquaLog returns a number of matches which are university entities but it does not specify how similar they are to the entity that is verified. This has caused a number of problems in the KMi portal scenario.

Our general goal for the future is to make our work more generic by providing a formal definition of what semantic data quality is and transforming our prototype into a generic framework for verifying semantic data.

Acknowledgements. This work was funded by the Advanced Knowledge Technologies Interdisciplinary Research Collaboration (IRC), and the Knowledge Sharing and Reuse across Media (X-Media) project. AKT is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. X-Media is sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC Grant IST-FF6-26978.

References

1. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34 – 43, May 2001.
2. C. Bizer. D2R MAP - A Database to RDF Mapping Language. In *Proceedings of the 12th International World Wide Web Conference*, Budapest, 2003.
3. P. Cimiano, S. Handschuh, and S. Staab. Towards the Self-Annotating Web. In S. Feldman, M. Uretsky, M. Najork, and C. Wills, editors, *Proceedings of the 13th International World Wide Web Conference*, pages 462 – 471, 2004.
4. A. Dingli, F. Ciravegna, and Y. Wilks. Automatic Semantic Annotation using Unsupervised Information Extraction and Integration. In *Proceedings of the KCAP-2003 Workshop on Knowledge Markup and Semantic Annotation*, 2003.
5. C. Fellbaum. *WORDNET: An Electronic Lexical Database*. MIT Press, 1998.
6. E. Hyvonen, E. Makela, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. MuseumFinland – Finnish Museums on the Semantic Web. *Journal of Web Semantics*, 3(2), 2005.
7. Y. Lei. An Instance Mapping Ontology for the Semantic Web. In *Proceedings of the Third International Conference on Knowledge Capture*, Banff, Canada, 2005.
8. V. Lopez, M. Pasin, and E. Motta. AquaLog: An Ontology-portable Question Answering System for the Semantic Web. In *Proceedings of ESWC*, 2005.
9. P. Mika. Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks. *Journal of Web Semantics*, 3(2), 2005.
10. N.F. Noy, M. Sintek, S. Decker, M. Crubezy, R.W. Fergerson, and M.A. Musen. Creating Semantic Web Contents with Protege-2000. *IEEE Intelligent Systems*, 2(16):60 – 71, 2001.
11. B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. KIM - Semantic Annotation Platform. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The SemanticWeb - ISWC 2003, Second International Semantic Web Conference, Proceedings*, volume 2870 of *LNCS*. Springer-Verlag, 2003.
12. M.C. Schraefel, N.R. Shadbolt, N. Gibbins, H. Glaser, and S. Harris. CS AKTive Space: Representing Computer Science in the Semantic Web. In *Proceedings of the 13th International World Wide Web Conference*, 2004.

13. A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke. Semantic Content Management for Enterprises and the Web. *IEEE Internet Computing*, July/August 2002.
14. L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the semantic web. In *Proceedings of the 17th ACM symposium on applied computing (SAC)*, pages 1100 – 1107. ACM Press, 2002.
15. Y. Sure, H. Akkermans, J. Broekstra, J. Davies, Y. Ding, A. Duke, R. Engels, D. Fensel, I. Horrocks, V. Iosif, A. Kampman, A. Kiryakov, M. Klein, Th. Lau, D. Ognyanov, U. Reimer, K. Simov, R. Studer, J. van der Meer, and F van Harmelen. On-To-Knowledge: Semantic Web Enabled Knowledge Management. In N. Zhong, J. Liu, and Y. Yao, editors, *Web Intelligence*. Springer-Verlag, 2003.
16. F. van Harmelen. How the Semantic Web will change KR: challenges and opportunities for a new research agenda. *The Knowledge Engineering Review*, 17(1):93 – 96, 2002.
17. M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In *Proceedings of the 13th International Conference on Knowledge Engineering and Management (EKAW)*, Spain, 2002.
18. J. Zhu, V. Uren, and E. Motta. ESpotter: Adaptive Named Entity Recognition for Web Browsing. In *Proceedings of the Professional Knowledge Management Conference*, 2004.