Automatic Report Generation from Ontologies: the MIAKT approach

Kalina Bontcheva, Yorick Wilks

Department of Computer Science, University of Sheffield
211 Portobello St, Sheffield, UK S1 4DP
{kalina,yorick}@dcs.shef.ac.uk

Abstract. This paper presented an approach for automatic generation of reports from domain ontologies encoded in Semantic Web standards like OWL. The paper identifies the challenges that need to be addressed when generating text from RDF and OWL and demonstrates how the ontology is used during the different stages of the generation process. The main contribution is in showing how NLG tools that take Semantic Web ontologies as their input can be designed to minimises the portability effort, while offering better output than template-based ontology verbalisers.

1 Introduction

The Semantic Web aims to add a machine tractable, re-purposeable layer to complement the existing web of natural language hypertext. In order to realise this vision, the creation of semantic annotation, the linking of web pages to ontologies, and the creation, evolution and interrelation of ontologies must become automatic or semi-automatic processes.

Natural Language Generation (NLG) takes structured data in a knowledge base as input and produces natural language text (see [1]). In the context of Semantic Web or knowledge management, NLG can be applied to provide automated documentation of ontologies and knowledge bases or to generate textual reports from the formal knowledge. Unlike human-written texts, an automatic approach will constantly keep the text up-to-date which is vitally important in the Semantic Web context where knowledge is dynamic and is updated frequently. The NLG approach also allows generation in multiple languages without the need for human or automatic translation [2].

The main challenge posed for NLG by the Semantic Web is to provide tools and techniques that are extendable and maintainable (the majority of existing NLG applications can only be modified and extended by NLG experts).

This paper presents the MIAKT (Medical Imaging and Advanced Knowledge Technologies) project where NLG is used to generate automatically reports from knowledge encoded in the domain ontology (Section 2). Section 3 discusses the MIAKT generator, followed by performance evaluation results (Section 4) and a discussion on domain portability (Section 5). Finally, Section 6 presents some related work, and Section 7 outlines directions for future work.

2 MIAKT

This work was carried out as part of the e-science project MIAKT[1], which aims at developing Grid enabled knowledge services for collaborative problem solving in medical informatics. In particular, the domain in focus is Triple Assessment in symptomatic focal breast disease.

The role of NLG in the project is to generate automatically textual descriptions from the semantic information associated with each case - patient information, medical procedures, mammograms, etc. The reports are aimed at the medical professionals involved in the diagnosis and treatment, therefore it is essential to convey in the generated report the complete information available in the ontology about the patient.

The majority of semantic information is encoded in the domain ontology, which is a formal description of the breast cancer domain [3] and is encoded in DAML+OIL [4]. In addition, each case has a case-specific, i.e., instance knowledge, which is encoded in RDF [5] and specifies information about this particular case, e.g., which medical procedures were undertaken, sizes and locations of lesions, diagnosis. The domain ontology was engineered manually as part of the project and does not contain inconsistencies. This is frequently the case with medical ontologies and terminological lexicons as they are then used as standards in the community (e.g., UMLS [6]).

In order to avoid the cost of having to parse and represent ontologies in each of these formats (DAML+OIL and RDF) in MIAKT, we used GATE's ontology tools [7] that can parse these formats and convert them into a common object-oriented model of ontologies with a unified API (Application Programming Interface). Consequently, our generator was developed to work from this common representation, in isolation from the concrete ontology implementation. The benefit of this approach is that if a new ontology format needs to be added at a later date (e.g., OWL), the generator would not need to be modified.

We used the NLG lexicon tools [8] to create a lexicon of 320 terms in the domain of breast cancer and map them to the 76 concepts and 153 instances in the MIAKT ontology. These terms were collected manually from the BIRADS lexicon of mammography terms[2] and NHS documents[3], then verified and enriched manually with synonyms from online papers, Medline abstracts, and the UMLS thesaurus [6].

3 The GATE-based MIAKT Generator

The MIAKT generator takes as input the medical ontology, an RDF description of the case, and the MIAKT NLG lexicon. The output is a textual report

[1] Project Web site: http://www.aktors.org/miakt. MIAKT is supported by the UK Engineering and Physical Sciences Research Council as part of the MIAKT project (grant GR/R85150/01), which involves the University of Southampton, University of Sheffield, the Open University, University of Oxford, and King's College London.

