Blog Navigation via Semantic Analysis

Riley Patterson
Caltech

Fred Zhao
Caltech

ABSTRACT

In this paper we describe our approach to bring together linguistic semantics and the Semantic Web. We briefly explore the history and applications of the Semantic Web, then shift our focus to the area of online text — specifically, in the form of blogs. We describe a plugin we made for the widely used blogging platform Wordpress that analyzes the semantic relationships between different blog posts and creates a graph-based interface to visualize their relationships. This enables blog readers to access topics by semantic relations rather than the traditional form of chronological order, and accomplishes this in an automated fashion rather than manual tagging.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext / Hypermedia—*navigation*; I.7.4 [**Document and Text Processing**]: Electronic Publishing; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*web-based services*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*linguistic processing*

General Terms

Design, Human Factors

Keywords

Natural language processing, Semantics, Semantic Web, DBpedia, WordNet, FreeBase, Tim Berners-Lee, World Wide Web

1. INTRODUCTION

When Sir Tim Berners-Lee first invented the web, his primary goal was to help facilitate information sharing and management among researchers at CERN. [2] Recently, as the web has broadened to cover all aspects of our daily lives, Berners-Lee has called for developers to bring out the *linked* nature of web data in a new form of web content, coined Semantic Web, to enable information sharing and management for the whole world [3].

The linguistic semantics of a word describes its meanings and its relationships with other words; the Semantic Web explores this idea with respect to online data and models both how the data relate to each other and what those relationships mean to human users. The traditional web is comprised of webpages and hypertext that link them together, but the contents must be manually parsed for information; the Semantic Web builds upon those standards and defines structured data to directly model the actual ideas and the relationships between them, and those structured data can then be aggregated from distinct and separated sources. The implication is that we can eventually use the web to combine human knowledge in a meaningful way, rather than just bits and packets strewn across the internet.

We focus on a specific web application, the blog, as an area that can stand to benefit from these changing standards. In §2 we examine the growing importance of natural language processing in web applications today. In §3, we briefly consider the trend shift currently occurring in web standards, and describe both limitations and innovations of organization features currently available to popular blogging applications. We also describe some of the existing standards and APIs designed for Semantic Web applications in general, emphasizing those that affect blogs. In §4 we describe a plugin we created for the popular platform Wordpress. It combines various modules to process posts in a blog and produce a form of output that, as of our current knowledge, has not yet been done. We start with the general workflow and then go in detail to describe each module's role. In particular, we tried many different approaches to establish relationships between blog posts, so we will explain the functions we coded for this purpose–what worked, what didn't, and what we chose to include in our final product. To evaluate each approach's strengths and weaknesses for different blogging purposes, we look at example entities and relations extracted in Figures 3- 6. Finally in §5 we conclude with our view of future work that could be done with semantic blogs and the semantic web in general.

2. THE STATE OF NATURAL LANGUAGE PROCESSING

In recent years, it has become increasingly clear that natural language processing (NLP), in particular compilation,

organization, and curation of semantically-informed data, has tremendous potential for social benefit and commercial value. Thus, it has been the focus of a significant body of work in both research communities and in private companies with commercial interests.

Semantic resources are effectively a prerequisite for powerful natural language processing. Amazingly, some such resources predate the web, and these have arrived at a state of sufficient maturity to allow their use in larger NLP-based projects. Consider, for example, Princeton's WordNet, a highly-regarded and heavily-utilized semantic project that boasts origins dating back to 1985. WordNet, in particular, has been used from everything from Applied Semantics, Inc.'s early projects [8], which would eventually become Google's AdSense, to contemporary work that is much closer to our present project [11, 9]. AdSense helped increase relevance of ads in Google's search pages by analyzing the content of the search and matching it up with ads of similar topics; Ponzetto tried to map out a taxonomy for Wikipedia, also with *structure* in mind; and Jo graphed evolution of topics over time. All of these efforts would be way too time consuming to be done manually, and the data would become outdated long before the human analysis required to process them is completed.

Fortunately for our work, useable semantic resources have both multiplied and matured. We were able to benefit substantially by simply relying on existing time-tested resources to do many of our core NLP tasks. This luxury allowed us to focus more intently on the higher-level problem that we sought to tackle to begin with.

3. THE STATE OF THE WEB

3.1 Current Blog Features

The modern blog evolved from the fundamental notion of an individual's online diary, so it logically follows that the primary means of blog organization remains chronological. Also standard are some organizational features like tag clouds and categories, which help manually compile lists of related posts, and allow for some amount of "semantic" navigation. Yet, these interfaces continue to feel clunky and outdated. A tag cloud, while useful for manual categorization of blog posts, is limited in that it provides the user with no sense of how different tags are *related*; each tag is a discrete identifier which has no connection to any other tag. Categories, while providing for a hierarchical organization, cannot show multiple relations, or graphs more complicated than trees.

