CSAMOA and ConceptNet: An experimental application of a model of architecture

Jason B. Alonso
Tangible Media Group
MIT Media Laboratory
20 Ames St E15-354
Cambridge, MA 02139 USA
+1-617-452-5617
jalonso@media.mit.edu

ABSTRACT
The Common Sense Application Model of Architecture (CSAMOA) was created to organize applications, libraries, and corpora of common sense reasoning with a concise taxonomy and guidelines that encurage sustainable code. In this paper, I discuss the motivations for selecting ConceptNet for reimplementation, compromises made in the enforcement of the architecture, observations on the effectiveness of the architecture, the future of this experiment, and plans for additional experiments. This is an addendum to the original CSAMOA short paper submitted to the IUI 2007 Common Sense workshop.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Theory and Methods.

General terms: Standardization, Theory

Keywords: Common sense, application programming interface, software architecture

INTRODUCTION

The Common Sense Application Model of Architecture (CSAMOA) was created to organize applications, libraries, and corpora of common sense reasoning with a concise taxonomy and guidelines that encurage sustainable code. I am in the process of conducting an experiment wherein the CSAMOA model is tested by rewriting the ConceptNet database and library to fit this new form. Though the code port is not complete, I have made enough progress to draw some useful conclusions.

In this addendum to the original CSAMOA short paper, I begin by discussing the motivations for choosing ConceptNet. In the subsequent section I set forth the conditions of the experiment, and after this is an examination of the components that I have adapted to fit into the CSAMOA layer architecture. Then, I discuss the future directions of this experiment and subsequent experiments before presenting my conclusions.

MOTIVATIONS

ConceptNet belongs to a family of substantially incompatible OpenMind Common Sense (OMCS) -derivative representations colloquially known as the "X-Nets". Other representations include EventNet and LifeNet. It has been advanced by Smith in [6] that a desirable future direction for the X-Nets is interoperability, which indicates to me that the X-Nets should be examined closely to see if the CSAMOA model of architecture can bring a potential mechanism for interoperability to light. I also became aware of an effort to revise ConceptNet to add new features, like polarity of assertions, which was identified as a future direction for ConceptNet in [6], and the people engaged in this project were accessible to me. ConceptNet also lacked interchangeable software components, despite the changes being developed. With all of these things considered, ConceptNet appeared to be the optimal choice for testing CSAMOA.

THE FIRST EXPERIMENT

The first experiment on the use of the CSAMOA model was restricted to ConceptNet and the underlying OpenMind Common Sense (OMCS) corpus. This limited my ability to search for interoperability with the other X-Nets, as such a search would require the close examination of at least two X-Nets, but this limitation allowed the first experiment to test CSAMOA for less esoteric benefits. The benefits sought in this experiment were:

1. a clear separation of dependency relationships among the common sense software components; and
2. intuitively obvious assignment of any data model or procedure to an appropriate layer.

It must be noted here that, though interchangeability of software components *may* be enabled by CSAMOA, I do not expect enough alternative structures to be developed in the course of this experiment to test CSAMOA's capacity for it.

THE EXPERIMENTAL IMPLEMTNATION OF THE MODEL

In the case where all layers of the model need to be implemented, the model naturally lends itself to a bottom-to-top construction, starting with the Corpus layer. Encouraged as such, I began my implementation there and worked toward the Realm layer. It should be noted that, by itself, ConceptNet does not complete an application, so I do not expect to populate the Application layer with any code.

The original ConceptNet code is written in the Python [1] interpreted programming language. Given personal preferences and the fact that the ConceptNet redevelopment effort with which I was cooperating was continuing to use Python, I opted to keep the chosen language for this experiment the same. I anticipated the appropriateness of an strong database abstraction layer, wherein database tables and queries are generated automatically from naturally-written Python code, and so I selected the Django web development framework [2].

At the time of this writing, however, the conversion of the ConceptNet code to the CSAMOA model was not complete. I made it through the development of the corpus layer and much of the representation layer, with some conceptual work throughout the remainder. The particulars of my progress and observations follow.