[2] Available at http://www.acr.org/departments/stand_accred/birads/

[3] http://www.cancerscreening.nhs.uk/breastscreen/index.html

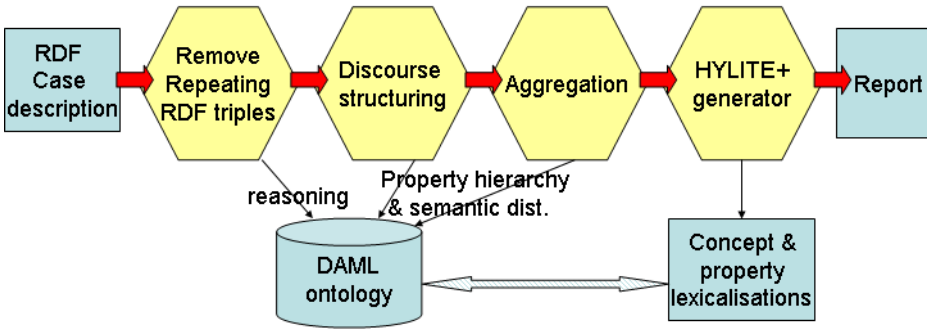

Fig. 1. The MIAKT Generator

verbalising the semantic information provided for the given case, enriched with information from the ontology and using the terms provided in the MIAKT lexicon.

The RDF case description is created by the medical professional who carried out the examination and made the diagnosis. MIAKT provides an ontology-based user interface where the medics can annotate the images (e.g., X-rays, MRI scans) with semantic information (e.g., shape of tumor, size, diagnosis) and also enter patient details, etc.

In order to generate natural language reports from RDF, there are several challenges that need to be addressed:

- The RDF can sometimes contain repetitive information, due to presence of inverse relations in the ontology, e.g., `involved_in_ta(01401_patient, ta-1069861276136)` and `involve_patient(ta-1069861276136, 01401_patient)`, or due to one fact entailing another. Therefore the generator needs to use a reasoner based on the ontology to filter out repetitive RDF statements.
- Need to order the RDF triples in well-structured discourse.
- If each triple is used to generate a separate sentence, the text becomes hard to read, with too many short sentences. Therefore, aggregation needs to be performed as part of the generation process to combine several triples into one sentence.
- Finally, if a sentence is being formed from more than one triple, the surface realiser needs to determine whether to use conjunctive or subordinate clauses; which attributive properties should be expressed as adjectives and which as nouns; when to use passive or active voice, etc.

The MIAKT generation architecture is shown in Fig. 1. The first stage is to remove already verbalised triples from the RDF (Sect. 3.2), then the remaining RDF triples are ordered to form well-structured discourse (Sect. 3.3), followed by an aggregation stage where similar RDF triples are combined to form one sentence, resulting into a more coherent text. Finally the text is produced by

the ontology-based verbaliser, which uses the lexicon and the property hierarchy from the ontology. We will focus on that process next.

3.1 The Ontology-based Realiser

The ontology-based realiser transforms RDF statements into conceptual graphs (a kind of semantic network) which are then verbalised by the HYLITE+ surface realiser [9]. The output is a textual report.

The surface realiser does not use templates (cf [10] for template-based verbalisation of RDF), i.e., have fixed expressions where the arguments of each relation are inserted. Instead, its input is the RDF statement and the concept which is going to be the subject of the sentence. Then it treats the RDF as a graph and finds a path through that graph, starting from the given concept and visiting all properties and their arguments. For further details see [9].

The HYLITE+ surface realiser already has a list of 40 different relations that it can verbalise. Some are linguistically motivated relations like `AGNT` (agent), `PTNT` (patient), and `OBJ` (object), while others describe attributes, locative relations, part-whole relations, etc. When these relations are compared to the properties in ontologies, there is a substantial gap, because properties are typically not linguistically motivated (e.g., `has_date`, `produce_result`). In order to verbalise such properties the surface realiser needs a lexicalisation for each one of them.

In order to make the ontology-based surface realiser more portable and reduce the need for specifying manually the lexicalisation of each property in the ontology, we defined a core set of 4 basic property types – `active-action`, `passive-action`, `attribute`, and `part-whole`. In the MIAKT ontology any other property is defined as a sub-property of one of these 4 generic ones.

Sub-types of attribute and part-whole properties can then be handled automatically, because they correspond to `PART_OF` and `ATTR` relations and the realiser already has grammar rules to verbalise them. New properties can be introduced in the ontology as sub-properties of either attribute or part-whole property and they will be handled automatically by the surface realiser.

