A Template-Based Markup Tool for Semantic Web Content

Brian Kettler, James Starz, William Miller, and Peter Haglich

ISX Corporation, 4301 N. Fairfax Drive, Suite 370, Arlington VA 22203 USA
{bkettler, jstarz, wmiller, phaglich}@isx.com

Abstract. The Intelligence Community, among others, is increasingly using document metadata to improve document search and discovery on intranets and extranets. Document markup is still often incomplete, inconsistent, incorrect, and limited to keywords via HTML and XML tags. OWL promises to bring semantics to this markup to improve its machine understandability. A usable markup tool is becoming a barrier to the more widespread use of OWL markup in operational settings. This paper describes some of our attempts at building markup tools, lessons learned, and our latest markup tool, the Semantic Markup Tool (SMT). SMT uses automatic text extractors and templates to hide ontological complexity from end users and helps them quickly specify events and relationships of interest in the document. SMT automatically generates correct and consistent OWL markup. This comes at a cost to expressivity. We are evaluating SMT on several pilot semantic web efforts.

1 Introduction

The Intelligence Community (IC), among others, is increasingly using document metadata to improve document search and discovery on intranets and extranets.[1] Document markup is still often incomplete, inconsistent, incorrect, and limited to keywords via HTML and XML tags. This can lead to poor search performance, even if the search tool has the ability to search structured metadata.[2] Tools to date have focused on metadata annotation by authors at document production time. Some (e.g., In.vision's Xpress Author [6]) integrate with MS Word and other popular document editing tools to create metadata during the document creation, review, and dissemination process. More recent IC efforts have focused on post-production markup in which a user other than the author creates markup, perhaps leveraging the output of an automated document classification tool. These tools tend to focus on pulling out known keywords (e.g., the names of countries), mapping them to terms in a controlled vocabulary (e.g., the ISO 3166 country codes), and then outputting matched terms as tags (e.g., in XML or HTML) in (or associated with) the original document. This is a ripe application area for markup in the Web Ontology Language

[1] This paper uses the terms "web" to refer to unclassified and classified intranets and extranets (as well as the World Wide Web).

[2] Google and similar search engines are still the predominant tools in operational use although extensions to these are the subject of advanced technology pilot projects.

Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 446 – 460, 2005.

(OWL) [11], which can capture the semantics of terms used in markup – including markup of content and the relationships within – and bridge diverse metadata vocabularies across the production community.

This paper describes our latest markup tool, the Semantic Markup Tool (SMT), which was developed for use on an IC pilot effort and applied on several other projects employing semantic web technology. The SMT has benefited from lessons learned in developing and using previous versions, some of which are also described in this paper. The SMT employs an innovative combination of automated text extraction technology (using a variety of commercial products), manual markup through a form-based interface, and the use of markup templates which hide ontological complexity from end users. The SMT outputs correct and consistent OWL markup that feeds other semantic web tools for content exploitation by machines and humans

A usable markup tool is becoming a barrier to the more widespread use of OWL markup in operational settings. The SMT explores a particular point in the tradeoff space between usability and expressivity. We are evaluating SMT on several pilot semantic web efforts. This paper also describes related and future work.

2 Motivation and Requirements

Metadata markup can improve information retrieval and discovery. Metadata needs to be done consistently: e.g., by using controlled vocabularies that are machine-interpretable. Metadata using HTML or XML tag sets can often be ambiguous, as the meaning of the tag is often built into the software that creates or exploits the tags. Although they can require significant effort to configure and maintain, automatic text extraction tools, such as Lockheed's AeroSwarm/AeroText [10], can pull out entities from the text and classify them according to a vocabulary. Some can also extract simple relationships (values for entity attributes, etc.). These tools are still limited by the state of the art in natural language processing in their understanding of the text and are thus unable to capture more complex relationships. The output of these tools can be automatically converted markup in HTML, XML, or even OWL.

In a previous IC project we were involved with, we helped build a manual tool for the capturing of more complex relationships in a document and representing them in automatically generated OWL markup.[3] This tool allowed a user (a subject matter expert) to add facts, represented in OWL, to a document using a form-based interface. This tool automatically generated OWL (in the RDF serialization) that was then combined with OWL-encoded assertions (statements) from multiple documents (and databases translated automatically into OWL) into a logically centralized Knowledge Base, which could then be queried and browsed by users through web-based tools.

A user could say almost anything about a document that the ontologies supported. Users could pick classes and properties of interest from one or more OWL ontologies presented in tree. The number of "root" classes and properties was in the hundreds. This made it hard for a user unfamiliar with the ontology to determine which classes and properties might be appropriate. Even a user familiar with the ontology would have to find those few classes and properties pertinent to the current document. A text

[3] The original version of this was based on the pioneering work resulting in U. Maryland's SHOE Knowledge Annotator [3].

search capability[4] would have been helpful in finding classes and properties in the ontology, but that would have been highly sensitive to how those were named by the original ontology author(s).

