> Semantic Web Use Cases and Case Studies

Case Study: Twine

Jim Wissner and Nova Spivack, Twine, USA

April 2009

Introduction

Twine helps people track, discover, and share content around topics they are interested in.

Twine is built on a semantic applications platform that combines W3C standards for RDF and OWL with natural-language processing, statistical analysis and graph analysis capabilities.

Twine is developed by Radar Networks, headquartered in San Francisco. Before developing Twine, Radar Networks had worked on the CALO (Cognitive agent That Learns and Organizes) project, a distributed research program focused on next-generation semantically-aware machine learning applications. The Twine product was initially shown in late 2007 and early 2008 and became publicly available in October 2008.

The Problem

The problem that Twine seeks to address is that of helping people keep up with their interests in the face of a growing information overload. The information explosion that is currently taking place is increasingly hard to keep up with. There are too many sources of information, too many types of information, and too much change for any individual to cope with. Solutions are needed to help people productively track their interests and discover content around them.

Solution

Radar Networks has developed a proprietary and highly scalable triple store. Above that they have built a full knowledge-base driven applications platform that also relies on semantics for much of its internal functioning.

For example, many application level concepts in the Radar Networks platform are defined ontologically: user accounts, application and data permissions, various types of content objects, social relationships between users, and even user-interface components. They have also built custom tools for Web data extraction and several levels of ontologies to map data objects to underlying meaning.

Twine.com is the companies user-facing product and is a Web-based application that leverages this platform.

In Twine.com, rather than tracking multiple sources individually, such as news sites and RSS feeds, Twine enables users to track groups that collectively gather content from a wider array of external sources than they could keep up with on their own. Thus by tracking a single topic group in Twine, a user may effectively leverage the collective intelligence of perhaps thousands of other mutually-interested people who are collaborating to gather relevant content from all around the Web.

These topic groups ("twines") are curated by Twine users who may function as contributors and moderators. The collective intelligence aspects of Twine are key to its functioning. Human intelligence still trumps artificial intelligence today, and by leveraging human editorial input, Twine is able to aggregate and organize content with a high level of subtle understanding of the nuances and needs of different topics and groups.

But in addition to human-curation, Twine also provides automated curation in the form of automatic entity-detection, summarization, full-text indexing, faceted navigation heatmaps, and related link generation. These services help to add further contextual information about the content of twines, and for each article that is added to a twine.

The application of artificial intelligence, statistical algorithms, and additional semantic metadata in Twine is designed not to replace, but rather to augment, the human intelligence that is the core of the system.

Twine uses Radar's technology to make sense of an individual's (or group's) database of content, including his or her documents, emails, and bookmarked web pages.

