CA Manager: a Framework for Creating Customised Workflows for Ontology Population and Semantic Annotation

Florence Amardeilh, Danica Damljanovic, Kalina Bontcheva

CA Manager: a Framework for Creating Customised Workflows for Ontology Population and Semantic Annotation

Florence Amardeilh
Mondeca
3, cite Nollez
75018 Paris, France
florence.amardeilh
@mondeca.com

Danica Damljanovic
Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello
Sheffield S1 4DP, UK
D.Damljanovic
@dcs.shef.ac.uk

Kalina Bontcheva
Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello
Sheffield S1 4DP, UK
K.Bontcheva
@dcs.shef.ac.uk

ABSTRACT

The aim of semantic annotation is to bridge the large gap between structured knowledge and the large volumes of unstructured text data that companies and people need to deal with daily. Alas, the process is very labourious and error-prone, even when performed semi- or fully automatically. Although widely researched, the two key steps in this process – semantic annotation and ontology population – still hold outstanding challenges. While there are numerous tools in existence, many lack compliance with recent standards, but more importantly, lack the flexibility to customise the annotation workflow. In this paper, we present the Content Augmentation Manager Framework, which bridges the gap between information extraction tools and semantic repositories by allowing easy plugin of various types of components. It is also capable of controlling the process of semantic annotation and ontology population by means of consolidation algorithms.

General Terms

semantic annotation, semantic repositories, ontologies, content augmentation

Keywords

knowledge acquisition, ontology population, software artefacts, key concept identification.

1. INTRODUCTION

Gartner predicted in 2002[1] that for the next decade more than 95% of human-to-computer information input will involve textual language. They also predict that by 2012 taxonomic and hierarchical knowledge mapping and indexing will be prevalent in almost all information-rich applications. There is a tension here: between the increasingly rich ontology-based, semantic models on the one hand, and the continuing prevalence of human language materials on the other. This process may be characterised as the dynamic creation of interrelationships between *ontologies* and unstructured and semi-structured textual content in a bidirectional manner.

However, transforming huge amount of unstructured text into the semantically interlinked knowledge space is a big challenge. Two key parts of this process are: 1) semantic annotation, and 2) ontology population. We define semantic annotation as a formal representation of content, expressed using concepts, relations and instances as described in an ontology, and linked to the original resource. Ontology instances are usually stored in a knowledge base independently from the annotated resource, as in the case of KIM [11] or ITM [3]. The process of adding new class instances or property values to a knowledge base is called *knowledge acquisition* or *ontology population*. As such, the *ontology population* looks at semantic annotation as means for data-driven enrichment of an existing knowledge base.

Although widely researched, both *semantic annotation* and *ontology population* tasks still remain a big challenge, as the role of human annotators remains paramount. Consequently, high-quality automation is one of the very important requirements in order to ease the knowledge acquisition bottleneck, particularly for annotating large collections of legacy documents [12]. While there are numerous tools in existence, many lack compliance with

SAAKM '09 Berkeley, California USA
ACM

[1] `http://www3.gartner.com/DisplayDocument?id=379859`

recent standards, but more importantly, lack the flexibility to customise the annotation workflow. Moreover, the workflow needs to also accomodate human, as well as automatic annotators.

In this paper, we present the Content Augmentation Manager Framework (CA Manager) which is capable of performing and controlling the process of semantic annotation and ontology population by means of consolidation algorithms. This framework supports ontology population from text (semi)automatically, by allowing easy plugin of various types of components including information extraction tools, customised domain ontologies, and diverse semantic repositories. In other words, this framework helps to bridge the gap between information extraction tools and the semantic repositories which are used to store the collected knowledge.

This paper is structured as follows. In Section 2 we present CA Manager. Evaluation results are presented in Section 3. Similar approaches are discussed in Section 4. Finally, we draw our conclusions and plan for future work in Section 5.

2. THE CONTENT AUGMENTATION MANAGER FRAMEWORK

The core philosophy of the CA Manager is to bridge the gap between the content augmentation tools, and the semantic repository tools. It is conceived as a middleware, meaning that it does not have the responsibility neither of the information extraction task by itself, nor of the knowledge storage, but it is flexible enough to adapt to any domain ontology, various content augmentation tools, semantic repositories and workflow requirements. It is, amongst other things, capable of controlling the quality and the validity of information extraction results against an ontology, matching them against existing resources (the application's knowledge base or repositories from the linking open data initiative for instance), and enriching them. To achieve that goal, the CA Manager relies on the recommendations formulated by the Semantic Web community (RDF/OWL languages, Service Oriented Architecture) combined with a UIMA-based infrastructure which has been enriched and customized to address the specific needs of semantic annotation and ontology population tasks.