This shortcoming of blogs is the primary motivation for our work. Our product aims to provide a truly semantically-based, rather than document-based, mode of navigation that addresses and accounts for many of the shortcomings of tag clouds. We will discuss more about our actual implementation of this in §4.

3.2 The Semantic Web

3.2.1 Resource Development Framework

The Resource Development Framework, or RDF, is a developing standard. The WWW Consortium (W3C) overlooks its development, as well as other standards part of the Semantic Web. In brief, it uses a simple model of connected "triples"–two entities and a relation between them–to allow differently structured data to be mixed and shared [6]. This allows database implementations to be wildly different below the surface, while sharing a common interface for the specific task of how they expose data.

3.2.2 Linked Data

With the Resource Development Framework as a common ground, many current websites have joined in on the collective effort to share data. Such data is referred to as Linked Data, to illustrate the concept that they are all linked by a common standard of access. Freebase and DBPedia are two such websites that rely on user efforts to expand, with their eventual goals being to cover as many topics of human knowledge as possible; however, there are *dozens* of other organizations that are all joining in by providing their linked data–including the ACM and, potentially, this paper itself. The result is that we see an impressively broad actual web of semantics that is being built on top of the traditional web [5]; most users just don't realize it yet.

We found great inspiration on how to utilize linked data in Linked Galaxy [1], an application that combines linked data from Freebase and DBPedia and the NLP functionality of Zemanta to produce a graphical output of related keywords. Linked Galaxy currently exists as just a proof-of-concept application; it processes a single chunk of text, and the output is temporary. However, from here, it was easy to see how we can use its idea to create a semantic blog.

4. THE SEMANTIC BLOG

In order to bring the benefits of the Semantic Web to the blog, we devised a navigation system inspired by tag clouds, but designed in such a way as to clearly represent *relationships* between "tags." We use the term "tag graph" to describe this new approach to blog navigation. More detail about the appearance and implementation of our tag graphs will follow in the visualization sub section. A full workflow of our Semantic Blog is shown in Figure 1.

4.1 Data Model

In the design of our data model, we focused on two main goals: simplicity and flexibility. We aimed for a very simple set of relations that could be applied on top of an existing unstructured data model from a blog, a crawler, or some other application that might provide data into our corpus collection (see Future Work in Section 5 for other ideas about where data might come from). While several important resources in the growing Semantic Web have enjoyed much of their success due to their insistence on purely structured data models, we believe that in the particular case of extending these ideas to the realm of blogs, flexibility is of utmost importance. Thus, our data model focuses on the idea of annotating unstructured data with information derived from semantic resources.

The current design involves two tables of information regarding entities, and one for relating these entities to arbitrary underlying unstructured data. Figure 2 shows our current data model. The `unstructured_table` refers to any form

[1]http://test.infoblow.zemanta.com/infoblow/galaxy/ *Open Galaxy*, Zemanta showcase