Often a user would desire to represent a fact involving multiple classes and properties: e.g., Bob Smith and Fred Jones attended a meeting in San Francisco, CA. The user would need to determine the following:

- Bob Smith and Fred Jones should be instances of class *Person* (or a subclass of *Person*).
- Their names should be represented as values for attributes (datatype properties) *firstName* and *lastName* of these instances.
- A meeting is represented as an instance of class *MeetingEvent* (or one of its subclasses).
- Attendance is represented by the property *hasParticipant* (defined for class *Event* and its subclasses).
- A meeting's location is represented as an instance, linked to an *Event* via property *hasLocation.*

The class instances (which we term "Knowledge Objects" after they are stored in our persistent repository) and properties involved are shown in Figure 1.

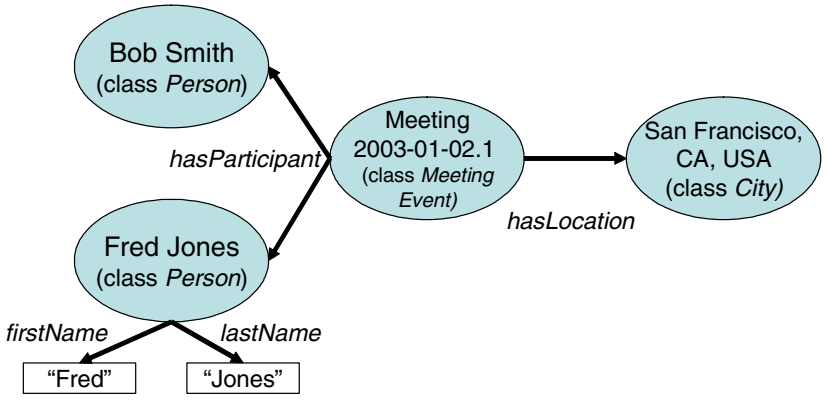

Fig. 1. An "Idiom" for Representing Knowledge Objects (Class Instances) for a Meeting

This is only one of the ways in which the above fact could be represented. Other ontologies would have different representations. Using the previous markup tool, this fact could take quite a while to enter, despite having pick lists, etc. for selecting classes and properties. Representing multiple facts in a document could take hours. The point is that having a user learn these ontology-specific "idioms" or patterns for representing facts burdens the markup process (even when the markup tool is handling the encoding into OWL/RDF/XML). Furthermore, the user is not insulated from the "raw" ontologies and thus potentially has to relearn these idioms when ontologies change.

[4] Such as Stanford's Protégé tool provides [11].

Thus while the previous markup tool provided great flexibility, we could not find many users willing to pay the price in effort to use it. This was especially true in IC organizations dealing with large volumes of highly varied, unstructured content (e.g., web and text documents) to process. The SMT's design, described in the following section, is an attempt to explore another point in the flexibility-usability tradeoff space.

Another lesson learned from our prior experience is the need to provide immediate added value to the user doing the markup, who may or may not be the document author. Although many of the users doing markup will also benefit from the markup later on (e.g., via using newly enabled search capabilities), there must also be some more immediate benefit. One technique investigated for SMT is the automated production of a document summary, which is often required for intelligence product publication, from the user's input. This is described in more detail in Section 3.3.

A major shift in our philosophy of markup has been to view markup as not replacing a document's content (even just its textual – versus multimedia – content), but rather providing a semantically grounded "index" entry for the document (in addition to any text indices for the document). The markup in a document will provide additional information – represented using one or more OWL ontologies – through which to find that document through higher precision, semantically grounded search. A user can then drilldown to the document's text to see its content that has not been modeled. An "index" here is stored in a knowledge base and contains multiple such entries. This index is unlike a book's index in that the former can itself be used to answer some queries, as well as located information. Thus the SMT is focused on support the manual markup of just a few key facts per document.

To summarize the motivation for SMT, our previous work has led us to a point where we are focused on automated markup for the generation of high volume, low "fidelity" assertions; more usable but less flexible tools for manual markup of key facts; and support for document search with some inferencing and question-answering potential via a "meaningful" index aggregated from these assertions. How the SMT does this is the subject of the next section.

3 The Semantic Markup Tool

This section presents additional details on the Semantic Markup Tool (SMT), the IC application in which it is embedded, and some initial results.

3.1 The Application Context

The IC application has the wider goal of providing web users (analysts and warfighters) with improved search and discovery capabilities by integrating techniques for keyword search (a la Google), metadata search, and retrieval of knowledge objects, assertions collected from facts in OWL markup of documents and data sources. The application collects and processes web documents (in HTML) to (1) extract and normalize (administrative) metadata (e.g., author, producing organization, date published, etc.), (2) convert the HTML to text, and (3) index the document's content by its keywords and ontologically described entities and relationships. Figure 2 shows this process and the components involved.