The UIMA (Unstructured Information Management Architecture) framework aims at providing a development platform for systems that analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user[2]. We adopted UIMA as the foundation of the internal architecture of the CA Manager, due to its ease of integration and composition of internal or external modules, and more importantly its wide acceptance by the Information Extraction (IE) community. However, although UIMA provides the building blocks to develop knowledge acquisition applications based on text-mining components, it does not give any guidelines as to which steps should be arranged in which order. Moreover, none of the existing components in the pipeline addresses the issue of controlling the quality and validity of the generated annotations/instances. Moreover, its Common Analysis Structure (CAS) defines a high-level annotation schema but it needs to be redefined for each new application need. Lastly, it uses a proprietary way to expose webservices (the Vinci IBM protocol), that makes it even more complex, as it is not reusing open Semantic Web standards.

Therefore, we implemented the CA Manager in order to develop a flexible architecture based on a combination of several UIMA Analysis Engines. UIMA provides an Eclipse plugin Eclipse facilitating the definition and customisation of each Analysis Engines (stored in an XML file) needed in the target application and their ordering as a workflow. Therefore, each step of the annotation workflow is a component that can be plug in or out according to the objectives of the final application. At the same time, we aimed at improving the UIMA infrastructure with the systematic use of Semantic Web standards We defined an RDF-based annotation schema dedicated to ontology population and semantic annotation tasks, composed of entities, properties, annotations and offsets. This annotation schema is produced after applying the first Analysis Engine and then enriched and controled by the following ones during the whole duration of the workflow process. We also provide a distributed service-oriented architecture relying on languages and protocols defined for the Semantic Web, especially for easing the integration with external components through the use of opened web services.

Furthermore, the CA Manager proposes a default workflow composed of a list of logical steps dedicated to semantic annotation and ontology population, see Figure 1:

- *Extracting* the valuable knowledge;
- *Consolidating* knowledge;
- *Storing*

These three phases allow the connection with information extraction tools (being defined as UIMA analysis engines plugin or providing a web service), semantic repositories and domain ontologies or corpus.

2.1 Extracting knowledge from text

The information Extraction component is made of two steps, *split* (optional) and *extract*. Split divides the in-

[2]UIMA website: `http://www.alphaworks.ibm.com/tech/uima`

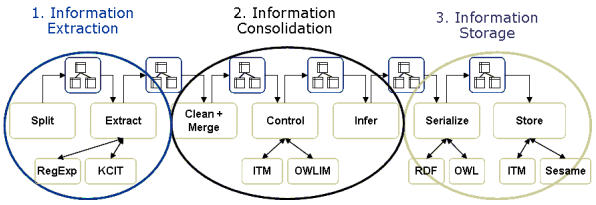

Figure 1: Specialized UIMA processing pipeline

put into multiple parts, for example, a corpus into a set of documents, or a single document into well-identified sections. The second, *extract*, step calls an Information Extraction tool, in order to process the text and find occurrences of the entities in the documents. To comply with the TAO project use cases, we developed the Key Concept Identification Tool (KCIT) to annotate software artefacts, based on the GATE [5] framework for semantic annotation and text mining.

2.1.1 KCIT

KCIT [7] automatically retrieves key concepts from legacy documents with regards to the domain ontology. These annotations are created based on the pre-condition that a specific part of a document is referring to a particular ontology resource if the *lemmas* of the two match. By matching lemmas, we ensure that all morphological inflections of the relevant terms will be matched. The KCIT process can be broken down into several steps:

- Building a list of relevant terms. Given an ontology, lexicalisations of all ontological resources (classes, instances, properties, property values) are lemmatised and added to a gazetteer list.
- Annotating the legacy content. The legacy content is first lemmatised, with a morphological analyser. It is then matched against the gazetteer list created in the previous step.
- Resolving conflicts. This step includes filtering annotations and solving ambiguity problems such as removing redundant annotations.

