Applying the Semantic Web as a Writer's Tool

Rick Thomas

Independent Software Developer

+1 404-966-5658

saaw2006@evenview.com

ABSTRACT

The process of writing and the practice of Semantic Web (SW) annotation are similar: each takes ideas and interprets them into progressively refined symbols. Drawing on this parallel, they may be mutually improved, conceptually and by the use of software tools. As an adjunct to writing, the SW will be more than a distributed data scheme and will be part of the creative process. Supported by the Semantic Web (SW), writing can be more flexible and include richer, non-linear structure.

The key insight is that the basic operation of dividing a resource and commenting on the relations exposed by the division recurs throughout the process of writing. This operation is rich enough semantically to serve as the basis of interpreting and annotating text for use in the SW.

This paper describes an experimental tool that supports this operation for a writer. Intentionally, the tool is minimal. The writer produces text while organizing it at the paragraph level into related, web-addressable boxes. The spatial relations of these boxes express a few general semantics, such as list membership, likeness and currency. Thus the writer annotates with informal structure, which enhances the use of annotation tools by giving identities to and relations among more narrowly focussed portions of text. The notes and relations established are useful for the writer's access to the text and provide a scaffold for revisions.

In this basic use, overt URIs and RDF are avoided, as is embedded markup for data and links. In the implementation, though, boxes also contain the normally hidden RDF/N3 that describes box properties and relations. For the writer comfortable with RDF, the boxes can be used for full annotation.

Looking forward, this approach lends itself naturally to support for finer-resolution collaboration. Perhaps such a technique will help writers to discover and formulate relations and thus allow texts to interoperate in the future SW.

1. INTRODUCTION

We often think of the Semantic Web as a database of metadata describing resources. This view will be challenged, though, as we progressively divide those resources and model their internal relations. For text resources, metadata will outgrow the text itself as we first treat paragraphs as resources and incrementally represent the explicit meaning of sentences.

Explicit language semantics has been pursued in many efforts: for translation, for literary studies, for legal argumentation and in computational linguistics. But there have been no common standards by which these techniques could interoperate. The SW, extended for the task, may serve this purpose and become the “data bus” for modeling text. But the SW is not yet able to serve this role. Its expressivity has been circumscribed to avoid logical undecidability. And its method of reference would be quite unwieldy if used at the word level.

As a step toward this future, this paper sketches an approach for generating and using ontologies in writing. Basically, the writer’s division of a text into paragraphs is a key opportunity to capture the meaning of the work-in-progress. Each such division implies relations among the paragraphs that may be made explicit. This structure, plus the writer’s comments about the divisions, become a scaffold for semantic annotation.

Use of this technique depends on software that can manage the newly explicit relations in the text, while hiding the technical details. Such a tool will offer direct benefits to the writer for organizing the work-in-progress while revealing resources that may be usefully addressed by the SW.

2. WRITING AND THE SEMANTIC WEB

A written work is meaningless in the SW until semantic annotations describe its otherwise opaque content. Metadata may describe its author and related documents; keywords and references may be extracted and indexed for easy access; commentary can be added to paragraphs in the text. But only with the use ontologies specific to the domain of the text can it be said to have meaning in the SW.

There is a gulf between SW annotation and writing: Annotation works with macro-scale resources and ontologies that are general-purpose and fixed. Writing addresses micro-scale resources and needs ontologies that are content-specific and formative.

Ontologies for annotation come from many processes: 1) conventional data ontologies describe embedded data within a resource such as people and places; 2) annotation ontologies associate comments and bookmarks with general resources; 3) general content ontologies mark the relations among portions of the work; 4) generic ontologies are used to label resources freely, which is one way to think of social tagging; and 5) emergent ontology models the domain with structure generated incidentally from some mass process.

These ontologies are similar in that they are external to the work and fixed relative to it. (This assumption is relaxed below.) On the other hand a formative ontology is initially implicit in the work and must be drawn out to model the content as it is developed.