Active action and passive action properties were introduced to help the surface realiser with mapping the arguments of such properties to semantic roles like agent, patient, and object. In general, the first argument of active properties is mapped to an agent role, whereas the first argument of passive properties is mapped to a patient role. For example the triple from case0140 (see Figure 2) involving the active property `produce_result`:

```
<rdf:Description rdf:about=
       'file:/...#01401_mammography'>
  <NS2:produce_result rdf:resource=
       'file:/...#image_01401_left_cc'/>
... </rdf:Description>
```

is mapped to two facts using the `AGNT` and `OBJ` relations and a concept `PRODUCE_RESULT`:

```
AGNT(Mammography: 01401_mammography, PRODUCE_RESULT)
OBJ(PRODUCE_RESULT, Medical_Image: image_01401_left_cc)
```

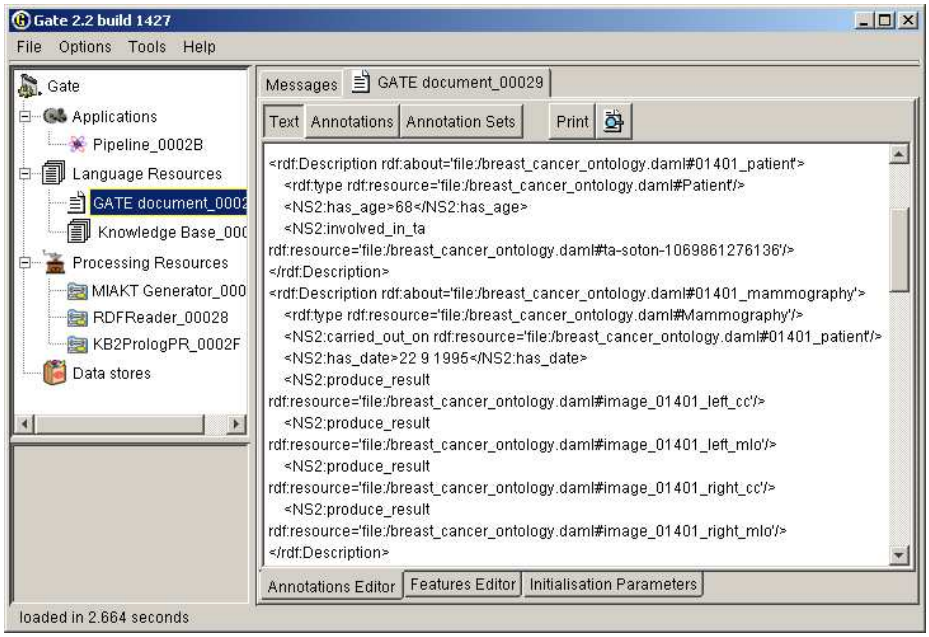

Fig. 2. The case information encoded as RDF (case0140)

The MIAKT lexicon already contains lexicalisations for concepts and instances that occur in the ontology, i.e., mammography and medical image in this case. Therefore, we additionally specified the lexicalisation of each active and passive action property (e.g., `PRODUCE_RESULT`). At present, there are only 4 active action and 3 passive action sub-properties in the ontology, so it was not a substantial overhead. If new active or passive action properties are added in the future, their lexicalisations need to be specified manually by the ontology builder in order to enable their verbalisation. However this is not a difficult task, as the user only needs to provide the verb that corresponds to the new property in its base form and the generator will deal with the morphological issues.

To summarise, the ontology-based realiser was made more generic and easier to adapt by introducing a property hierarchy in the ontology, based on 4 linguistically motivated basic types. This hierarchy also plays an important role in allowing the creation of more generic text schemas (Section 3.3). The decision to introduce linguistically motivated properties in the ontology can be seen as undesirable in some applications, so it is possible to introduce this classification only in the generator's input handler, leaving the original ontology intact.