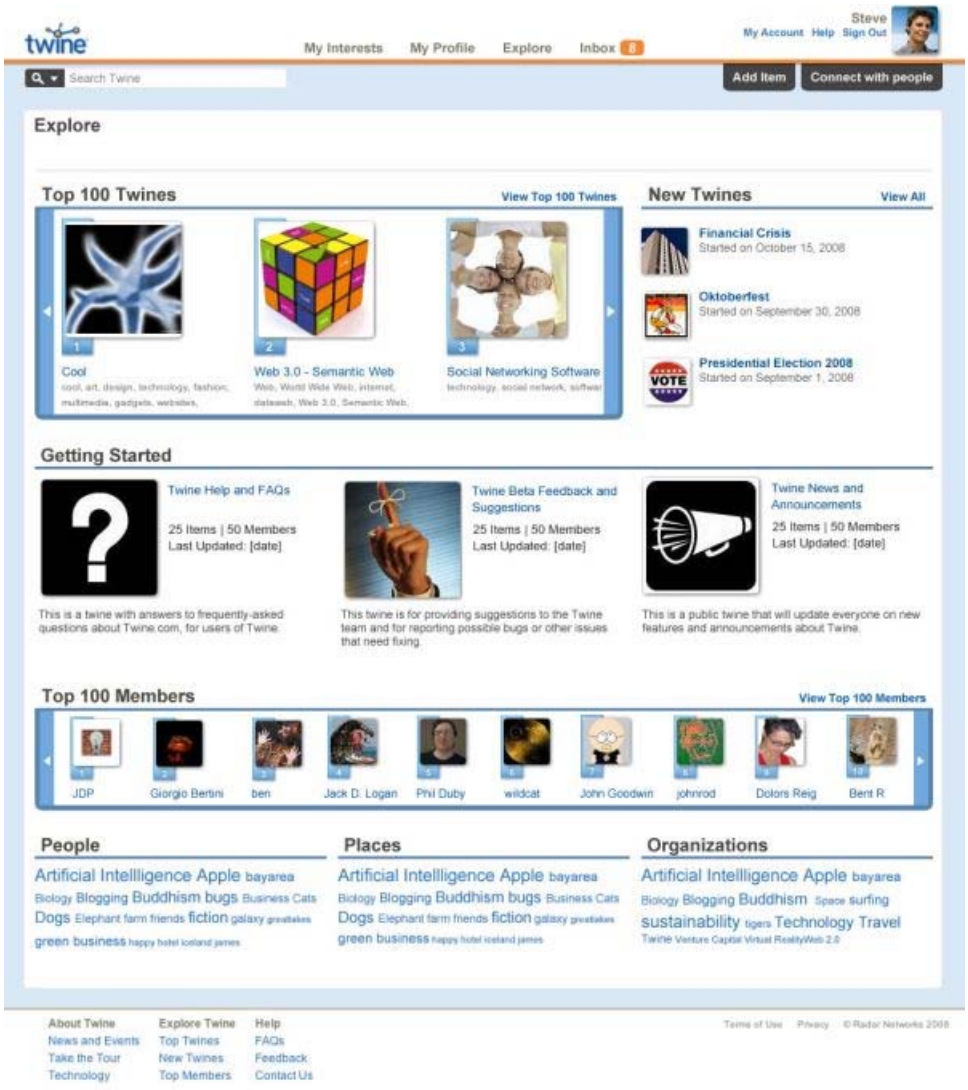

Figure 1: Some Top Twines (see the larger image)

A "twine" is a place for collecting, sharing, organizing and keeping up with interests. It's the next step beyond a file server, wiki, personal home page, or database. Users can create a twine for any group, team or community. Twines can be private and personal, private for groups, or public for groups and communities.

The most popular twines right now represent an array of interests, with names like Foodie Extraordinaire, Alternative Medicine, The Art of Filmmaking, Science Fiction Depot, Oddities Around The World, Sustainable Living, Humor and so on. The #1 most popular Twine is just called "Cool"—it has thousands of members who all contribute the "coolest" stuff they find, in a sort of massive, distributed, user-generated crawl of the Web.

There are also private Twines for conferences, school groups, corporate teams, families, etc. And there are thousands of Twines for more esoteric interests. In fact the smaller Twines are some of the company's more interesting use cases—there are only so many people in the world who are intensely interested in British cartoonists, but they seem to be finding each other using the service.

Since the company's inception, the development efforts of Radar's product team have been closely aligned with the standards emerging from W3C's Semantic Web working groups. Radar's platform makes use of OWL to describe its ontologies, it communicates data in the powerful RDF format, and Twine.com plans to make its users' data available via SPARQL endpoints.

The benefit of using OWL is that it enables both the development platform and the end-user application to have highly flexible data models. The Radar platform enables live changes to the underlying ontologies, that are immediately reflected in the live running application. Ontological inferencing is done at query time, therefore many changes to an ontology don't require data migration.