To build a list of relevant terms dynamically we first extract a list of the ontology resource names (i.e., fragment identifiers) and their assigned property values (e.g., label and datatype property values). Each item from the list is further processed, based on some heuristic rules derived from different ontology designs. Although there is no need to customise KCIT when using it with different ontologies, customisation might yield better results. By default, KCIT apply several rules such as replacing dashes and underline characters with spaces and also splitting camelCase words. In addition it applies some heuristic rules which are optional, as their usage depends on the lexicalisations available in the ontology. KCIT remains domain-independent, as all domain-specific issues are implemented through parameters. Typically, the output of a content augmentation tool is in some kind of XML format and is not connected directly to the ontology. In such cases, it needs to be transitioned to the generic Annotation Schema (AS) of the CA Manager through the use of a mapping. However, as KCIT produces ontology-based annotations, such a mapping is not necessary in this case, because the annotations are directly grounded in the ontology model.

2.2 Consolidating annotations and knowledge with an ontology repository

We studied in [2] the various possible cases of instances and annotation creation and identified two axes of consolidation:

1. the first axis defines the ontological element concerned, i.e. an instance of a class, a property value or a semantic annotation
2. the second axis defines the constraints to be checked, i.e. non redundancy, the domain and range restrictions and the element's cardinality

Each of the CA Manager consolidation algorithms takes into account these two axis. In the Information Consolidation component, they are performed through three steps, *merge*, *control* and optionally *infer*.

The first step, *merge*, eliminates the duplicates within the CAS (same entity or annotations occurring more than once in the CAS) and queries the semantic repository in order to retrieve the corresponding URI of the concerned entity or annotation if not present. These queries can be simple (class + string label) or multicriteria (class + set of required properties that identify unambiguously an entity in the repository). For instance, a *person* can be queried by its *name*. However, in cases of homonymy, looking at the person name is clearly not enough and one might want to query on particular properties such as the *date of birth* that can better discriminate several instances of persons sharing the same name. This multicriteraai search is built from the restricted properties where cardinality is minimum 1; we call these properties identifier properties as they are required to identify and define an instance

of a concept. In case of law case reports, the identifer properties are the court, the location of the court, the date of the decision and the decision itself. If it is not possible to disambiguate between two instances of the semantic repository because for example no identifier properties have been extracted and annotated as such, then the new entity or annotation is tagged with a metadata "invalid".

The *control* step verifies that the extracted entity or annotation is valid against the ontology model. This implies controlling domains and ranges, cardinalities, date formats and the temporal information, the number formats and metric systems, etc. For instance, if in the preceding step the extracted entity was merged with an existing instance, the algorithms look at the properties of the extracted entity: are these properties types authorized for the entity's class? do these properties already exist on the merged instance? do they have the same values or different values? then how do we know which value is the right one, especially when dealing with thesaurus values such as geographical locations, or with time value such as dates? The algorithms try to automatically resolve these issues and when not possible, they also mark the new entity or annotation with the "invalid" metadata. All invalid statements are stored in the semantic repository on the server so that they can be retrieved and presented to the end-user for manual validation, if required by the target application.

The last step of this component, *infer*, is optional. It applies inference rules through a reasoning engine in order to discover new entities or new relations between them and also to control the overall coherence and quality of the semantic repository.

2.3 Storing annotations and knowledge in repositories

The Information Storage component has two steps, *serialise* and *store*. The *serialization* step parses the enriched and consolidated annotation schema in order to generate an output in the requested applicatoin format (XML, RDF, OWL, NewsML, CityGML, etc.). The second step, *storage*, is optional as it depends if the application directly digests the serialised format or stores the results in a knowledge store (such as ITM) and/or in a dedicated annotation server (such as Sesame).

The CA Manager framework is open-source and available on SourceForge.net[3]. The release includes a documentation describing how to install the CA Manager, how to configure workflows, how to develop new plugins as for connecting a new information extraction engine for example, and so on. Moreover, the above pipeline is also exposed as a web service, and a testing web client application is available from `http://client2.mondeca.com/scan/`.

3. EVALUATION

Within the TAO project[4] we implemented different workflows (a combination of different ontologies, semantic annotation tools, semantic repositories and corpora) for evaluating the *flexibility* and the *scalability* of the CA Manager framework. One of these was composed of loading the upper-level PROTON Ontology[5] in the Sesame RDF repository and applying simple regular expressions on the corpus. The other two were based on the GATE Ontology developed within the TAO project, both using KCIT as the content augmentation system but one semantic repository being ITM[6] and the second one being Sesame RDF repository. Moreover, we have challenged flexibility of CA Manager beyond the TAO project by trialing it in different environments with various information extraction tools as well. For more details, see [6].