Linguistically oriented ontologies have already been used as interface methods between generators and formal knowledge of the domain in some NLG systems(e.g., ONTOGENERATION [2]). The most frequently used on is the Generalised Upper Model (GUM) [11], which is a linguistic ontology with hundreds of concepts and relations, e.g., part-whole, spatio-temporal, cause-effect. In con-

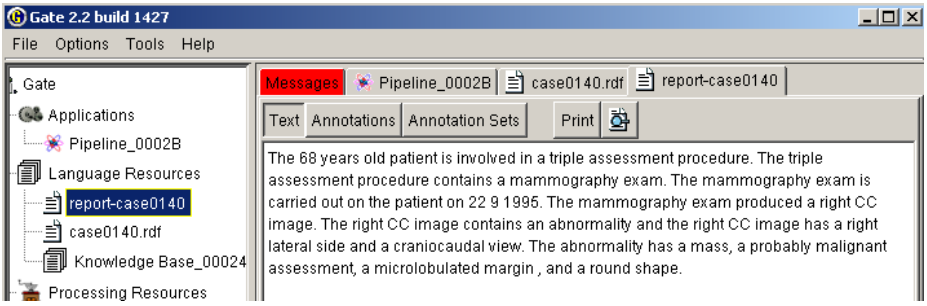

Fig. 3. The generated report for case0140

trast, in MIAKT we chose to use few, very basic distinctions in order to enable knowledge engineers rather than NLG experts to perform the mapping between properties and their linguistic expression. Our experience indeed showed that the ontology engineer found the distinction between the 4 basic types intuitive to understand and use [3]. However, the choice of only few basic types has an impact on the variability and expressivity of the generator. This has not posed a problem in our domain, because the majority of the information is conveyed in the noun phrases and also the reports tend to have a fairly regular structure.

3.2 Filtering Out Repetitions using the Ontology

The report generation process starts off by being given a set of triples describing a case, in the form of an RDF file (see Figure 2). Since there is some repetition, these triples are first filtered to remove already said facts. In addition to triples that have the same property and arguments, the system also removes triples involving inverse properties with the same arguments, as those of an already verbalised one. The information about inverse properties is provided by the ontology – both DAML+OIL and OWL have this primitive for properties.

For example, the MIAKT ontology states that these two properties are inverse:

```
<daml:ObjectProperty rdf:about=
        "file:/...#involved_in_ta">
   <daml:inverseOf rdf:resource=
        "file:/...#involve_patient"/>
   ...
```

At the start of the report, the generator verbalises the triple `involved_in_ta(01401_patient, ta-soton-1069)` as part of the first sentence (Figure 3) describing the patient. Then the following triples are provided in the RDF input for the given instance of triple assessment procedure:

```
involve_patient(ta-soton-1069, 01401_patient)

consists_of_subproc(ta-soton-1069, 01401_mammography)
```

Since the ontology specifies that `involve-patient` and `involved_in_ta` are inverse properties and the triple `involved_in_ta(01401_patient, ta-soton-1069)` has already been verbalised, the `involve-patient` triple is removed as already known.

In this way, the ontology plays an important role in detecting repetitive input. Future work will experiment with removing implied information, which logically follows from already said facts. This will be done by using a reasoner and axioms from the ontology, in order to determine which statements follow from other ones. Such functionality is often provided by semantic repositories such as Sesame (`http://www.openrdf.org/`).

3.3 Discourse Planning and the Property Hierarchy

The main effort in porting a generator to a new domain is in the adaptation of its discourse planning component. The example report in Figure 3 shows that typically the MIAKT reports start off by describing the patient, then move on to the medical procedures and their findings. The RDF input for the case however (Figure 2) does not specify any ordering between the RDF triples. In order to bridge this gap, we created discourse patterns that are applied recursively by the discourse structuring algorithm and capitalise on the property hierarchy.

The top-level schema looks like:

```
Describe-Patient ->
    Patient-Attributes,
    Describe-Procedures
```

where `Patient-Attributes` and `Describe-Procedures` are recursive calls to other schemas. For example, the `Patient-Attributes` schema collects recursively all properties that are sub-properties of the `attribute-property` and involve the given patient:

```
Patient-Attributes ->
    [attribute(Patient, Attribute)],
    Patient-Attributes *
```

As can be seen, the discourse schemas exploit the property hierarchy in order to be independent of property names in the given ontology. However, if more specialised treatment of some properties is required, it is possible to enhance the schema library with a new pattern, that applies only to a specific property.

In the example shown in Fig. 2, only one attribute property involving the patient is matched by the `Patient-Attributes` schema: `has_age(01401_patient, 68)`. The `Describe-Procedures` schema then matches the `involved_in_ta(01401_patient, ta-soton-1069)` triple and continues by recursively describing each of the procedures from the matched triples (in this case – one triple assessment procedure).