The entities and relationships will be described in OWL markup that references OWL ontologies. Because of the projected volume of web documents to process, a major requirement for a markup tool was the minimization of human effort required. The initial concept of operations was that most documents would be processed through an automatic extractor to capture entities and simple relationships from the text. Only a fraction of documents would be augmented with manual markup to describe more complex relationships beyond the capabilities of text extractors to find. The application uses a web service wrapping one or more commercial text extractors.

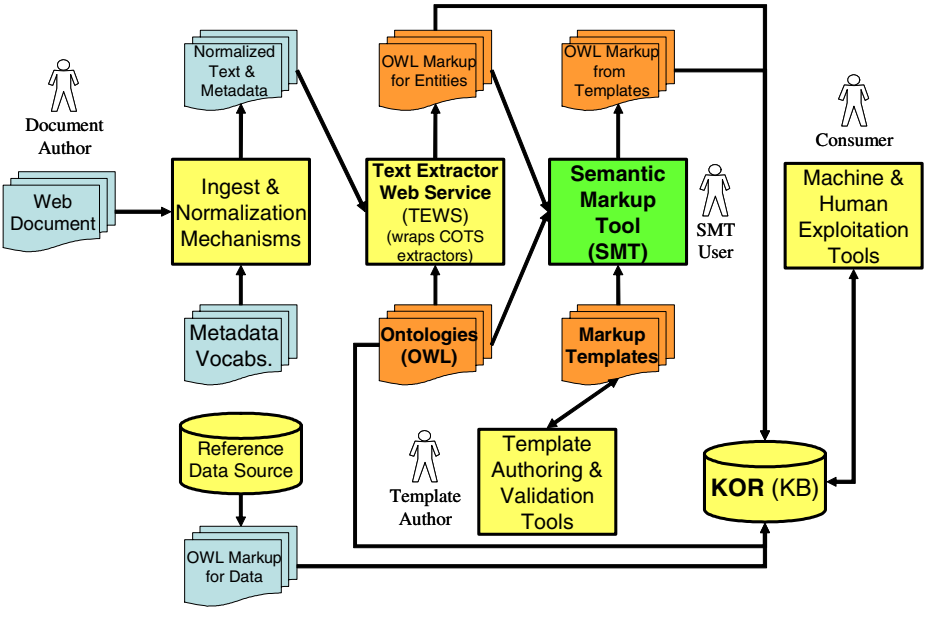

Fig. 2. Functional application architecture in which SMT is embedded

The Semantic Markup Tool (SMT) supports the markup of OWL documents by subject matter experts (SME's). It uses a hybrid manual-automatic approach to markup of document content in OWL. Steps include:

1. A document is ingested from the web. This document can be in HTML and several other types. The document is converted to text. Metadata in the document (e.g., in HTML tags) extracted and normalized (e.g., dates, country codes, etc are converted to a canonical format). Statistical categorizers assign topic codes) to the document if necessary. The document is indexed by its keywords using a commercial text indexer.
2. A document is then processed by the commercial text extractors, wrapped by the Text Extractor Web Service (TEWS).[5] The TEWS returns a list of entities (e.g., "Prime Minister Berlusconi", "Italy"), entity types (classes in an OWL ontology: e.g., Person, Country), and entity offsets with the

[5] This service was developed by another IC organization, which hosts it for multiple clients (including our application).

document. The TEWS also extracts some relationships (e.g., for an event) and returns entity offsets within the original document text. The TEWS outputs OWL markup describing the entities and relationships extracted.

3. The SMT receives a queue of ingested and normalized along with the OWL markup of any entities and relationships automatically extracted. The OWL markup produced automatically by the TEWS is automatically saved for storage in a knowledge base, the KOR (Knowledge Object Repository). This markup is higher volume (i.e., more assertions) than that produced through manual markup via the SMT, but is generally of lower quality and accuracy. The latter is due to errors made in the extraction of entities and their classification by type (ontology class).
4. An SMT user can then select a document and one or more markup templates (Section 3.2). Markup templates are used to describe events and other complex entities of interest mentioned in the document. The SMT can recommend markup templates to the user based on the entities extracted from the document. The types (ontology classes) for these entities are matched to the types of classes specified as legal values for template slots in a template's definition.
5. A user fills out ("instantiates") a markup template using entities from the TEWS, values he supplies, etc. The SMT's graphical user interface provides several ways to fill in templates (see Section 3.3). Uses may also choose to augment or change the type classification of one or more entities made by the TEWS.
6. Once a template is filled in, a user can save it. This causes OWL markup to be automatically generated. Currently this markup is not stored within the original web document but rather logically linked to the original web document through metadata (including the source document's URL) a we save on a per-assertion basis (via RDF's reification mechanism).[6]

A number of OWL-aware semantic web applications can process the OWL markup generated by the SMT. In our application, this markup is stored in the Knowledge Object Repository (KOR) for later exploitation by humans (via visualization, navigation, and query tools) and machine agents. The KOR is built on top of the Sesame triple store [2] and is one of the tools in ISX's Semantic Object Web framework [8].

