A Machine Learning Approach to Linking FOAF Instances

Jennifer Sleeman and Tim Finin
University of Maryland, Baltimore County
Baltimore MD 21250
{jsleem1,finin}@umbc.edu

Abstract

The friend of a friend (FOAF) vocabulary is widely used on the Web to describe individual people and their properties. Since FOAF does not require a unique ID for a person, it is not clear when two FOAF agents should be linked as co-referent, i.e., denote the same person in the world. One approach is to use the presence of inverse functional properties (e.g., foaf:mbox) as evidence that two individuals are the same. Another applies heuristics based on the string similarity of values of FOAF properties such as name and school as evidence for or against co-reference. Performance is limited, however, by many factors: non-semantic string matching, noise, changes in the world, and the lack of more sophisticated graph analytics. We describe a supervised machine learning approach that uses features defined over pairs of FOAF individuals to produce a classifier for identifying co-referent FOAF instances. We present initial results using data collected from Swoogle and other sources and describe plans for additional analysis.

Introduction

One of the most widely used Semantic Web ontologies is FOAF (Friend of a Friend) which defines classes and properties useful for describing people, their attributes, and their relationships to other people, organizations and objects. FOAF's popularity is evident in the social networking sites that publish profiles using FOAF, the number of RDF documents using the FOAF namespace, the number of foaf:Agent instances or the volume of RDF triples using FOAF terms (bil ; Ding *et al.* 2005). The widespread use can be explained by the common need to publish, find and reason with basic data on people and also the lightweight and practical design of the FOAF vocabulary.

A person new to the Semantic Web studying the design of the FOAF vocabulary might initially be perplexed. One of the principles underlying the Semantic Web is that it enables us to give concepts and individuals URIs that serve as unique identifiers, removing much of the ambiguity that comes with using human language and representation systems that are not designed to be distributed and open. The FOAF ontology allows one to create a foaf:Agent instance that indeed represents a single, unique individual and to describe its properties and relations. What it does not include is a property that represents a globally unique identifier (GUID) that can be used to recognize when two foaf:Agent individuals are co-referent, i.e., refer to the same individual whether real or fictional. Though recently it has become popular to use the FOAF instance URI as the GUID, this does not guarantee that a person cannot be represented by more than one URI.

What we are left with is a problem that is common to other representation systems, including natural language descriptions, database address records, and even official government records – we describe individuals by enumerating their relevant attributes and properties. Given two such descriptions and depending on the context and task at hand, we may find enough evidence to conclude that the individuals are or are not co-referent. If there is enough supporting evidence to conclude that individuals are co-referent, the process of 'smushing' (http://wiki.foaf-project.org/w/Smushing 2010) can be applied, combining the profiles. 'Smushing' FOAF profiles can bring together information from various sources that are determined to represent the same 'person'. One can choose to rely solely on the presence of owl:sameAs, as this property is meant to link individuals (Bechhofer *et al.* 2004). However, relying on this property alone to 'smush' data is not effective, as its presence is not always found and it can also be represented inaccurately. There are multiple techniques used to both identify co-referent FOAF profiles and that perform some type of 'smushing' (Shi *et al.* 2008; Price, Rawles, & Flach 2004; Hogan, Harth, & Decker 2007).

OWL's InverseFunctional property class (IFP) can help provide a way to recognize co-referent profiles, but it does not offer a complete solution. The FOAF vocabulary defines, for example, foaf:homepage and foaf:mbox to be inverse functions, providing strong evidence that two FOAF agents are co-referent if they share an identical value for either of those properties. However, with the popularity of social networking sites that support FOAF extraction of profiles, extracted FOAF profiles do not always include such an inverse functional property and sometimes these properties can be misused. This was discovered by previous work (Shi *et al.* 2008; Hogan, Harth, & Decker 2007; http://blog.aidanhogan.com/2008/10/god-entity.html 2008). For example, (http://blog.aidanhogan.com/2008/10/god-entity.html 2008) describes how a list of FOAF profiles produced by exporters with empty foaf:mbox values all contained duplicate foaf:mbox_sha1sum values. In (Hogan, Harth, & Decker 2007), it was discovered that a large portion of FOAF profiles which contained foaf:weblog property, contained a duplicate value for each profile, representing the community web logs. In (Shi *et al.* 2008), foaf:weblog and foaf:homepage in particular were found to be representing collective sites. In our own data, we have found foaf:homepage values representing community web sites and we have also found FOAF profile with the absence of inverse functional properties altogether. It is not uncommon for several people to share a common value that is contained within an inverse functional property.

In the rest of this paper we discuss the problem of deciding when RDF descriptions of two FOAF agents are likely to be co-referent. One common approach is to use the presence of FOAF properties declared to be inverse functions (Shi *et al.* 2008; Golbeck & Rothstein 2008; A. & H. 2005; Hogan, Harth, & Decker 2007) (e.g., foaf:mbox) as a validation that two individuals are the same. A second (Price, Rawles, & Flach 2004) applies heuristics based on the string similarity of values of FOAF properties such as name and school as evidence for or against co-reference. We describe a supervised machine learning approach using features defined over pairs of FOAF individuals to produce a classifier for identifying co-referent FOAF instances. Our approach is comparable to (Price, Rawles, & Flach 2004), we rely not strictly on the semantics of FOAF but a combination of semantics and the data itself. We take this one step further by using a Support Vector Machine (SVM) to classify the data rather than using a heuristics-based approach. This enables us to generalize our method such that new co-referent examples can be discovered without requiring the heuristics to frequently change. We present initial results using data collected from a number of sources on the Web. We also describe plans to add additional features that should improve performance.