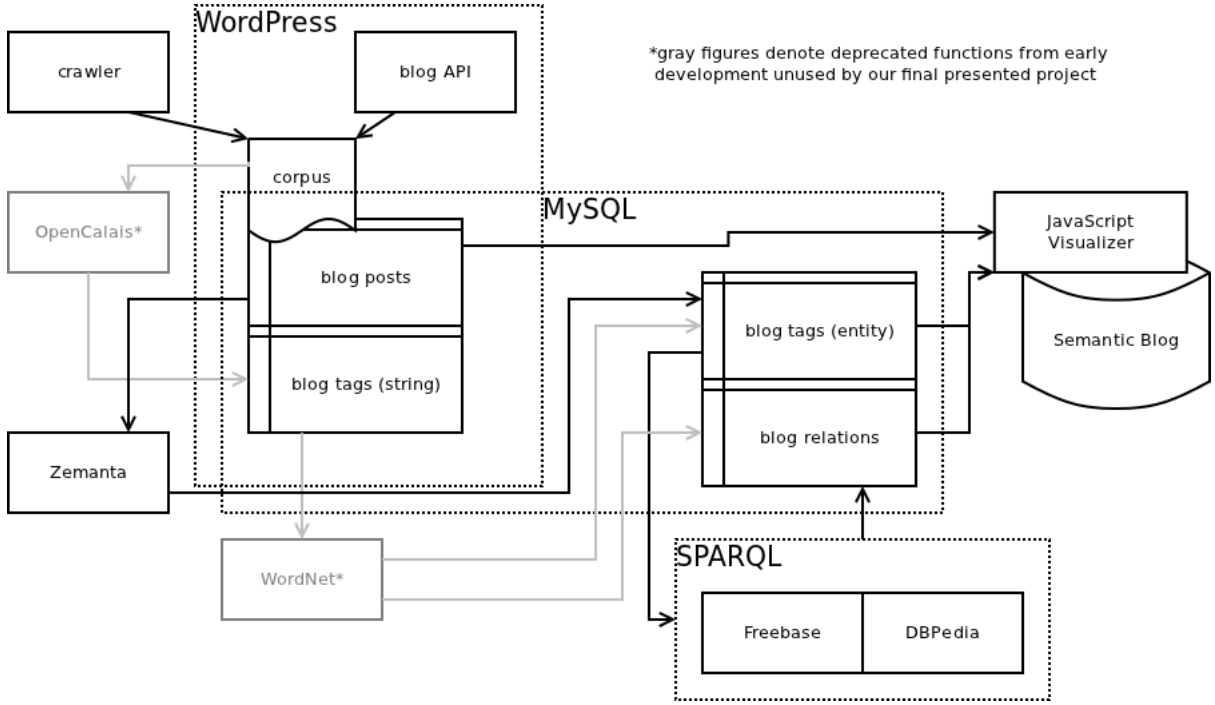

Figure 1: The Semantic Blog Workflow. This diagram describes the workflow involved when specifically considering the WordPress plugin.

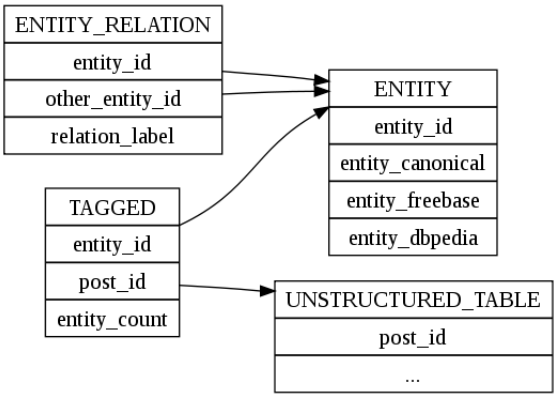

Figure 2: Generic Data Model. The unstructured table is simply a generic placeholder for the data provided from the corpus collection. The other three tables together make up our data model.

of storing unstructured data; all that is required is that it have a unique integer identifier for each chunk of text, and a lack of this could of course be mitigated with an intermediate relation. This design enables us to change or add to our data sources with ease. The Freebase and DBPedia columns are meant to store URL's for the given entity in those two resources. As the product grows, URL's for other resources could be added to this table.

4.2 Corpus Collection

We developed two methods for our software to gather its initial, unstructured blog post data: a web crawler and a WordPress plugin. As described above, our data model is flexible enough to allow future development of other methods with ease. For the purposes of this paper, however, we focus on WordPress.

The web crawler simply crawls and organizes a WordPress blog using the interface presented in HTML. It implements a simple parser that seeks common elements found in WordPress pages in order to find page text, titles, dates, and URL's. This means of gathering data is much more flexible than direct use of the WordPress API, as it only requires read access to the HTML presentation of a WordPress blog, as opposed to administrator privileges to run a plugin. However, it also has its drawbacks, as the process is error-prone and WordPress's output is not as standard as one would like it to be. One would also like this approach to function across several different blog software platforms, and this would certainly be possible, but each one would require quite a bit of fine-tuning. The crawler is included in the package released at the time of this paper, but it is not the currently active means for gathering data.

The preferred data source in our implementation is direct access to the WordPress database by way of the WordPress plugin API. This means of access entails two main limitations, first that the blog must be implemented on a WordPress platform, and second that the author himself must install and run our software in order to allow his users to benefit from its services. However, this approach is not a huge limitation, as Wordpress is very popular, and the API opens the software up to quite a few conveniences and optimizations, including incremental updates to the tag graph (as opposed to a full reconstruction) immediately upon publication of a new blog post, and an elegant way to integrate the interface with WordPress's default interface. For the remainder of the paper, the WordPress plugin API will be our assumed means of access to unstructured blog post data.

4.3 Natural Language Processing and Tagger

As we have already discussed, the existing body of work in NLP is substantial. Among the projects out there is the General Architecture for Text Engineering (GATE), which exists primarily to build a common community and API layer on top of dozens of specific-purposed open source NLP solutions [2].

After researching solutions such as GATE, we came to the conclusion that we would benefit from using web services for our NLP procedures. This decision has several benefits. First, it limits the number of dependencies our product has. It is also the case that the web services providing these solutions tend to have a better sense of the semantic web in general; they usually provide linked data to semantic resources like FreeBase, DBPedia, and WordNet. Finally, it is consistent with the spirit of the Semantic Web which inspired this project to share in the growing community surrounding its ideas.