Additional tools (such as Stanford's Protégé tool [12]) are used to author and validate the ontologies used. This markup can then be loaded into a knowledge base, such as our Knowledge Object Repository (KOR). Facts from the markup will be integrated with existing knowledge loaded from reference data sources (whose data has been converted to OWL). The KOR can be exploited by visualization, browsing, and navigation tools for human use and by software agents for knowledge discovery and other applications.

3.2 Markup Templates

As mentioned previously, the SMT's predecessors presented users with the complexity of all the classes and properties in a set of ontologies to use. Providing a

[6] In our IC environment, we are not allow to modify the source web documents.

usable interface for quickly visualizing, navigating, and selecting classes and properties from large, graph-structured ontologies proved difficult.

An innovation in the SMT was the use of markup templates to hide ontological complexity from users. For example, a user describes a meeting event by picking the class Meeting and selecting and filling out (allowable) properties. The values of properties could be literals (e.g., numbers and strings) or entities that in turn could have their own properties. This also makes document markup easier and faster. Markup templates can be viewed as somewhat analogous to database forms and database views, which serve to hide the complexity of multiple underlying database tables from a user entering data or performing queries. A database view can span multiple tables, much as a template can span multiple classes in one or more ontologies.

The use of templates comes at a cost of limiting the kinds of facts that can be expressed about the document. For example, to describe reports of meetings, a Meeting template might be defined with slots for participant, location, topic, start date, end date, etc. An SMT user fills out a slot by supplying a value, ideally by selecting one or more entities automatically extracted from the document by the TEWS. These slots are the *only* things a user can say about the meeting, for example. This is the expressivity limitation that is due to our approach of shielding a user from the full complexity of the OWL ontologies. If the user wishes to specify the weather during the meeting, he can either bring up a weather template (if available) or utilized a "power user" features that lets him add a single assertion (specified using terms from one or more ontologies) about one or more entities.

The SMT generates OWL markup from the filled in template, which, when loaded in the KOR, will result in a knowledge object (KO) of type (ontology class) Meeting. Other KO's may also be created. From a meeting template, for example, KO's and the links between them – including those shown in Figure 1 – are created.[7]

Template definitions thus contain the mappings from instances (KOs) and slots to classes and properties in one or more OWL ontologies. Thus if an SMT user supplies a value for participant (e.g., "Fred Jones"), a KO for that participant will be specified in the markup sent to the KOR. The KO will have properties that correspond to the slot values: e.g., filling slot Location with entity "San Francisco" will result in the KO for the meeting being linked to the KO for *SanFrancisco* (an instance of class *City*) via the property *hasLocation*.

Templates can also contain metadata (template author, version, etc.) and constraints. The latter can be used to populate and validate a template. Implicit constraints come from the ontologies referenced. For example, the ontology might specify that the property *hasLocation* (corresponding to slot *Location*) can only have a single value of type Location. Explicit constraints can state a relationship that must hold between the user-supplied values for slots: e.g., value for slot *Start Date* must be less than value for slot *End Date*.

[7] KO's for cities and countries, for example, will likely already exist in the KOR. In this case, a collection of heuristic matching techniques are used to match KO's asserted in the newly created markup with those already in the KOR. We term this process "co-reference resolution" between KO's. This involves comparing KO's potentially populated from multiple documents (and reference data sources). Entities within a *single* document are co-referenced using by the TEWS using synonym lists, anaphoric binding techniques from linguistics, etc.

Template definitions are stored as XML files and validated using an XML schema. These can be created and edited by trained subject matter experts using XML authoring tools or ISX's DTV tool.[8] We envision templates being created for individual production organizations, communities of interest, and community-wide use. Such templates can be used to enforce markup standards such as required document metadata, etc.

3.3 Semantic Markup Tool User Interface

Figure 3 shows the SMT's graphical user interface (GUI) in the midst of markup by a user for a test document (and test markup template) about a meeting between Italian PM Silvio Berlusconi and U.S. envoy James Baker on January 19, 2004. The upper right-hand (Document) pane shows the document with the entities extracted by the TEWS underlined (entities applicable to the selected slot are shown in red). A user can quickly check the output of the TEWS. The user can mouse-over an entity is to view its type (ontology class), shown as a tool tip. He can change its type or even designate a selection of text as an entity in the event the TEWS missed it. Thus, the user can optionally correct the output of the TEWS.

After reading the document, the user next selects a markup template. The SMT can recommend templates to the user based on the entities the TEWS found. These recommendations are presented in relevance-ranked order. The user can also choose a template that is not recommended. We anticipate that there will be about a dozen templates a user might use on a regular basis. Each template would have about 5-10 slots. This is in contrast to the direct use of the ontologies in which the user might have to select from among hundreds of classes and properties.

