Approaches to Integrating Semantic Applications

Martin Dzbor[1], Philipp Cimiano[2], Victoria Uren[1] and Sam Chapman[3]
[1] Knowledge Media Institute, The Open University, UK
[2] Institute AIFB, University of Karlsruhe, Germany
[3] Natural Language Processing Group, University of Sheffield, UK

Abstract

We have integrated downstream tools designed for knowledge engineers with more user-centered, upstream tools. This approach reflects the fact that we should not place the needs of developers and their tools in a strict opposition to those of end users and their tools. We believe that adoption of the semantic web depends on satisfying the different needs of different users. Focusing solely on end users is rewarding in the short term, but in addition to the instant, often shallow and short-lived rewards, semantic applications should offer sustainable, "delayed gratification".

1 Motivation

As a scenario, take a preparation of literature or research review in a field. A large part of the review is about retrieving additional knowledge about the existing facts. In terms of a Magpie-based semantic application, we can include in this category services like "Find papers on theme Y" or "Find experts in X". However, in addition to recalling the existing references and reusing them, the review writing scenario has a hidden but important *exploratory component* whereby we create new knowledge correlated with the existing facts. We want to create knowledge of "People with similar expertise" or of "What has argument A to do with person P". The state-of-the-art tools support either knowledge retrieval and *reuse*, or knowledge *creation*, but not both. We need to support both phases since together they constitute an important knowledge maintenance process.

1.1 End user's perspective: Towards more robust knowledge navigation

In poster we show an extract of a web page annotated using a user-selected lexicon populated from internal databases of research activities. As can be expected, concepts like "*Magpie*" or "*BuddySpace*" are highlighted because they were explicitly defined in a KB. However, a few potentially relevant concepts are ignored (e.g. "*web services*" or "*human language technology*"). These are closely related to the existing terms but are not in the KB. This is a sign of (i) an incomplete lexicon and (ii) ontology brittleness (i.e. rapid degradation of performance outside the intended domain).
To overcome brittleness, the tool shows a collection of additional relevant concepts discovered by an offline IE tool. These relate to the 'discourse' or domain of the user-selected lexicon, yet do not currently exist in the KB, from which annotation lexicons are generated. New concepts are mostly on the level of instances, but the IE tool (which supplies these facts) also proposes a finer-grained classification using discovered classes such as "*Activity*" or "*Technology*". Discovered instances (e.g. "*browsing the web*" or "*workflow and agents*") can be rapidly incorporated into the KB and used for annotation. Importantly, items such as "*workflow and agents*" are not only highlighted as "*Research activities*", but the user can also invoke associated services to obtain additional knowledge about these discoveries. The user can then interact with the instance using semantic menu – e.g. to invoke service "*Find similar areas*" for the newly discovered "*workflow and agents*" instance.

1.2 Engineer's perspective: Towards automated knowledge maintenance

The challenge arising from section 1.1 is how to avoid having the user manually entering and committing the proposed discoveries into a KB. Such revisions are, admittedly, not what most users would have the privileges or expertise to do. How can this be achieved in a way that would still deliver rewards to the user? The extended lexicon provides instant benefits to the users, but not to the knowledge engineers. Having a lexicon/KB that is evolving and exhibiting some learning and adaptation would obviously justify using the adjective "semantic" for the application supporting the literature review task. From engineer's perspective, the downside is the manual filtering, validation and inclusion of the suggestions from IE into the KB.

Such manual revision can be done (provided we are talking about a *small number* of concepts and *occasional* amendments), but it is not scalable. In our experiments with the semi-automated generation of lexicons using IE tool PANKOW, the existing lexicon of 1,800 concepts was extended by discovering 273 organizations, 80 events, 176 technologies, etc. The engineer had to manually adjust classification for 21-35% of discoveries in each category. Linking PANKOW to another (validating) IE technique reduced the need for adjustment (e.g. for events) to 8%. Thus, by integrating the pragmatic (or as Brown says 'scruffy') IE techniques and by making them accessible from a semantic browser we achieved benefits and reward for developers and other KB users – in terms of evolving knowledge and assuring its quality.

2 Approaches to Application Integration

Architecturally, there are several ways to realize the strategic need for integrated applications, originally developed as standalone applications. One is as a channel for delivering newly acquired and potentially relevant facts to the user. We call this approach *integration through aggregation*. The other is more complex but unlike the former one, it addresses the issue of knowledge validity not only relevance. This strategy links standalone applications by *integration through choreographing*.