The two main web services that caught our attention for keyword production were OpenCalais and Zemanta. Both of these services take as input an unstructured block of text (i.e. a blog post) and produce a set of keywords. Both services also use context to link these keywords to entities in semantic resources, which is highly convenient for our purposes. In the end, we settled on Zemanta for our product as its RDF output is a much more powerful and direct format for our purposes.

4.4 Entity Extraction

Once we produced keywords from the previous tools, the next step was to link the keywords together. For this we needed to transform plain string data into linked data, so we turned to some well-known existing databases that were designed for this. First we began with WordNet, which has a history of development older than the web itself. However, we saw that it was not well-suited for blogs. We then found inspiration in Open Galaxy, which uses Freebase and DBPedia, which are recent developments that produced much more satisfactory results.

4.4.1 WordNet

WordNet stores for each word several *synsets*, or "synonym sets", each of which represents a different meaning for the word [10]. Since synonyms are related by meaning, a synset represents a disambiguated meaning for a word by presenting other words that could also have the same meaning. The chief distinction is in the direction of mapping between string and meaning: a single *word* maps to several synsets if it has several different usages; a single *synset* maps to several words if those words all share the same usage. As Miller *et. al* mention in their paper on WordNet, the task for a speaker or writer is to choose a member of a synset to use as a word, while the task for a listener or reader is to be presented a word and deduce which synset it came from; therefore, the entity we wish to extract is the correct disambituated synset, given the word.

We do this for a keyword depending on other keywords present in the same text. As an example, the word "net" may refer to the internet, the goal used in a game of soccer, or many other meanings. However, if the same text also uses words such as "web", "computer", "cyberspace", etc., it would be very reasonable to guess that the word "net" is referring to the internet; and similarly, if the text uses words such as "ball", "referee", "player", etc., it would be reasonable to guess that it refers to the game equipment. To find this "closeness" in meanings of words, we use a Python package called `nltk` that includes many useful functions for working with synsets. Among them are several "similarity" functions that operate on pairs of synsets and returns a numerical value based on their relative structural locations in WordNet [3]. In the examples above, we would see that the first group of words relating to the "internet" definition all relate to the concept of "technology", while the second group of words relating to the "game equipment" definition all relate to the concept of "recreational game". These broader terms, or *hypernyms*, are what the `nltk` functions use to for their similarity calcuations, and we will discuss the specifics of this and other relational terms in the next section. For now we outline our algorithm for extracting synsets from strings.

Algorithm 1 <Synset Extraction via Pairwise Similarity>

S={unextracted strings}=input
Y={extracted synsets}={}
while |S|>1 **do**
  for each $s_\alpha \in S$ **do**
    for each $s_\beta \in S$, $s_\beta > s_\alpha$ **do**
      $(i^\alpha_*, j^\beta_*) = \max_{\arg i^\alpha, \arg j^\beta} sim(y_{i^\alpha}, y_{j^\beta})$
      {for $y_{i^\alpha} \in synsets(s_\alpha), y_{j^\beta} \in synsets(s_\beta)$}
    end for
    for each $y_k \in Y$ **do**
      $i^\alpha_* = \max_{\arg i^\alpha} sim(y_{i^\alpha}, y_k)$
      {for $y_{i^\alpha} \in synsets(s_\alpha)$}
    end for
  end for
  $i^+ = max_{\arg i^\alpha_*} sim$(all $(y_{i^\alpha_*}, ?)$ pairs this iteration)
  Remove s_α that corresponds to i^+ from S
  Add y_{i^+} to Y
end while

That is, we compare the ambiguous synsets of each unextracted string with those of other unextracted strings (using lexicographical ordering to avoid repeats), as well as with

[2] http://gate.ac.uk/overview.html *Overview*, GATE

[3] http://nltk.googlecode.com/svn/trunk/doc/api/index.html *Module wordnet*, package `nltk`

the set of extracted, disambiguated synsets already found so far. This latter process, seen as the second inner for-loop above, was added after experimental error; a test of "cat", "dog", "bird", and "man" kept removing "cat" and "dog" first (due to their close relation as carnivores), then extracted the obscure connotation "lady" from "bird" since it was the closest in meaning with "man". After we added comparisons with the synsets for "cat" and "dog", the algorithm pinpointed the correct synset for "bird" easily (via the close relation of all three as animals). In other words, this extra check helps us incrementally build up an understanding of the corpus using the best available information at any given time, which improves as we extract more synsets.