The template is displayed in the left-hand (Template) pane: e.g., a template for a Meeting. When a template comes up, the SMT tries to fill in as many of the template's slots as it can to reduce the burden on the user. For a given slot, the SMT matches its value type – an ontology class specified in the template definition – to the types (ontology classes) of entities from the TEWS. For example, for the slot *Participant* (of the Meeting template) can be filled by entities of type (class) *Person* or *Organization* extracted from the document such as "James Baker", "President Bush", "Prime Minister Berlusconi". If only one entity and appropriate type for a slot is found, the SMT fills it in as the default value for slot *Participant* (a lightning bolt icon is displayed next to the slot to indicate this). In the example, multiple entities could fill the *Participant* slot, so the SMT lists those (and only those) entities in a pull-down menu for value for slot *Participant* and the user can select from among these (or supply an alternative value by designating another entity in the document as a location).

For some slots (e.g., *Event Title*), the user can type in values directly. The SMT can validate these values generally: e.g., flag a non-date value entered for one of the *Date* slots. Some slots are required (designated by a "*"). Users can select "Unknown" for a value. Some slots can have multiple values (as specified in the template definition). In the example shown, slot *Participant* can have multiple instantiations (the user clicks on the + to add additional ones), each with a different value (i.e., a facility).

[8] The DIONE Template Versioning Tool (DTV) supports the authoring and automatic validation of templates against changing ontologies. ISX developed DTV as part of the ontology versioning work with Lehigh University for DARPA's DAML Program.

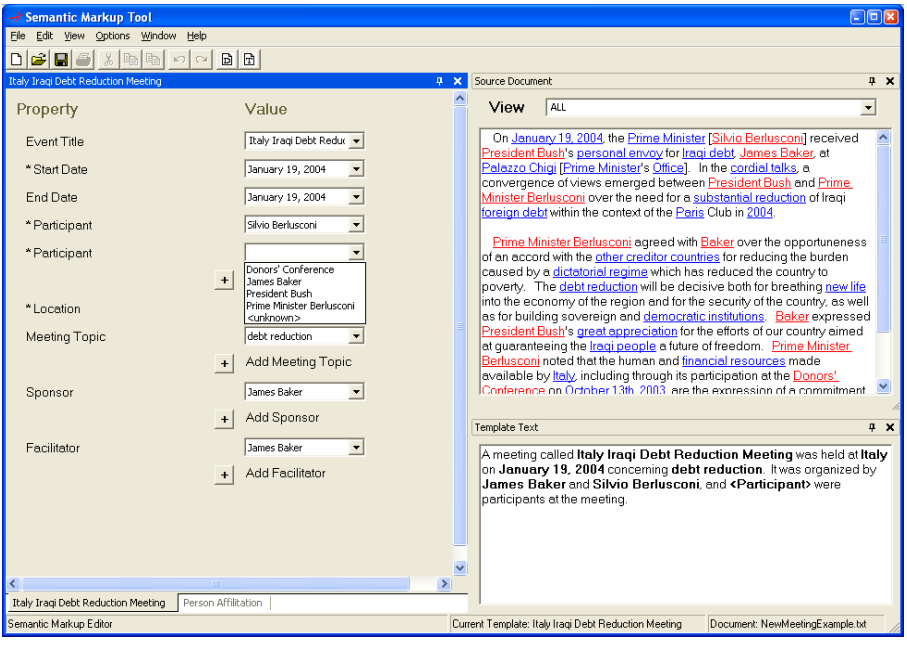

Fig. 3. Semantic Markup Tool's Main Graphical User Interface

The SMT tries to apply additional rules to fill in slots automatically. These include constraints specified in templates. For example, the Meeting template has the constraint mentioned above that the value for slot *Start Date* must be earlier than the value for slot *End Date*. Assume there are 3 date entities extracted from the document: "1/17/2004", "1/18/2004", and "1/19/2004". If the user selects 1/19/2004 from the pull-down menu for the value of slot Start Date, then the pull-down menu for the value of slot End Date will default to "1/19/2004" (since "1/17/2004" and "1/18/2004" are earlier than "1/19/2004" and hence invalid for *End Date* given the constraint). By managing the contents of pull-down menus and populating default values, the SMT tries to minimize user effort required to populate the template. More sophisticated business rules for computing default values and validating values filled in could be implemented. We are investigating the use of SWRL [5] for representing such rules, perhaps using the Jess [15] engine to execute them.

A user can fill in slots directly from the document view by right clicking on an entity. The SMT displays a list of slots that that entity could fill. For example, right clicking on the entity "James Baker" would list slots *Sponsor* and *Facilitator* as alternatives that could be filled by that entity. The SMT has no special domain (or linguistic) knowledge built in to allow it to determine that this entity might be more likely the value of slot *Facilitator* rather than slot *Sponsor* or vice versa. In some cases, the TEWS can apply linguistic rules to determine from the text describing an event (e.g., a Bombing) who the victim, perpetrator, etc. are. When such knowledge is available, the SMT can leverage it to automatically fill in one or more slots. Thus as text extractors improve, the SMT can leverage the improvements to populate more of the template automatically.

As a template is populated, the lower right-hand (Template Text summary) pane shows a stylized English text with placeholders corresponding to some of the template's slots. As slots are filled in, the text is fleshed out. A user can edit the sentences directly by typing into the sentence pane. A user can also fill in slots from this pane by right clicking on a slot name and selecting a value.