2.1 Integration through aggregating

Trivial knowledge maintenance through aggregation occurs by employing an information extraction (IE) service. The IE service is implemented as using Magpie's trigger services facility. The IE service is informed about the web page the user visited by means of an asynchronous message. If new information is extracted from that page, the user is informed – the results are automatically pushed to a trigger service interface, where the user can interact with the discovered instances. The IE engine that has been published as a trigger service for Magpie and tested in this role was C-PANKOW. The information aggregation and immediate delivery strategies made the semantic browsing more robust, but this solution has limited usability. There is a considerable delay in waiting for the results with no additional knowledge about the discovered instances. This is not a fault of C-PANKOW as an IE tool, but with aggregation in general. It is about taking in account not only the quantity of the discovered knowledge but also its quality, validity and extensibility.

2.2 Integration through choreographing

Instead of instant knowledge presentation, the emphasis shifted to *knowledge validation*. To validate knowledge we need to go beyond mere discovery of additional facts in a corpus and their shallow classification. In order to create relations and properties for the discovered instances, we remove the instant notification about discoveries into Magpie's collectors. Instead, the IE tool for validation catches the output of the initial fact discovery step. Thus, we do not aggregate additional services with Magpie, but create a choreographed sequence of standalone applications/services. We used PANKOW as a first-pass IE service for discovering potentially relevant instances in a given web page, and delegated the results of the initial discovery to the Armadillo service for further processing. Armadillo was originally a standalone application that has been published into Magpie's open services framework as a special type of service. This service was triggered by other services rather than the user's interaction with a web page. It maintained a persistent record of discovered facts in a local RDF store, which was treated as potentially relevant knowledge.

3 Discussion

Integration based on loose choreography of the independent services into a semantic application is the way for meeting pragmatic challenges for the semantic web. It draws on the pluralistic notion of the semantic web being an open knowledge space with a variety of resources, which always mutually interact. If we want to highlight the primary benefit of choreography over aggregation, it would concern the aspects of *ownership*, *scalability* and an opportunity to bring in *social trust*. Because there is no need to design a service overseeing the interaction of the component tools, this simplifies the application development. In our case, enabling loose trigger handlers in Armadillo was easier than integrating it tightly with C-PANKOW. Significant time was saved by the developers, who would otherwise need to re-design all tools involved (i.e. PANKOW, Armadillo and Magpie).

We can argue that in respect to knowledge validity, our choreographed integration is an extension, which is superior to the solutions provided by the individual component tools. Improvement is also observable on the level of knowledge maintenance. While there are many techniques and strategies for knowledge discovery, representation and reuse, the maintenance of knowledge is in its infancy. *We are not aware of any major research into robust and scalable knowledge maintenance.*

References

[Cimiano *et al*, 2005] Philip Cimiano, Gunther Ladwig & Steffen Staab. *Gimme' the Context: Context-driven automatic semantic annotation with C-PANKOW*. In *14th Intl. WWW Conf.* 2005. Japan.

[Ciravegna *et al*, 2003] Fabio Ciravegna, Alexi Dingli, *et al. Integrating Information to Bootstrap Information Extraction from Web Sites*. In *IJCAI Workshop on Information Integration on the Web*. 2003. Mexico.

[Dzbor *et al*, 2004] Martin Dzbor, Enrico Motta & John Domingue. *Opening Up Magpie via Semantic Services*. In *3rd Intl. Semantic Web Conf.* 2004. Japan. p.635-649.

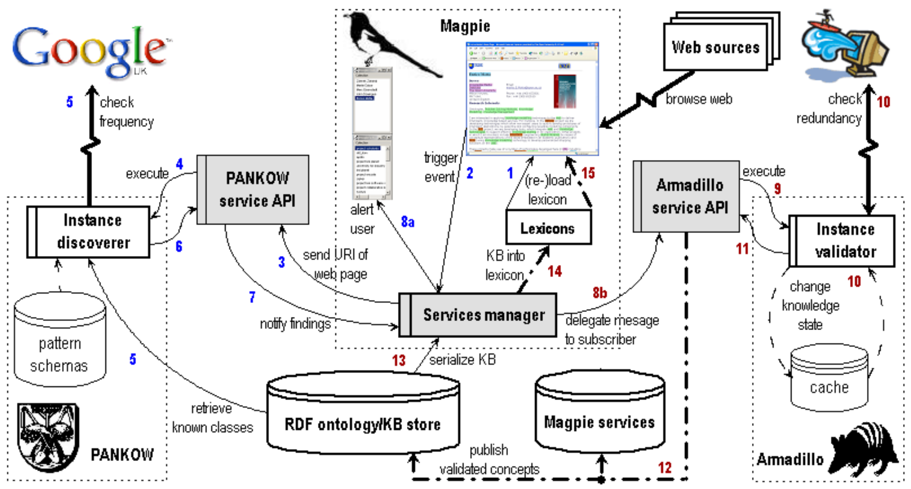

Fig. 1. Magpie – PANKOW – Armadillo interactions: *aggregation* (1-8a) and *choreography* (1-8b-15)