Simplifying knowledge acquisition from end-users on the Semantic Web

Max Van Kleek
MIT CSAIL
32 Vassar St.
Cambridge, MA 02139 USA
emax@csail.mit.edu

Michael Bernstein
MIT CSAIL
32 Vassar St.
Cambridge, MA 02139 USA
msbernst@mit.edu

Paul André
Electronics and Computer Science
University of Southampton, UK
pa2@ecs.soton.ac.uk

Mikko Perttunen
University of Oulu, Finland
mikko.perttunen@ee.oulu.fi

David Karger
MIT CSAIL
32 Vassar St.
Cambridge, MA 02139
karger@mit.edu

mc schraefel
Electronics and Computer Science
University of Southampton, UK
mc@ecs.soton.ac.uk

Abstract

In this position paper, we argue that improved mechanisms for knowledge acquisition on the semantic web (SW) will be necessary before it will be adopted widely by end-users. In particular, we propose an investigation surrounding improved languages for knowledge exchange, better UI mechanisms for interaction, and potential help from user modeling to enable accurate, efficient, SW knowledge modeling for everyone.

Keywords

Semantic web, user interaction, knowledge acquisition and access, natural language interfaces, guided input

ACM Classification Keywords

H5.m. Information interfaces and presentation, H5.2 User Interfaces

Introduction

The success of Web 2.0 in turning the web into a massive participatory medium has demonstrated how making content contribution easier and more accessible to the general public can revolutionize a medium, making it pervasive and opening it up for new applications previously unimagined. We believe that if the Semantic Web is to enable every individual to participate in the creation and use of a vast ecosystem of information and new applications hitherto unimagined, we must attempt to make it possible for everyone to easily and effectively contribute and use knowledge from it. Furthermore, substantial evidence suggests that reducing the "input and access bottleneck" of information tools can improve a tool's perceived usefulness even among the most skilled knowledge workers. This has been observed repeatedly in several domains, including, Personal Information Management (PIM) [1, 2], physicians' record keeping tools [3], and in journalists' investigative processes [5].

Today, we see knowledge acquisition and access (KA&A) on the SW still a particularly labor-intensive process with the current tools. We believe this stems primarily from the following problems:

HIGH KNOWLEDGE BARRIER - Many "end-user" SW KA tools such as [10] were designed with knowledge engineers in mind - who are comfortable with terminology and concepts related to representational formalisms such as description logics. We argue that while the expressivity of ontology languages and description logics allow skilled information architects to construct powerful representations for building complex systems, these concepts are foreign and unsuitable for most end-users.

ONTOLOGICAL CONSIDERATIONS FIRST, INSTANCES SECOND - Most existing SW tools for encoding knowledge either assume that the user will be writing against a set of pre-existing ontologies, or that the user will construct the ontology herself prior to encoding instances. We argue that putting such ontological considerations first impedes capture by requiring substantial work on the part of the user upfront to locate, understand, and figure out how to transform their knowledge into representations suitable for the set of existing ontologies they wish to use, or, alternatively to design appropriate ontologies from scratch abstractly prior to encoding knowledge. Both activities are challenging today on the SW even for experienced knowledge engineers.

THE NEED TO BE FULLY EXPLICIT - In addition to grounding all knowledge in ontologies, these tools assume the user will explicitly and completely specify all knowledge in the SW as unambiguous, logically consistent statements.

In this paper, we argue that the next generation of SW KA&A tools should make primary considerations surrounding barriers to use – seeking out solutions that include more users, and reduce time and effort required to access and create knowledge. In particular, we propose a three-fold attack to this problem: first, considering new languages for expressing knowledge that are more natural and accessible to more people; second, exploring interface-level user interaction mechanisms for accelerating knowledge exchange between humans and SW KBs, and finally, investigating modeling mechanisms to provide make knowledge exchange more efficient, such as by reducing the amount of explicit information that needs to be exchanged with every interaction. We briefly go into each of these dimensions next.

Dimensions of knowledge capture: Language, Interface, and Modeling

End-user tools for letting users create new content for the SW (as opposed to exposing content from existing legacy databases, a topic we do not address here) have taken roughly two approaches. The more common has been the "graphical knowledge workspace" approach (such as [9,10]) which consist of GUI-rich environments for the construction of SW schemata (ontologies) and instances. The other approach has focused on transforming user-generated textual or other linguistic representations of knowledge into forms suitable for the SW.