4.4.2 Freebase and DBPedia

Both Freebase and DBPedia represent entities as collections of facts, and relations are represented as links to other entities. An entity can be identified by one or more URLs; additional URLs are equated using the `owl:sameAs` relationship, a standard from the Web Ontology Language [1]. As both databases have had time to develop, many entities exist in both of them; and as expected, those corresponding URLs all reference each other via `owl:sameAs`. Both databases also reference equivalent listings in WordNet, though there is no reverse link back from WordNet to them.

What is interesting is that a URL doesn't look the same to a human as it does to a machine. The raw data that contains these relations is in forms easily accessible to parsers, but it would look as jumbled and senseless to readers as an XML file. Both websites perform browser redirects based on the user agent detected, and the result is that humans navigating the sites feel like they're browsing through a redesigned Wikipedia, while machines crawling through the pages can easily parse through all the links and relations and produce graph objects.

Clearly, finding the correct URL is all we need to match to an entity; therefore the task of finding entities becomes deducing the correct URL given a large chunk of text. It appears that we again have the problem seen in working with WordNet: how do we match up an input string to its corresponding entity, if that string may be relevant to multiple entities? For this, we turn to existing APIs for help. Fortunately, Zemanta's query function does this for us: it takes input context into account and returns a list of possible relevant URLs, each with two important numbers: relevance and confidence, each a fraction between 0 and 1. Our task of matching entities to strings was therefore significantly facilitated by the Zemanta API, and all we needed was to filter the results by a threshold. In our practice, a filter of $confidence \times relevance > 0.5$ produced tags that were relevant and unique: each "anchor text", or actual string in the original input, was matched with exactly one entity. The above formula can also be easily adjusted by replacing the filter function with any other boolean function that takes in the two values, a design intended to help extend our application in further development.

4.5 Relation Building

As our entities extracted from the previous section are in different forms depending on the implementation (WordNet vs. Freebase/DBPedia), we divide this section similarly. Though our final implementation uses the latter, we show the strengths and weaknesses of both methods, and retain the now-deprecated WordNet functions in our final project as modules that could easily be re-integrated.

4.5.1 WordNet

We now return to define some basic terms used by WordNet in its classification hierarchy of the words in its database. A *hypernym* is a superclass of entities that its corresponding *hyponyms* belong in. Table 1 lists other classification terms for nouns (similar terms also exist for other parts of speech, though we focus on nouns because they are most relevant for tagging). The table also lists equivalent relational terms in graph theory, so that our following descriptions using concepts such as "root" and "parent" can have a clear analogy in the WordNet hierarchy.

We build relations between synsets by reproducing these hierarchies using the `nltk` function `lowest_common_hypernym`, which finds the closest parent synset between any two synsets [4]. In our case, the two synsets are the pairs we compared in Algorithm 1. The hypernym would then be used to create a parental node that points to the children, which helps us easily produce a hierarchy as defined by WordNet.

Unfortunately, this also introduces the main shortcoming of this entity-relation building method: the relevance of the method is restricted by the limitations of what WordNet defines with its hierarchies. As WordNet was developed with a thesaurus as its model, the hierarchies we observed seem to be very strict, scientific, and static. While our experiments with words such as "cat", "dog", "bird", "man", etc. were very promising, we were quickly disappointed when we tried to use actual blog posts. For example, a CS146 paper summary produced many tags from OpenCalais, but WordNet failed to recognize all the technical terms; the only entities it recognized were "California" and "Wyoming", with "state" being the hypernym that linked the two. We quickly realized that recent technologies, pop cultural references, etc., would all fail to be detected as well. Knowing that many blogs are about recent news and developments, we abandoned the WordNet module, but strongly note here that it has the unique advantage of a very standardized hierarchy that will more likely remain through time.

4.5.2 Freebase and DBPedia

It was easy to build relationships using Freebase and DBPedia, since relations are a fundamental part of these two databases. For both databases, relationships can be queried via the common RDF query language, SPARQL. This is a powerful language that operates on relationship triples in the form `subject predicate object` [12], with the biggest difference from standard querying languages being that *all three fields* can be variables. We see the importance of this flexibility when we consider that, aside from standard relations such as `owl:sameAs` mentioned previously, each database following the RDF standard is also allowed to define its own set of relations. This means the relations are as dynamic as the entities they connect, so a constant set of predicates (akin to hypernyms, hyponyms, etc., seen in WordNet) will quickly become outdated. Therefore we may write a query of the form of the first line of Algorithm 2, which translates to *any entity, related by any relation, to*

Table 1: Semantic Relations in Wordnet Synsets [10]

Relation	Graph Relation	Example
hypernym	parent (classification)	*tree* is a hypernym of *maple*
hyponym	child (classification)	*maple* is a hyponym of *tree*
coordinated term	parallel (classification)	*dog* is a coordinated term of *wolf*
holonym	subset (composition)	*wheel* is a holonym of *car*
meronym	superset (composition)	*car* is a meronym of *car*

any other entity. But this will also return *any* (subject, predicate, object) triple, effectively being a wildcard condition. To filter the relations down, we add the additional database-specific clauses in the rest of the algorithm, which help pick out the actual DBPedia or Freebase entities.