Corpus

In the Corpus layer, I implemented models and control code to manage the OMCS corpus. Though the code developed is essentially complete and functioning, it does not include any knowledge elicitation code as would be required to have a complete OMCS system [5].

In the original ConceptNet code, an autocorrector for fixing common typographical errors was included in the code that parsed sentences. As this operation produces data in the same form as the original data in the corpus, I determined that autocorrection code should be separated from the parsing code and assigned entirely to the Corpus layer. This code became a generalized autoreplace engine, with distinguishable rulesets, when I discovered "swaplists" in the ConceptNet code with subtle semantics-altering effects deeper in the parsing code. It should be noted that the "swaplists" were applied indiscriminately in the parsing layer, such that it would be equivalent to put the substitution rules in the Corpus layer.

The models that I created for this layer include the supporting elements Language (for keying multilingual dictionaries and rule sets), AutoreplaceRule (for providing the aforementioned autoreplace rulesets), and Activity (for representing the knowledge elicitation method). The layer also includes the core elements Source (for storing profile data on the provider of a piece of common sense knowledge) and Sentence (the smallest unit of corpus data).

Representation

The Representation layer, as anticipated, contains the bulk of the ConceptNet transformation. The models were easy to construct, with some exceptions that will be discussed.

Parsing/Encoding ConceptNet used a number of parsing dictionaries for storing word sets like stop words and words that invert meaning–this was easy to accomodate, but the use of these dictionaries was so far removed from the publicly-exposed models of this layer that I was encouraged to place them into their own "tools" subcomponent of the Parsing sublayer.

Models that I created in this sublayer include the defining models Frame (for storing elicitation frames) and RawPredicates (for storing binary predicates extracted from a Sentence).

Reasoning The Reasoning sublayer is dominated by the reasoning process, and no supporting models, like dictionaries, have warrented themselves necessary yet. The algorithms in this sublayer are being redeveloped by Speer [7], but the work I have seen so far seems to indicate that the Reasoning sublayer can rely entirely on the models provided by the Parsing and Presentation sublayers with their respective tool kits.

Presentation

Realm

Application

FUTURE DIRECTIONS

As the experiment described in this paper is not complete, I plan on finishing the transformation of ConceptNet into the CSAMOA model, using GOOSE [4] (the goal-oriented search engine assistant) or Empathy Buddy [3] (the common-sense-driven affect-sensing system) for the application layer.

CONCLUSIONS

ACKNOWLEDGEMENTS

I would like to thank Rob Speer and Catherine Havasi for their understanding as I scrutinized everything, and continue to do so, while they worked on reviving the ConceptNet code. I would also like to thank Henry Lieberman, Junia Anacleto, and Dustin Smith for their insights on the structure and future of ConceptNet.

REFERENCES

1. Python Software Foundation. Python programming language. Python web site, 2006. http://www.python.org.

2. Lawrence Journal-World. Django, the web framework for perfectionists with deadlines. Django web site, 2006. http://www.djangoproject.com.

3. Henry Lieberman, Hugo Liu, Push Singh, and Barbara Barry. Beating common sense into interactive applications. *AI Magazine*, 25(4):63–76, 2004.

4. Hugo Liu, Henry Lieberman, and Ted Selker. Goose: a goal-oriented search engine with commonsense. In *Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, LNCS 2347, pages 253–263. Springer, 2002.

5. Push Singh. The public acquisition of commonsense knowledge. In *Proceedings of AAAI Spring Symposium: Acquiring (and Using) Linguistic (and World) Knowledge for Information Access*. AAAI, 2002.

6. Dustin Smith. The future of commonsense reasoning. Presentation, March 2006.

7. Rob Speer. Openmind commons: A new framework for acquiring common sense knowledge. M.Eng. thesis proposal, 2006. http://torg.media.mit.edu/rob/index.php/Research.