3.4 Ontology-based Aggregation

Once the information from the ontology is structured using the schemas, the next stage is to perform aggregation and join similar RDF triples. This aggregation is done at discourse level [12] by joining adjacent triples that have the same

first argument and have the same property name or if they are sub-properties of `attribute` or `part-whole` properties. For example, in the last sentence in Figure 3 we have 4 triples with the same first argument (the abnormality) and all properties are attribute properties. Therefore, they are joined together as one proposition, which gives rise to the last sentence. The aggregated proposition is which is then passed to the surface realiser (Sect. 3.1):

```
ATTR(Abnormality: 01401_abnormality, Mass: 01401_mass)
ATTR(Abnormality: 01401_abnormality, Margin: inst_margin_microlob)
ATTR(Abnormality: 01401_abnormality, Shape: inst_shape_round)
ATTR(Abnormality: 01401_abnormality, Diagnose: inst_ass_prob_malig)
```

Without this aggregation step, there will be four separate sentences, resulting in a less coherent report:

```
The abnormality has a mass. The abnormality has a microlobulated
margin. The abnormality has a round shape. The abnormality has a
probably malignant assessment.
```

4 Robustness Evaluation

The robustness of the MIAKT generator was evaluated by running it on 350 different cases without modifications in between the runs. None of these 350 cases were used during the customisation of the ontology-based generator to the MIAKT domain, i.e., this was unseen data.

The experiment showed that there are no missing lexicalisations in the MIAKT terminological lexicon. Also, the MIAKT generator successfully produced reports for all 350 cases. There were some problems with the consistency of the RDF input in some cases, but the MIAKT generator proved robust in face of such noisy input.

The most frequent problem was that some RDF descriptions were incomplete. In these circumstances this information was not included in the report. For example, some RDF files would list 4 images as the result of a mammography exam (see Figure 2), but would not provide corresponding `<Description>` entries for them further down defining these instances, so the generator cannot refer to their location or their properties. For such instances without definitions the system currently chooses to omit them completely, e.g., the corresponding report in Figure 3 mentions only one image, rather than 4, because the other 3 instances are undefined. A warning message is produced in the GATE Messages tab alerting the user to the detected problems with the input.

In terms of performance, the generation of one report takes, on average, less than 20 milliseconds on a PIII 800MHz computer with 512MB memory, running Windows 2000 and Java as part of the JBuilder development environment. Batch processing in a command prompt and without a development environment is some 10% faster. The average length of reports on these 350 test cases is 10 sentences. Performance evaluations of HYLITE+ on longer texts have shown comparable results [13].

5 Domain Portability

The ontology-based generator relies on a lexicon for all concepts and instances in the ontology and also on lexicalisations for all active action and passive action properties. In MIAKT they were all acquired manually and part of the porting effort to a new domain lies in the creation of these resources. However, this process can be facilitated substantially by importing information from existing resources such as UMLS and its SPECIALIST lexicon [6].

A new domain also means a different ontology. The main requirement towards the input ontology is to provide the hierarchical property structure discussed in Section 3.1. Further refinement of the `part-whole-property` into finer grained types such as `member-of-property` and `made-of-property` will increase the generator's expressiveness, but is not required.

If the ontology does not already have such a hierarchy, then it might be possible to create one semi-automatically, if there are well-defined naming conventions. For example, the UMLS Semantic Net [6] provides 54 links[4] classified into a 3-level hierarchy with 5 top categories – `Spatially RelatedTo`, `Conceptually RelatedTo`, `Physically RelatedTo`, `Functionally RelatedTo`, and `Temporally RelatedTo`. Sub-links of each of these 5 top-level links tend map to at least 2 of our 4 generic properties. For instance, there are 4 links under `Spatially RelatedTo` – `Has_location`, `Adjacent_to`, `Surrounded_by`, and `Traversed_by`. From NLG perspective the first one is attributive, the second – an active action, and the third and fourth – passive action properties. Since UMLS links are named following certain conventions, it is possible to implement heuristics automatically mapping links with names starting with `has_` as subproperties of `attribute-property` and those ending with `ed_by` as subproperties of `passive-action-property`.

As discussed in Section 3.3, the patterns used for text structuring are typically domain and application specific. Therefore, the main effort in porting the generator to a new domain is in defining and implementing these patterns. Typically this involves the creation of a corpus of target texts and its analysis by a language generation expert, unlike the lexicon building and ontology modification tasks which can be carried out by knowledge engineers.