This provides a third alternative way (counting the template and document panes) by which a user can populate a template. We believe giving users these choices will allow them to use whichever method proves the easiest or fastest for them. The user can hide an unused pane, for example, if he finds it distracting. The sentences generated are stored with the document metadata as a summary to be used later (e.g., displayed in list of documents, etc.).

The user can instantiate multiple templates for a document. For example, the document might describe a meeting and a bombing, requiring two kinds of templates to mark up both events. Each template is shown on a separate tab panel. The user can easily switch between these tabs to go back and forth between filling out templates.

Some templates may be linked: i.e., the value of one template may determine the values in another template. For example, a Meeting template has slot Location. A user can link a Location Detail template to that slot to describe additional properties about the location besides just its name (e.g., perhaps its address, lat/lon, etc.). These additional properties are slots in the Location Detail template.

Once the user fills in the template(s), the SMT generates OWL markup using information from the template definition(s), which links slots to ontology properties, etc. The OWL markup generated can be quite sophisticated. This operation is hidden from the user. OWL markup from the TEWS (or another extractor) for entities is also saved. This is tagged as non-user-validated (versus markup specified through an SMT template).

For most uses, we anticipate the user will open a document, choose a template from a small set of frequently used templates, fill in a few slots (in addition to those the SMT auto-populates from the TEWS), and save the result.

The SMT is implemented in C# in Microsoft's .NET environment using Infragistics' GUI components.

3.4 Applications and Results

On several projects, we have demonstrated that the SMT can produce markup quickly that is more consistent, correct, and complete than our previous methods. Templates can enforce slot entry, helping to ensure the markup generated will be complete. The SMT can validate slots against constraints and business rules, helping to improve correctness. The entities a user puts in a slot must match the slot definition in type. The output of OWL is handled automatically, reducing the potential for syntactic errors, etc. Organizational and community standards can be embodied in markup templates to improve consistency of markup across users and organizations, respectively.

A trained SMT user can mark up most documents in well under a minute by filling in a template. In fact, in many cases the limiting factor on using the SMT tends to be the time for the user to read the actual document. The SMT has been vetted with several intelligence analysts, and their feedback incorporated into its design.

4 Related Work

Markup tools differ in the information they aim to capture (administrative and/or content metadata), their output language (e.g., HTML, XML, OWL, etc), and the level of automation provided. Several tools are embedded with common user applications to create markup as a by-product of authoring. These include Teknowledge's prototype MS PowerPoint-based tool (Briefing Associate) and MS Word-based tool (Semantic Word) from DARPA's DAML program that produce OWL markup [16]. Adobe's XMP uses RDF and metadata templates for metadata markup [1]. In.vision's Xpress Author for Word product integrates with MS Word to generate (XML) markup automatically [6]. XML publishing tools such as Arbortext's products also support metadata capture.

Table 1 compares several tools that use ontologies to generate markup.[9] The table was populated primarily based on documentation found at the various web sites. In some cases (e.g., OntoMat, Protégé, and SMORE) the tools were installed and run. Thus we apologize in advance for any inadvertent misrepresentation of the tools. The reader should refer to the web sites for the latest features. The tools include: (ISX), KIM (OntoText) [9], Melita (U. Sheffield) [3], MnM (Open Univ.) [13], OntoMat-Annotizer (U. Karlsruhe) [17], Protégé (Stanford)) [12], Semantic Word (Teknowledge) [16], SMORE (U.Maryland) [20], and SMT (ISX).

The table shows 3 sets of features including Input Parameters (unshaded rows at top), User Interface (shaded rows in middle), and Output (unshaded rows at bottom).

Most of these tools mark up web pages using OWL or RDF Schema ontologies. About half are Java-based. Some support the creation of ontologies on the fly during markup, including the finding and reuse of existing web ontologies. Some pull instances from a knowledge base (KB). Most allow either form-based specification of assertions (triples) and many support drag-and-drop from the document's text (and/or selecting text then selecting a class or property). Most support domain and range validation on assertions. About half interface with a text extractor to automatically find named entities in the text that can be used in assertions. Several tools (e.g., MnM and Melita) use learning techniques to improve automatic extraction/annotation. The tools differ in how OWL markup (typically containing individuals and their relationships) is output or stored.

Only 3 tools support templates, which facilitate the entry of assertions and can hide ontological complexity from end users. Protégé's templates are really custom forms that support the easy entry of property values for single (versus multiple) individuals. Templates can, as previously discussed, limit what can be expressed in the markup generated. To our knowledge, the SMT provides a unique combination of features including template-based markup leveraging the output of an automatic entity extractor, several GUI markup "modes", and maturity beyond "research-ware".

See, for example, [14] for another comparison of semantic annotation tools.

[9] Most of these were listed on http://annotation.semanticweb.org/tools/.