The GATE domain ontology[7] contains knowledge about GATE components, modules, plugins and language processing resources and was used by KCIT to annotate software artefacts about GATE software[8]. In the rest of this section, we present evaluation of the two main steps described above: *information extraction* and *consolidation*, based on the described workflow configuration, using GATE ontology, KCIT annotation tool and ITM semantic repository.

We have first evaluated the KCIT information extraction tool, as consolidation is entirely dependent on the quality of the KCIT output. As KCIT is primarily a semantic annotation tool, we have measured its performance using standard information extraction measures, namely precision and recall. We collected 64 GATE software artefacts of various types, namely source code, source documentation, GATE user manual, publications and forum posts from the GATE mailing list. Then, we organised an annotation exercise with 12 human subjects to whom we gave a manual on how to proceed in validating annotations and creating a gold standard. Each document has been assigned to two participants so that we can calculate inter-annotator agreement.

3.1 Inter-annotator agreement

Inter-annotator agreement is used to establish the upper bounds on performance of information extraction tools such is KCIT. Table 1 shows precision, recall and F-Measure values, based on the results of this experiment. For computing these measures between two annotation

[3]CA Manager release: `http://sourceforge.net/projects/scan-ca-manager`

[4]TAO website: `http://www.tao-project.eu`

[5]PROTON website : `http://proton.semanticweb.org`

[6]ITM website: `http://mondeca.com/index.php/en/intelligent_topic_manager`

[7]`http://gate.ac.uk/ns/gate-kb`

[8]GATE website: `http://gate.ac.uk`

sets, one can use one annotation set as gold standard and another set as system output. One can switch the roles of the two annotation sets. The Precision and Recall in the former case become Recall and Precision in the latter, respectively. But the F1 remains the same in both cases.

corpus	precision	recall	f1
user manuals	0.90909094	1	0.95238096
publications	0.90909094	1	0.95238096
Web pages	1	0.8	0.88888896
source code	0.71428573	0.625	0.6666667
Java doc files	0.64	0.9411765	0.7619047
forum posts	1	1	1
	0.862077935	0.89436275	0.87037038

Table 1: Inter-annotator agreement

Disagreements between annotators occurred for all document types excluding forum posts, with the source code being most problematic. In some cases, annotators did not agree on the annotation type. For instance, *resources* was annotated as a mention of *gate:GATE-Resource*[9] by one annotator, whereas another annotator labeled it as *gate:ResourceParameter*. In some cases, annotators decided to delete an annotation such as in the sentence *Couldn't get resource data for gate.creole.morph.Morph*, although the context indicates that *resource* refers to the GATE-Resource concept.

Another interesting example is annotating *AnnotationSchema()* with brackets included, or *protected FeatureMap features* annotated as a whole, not only *features* part as done by the other annotators.

We used the human-annotated corpus mentioned above to optimise the KCIT tool. In the first iteration we compared the automatic results of KCIT against the gold standard. Then, the mistakes were examined manually and rules for improving KCIT performance were derived and implemented, mostly concerning the filtering and disambiguation phase.

After improving KCIT, 4956 annotations (out of 6175 in total) were correct, 49 partially correct[10], 347 annotations were missing (manually added by participants who created the gold standard), and 823 annotations were wrong (manually deleted by the participants who created the gold standard). These were used in the calculation of the precision and recall figures reported in Table 2.

We conclude that with the proper configuration and tailoring the filtering phase, very good results can be achieved. The quality of the KCIT annotations approaches that of human annotators, while offering significant gains in terms of annotation effort required.

3.2 Evaluation of the information extraction and consolidation phase

We selected another 20 documents to serve as a representative corpus of GATE software artefacts, on which to evaluate KCIT's performance. This selection is made in order to cover not only the knowledge about GATE, but also different types of documents (structured, semi-structured, and unstructured). As before, these were annotated manually and then used to calculate precision and recall values as presented next.

3.2.1 Information extraction

To evaluate the information extraction phase, the automatic annotations produced by KCIT were compared against the gold standard, using the GATE Benchmarking tool [4]. The results are shown in Table 3.