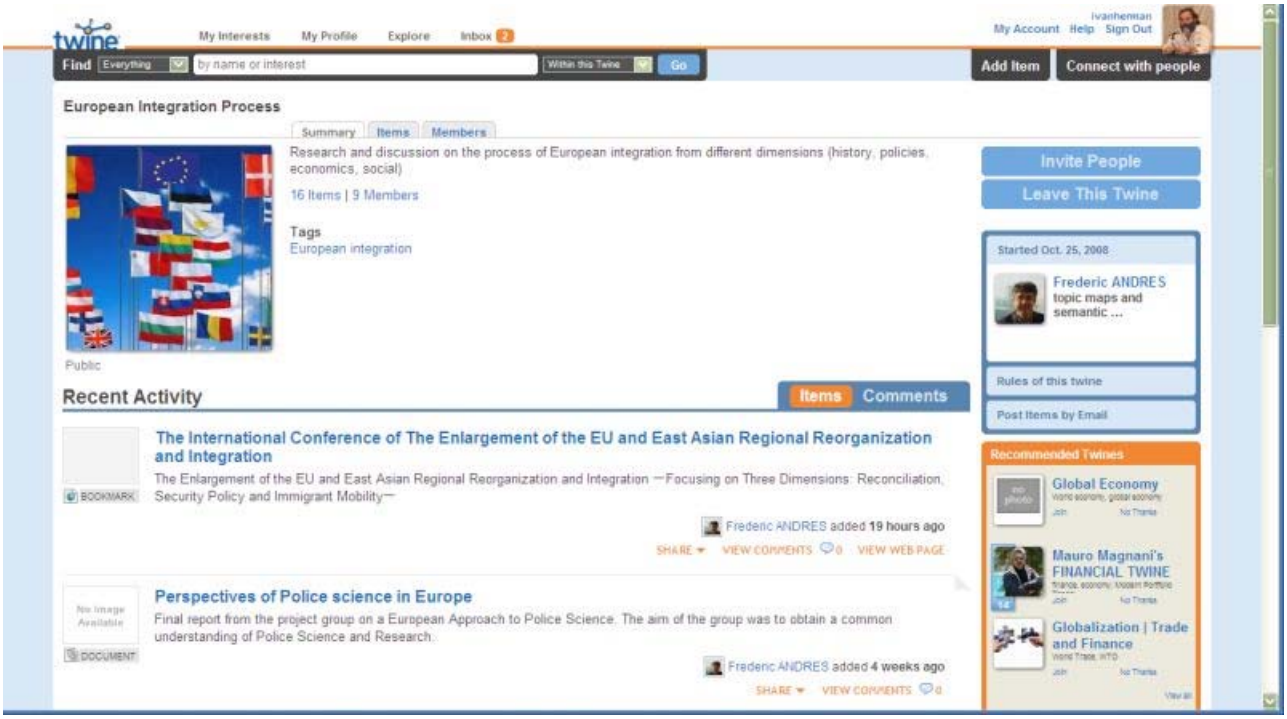

Figure 2: Example for a twine (see the larger image)

New classes of content, new properties, and new relationships may be added to Twine in real-time and are immediately usable within the live running application. Notably, this may be accomplished without programmers having to change the underlying database schema. This is a result of storing the ontology (and the data it describes) in a generic, tuple-centric schema. There are pros and cons of this approach. From the standpoint of delivering a web-scale consumer application, the primary pro is that application developers have more flexibility to develop logical data models without the usual concerns about the physical data model. That streamlines the development process and allows the application to be retooled and/or extended in a much shorter time frame. The downside to this approach is that it is much more difficult to scale a generically-defined schema than it is one that is highly tuned to a specific dataset. Storing data in a small number of long tables with billions of rows and only a few columns makes queries and other database operations very complex (due to large numbers of self-joins that must be executed). The principal trade-off is therefore between flexibility and performance.

The current generation of the Radar Platform (as of Q1 2009) has now seen several iterations to this architecture, and as Twine.com has grown, we have made meaningful adjustments to the schema to grow along with it. We have employed several additional tables to the underlying schema that are optimized around certain classes and predicates, to improve performance further. The next-generation of the architecture (slated for Q3 of 2009) will embody cloud-computing principles to enable open-ended scaling and many orders of magnitude higher performance.

In addition to the benefits of OWL on the database-level, OWL also enables data modeling to more closely resemble various domains being modeled. The object-oriented nature of OWL data models is more easily understood by non-technical domain experts and is therefore easier for them to use when developing large domain models. OWL also makes the underlying schema of data more machine understandable, and enables the schema to “travel with the data” wherever it goes.

Beyond OWL, the use of RDF as a data representation also has numerous advantages which are detailed elsewhere. However some of the main advantages are that data sets from disparate sources can more easily be merged, and that data and meta-data are more easily syndicated on the Web and shared between Web services.