In this paper, we focus on this latter approach for several reasons. First, text is already used for the communication of knowledge everywhere, and is easy for humans to store, create, and manipulate, whereas many people have never seen (and would possibly be intimidated by) graphical ontology and instance editors. Second, language is often used to convey extremely heterogeneous information, and is therefore designed to be versatile and efficient simultaneously, unlike database data-entry UIs, which are typically optimized for either versatility (i.e., graphical instance construction) or for facilitating encoding of many regular instances of the same type (i.e., data entry forms). Finally, as we discuss in the remainder of this section, focusing on text lets us examine a whole continuum of languages varying in natural-ness, flexibility, and machine interpretability. (For a more complete argument of text vs GUI approaches to conveying knowledge, please see [11].)

1. Languages for communicating knowledge: Balancing interpretability, flexibility, and naturalness
Previous efforts surrounding using languages for conveying knowledge to the SW have taken a wide variety of approaches. On one hand, artificial structured languages have been created that make it easier for people to read and write RDF verbatim, such as Notation3 (N3). N3 is easily machine-interpretable, but requires an understanding of RDF, ontologies, and exacting precision to write. On the other hand, many efforts have surrounded extracting knowledge directly from unconstrained human natural language. [9] This approach saves users from having to know anything about RDF or the target ontologies, but is still very error-prone due the inherent complexity of natural languages.

Between these two extremes, many have proposed a middle ground -- simplified languages designed to resemble natural languages, but which are generally easier for machines to interpret [4,8]. This has created an enormous design space, which has shown great promise but (as yet) no structured systematic exploration. We argue that in order to proceed, we should identify characteristics similar and different in each of these languages, and begin towards an investigation of how particular language features correlate with end user perception, flexibility, and ease of interpretability.

Our exploration in this space have surrounded a set of simple data languages we call *pidgin languages* to convey that they borrow aspects of structured and natural languages in order to serve as common modes of communication between humans and machines.

	description	Example
tame pidgin	hand-written grammars for common domains, with semi-open SW-KB defined lexicon, and support for nested expressions. Not user-extensible or re-orderable.	`Meet with Jane phone 617-123-4567 tomorrow at diesel cafe about SWUI submission`
clay pidgin	User-defined N3 macro language using "means" templates written by the user. Support for nesting. No re-ordering clauses.	*Template:* `"meet when with whom about what" means [ a :Meeting; vcal:start "when"; xcal:attendees "whom"; xcal:description "what"].` *Example:* `meet 3pm with jane smith about swui.`
n3+res pidgin	N3 with entity and property and value resolution. Uses a colon or dash to delimit multi-word properties from their values, and semicolons to delimit clauses.	`swuimtg a Meeting; starts at: 3pm tomorrow; with jane; location Diesel Cafe.`
sloppy pidgin	"sloppy parsed" to allow out-of-order matching and recursive nesting of typed templates	`jane 3pm diesel café`

Table 1. The *Pidgin family*, simple languages for expressing knowledge and their properties, introduced in [11].

Rather than invent anything radically new, we began by re-implementing several of the ideas suggested by previous artificial languages and by adding “small tweaks” to languages such as N3. See Table 1 for a few examples. From these pidgins, we identified a few design features we believe are the most important:

- *General/ontology*-specific – whether the language contains domain specific representations (e.g., events, contacts) or generic entities and relations

- *Resolution of entity and property names* – the ability for the user to refer to entities using short/familiar names instead of only by URI.

- *Literal type deduction* – the ability to automatically determine the types of literal values without extra work by the user, e.g., to parse relative dates, coerce numeric values

- *End-user extensibility*– letting the user extend the language to new forms

- *Nesting of expressions* – allowing for the embedding of one statement within the clause of another

- *Reorder-ability/Optional clauses* – supporting reordering/optional clauses

- *(Lack of) Mandatory delimiters* – not requiring the user to adhere to strict syntactic rules

- *(Lack of) Syntactic ambiguity* – whether the strings in the language can have multiple valid interpretations

Although we lack empirical statistics regarding the relative importance of these (suggestions on how to capture such statistics are welcome), based on informal feedback from users and a trial deployment of Jourknow [11], we have identified that a number of these features seem to be essential if not highly convenient for most users. In particular, supporting familiar references to entities and properties, and literal type deduction should be considered mandatory. We noticed that there is often a delicate tradeoff between naturalness and unambiguous interpretability. For example, syntactic delimiters were seen in general to be somewhat onerous (e.g. requiring quotation marks around literal expressions); however, relaxing the syntactic requirements too far often resulted in an explosion in syntactic ambiguity – which is perceived to be far worse.

A study by colleagues [6] demonstrated that being flexible regarding word/phrase order in expressions might be important as well; their study revealed that nearly 50% of expressions entered by users in their sloppy-programming [7] based language were out of order in some way, despite immediate graphical assistive feedback.