Document group	Number	Precision	Recall
Forum posts	4	0.986	1.0
GATE User Manual chapters	7	0.96	0.96
Web pages	2	0.944	0.951
publications	3	0.898	0.958
java classes	3	0.973	0.989
GATE developer's guide	1	0.965	0.984
Total	20	0.943	0.969

Table 3: Precision and recall measures for the 20 GATE software artefacts

For the 20 selected documents, 4523 created annotations were correct, 41 annotations were partially correct, 126 annotations were missing, and 255 were spurious. Further inspection of the annotated documents revealed that the majority of the spurious/missing annotations were due to errors by the *GATE morphological analyser*. For example, it could not extract correctly the roots of acronyms and camelCased words: the root of *LanguageResources* remained *LanguageResources*.

In addition, there were many wrong annotations of the word *learn*. The problem arises from KCIT not being able to disambiguate correctly based on the local context, resulting in each appearance of the word *learn*, e.g., *learn GATE using movie tutorials*, being annotated as referring to the GATE maching learning plugin. Similarly, many annotations were created for each mention of the word *resources*, even though there was no reference to GATE resources. For example, when reporting the problems on the mailing list, a user said *I cannot waste too much resources on the server...*, meaning computational time, not GATE components. Nevertheless, as demonstrated by the overall results, such problematic cases were quite infrequent.

[9] gate: is used instead of the full namespace which is `http://gate.ac.uk/ns/gate-ontology`

[10] Manually modified by the participants who created the gold standard, where modifications include for example extending the annotation to refer to the longer string; for instance, if *ian_roberts* is not annotated as a whole, but only *ian* is, then this is considered partially correct.

Document group	Number of documents	Iteration 1 precision	Iteration 1 recall	Iteration 2 precision	Iteration 2 recall	Difference precision	Difference recall
user manuals	*10*	0.072878	0.074256	0.944312	0.927389	0.871434	0.853133
publications	*2*	0.07385	0.06655	0.8992	0.85415	0.82535	0.7876
web pages	*6*	0.929883	0.883333	0.929883	0.883333	0	0
source code	*19*	0.711411	0.930384	0.722607	0.94087	0.009433	0.010485
javadoc files	*2*	0.048976	0.05625	0.61	0.8505	0.557625	0.79375
forum posts	*25*	0.391792	0.479632	0.714	0.88	0.322008	0.397043
	64	0.371465	0.415067	**0.803334**	**0.889374**	0.430975	0.473669

Table 2: Average precision and recall values for all documents

The third class of mistakes arose from overlapping annotations not being filtered out properly by KCIT. For example, *ANNIE NE Transducer* was annotated as the whole string refering to the eponymous processing resource, but also *Transducer* was annotated as a mention of *JAPE Transducer*, which again should have been filtered out as redundant.

Overall, we can conclude that the performance of KCIT is of a good quality, especially on domain-specific documents and can be a good base for evaluating the consolidation algorithms. For example, forum posts and java classes were annotated with a very high precision and recall. As documents get more abstract and generalised, more ambiguities creep in (e.g., peer-reviewed publications) and KCIT's performance degrades.

3.2.2 Consolidation phase

To evaluate the consolidation algorithms, we applied recall and precision measures on the same corpus using the GATE ontology, KCIT exposed as a service, and ITM repository for storage.

Here the recall and precision measures are applied to the semantic annotations produced by KCIT exploited for evaluating the two following tasks:

- ontology population (knowledge instances newly created from annotations) and
- semantic annotation (semantic annotations controlled with regard to the instantiated concepts in the repository).

Hence, we obtain the two following adapted measures:

- Precision measures the number of annotations/instances correctly acquired divided by the total number of annotations/instances acquired.
- Recall measures the number of annotations/instances correctly acquired divided by the number of annotations/instances returned by KCIT.