Table 1. Comparison of Related Markup Tools

Tool / *Feature*	**KIM**	**Melita**	**MnM**	**OntoMat Annotizer (V0.8)**	**Protégé (w/ Custom Forms)**	**Semantic Word**	**SMORE**	**SMT**
Version / Status / Availability	Plug-in (Free for Research)	Limited Distrib.	2.1 Open Source	V0.8 (Free)	3.1 Open Source	1.0 Alpha (free)	V5.0 (free)	V1.0 Prototype (GOTS)
Platform	MS IE Plug-in & Server	Java Client & Server	Java	Java	Java	Visual Basic plug-in to Word, Java	Java	MS .NET (C#) Application
Object of Markup	web page	web page	web page	web page	any (doc not visible)	MS word doc	web page	web page
Kind of Ontologies Supported?	KIM Ont. (RDFS)	yes but kind??	RDF, DAML+OIL	OWL	OWL	OWL	OWL	OWL
Ontology Creation Supported?	no	no	no	yes	yes	no	yes	no
Ontology Web Search Capability?	no	no	no	no	no	no	yes	no
Templates Supported?	no	no	no	no	yes (forms)	yes	no	yes
Instance KB Access?	yes	no	yes	no	yes	no	no	no
Pre-population of Assertions or Slots?	entities from Text Extractor (GATE)	entities from Text Extractor (Amil-care)	entities from Text Extractor (various)	no	no	entities from Text Extractor	no	entities & relationships from Text Extractor (various)
Entry/Editing modes	only auto-generated (named) entity annotations supported	text entity select;	select/click text	form, doc text drag/drop, OWL editing	form, copy/paste, wizards	doc text select/click	doc text drag/drop or select/click; triple formats	form, doc text drag/drop, text entity
Constraint Checking of Assertions or Slots?	N/A	N/A	yes (domain & range)	yes (domain & range)	yes (domain & range)	yes (domain & range)	yes (domain & range)	yes
Content of Markup Generated	individuals & doc metadata	individuals	individuals, & relationships	classes, individuals, relationships	classes, individuals, relationships	individuals & relationships	classes, individuals, relationships	individuals, relationships
Expressivity of Markup	all classes in the ontology	all classes in the ontology	all classes & props in the ontology	all classes & props in the ontology	all classes & props in the ontology (without forms)	all classes & props in the ontology	all classes & props. in the ontology	limited to classes & props in templates
Format of Markup	output stored in KB	OWL	XML, RFD, DAML+OIL	OWL (embedded in HTML)	OWL	MS Word file or XML	OWL	OWL (in separate doc)
Other Features		system learns annotation rules	KB population; learning to extract		many plug-ins provide additional functionality	markup within MS Word	supports HTML editing	generates doc summary text

5 Future Work

We are investigating the SMT in a number of other semantic web applications. For our primary IC application, SMT has not yet entered operational use, however. This is primarily due to a recent change of program direction to explore how far fully automated (yet "lower grade") markup can take us, given the high volume of documents that must be processed. Another issue to be resolved is organizational and concerns who in the current production workflow will be responsible for doing any manual markup (presumably on a high-value subset of those documents).

Other applications being investigated for SMT include the CAST project for an IC organization to provide tools for analysts that allow them to capture complex relationships in documents (via OWL) and use those relationships to organize the documents around their task context: e.g., current set of analysis tasks and hypotheses. The AFRL Effects-Based Operations-Center of Gravity Analysis (EBO/COG-A) project is planning to use the SMT to extract from documents information about the interrelationships of complex target systems to aid military planners. We have proposed using the SMT markup templates for querying a knowledge base of markup as well.

Proposed enhancements to the SMT include more sophisticated automation to select template and populate slots; integration of the SMT with a knowledge base to assist in template population and reasoning; a richer language (e.g., SWRL (Horrocks *et al*. 2004)) to express constraints; and numerous usability enhancements. We are working to handle additional document types without having to convert them to text first (e.g., HTML documents).

We have begun to extend the SMT to handle the markup of images. In this version of the SMT, the TEWS-extracted text entities are instead image "primitives" extracted by feature extractors (perhaps with assistance from human photo/imagery analysts): e.g., Image X contains a truck, road, security fence, and 4 persons. The SMT could then be used to instantiate a template by filling in slots with these extracted features: e.g., a template for an arms delivery event. The template would assign roles (via slots) to the components of the image: e.g., arms dealer, truck driver, security goon, and arms recipient. A template could markup several related images to show different temporal slices of an event. The OWL generated from the template would support image retrieval, real-time monitoring, and other applications.[10]

6 Conclusions

This paper has presented the SMT, a template-based tool for the markup of SMT documents. The SMT provides a hybrid manual-automatic markup tool for the rapid use specification of relationships in the document content in addition to the entities extracted automatically. Via templates, the SMT generates OWL markup that is more complete and consistent than previous methods. The SMT has been vetted with several intelligence analysts and is being used on a several pilot applications.