Algorithm 2 SPARQL query: relational WHERE clauses

?a ?b ?c	{(General)}
?a rdfs:label ?aname	{(DBPedia)}
?a rdfs:label ?aname	{(Freebase)}

The output is a list of the triples in the `subject predicate object` order. We then store these as entries in a lookup table, with the subject as the key and a list of `(predicate, object)` pairs as the value. Finally, each `object` member of a list and its corresponding `subject` are then written to our `blog relations` MySQL table, where the relations can be visualized via JavaScript with the aid of a MySQL-PHP interface.

4.6 Visualizer

Our primary design goal is the output of a "tag graph," which is quite simply a graph of tags, with each tag providing links to blog posts that pertain to it. Due to the nature of the product's interaction with web applications, we also desired that this graph be a JavaScript-based interface for navigating the blog from within the browser itself. Fortunately, there is a significant body of work in the area of graph visualization, and we were able to find several generic JavaScript graph visualizers.

We ended up settling on the Dracula Graph Library, which produces very elegant graphs using pure JavaScript and Scalable Vector Graphics (SVG) [4]. We then modified the library to allow association of blog posts with nodes, including an interface for displaying blog post titles upon clicking a node. We built a php script for querying our MySQL database and producing the JSON data with which to build the graph. We integrated the interface into our WordPress plugin to allow it to be a fluid part of the blog navigation experience. For a graphical look at our visualization, see Figures 3- 6.

5. FUTURE WORK

As Linked Data and Semantic Web begin to garner buzzword-level awareness, it is important to note challenges they face. As with all developing standards, they are still far from ripe, and are still characterized by bugs and nuances. For example, our implementation struggled with a minor detail in the Freebase standard: some URLs replace underscores in the terminal portions of URLs with periods, but both sets of URLs are intended to represent the same entity. They will seem identical to human navigators, who will be redirected to the same human-readable page from either URL. For direct string comparison in programs, however, they pose a problem. Our solution simply converted such strings, but it does seem counterintuitive to require such a process to standardize the method to express a single entity.

Even more difficult to manage is managing entities across different services altogether, since the `owl:sameAs` relation is often human-made, and thus error-prone. As an example, consistency between WordNet and DBPedia is checked on an annual basis, and the results show that it is indeed a nontrivial problem, and many relations and entities are *not* properly mapped across the two [7]. However, this should improve as the RDF standard matures and developers realize the importance of adhering to such standards. As more of the web is turned to linked data, such errors may also be correctable by machine algorithms.

Also, there are still plenty of semantic resources on the web that we did not get to take advantage of in our project so far. We experimented with a heuristic-based approach for deciding on strength of relationship between two entities, using contributions from a multitude of resources. In order to be compatible with our end product of a graph visualization, this approach requires a threshold for deciding whether or not two nodes should be connected by an edge. In order for such a threshold-based algorithm to reach stability and reliability in prediction, we would require the use of significantly more resources than we are currently able to link to. However, the resources are out there, and the availability of linked data is growing, so there is potential for significant improvement to the quality of the produced tag graph by taking such an approach.

An additional important future project is the development of client-side solutions for performing these tasks. Plugins for the major blog platforms are arguably the ideal implementation of this work, but they require action on the part of each blog's author or administrator in order to allow users to make use of the tag graphs. A client-side solution would allow users who enjoy navigation using the tag graph to do so on top of any blog they choose to browse, regardless of the author's knowledge of the tag graph's existences. The core functionality can easily be run client side, but the gathering of blog data and the visualization pose a slight problem. We believe the best way to implement this on the client side is to build plugins for the major internet browsers. They all have built-in HTML parsers which the plugin could make use of to extract the data it needs, and they also provide the natural interface for tag graph navigation; the visualization could simply be overlayed on top of the relevant blog using the