Recent work in language generation has started experimenting with machine learning to induce text structuring patterns from example texts annotated with the underlying semantic propositions. So far, only small semantically annotated corpora have been created because semantic annotation is time consuming since it requires a high level of detail, i.e., first annotating concepts and then the relations between them. For example, [14] have collected an annotated corpus of 24 transcripts of medical briefings. They use 29 categories to classify the 200 tags used in their tagset. Each transcript had an average of 33 tags with some tags being much more frequent than others. Since the tags need to convey the semantics of the text units, they are highly domain specific, which means that this training data is specific to their application and domain and any model

[4] Links in UMLS Semantic Net are equivalent to properties in Semantic Web ontologies.

learned from this corpus will not be applicable in other circumstances. Future work on language generation in MIAKT aims at investigating how to lower the cost of adapting the discourse structuring component to new domains.

6 Related Work

NLG systems that are specifically targeted towards Semantic Web ontologies have started to emerge only recently. For example, there are some general purpose ontology verbalisers for RDF and DAML+OIL [10] and OWL [15]. They are template-based and follow closely the ontology constructs, e.g., *"This is a description of John Smith identified by http://...His given name is John..."* [15]. The advantages of Wilcock's approach is that it is fully automatic and does not require a lexicon. In contrast, the MIAKT approach requires some manual input (lexicons and domain schemas), but on the other hand it generates more fluent reports, oriented towards end-users, not ontology builders.

On the other end of the spectrum are sophisticated NLG systems such as TAILOR [16], MIGRAINE [17], and STOP [18] which offer tailored output based on user/patient models. In MIAKT we adopted a simpler approach, exploring generalities in the domain ontology, because our goal was to lower the effort for customising the system to new domains. Sophisticated systems, while offering more flexibility and expressiveness, are difficult to adapt by non-NLG experts. Our experience in MIAKT showed that knowledge management and Semantic Web ontologies tend to evolve over time, so it is essential to have an easy-to-maintain NLG approach.

The ONTOGENERATION project [2] explored the use of a linguistically oriented ontology (the Generalised Upper Model (GUM) [11]) as an abstraction between generators and their domain knowledge base. The project developed a Spanish generator using systemic grammars and KPML [19]. The main difference from our approach comes from the number of concepts and relations used to abstract the generator from the concrete domain ontology. In MIAKT we chose only 4 basic properties, in order to make it easier for non-linguists to carry out this task. The size and complexity of GUM make this process more difficult for non-experts. In general, there is a trade-off between expressivity and the number of linguistic constructs in the ontology. Therefore our approach is mainly suitable for applications where more schematic texts are sufficient and the goal is to have non-linguists being able to customise the generator for new domains.

This work also bears similarities with the problem of building portable and customisable NLG systems from relational databases [20]. Both our and ILEX approaches require a formal definition of domain knowledge as taxonomy or ontology and a mapping of ontology items to their lexicalisations. In the case of Semantic Web ontologies, the information about domain types and data types of the slot fillers is already formally specified, unlike in databases. Our approach differs from that in [20] with its use of reasoning and a property hierarchy to avoid repetitions, enable more generic text schemas, and perform aggregation.

Work on ILEX is complementary because it focused on low-cost methods for providing adaptivity and generation of comparisons.

7 Conclusion

This paper presented an approach for automatic generation of reports from domain ontologies encoded in Semantic Web standards like OWL. The novel aspects of the MIAKT generator are in the use of the ontology, mainly the property hierarchy, in order to make it easier to connect a generator to a new domain ontology. It also comes with a number of user-friendly tools for providing lexicalisations for the concepts and properties in the ontology [8], thus making it easier for non-specialists to customise a generator to their application. Our main contribution is in showing how existing NLG tools can be adapted to take Semantic Web ontologies as their input, in a way which minimises the portability effort while offering better output than template-based ontology verbalisers (e.g., [15]).

The system is still under development[5] and is about to undergo user-based evaluation where medics will be asked to provide qualitative feedback on the readability and utility of generated reports. Preliminary feedback from the medical researchers involved in the project has indicated that such reports are perceived as well structured and understandable.

Future work will also aim to address the problem of generating tailored reports, depending on the user and the context. In the MIAKT domain the application required that all the information from the ontology about a given patient is included in the generated report. In other applications this might lead to overly verbose reports and thus methods for selecting only part of the available information will be required.