The work is annotated incrementally, starting with basic ontologies and by gradually developing and applying more specialized ontologies. This modularity is the great strength of the SW. Data-oriented ontologies stand with ontologies that model the work directly. Multiple ontologies describe different views of the same resources, and they work together because they describe a network of relations linked by these pivotal resources. More classes of resources and more relations are distinguished among the resources, forming a network of meaning.

But if we require that the writer does nothing with explicit ontology, how can we capture structure that can be formed into peer ontologies? The premise here is that as the writer divides the initial opaque text and comments on the divisions, the significant structure of the work is revealed. The comments and relations are then used to guide the formation of provisional ontologies and the application of SW annotations.

If this process of revealing implicit meaning is to be guided by SW technologies, we need a combination of natural language and the SW that offers the best of both and makes them mutually supportive. Three areas must be addressed: expressivity, reference and usability.

A writer commands natural language, a system of unlimited combinatorial expression. Language freely mixes resources, statements, meta-statements, and ontology, though awkwardly and unsystematically. The process of writing depends extensively on knowing the provenance, standing and relations of statements at a detailed level. The language of the SW is, by contrast, limited to a subset of first order logic. Resources are separate: statements and ontology are about resources, while the conventions to treat statements as resources are ad hoc.

But this approach to using the SW for writing immediately requires friendly reification. The writer divides the text and comments on the now explicit relation between two paragraphs. In other words, the elemental structure involved is statements about statements. While this brings the expressivity of the SW in line with language's all-in-one nature, it compromises the logical design of the SW. Yet much can be done without the use of inference and perhaps in practice the use of reification can be isolated.

Reference in natural language is highly constrained because its origins are verbal, thus linear. Expressing complex structure clearly is difficult: reference is either relative and brittle or precise and unwieldy. Pronouns are ambiguous, clause structure is confusing, and cross reference is ugly and inconvenient. The SW solves these problems by using universal identifiers, URIs. This consistency allows for ubiquitous definitions, relations, comments and paraphrases (once its overhead is accepted). The SW is good with structure, but inelegant (to say the least) for linear expression. The approach here is to present the writer with local reference like natural language, but to undergird it with absolute URIs.

Writers cope with many fragments and much iteration on the way to a finished, linear work. The SW can easily represent the complexity, but this would be of little use without ways of manipulating writing-specific structures. This design must present views of the work that are clear and persistent, yet reflect the detailed structure of the content.

3. ANNOTATION + WRITING = SCIENCE

Before discussing design, it is helpful to explore the flexible and dynamic use of ontology by analogy with the scientific method. This analogy will expose the implicit activities in annotation and writing. In turn, that will illuminate the synergies between writing and the SW to guide the design of software.

The scientist works in several phases: observation, hypothesis formation, prediction of results and experimentation to test the predictions. Observation begins by selecting a manageable domain and collecting data that characterizes it. Hypotheses are formed to describe the regularities within the observations. Predictions project the data and hypotheses to new outcomes and experiments test the hypotheses and contribute to the stock of observations.

Consider the parallel with annotation. Annotation starts with an interest in a domain of resources (Observation). Ontologies are chosen that will structure that domain (Hypotheses). Annotation is applied and presupposes that the annotation is correct (Prediction), and that is confirmed by a consensus on its accuracy or usefulness (Experiment).

The analogy brings out a lot that is implicit in the simple act of annotation. It first requires an active choice of resources. This selection, which may be incorrect and incomplete, colors all the annotation. It requires a distinction to be made and justified by the annotator.

Annotation requires a choice of ontologies. Ontology is usually taken to be fixed prior to its use in annotation. Sometimes though, the annotator may not find an ontology that fits well, or may find competing ontologies, or may be working in an emerging domain, or may just prefer a casual approach. As annotation proceeds the ontology is tested, and may require changes to refine terms and add new relations, or may be replaced altogether. Thus ontology is formative in general. Likewise, each annotation is conditional and subject to revision.

All these steps depend on the care, clarity and curation of the annotator. In other words, the ontologies, the annotations, and the users' expectations are a system that must be kept tuned. These stand as important criticisms of the SW, but this is not intended to be dismal. Many domains are intuitive and stan-

dard, but fundamentally, semantics are negotiated in a *process* of annotation where divisions are made and conventions about those divisions are formed. Thus SW implementations should support revisability.