Table 4 shows results according to the two different tasks. We want to emphasize the fact that we are not evaluating the KCIT results (done in previous section) or the quality of the ontology model but the performance of the consolidation algorithms themselves. First, it is important to notice that the consolidation algorithms are reducing the final number of knowledge instances or annotations based on the KCIT annotations. That means that the merging step is successful in resolving references to different annotations towards the same instance in the knowledge base. For example, in the movies.xml file, on 129 annotations generated by KCIT, CA Manager created only 46 knowledge instances and 27 semantic annotations. For instance, the KCIT annotated the two terms ontology tool and Ontology Tool that in fact correspond to the same instance of the class GATEPlugin, with both labels as names and the following URI, `http://gate.ac.uk/ns/gate-ontology\#Ontology\_Tools`. This URI is then used as the reference link to create the semantic annotation between the document and this instance. So instead of having two annotations, only one is produced in this case and the labels have been consolidated on the knowledge instances. The same occurs with for example the term ANNIE which is annotated 9 times by KCIT whereas at the end of the CA Manager pipeline, there is only one semantic annotation referring to the same instance ANNIE of class "GATEPlugin"

It appears that sometimes the consolidation of the semantic annotation on one label has not been correctly achieved by the CA Manager, producing non pertinent annotations or instance references as some duplicates remained. For example, the terms "pipeline" and "Pipeline" produced four knowledge instances: two with the URI `gate:Pipeline` and two with the URI `gate:GATEController`. At most, we would like to have one of each URI if the CA Manager could not disambiguate to which instance the annotation refers to. But it should aggregate the two labels "pipeline" and "Pipeline" on each URI value.

This leads to only 76.5% of precision for creating knowledge instances first and as the semantic annotations are referring to these instances with their URI, the precision result is slightly better, reaching 93.3%. Indeed, if in the knowledge instances results, we obtain four instances with the same URI, each having their own label,

Table 4: Performance results on the representative corpus of the GATE case study

Element type in the ontology	Number of correct elements (A)	Number of missing elements (B)	Number of spurious elements (C)	Recall (A/A+B)	Precision (A/A+C)	F1-measure (R*P)/0.5 (R+P)
Kb instances	208	0	64	1	0.765	0.867
Annotations	168	0	12	1	0.933	0.965

in the semantic annotation we point to the URI, and not the individual labels. As such, the different knowledge instances possessing the same URI are merged into the one with now four labels and the semantic annotation can refer to this federated instance, improving the results. In fact, the more the CA Manager is used, the better it enriches the knowledge base and therefore the semantic annotation results.

The CA Manager obtains 100% recall on both tasks, hence there is no loss of information after processing the KCIT annotations to produce the final semantic annotations and knowledge instances. The CA Manager consolidation algorithms that deal with merging need to be improved in order to eliminate duplicated terms with different orthographic labels such as "datastore", "data store", "DATASTORE", and "Data Store" that must be merged together in the same knowledge instance and thus producing only one semantic annotation referring to the one particular instance. On the other hand, the consolidation algorithms that control the ontology model are performing nicely which is not very difficult in the GATE case study as we mostly refer to the class instances. There are no annotations which refer to the relation between knowledge instances for example. For doing this, we need to improve the linguistic analysis to support that feature and thus evaluate properly this functionality. This is the case in the research and industrial projects in which the CA Manager is now used.

4. RELATED WORK

KIM [11] performs semantic annotation and ontology population automatically in respect to their ontology, by identifying the Key Phrases and Named Entities (NE). As NE they consider people, organizations, locations, and others referred to by name. They use GATE [4] for NE Recognition. Our approach is more flexible than KIM, in the sense that the CA Manager is a generic framework and can thus support semantic annotation based on any information extraction tool, and also any ontology, or semantic repository.

Other similar work includes frameworks such as S-Cream [9], MnM [10], Artequakt [1] and OntoSophie [13]. We can notice important differences between the CA Manager framework and similar approaches: some of them use machine learning techniques, some are based on high-level generic ontologies such as PROTON[11] rather than domain-oriented ontologies, or they perform either ontology population or semantic annotation. The main difference between those platforms and the CA Manager is the fact that the CA Manager preserves the independence between the content augmentation tool, such as KCIT, and the semantic repositories (Sesame[12], ITM[13]). It acts as a mediator, providing greater flexibility and adaptability capabilities as required by real world applications. For more details about the existing semantic annotation platforms we refer the reader to the survey in [12].