[4]http://www.graphdracula.net/

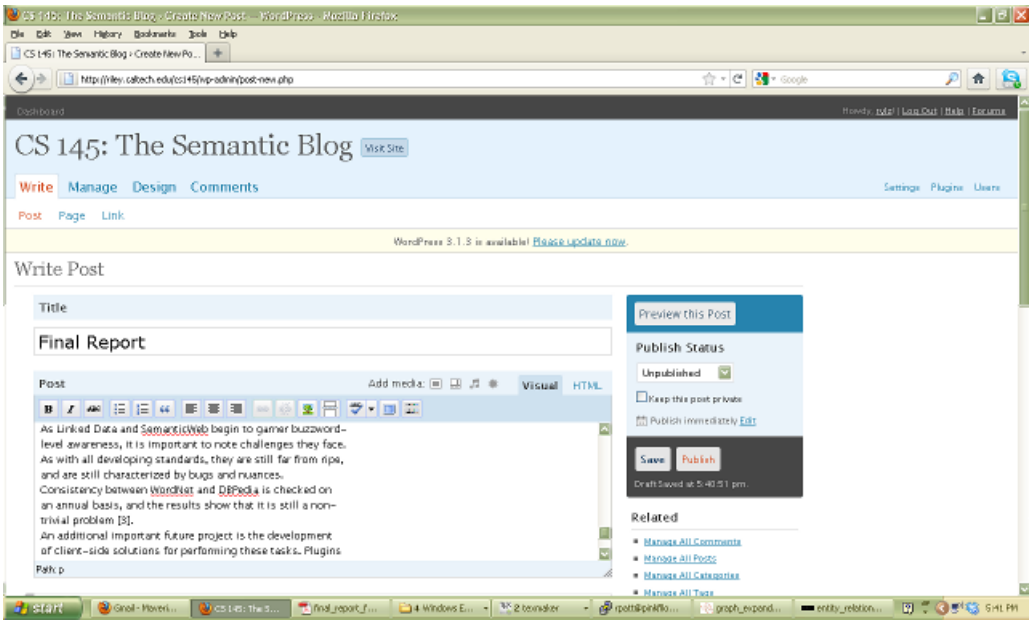

Figure 3: Entity Extraction. For the author, it's as easy as publishing in WordPress. The installed plugin then activates and does all of the work behind the scenes.

entity1	entity2
WordNet	Natural language processing
Semantic Web	DBpedia
DBpedia	Semantic Web
National Science Foundation	WordNet
Eric H. Davidson	Gene regulatory network
Tim Berners-Lee	World Wide Web

Figure 4: Entity Relationships. The plugin deposits the discovered relationships into a database table shown here. These relationships are to be represented as edges in the tag graph.

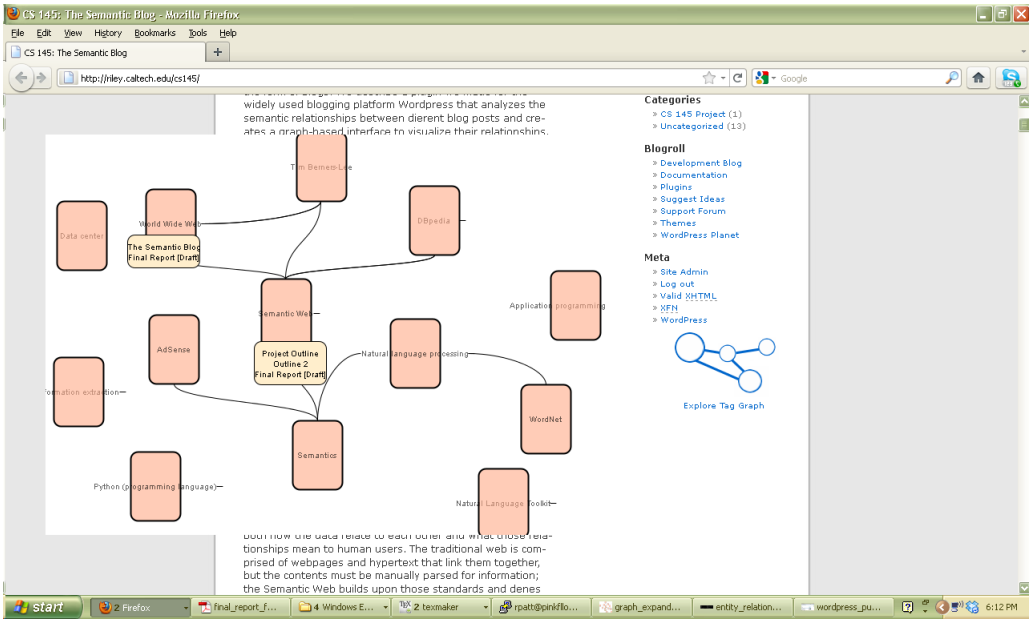

Figure 5: JavaScript Interface. When the user requests to navigate using the graph, the plugin draws an SVG-based graph on top of the current page. Users can then click on the nodes to activate lists of blog links. These links allow them to navigate to different pages.