2. User interface mechanisms for assisting input
Since most artificial languages have rules (grammars) that restrict valid statements to very limited subsets of natural language, users are likely to stray often outside the rules of the language without additional guidance. Furthermore, many of the desirable capabilities (e.g., named entity resolution) require the language interpreter to cope with ambiguity, i.e., to determine intended interpretation. In either case, forcing the user to have to correct expressions post-hoc could greatly hinder the experience. For these reasons, we perceive designing user interface mechanisms that, at capture time, guide the expression of statements and allow users to quickly and easily resolve ambiguities as they arise as being critical to the exploration process.

In particular we see the following roles of assistive UIs for artificial languages:

- Staying within the language
- Interactive entity/property disambiguation
- Input acceleration - (predictive auto-complete)
- Feedback - (displaying the interpreted result as confirmation of successful capture)
- Enforcing consistency - (preferring expressions similar to those made in the past)

We have implemented simple predictive auto-complete entry boxes for various pidgins in Jourknow, and have found that it is a familiar metaphor for most users today due to the semblance to the simple auto-complete mechanisms present in many web browsers.

3. "You know what I mean" Applying user modeling to aid disambiguation
Humans use contextual and historical information fluidly in conversation to help disambiguate each others' utterances and entity references. Using this as inspiration, we have begun investigating ways that KA could be made smoother by taking into account past interactions and aspects of the user's context, such as location, activity, and time during interpretation. In

particular, we are first looking at ways of improving the entity and interpretation disambiguation mechanisms mentioned previously to provide better defaults/ and rankings that are most likely to correctly prioritize the user's intended meaning. In one approach we are pursuing, we simply rank interpretations according to how similar the user's situational and activity contexts are (perceived through [12]) at the time of capture to those of past instances that they captured information. The intuition behind this ranking heuristic is that references to real-world entities (e.g. people, places and things) tend to be relevant to particular situations and less to others; for example, "John" at work is likely to refer to a different person than in another context. We are currently evaluating this heuristic and determining which aspects of situation are most useful for entity disambiguation.

Conclusion

Although significant challenges remain before knowledge creation and access on the SW becomes simple enough to be as routine for everyone as accessing the web, we believe that such a goal is both within reason and essential for the web's evolution. We hope that by providing one possible research agenda towards fulfilling this goal and starting to pursue it ourselves, we have and will continue to contribute to its eventual achievement.

Acknowledgements

We would like to thank Ora Lassila, Deepali Khushraj and Jamey Hicks at Nokia Research Center Cambridge. This research is sponsored by the Nokia-MIT alliance, and is a Web Science Research Initiative (WSRI) collaboration.

References

[1] Bedersen, B. Interfaces for Staying in the Flow. *ACM Ubiquity*, 5. 28 (Sept. 2004).

[2] Bellotti, V., Duchenaut, N., Howard, M., and Smith, I. Taking email to task: the design and evaluation of a task management centered email tool. *In Proc.* CHI 2003, ACM Press (2003), 345-352.

[3] Berg, M. *Rationalizing Medical Work: Decision Support Techniques and Medical Practices*. MIT Press, Cambridge, MA, 1997.

[4] Bernstein, A., and Kaufmann, E. GINO – Guided Input Natural Language Ontology Editor. *In Proc.* ISWC 2006, LNCS 4273, Springer 2006, 144-157.

[5] Blandford, A., and Green, T., Group and Time Management Tools: What You Get is Not What You Need. *Personal Ubiquitous Computing*. 5, 4 (Jan. 2001), 213-230.

[6] Chou, Vikki. *Inkey: Internet keyword commands with user feedback.* M.Eng Thesis, MIT, Feb 2008.

[7] Little, G. and Miller, R. C. Translating keyword commands into executable code. *In Proc*. UIST '06. ACM, New York, NY, 135-144.

[8] Livingston, K., and Riesbeck, C., Knowledge Acquisition from Simplified Text. Livingston, K. and Riesbeck, *In Proc.* IUI '07. ACM, New York, NY, 198-205.

[9] McCallum, A. "Information Extraction: Distilling Structured Data from Unstructured Text". ACM Queue. 3, 9, (Nov. 2005), 48-57.

[10] The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu.

[11] Van Kleek, M., Bernstein, M., Karger, D., schraefel, m., GUI- Phooey!: The case for text input. *In Proc.* of UIST '07. ACM, New York, NY, 193-202.

[12] Van Kleek, M., Shrobe, H. A Practical Activity Capture Framework for Personal, Lifetime User Modeling. *In Proc.* UM'07, LNCS 4511, Springer 2007