Moreover, as the consolidation phase is a very important part of the CA Manager, we would like to emphasize that the tools for ontology population or semantic annotation which describe, or even mention, the consolidation phase in their workflows [1], are rare. However, this phase is extremely important to maintain the integrity and the quality of the application's referential. In fact, most of them rely on manual validation only, to check the generated annotations or instances. In the knowledge acquisition field, the *ArtEquAkt* project [1] was concerned with the consolidation phase where Alani et al. defined four problems related to the integration of *new* instances in a knowledge base: duplicated information, geographical consolidation, temporal consolidation and inconsistent information. Their approach consists of instantiating the knowledge base based on the information extracted from the documents. They apply a consolidation algorithm driven by a set of heuristics and methods of terminological expansion based on the WordNet [8] lexical base.

Contrary to their approach, we are convinced that in order to preserve the integrity of the knowledge base, this consolidation phase must be carried out before the creation of the instances in the repository. Thus, only new and consistent information is created, yet preserving the integrity of the referential and thus improving the quality of the target application.

5. CONCLUSIONS

[11]PROTON website: `http://proton.semanticweb.org/`
[12]Sesame website: `http://www.openrdf.org`
[13]ITM website: `http://mondeca.com/index.php/en/intelligent_topic_manager`

We presented the CA Manager framework which serves as a mediator between semantic annotation and ontology population, and is capable of consolidating and controlling this process while allowing human annotators to be involved, if required. We created various workflows to evaluate the flexibility and scalability of this framework, which offers adapted workflows for ontology population and semantic annotation based on Semantic Web standards and UIMA concepts. The main contribution of the CA Manager in comparison to other similar tools is that it allows easy plugin of information extraction tools, semantic repositories and ontologies.

We have created a gold standard corpus in the domain of software engineering, based on which we could calculate Inter Annotation Agreement, and also precision and recall values of automatically processed results. First, we have used this corpus to calculate the performance of KCIT - a GATE-based information extraction tool which has been exposed as a Web service and used by the CA Manager; then we calculated the performance of the consolidation algorithms based on the same corpus. The automatically produced annotations reach the level of human annotators and are therefore suitable for practical applications.

6. ACKNOWLEDGMENTS

This research was partially supported by the EU Sixth Framework Program project TAO (FP6-026460).

7. REFERENCES

[1] H. Alani, S. Kim, D. Millard, M. Weal, W. Hall, P. Lewis, and N. Shadbolt. Web-based Knowledge Extraction and Consolidation for Automatic Ontology Instantiation. In *Proceedings of the Knowledge Markup and Semantic Annotation Workshop (SEMANNOT'03)*, Sanibel, Florida, 2003.

[2] F. Amardeilh. Semantic annotation and ontology population. In J. Cardoso and M. Lytras, editors, *Semantic Web Engineering in the Knowledge Society*. Idea Group Publishing, 2008.

[3] F. Amardeilh and T. Francart. A semantic web portal with hlt capabilities. *Veille Stratégique Scientifique et Technologique (VSST04)*, 2:481–492, 2004.

[4] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[5] H. Cunningham, V. Tablan, K. Bontcheva, and M. Dimitrov. Language Engineering Tools for Collaborative Corpus Annotation. In *Proceedings of Corpus Linguistics 2003*, Lancaster, UK, 2003. `http://gate.ac.uk/sale/cl03/distrib-ollie-cl03.doc`.

[6] D. Damljanovic, F. Amardeilh, and K. Bontcheva. CA Manager Framework: Creating Customised Workflows for Ontology Population and Semantic Annotation. In *Proceedings of The Fifth International Conference on Knowledge Capture (KCAP'09)*, California, USA, September 2009.

[7] D. Damljanovic, V. Tablan, and K. Bontcheva. A text-based query interface to owl ontologies. In *6th Language Resources and Evaluation Conference (LREC)*, Marrakech, Morocco, May 2008. ELRA.

[8] C. Fellbaum, editor. *WordNet - An Electronic Lexical Database*. MIT Press, 1998.

[9] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM — Semi-automatic CREAtion of Metadata. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 358–372, Siguenza, Spain, 2002.

[10] E. Motta, M. Vargas-Vera, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 379–391, Siguenza, Spain, 2002.

[11] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. KIM – Semantic Annotation Platform. In *2nd International Semantic Web Conference (ISWC2003)*, pages 484–499, Berlin, 2003. Springer.

[12] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):14–28, January 2006.

[13] A. G. Valarakos, G. Paliouras, V. Karkaletsis, and G. Vouros. Enhancing ontological knowledge through ontology population and enrichment. In *Engineering Knowledge in the Age of the SemanticWeb*, pages 144–156, 2004.