References

1. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Journal of Natural Language Engineering **Vol. 3 Part 1** (1999)
2. Aguado, G., Bañón, A., Bateman, J.A., Bernardos, S., Fernández, M., Gómez-Pérez, A., Nieto, E., Olalla, A., Plaza, R., Sánchez, A.: ONTOGENERATION: Reusing domain and linguistic ontologies for Spanish text generation. In: Workshop on Applications of Ontologies and Problem Solving Methods, ECAI'98. (1998)
3. Hu, B., Dasmahapatra, S., Shadbolt, N.: From Lexicon To Mammographic Ontology: Experiences and Lessons. In Calvanese, D., De Giacomo, G., Franconi, E., eds.: Proceedings of the International Workshop on Description Logics (DL'2003). (2003) 229–233
4. Horrocks, I., van Harmelen, F.: Reference Description of the DAML+OIL (March 2001) Ontology Markup Language. Technical report (2001) `http://www.daml.org/2001/03/reference.html`.

[5] It is available online as a web service. To obtain a demonstration client or further information contact the first author.

5. Lassila, O., Swick, R.: Resource Description Framework (RDF) Model and Syntax Specification. Technical Report 19990222, W3C Consortium, `http://www.w3.org/TR/REC-rdf-syntax/` (1999)
6. NLM: Unified Medical Language System (UMLS). Technical report, National Library of Medicine, (http://www.nlm.nih.gov/research/umls/umlsmain.html)
7. Bontcheva, K., Kiryakov, A., Cunningham, H., Popov, B., Dimitrov, M.: Semantic web enabled, open source language technology. In: EACL workshop on Language Technology and the Semantic Web: NLP and XML, Budapest, Hungary (2003)
8. Bontcheva, K.: Open-source Tools for Creation, Maintenance, and Storage of Lexical Resources for Language Generation from Ontologies. In: Proceedings of 4th Language Resources and Evaluation Conference (LREC'04). (2004)
9. Bontcheva, K.: Generation of multilingual explanations from conceptual graphs. In Mitkov, R., Nicolov, N., eds.: Recent Advances in Natural Language Processing: Selected Papers from RANLP'95. Volume 136 of Current Issues in Linguistic Theory (CILT). John Benjamins, Amsterdam/Philadelphia (1997) 365 – 376
10. Wilcock, G., Jokinen, K.: Generating Responses and Explanations from RDF/XML and DAML+OIL. In: Knowledge and Reasoning in Practical Dialogue Systems, IJCAI-2003, Acapulco (2003) 58–63
11. Bateman, J.A., Magnini, B., Fabris, G.: The Generalized Upper Model Knowledge Base: Organization and Use. In: Towards Very Large Knowledge Bases. (1995) 60–72
12. Reape, M., Mellish, C.: Just what *is* aggregation anyway? In: Proceedings of the European Workshop on Natural Language Generation (EWNLG'99), Toulouse, France (1999) 20 – 29
13. Bontcheva, K., Dimitrova, V.: Examining the Use of Conceptual Graphs in Adaptive Web-Based Systems that Aid Terminology Learning. International Journal on Artificial Intelligence Tools – Special issue on AI Techniques in Web-Based Educational Systems (2004) Forthcoming.
14. Duboue, P.A., McKeown, K.R.: Empirically estimanting order constraints for content planning in generation. In: Proceedings of ACL-EACL, Toulouse (2001)
15. Wilcock, G.: Talking OWLs: Towards an Ontology Verbalizer. In: Human Language Technology for the Semantic Web and Web Services, ISWC'03, Sanibel Island, Florida (2003) 109–112
16. Paris, C.L.: Tailoring object descriptions to the user's level of expertise. Computational Linguistics **14 (3)** (1988) 64–78 Special Issue on User Modelling.
17. Mittal, V.O., Carenini, G., Moore, J.D.: Generating Patient Specic Explanations in Migraine. In: Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care, McGraw-Hill Inc. (1994)
18. Reiter, E., Robertson, R., Osman, L.: Lessons from a Failure: Generating Tailored Smoking Cessation Letters. Artificial Intelligence **144** (2003) 41–58
19. Bateman, J.A.: Enabling technology for multilingual natural language generation: the kpml development environment. Journal of Natural Language Engineering **3** (1997) 15 – 55
20. O'Donnell, M., Knott, A., Oberlander, J., Mellish, C.: Optimising text quality in generation from relational databases. In: Proceedings of the International Natural Language Generation Conference (INLG'00). (2000) 133–140