This potential for variation helps to understand writing, which is by comparison unstructured. The writer reorganizes and rewrites, iteratively. Source materials and drafts are associated and ranked, and are folded together with the writer's thoughts.

Compared to science then, writing begins with a selection of sources as context and inspiration (Observations). A plan for the work defines terms, states themes and sketches arguments (Hypotheses). The writer then drafts text to support the plan, drawing support from the context (Prediction). Reading the draft is the test that leads to a revised plan and a new draft (Experiment). (Editors and readers are essential participants, but the focus here is on the sole writer.)

The analogy illuminates the planning and testing phases of writing. In planning, the writer intuitively organizes the domain with borrowed, discovered and invented regularities. It is like ontology, but it initially has only one instance - the text itself. It is not necessarily explicit or shared. If it is explicit, it is expressed informally within the text.

Each new draft influences this intuitive ontology, which in turn influences the next draft. This formative understanding of the domain in effect carries the work's meaning from version to version: text is held constant while the writer ponders meaning; meaning is held constant while language is changed. This is a process of interpretation, where the meaning is represented with one set of symbols and then recast to new symbols - more clear, more familiar, more appropriate. This evolution is central to writing. As for general annotation, the process generates distinctions that are the basis of formative ontology and may also be incorporated into the text.

This is not to suggest that a writer will model the work explicitly (though for some works, like technical papers, this may be feasible and desirable). Even so, a tool can help manage the distinction between plan and text. If the outlines of the intuitive ontologies are revealed by the writer's actions, they can be caught, refined and support the fluid process of writing.

4. AN INTERFACE FOR WRITERS

The strategy for the design is simple: 1) Provide a way to partition the text of a work-in-progress into small resources. 2) Describe the resources in terms of a simple ontological structure. 3) Capture the structure of the relations among the resources and use it to guide semantic annotation.

The interface is simplified to one primary element - a box, which contains text (and hidden RDF) and which is presented on a page with other boxes. As far as possible the interface is limited to editing text in a box and positioning boxes to indicate relations.

The writer reorganizes, ranks and associates text using the boxes as proxy, and then rewrites within boxes while referring to related text. This provides a frame for the iterative writing process.

Boxes serve four purposes in the design: text editing, SW resource identity, spatial relations representing semantics and spatial relations to engage the user's visual thinking.

4.1 Text editing

Text editing is in a series of vertically stacked boxes, somewhat like writing in the cells of a spreadsheet or outliner. These boxes are in a list relation. In addition, boxes may be moved freely within the page. Two boxes may be related, which encompasses their spatial relation and a comment.

One of the goals of the design is to understand the interplay between text editing in boxes and short-range reorganization of boxes. Boxes outside the linear flow of text provide a convenient way to store snippets, but the mechanics of editing are still less convenient than a text editor. Support for moving text between boxes based on box relations would be helpful.

4.2 Semantic Web resource identity

Boxes are optimized for a division of text to the paragraph level, but not finer to the word level. The box is a proxy for its contents and so only relations to the box are available represent relations to the content. This compromise avoids references directly into the text - a text that is frequently changing. This limits modeling of details and may be ambiguous, but this is moderated by the limited scope of the box. (RDF within the scope of the box may be used but this is not supported by the interface and may be difficult to maintain.)

On the other hand, at the paragraph level, relations addressing the whole resource may be accurate enough, so the need for links embedded in the text is reduced. Giving up precise location of reference yields a simpler, consistent structure and easier editing. The reference is narrower than, say, the page reference in a book index, but not as narrow as we are accustomed to with an embedded hypertext link target.

As the primary resource in this SW application, boxes have URIs, but they don't appear in the user interface. The writer recognizes a box by its text and its individual appearance - shape, position on its page and relation to other boxes - and references a box by gestures toward this visual representation.