The XML-based representation of templates is general purpose. Template hide ontological complexity from end users, enable automation (using constraints and rules), and generate correct and complete OWL markup. End users, leveraging the output of an automated text extractor such as the TEWS, can fill in templates quickly. In addition to OWL markup (using the RDF/XML serialization), the SMT could be easily modified to generate XML markup. Additional tools to support the easy authoring and validation of templates (against evolving ontologies) have been developed.

The machine-understandable OWL markup generated by the SMT can be exploited by a growing set of semantic web tools and applications to provide improved search, discovery, and knowledge management capabilities. Providing users with usable tools

[10] There are several other tools that image markup: e.g., PhotoStuff [12].

for human augmentation of semantic markup and with incentives to use them – both in direct value-added functionality from the markup tools themselves and from exploitation tools to be used later – will be essential for the semantic web to move towards widespread adoption.

Acknowledgments

The authors wish to thank the IC sponsors of this work. Much of this work was done under a subcontract to ISX Corporation from Computer Sciences Corporation (CSC). Dr. Joseph Rockmore of Cyladian Technology Consulting provided valuable inputs into the design, as did Gary Edwards, Mark Hoffman, and Joe Roberts of ISX Corporation. Previous versions of the SMT were funded by DARPA under the DARPA Agent Markup Language (DAML) and benefited from the insights of Dr. James Hendler, Mr. Murray Burke, and Dr. Mark Greaves. Dr. Greaves has sponsored the work on the DIONE project which has supported the SMT work through the development of template authoring and validation/versioning tools (among others). Several of those versions were developed by NGIT. Dr. Jeff Heflin (now at Lehigh University) developed the SHOE Knowledge Annotator which was one of the first ontology markup-based tools for the Web, as part of the pre-DAML work done by the University of Maryland under the direction of Dr. James Hendler.

References

1. Abobe, Inc. Extensible Metadata Platform (XMP). http://www.adobe.com/products/xmp/main.html.
2. Broekstra, J. et al., 2002. Sesame: A generic architecture for storing and querying RDF and RDF schema. In *The Semantic Web - ISWC 2002*, volume 2342 of *Lecture Notes in Computer Science*, Springer Verlag.
3. Dingli, A. (University of Sheffield). Melita. http://www.dcs.shef.ac.uk/~alexiei/WebSite/University/Melita/index.htm
4. Heflin, J., and Hendler, J., 2000. Dynamic Ontologies on the Web. In *Proceedings of American Association for Artificial Intelligence Conference (AAAI-2000)*. Menlo Park, Calif.: AAAI Press.
5. Horrocks, I. *et al.*, 2003. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. World Wide Web Consortium Submission, 19 November 2003. http://www.daml.org/2003/11/swrl/
6. In.vision Research. Xpres Author for Word. http://www.invisionresearch.com/xpress.htm
7. Kalyanpur, A. et al., 2004. Hypermedia Inspired Ontology Engineering Environment: SWOOP. In *Proceedings of the International Semantic Web Conference (ISWC)*.
8. Kettler, B. *et al.*, 2003. The Semantic Object Web: An Object-Centric Approach to Knowledge Management and Exploitation on the Semantic Web. ISX Corporation Whitepaper. Presented as a poster at the 2nd International Semantic Web Conference. http://www.semanticobjectweb.isx.com
9. Kiryakov, A. et al. (Ontotext). KIM Semantic Annotation Platform, http://www.ontotext.com/kim
10. Lockheed Martin, AeroSWARM http://ubot.lockheedmartin.com/ubot/hotdaml/aeroswarm.html

11. McGuinness, D. and van Harmelen, F. 2004. Web Ontology Language (OWL) Overview. World Wide Web Consortium Recommendation, 10 February 2004. http://www.w3.org/TR/owl-features/
12. Noy, N.F., M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen, 2001. Creating Semantic Web Contents with Protege-2000. *IEEE Intelligent Systems* 16(2):60-71. http://protege.stanford.edu
13. Open University. MnM. http://kmi.open.ac.uk/projects/akt/MnM/
14. Reeve, L. and H. Han, 2005. Survey of Semantic Annotation Platforms. In *2005 ACM Symposium on Applied Computing*.
15. Sandia National Labs. Java Expert Systems Shell (JESS). http://herzberg.ca.sandia.gov/jess/
16. Tallis, M. et al., 2001. The briefing associate: A role for cots applications in the semantic web. In *Semantic Web Working Symposium (SWWS)*, Stanford, California, USA. http://mr.teknowledge.com/daml/software.htm
17. Univ. of Karlsruhe. OntoMat-Annotizer. http://annotation.semanticweb.org/ontomat.html
18. Univ. of Maryland at Baltimore County. Swangler. http://swangle.projects.semwebcentral.org/
19. Univ. of Maryland at College Park (MINDSWAP Lab). PhotoStuff. http://www.mindswap.org/2003/PhotoStuff/
20. Univ. of Maryland at College Park (MINDSWAP Lab). SMORE. http://www.mindswap.org/2005/SMORE/