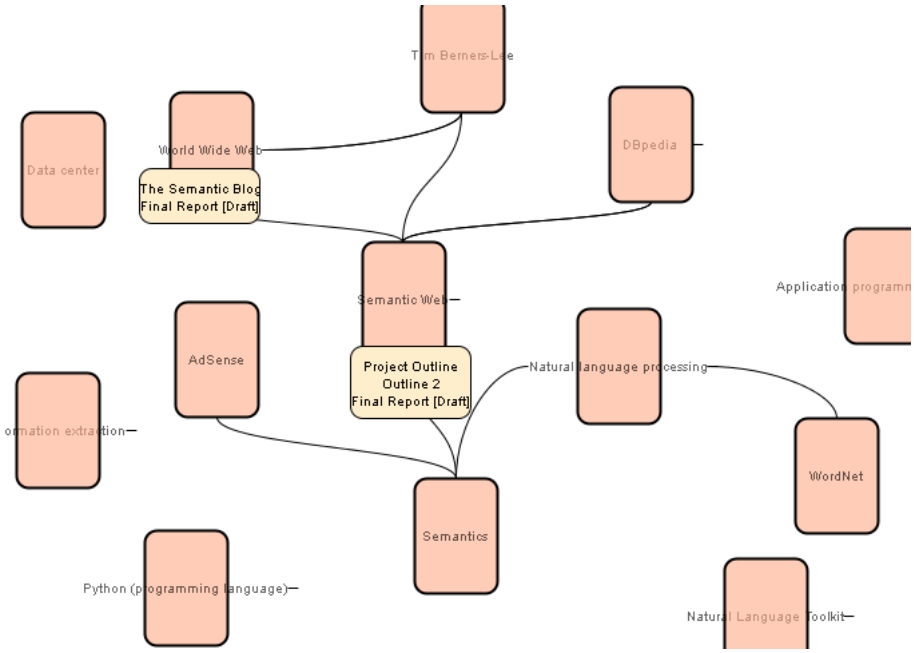

Figure 6: Tag Graph. Each node represents an entity found in at least two posts in the blog. The graph is connected semantically.

browser plugin. A client-side implementation would have the additional benefit that users could build super graphs linking similar blogs with the same extracted entities, providing services similar to RSS aggregation but for semantic organization rather than chronological.

Finally, assuming the existence of a client-side implementation, a central server to cut down on redundancy in computation and in crawling bandwidth would be very useful. The central server could also provide a social service in which users could share the topics they like to browse the most with each other and combine subsets of their generated tag graphs. This could also potentially pave the way for a business model supporting this service in the long term. Such business models are currently somewhat absent in the growing Semantic Web. They will be essential if this paradigm is to move forward.

6. ACKNOWLEDGEMENTS

We would like to acknowledge Professor Steven Low for his helpful mentorship from planning stages to completion of this project. Professor Adam Wierman gave us very useful advice in the planning stages as well. Additionally, Caltech alumni Rishi Chandy and Julian Panetta were quite helpful in our development of our approaches to entity extraction and relation building, and of our understanding of the Semantic Web in general.

7. REFERENCES

[1] S. Bechhofer, F. Van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein, et al. Owl web ontology language reference. *W3C recommendation*, 10:2006–01, 2004.

[2] T. Berners-Lee. Information management: A proposal. 1989.

[3] T. Berners-Lee. The next web. *TED.com*, 2009.

[4] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Series. O'Reilly, 2009.

[5] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[6] D. Brickley and R. Guha. Resource description framework (rdf) schema specification 1.0. *W3C Candidate Recommendation*, 27:2001–03, 2000.

[7] J. Euzenat, A. Ferrara, L. Hollink, A. Isaac, C. Joslyn, V. Malaisé, C. Meilicke, A. Nikolov, J. Pane, M. Sabou, et al. Results of the ontology alignment evaluation initiative 2009. In *Fourth International Workshop on Ontology Matching, Washington, DC*. Citeseer, 2009.

[8] P. Hane. Beyond keyword searching-oingo and simpli. com introduce meaning-based searching. *Information Today*, 17(1):57–68, 2000.

[9] Y. Jo, J. E. Hopcroft, and C. Lagoze. The web of topics: discovering the topology of topic evolution in a corpus. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 257–266, New York, NY, USA, 2011. ACM.

[10] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[11] S. Ponzetto and R. Navigli. Large-scale taxonomy mapping for restructuring and integrating wikipedia. *Proc. of IJCAI-09*, pages 2083–2088, 2009.

[12] E. PrudâĂŹhommeaux and A. Seaborne. Sparql query language for rdf. w3c recommendation 15 january 2008. *World Wide Web Consortium*, 2008.