4.3 Spatial relations for semantics

Persistent spatial relations of boxes on pages are associated with logical relations. These relations are made by user gestures. Two boxes are related by their relative position, and a comment applied to the relation in the form of a third box. These relations are simple and general in the interest of refining the user interface gestures and learning how expressive a simple interface can be.

There are two relations: List and Like. List locates a box in a sequence of boxes. Visually, boxes are connected bottom to top down the page, making explicit the normally implicit relation between paragraphs. Like is shown by proximity and stands for any relation between boxes (other than List). Unlike graphs that depict RDF there are no arcs that show the relation. Instead, pairs of like boxes are be highlighted together and a third box commenting on the relation is shown. The comment

text can say anything (and its RDF may detail the semantics).

The comment box (which may also be used with the List relation) is the key to capturing the structure of the work as a scaffold for annotation. The relation between boxes is a stand-in for any possible RDF statement. The comment is a stand in for any possible ontology.

It is worth noting that the boxes provide a context for both text and RDF, useful for controlling scope for search and inference as well as for text indexing.

4.4 Spatial relations for thinking

Boxes are arranged on a page, implemented as a web browser tab. Pages are organized as notebooks in the order that they are created. The page is a neutral presentation area for boxes for assembly of the work. It is also a boundary for cognitive scope, a working context. The unique visual layout of its boxes engages the writer's spatial memory and reasoning. Boxes also give the text a stronger identity by location. Even if the text scrolls within its box, the box gives a better spatial cue than remembering that a paragraph is, say, two thirds through the document. It adds an additional, natural means of recall and orientation.

The original impetus for this application was to import the content of handwritten notebooks into the computer. Scanning and transcribing is easy, but capturing the relations intended by sketches and side notes lead to this approach. The spatial relations of these notes imply semantic relations.

When the box layout represents a physical page the number of relations is limited and the layout can be fixed until the user changes it. This persistence is important for later recall.

On the other hand, this annotation approach leads to the use of many more relations than resources. To work within the page, temporary layout transforms are needed. Managing these transforms is the most difficult part of this design.

Here are several cases: 1) With a large number of small boxes there may not be enough room to see the text when editing. In this case the current box is enlarged while surrounding boxes keep their same relative positions. 2) A box may be in more than one List and have many Like boxes so the relations would be obscured if they are shown all together. Temporary or permanent selective views allow focus. 3) Comments on relations between boxes are placed on a different plane and are shown only when the boxes are addressed, unless the box is “reified” as a permanent box on the page. 4) A query of either box text or RDF yields a collection of boxes that are displayed together, on a new page if needed. 5) Importing data presents a similar problem, for example, a directory of links and files with associated metadata.

5. IMPLEMENTATION AND DEMO

A prototype of the user interface is implemented using Javascript in the Firefox browser. Python in a local server implements the RDF processing and text indexing.

The text is formatted with a plain text markup syntax, which is extended to carry RDF/N3. Preprocessing is used to reduce repeated syntax, to expand references to box and page, and to convert some simplified notations to RDF/N3.

Current implementation is experimental, though useful, and is being used to refine the user interface design and concept.

6. CONCLUSION

This project takes a step toward a SW for less structured activity. The concepts and prototype serve to explore how SW machinery can stay invisible and still help the writer.

The key insight is that the basic operation of dividing a resource and commenting on the relations exposed by the division recurs throughout the process of writing. This operation is rich enough semantically to serve as the basis of interpreting and annotating text for use in the SW.

Looking forward, this approach lends itself naturally to support for finer-resolution collaboration. Perhaps such a technique will help writers to discover and formulate relations and thus allow texts to interoperate in the future SW.

7. REFERENCES

The influences for this work are too numerous and diffuse to acknowledge in a small space. Also, important prior work has no doubt been neglected. In the interest of correcting these omissions and continuing to improve these ideas, online references are available at http://www.evenview.com/saaw2006/

Topics include: Emergent and formative ontologies; Contexts and reification; Merging and ontology; Tools for writers; Evolutionary epistemology.

8. ACKNOWLEDGEMENTS

Thanks are due to the reviewers of this paper. They have suggested important corrections and clarifications.