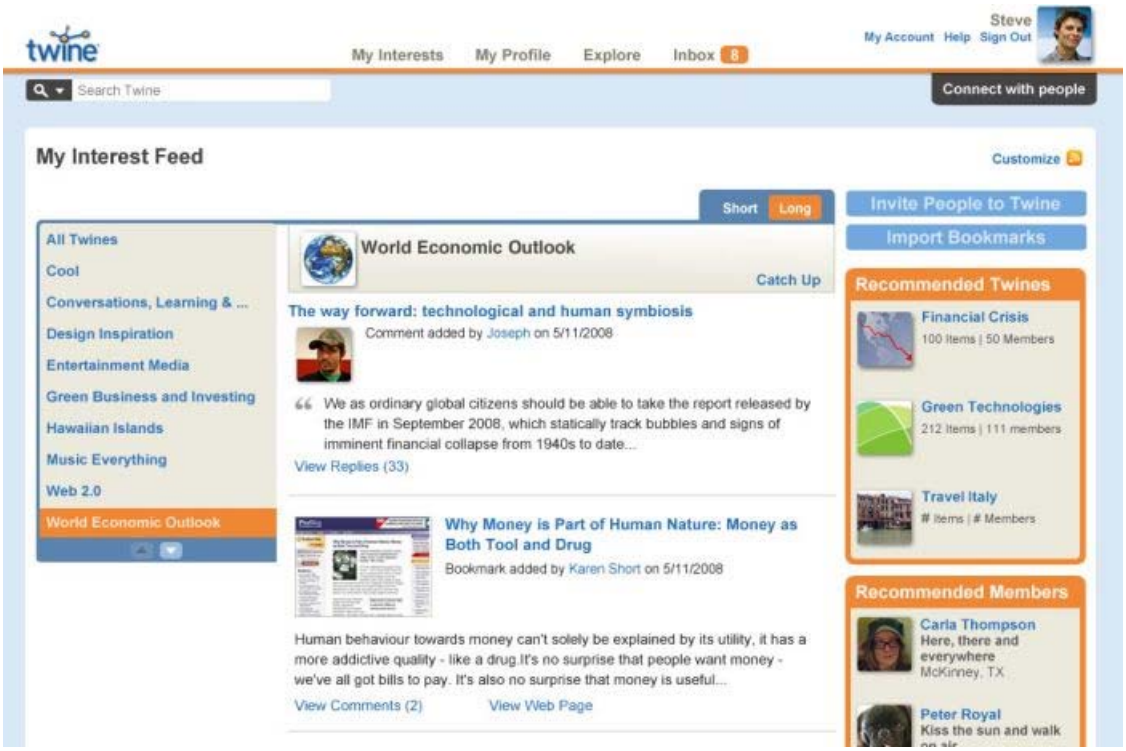

Figure 3: Example for a user's "interest" page on Twine (see the complete and larger image separately)

Conclusion

By extracting and interpreting the data embedded within different types of documents—including web pages, emails, and Word files—Twine significantly reduces the effort required by a user to curate and later re-use information. Documents and data about a given person or concept are grouped together automatically, and synergies in the relationships between different types of data become apparent.

When you put information in, Twine looks at it, tries to understand it, tries to enhance it, auto-tag it, connect it to other related things, and actually starts to learn and build a semantic graph, about you, your group(s), and your information.

Twine also learns from each user and group and begins to make personalized recommendations—using its knowledge of both the social and semantic graphs it contains.

By combining social media best-practices with the Semantic Web, and a significant amount of statistics and analytics, Twine has developed a service that is rapidly gaining traction among users on the Web.

As of March 2009, Twine had nearly 1M monthly unique visitors, 100K registered contributors, 4 million pages of Web content contributed, 3 million synthetically generated semantic tag objects created automatically, millions of relationships within it, and an approximately 80% month-over-month unique user growth rate (Sources: Compete.com and Twine.com). While it is still early in the roll-out of the service, Twine appears to be well on its way to becoming a popular Web application.

Nova Spivack says that W3C standards are required to ensure that previously disconnected databases can share data freely; "Approaches that don't use the W3C standards are putting data on the Web, but they're not making a true 'web of data.' These approaches lack the flexibility and extensibility that are available with RDF and OWL. They recreate the problem we already have: if you change all these brittle disconnected databases, you have to change all the applications that depend on them. The Semantic Web will solve that issue, if everyone adopts the standards."

Key Benefits of Semantic Web Technology

- Flexible database schema—easy to change or modify
- Programmers do not need to think about the underlying database
- Little to no database refactoring necessary as applications evolve
- Query-time inferencing on new schemas and data possible in live running application
- Application and service layer components easy to define and modify ontologically
- Domain experts are able to more easily model their domains
- Semantic tagging enables semantic faceted search and semantic discovery of content
- Semantic metadata within application supports deeper analytics and recommendations
- Open-standards metadata about content is more easily re-usable in 3rd party applications