Other approaches as mentioned above are successfully determining co-referent profiles by using different methods but they may require changes to heuristics based on newly discovered patterns in the data. For example, using a strict IFP approach, one has to learn the cases of when files are not properly using IFPs as intended. In purely heuristic string similarity approaches, common properties are not always present in both profiles and noise in the data can obfuscate the fact that two profiles represent the same person. Noise and anomalies in the data can feasibly be captured in a classification model.

Our technique could also be abstracted and applied to other domains, in particular it can be used to disambiguate people on the web. On the web, a person's name can be common enough that if one were to perform a search for a person, they would receive results that are relevant to a number of entities that share that common name. Ideally one would be able to cluster data that is pertinent to each entity relating solely to that entity. This is a common problem currently addressed by a number of 'Web People Search Engines' that attempt to combine and cluster information representing a single entity (Artiles, Gonzalo, & Sekine 2009). We would like to note that clustering is part of our approach but an in-depth discussion on this topic goes beyond the scope of this paper.

In this paper we refer to a FOAF individual as a FOAF instance or a FOAF profile interchangeably. We do not presume a FOAF document is equal to a single FOAF instance. We do however validate FOAF documents before processing them, using a set of rules, as defined later in this paper. We also tend to currently use documents that are typically representing a single person, however the next version of our software will use multi-person documents as well.

Approach

The overall approach we are developing is similar to the one used in other co-reference problems, such as (Volz *et al.* 2009; Mayfield *et al.* 2009). Given a collection of FOAF instances to compare, we would like to cluster them into sets that we believe refer to the same person in the world. This process is divided into three stages: (i) generating candidate pairs, (ii) classifying the candidates as co-referent or not, and (iii) creating clusters.

Overview

Generating candidate pairs. Given a potentially large collection of N FOAF instances we could proceed by testing each of the N*(N+1)/2 - N possible pairs to see which are co-referent. Since the vast majority of the pairs will not be matched and the co-reference test will be relatively expensive, we start by filtering the possible pairs to produce a smaller set of candidates using a computationally simple heuristic test. The result is a set of FOAF instance pairs that can be used for both training and generating a test set that will be run through the classifier in step two.

Classification. The co-reference classifier takes a pair of FOAF instances and decides if they are co-referent or not. The construction and training of this classifier is described in the next section. Depending on the system used, our classifier might return a Boolean answer (co-referent or not) or an answer together with a score that can be used as certainty measure.

Clustering. Once the candidate pairs are classified, we are left with a set of pairs thought to be co-referent. We can consider this to be a graph where the nodes are the original set of FOAF instances and an edge exists between two nodes just in case they were a candidate pair and the classifier determined that they were co-referent. The final step is to find connected components in the graph, each of which represents a set of FOAF profiles that represent the same person.

In the remainder of this section we describe the process of generating candidate pairs and classification in more detail. The clustering step is still under development.

Generating Candidates

Candidates are generated based on a simple heuristic that is used to generate potential pairs that are likely to match. Using the properties common among pairs of FOAF instances, we calculate a simple score based on string matches. The objective of this process is to produce a subset of potential candidates that is smaller than the total set, thereby removing any unnecessary matching attempts that would result in a non-match. On average, we generate a candidate set for each url of 9 potential matches based on processing 60,000 FOAF profiles of which 10,000 did not pair at all.

Co-reference Classifier

We use a Support Vector Machine (SVM) for classifying our data. A SVM provides a way to classify both linear and nonlinear (Joachims 2002). We chose to use a SVM due to its accuracy and performance (Joachims 2002). SVMs were successfully applied to other text classification problems (Mayfield *et al.* 2009) that were similar to this problem. We could potentially have a large number of relevant features and SVMs are known for supporting such a requirement (Joachims 1998). We used a simple linear kernel and the feature set is based on FOAF properties.

We designated the FOAF properties shown in table 1 to be of relevance to our classification. We chose these properties based on statistics of our dataset and based on the likelihood that they would provide data that would be distinguishable. A distance measure is calculated for each property and used as a feature. In addition, for each property, a feature is generated indicating whether the feature is inverse functional. In the case of foaf:knows, matches are calculated for the set of 'Persons' that an instance 'knows' and used as a feature. We are continually adding features as the system matures.

Current FOAF properties used as features
foaf:family_name
foaf:name
foaf:mbox_sha1sum
foaf:phone
foaf:birthday
foaf:nick
foaf:firstname
foaf:givenname
foaf:surname
foaf:mbox
foaf:jabberID
foaf:msnChatID
foaf:yahooChatID
foaf:weblog
foaf:aimChatID
foaf:homepage
foaf:icqChatID
foaf:accountname
foaf:img
foaf:schoolhomepage
foaf:publications
foaf:knows

Table 1: These FOAF properties were used to develop features for the FOAF instance co-reference classifier.

Implementation

The system is written in Java and uses a MySQL database to store data. It performs Swoogle ingestion and other ingestions based on crawled data. Triples are parsed using Jena (Carroll *et al.* 2003) and stored in a MySQL table. In addition, a hash of FOAF properties is stored, along with general information. Candidate pairs are generated by using a heuristic that simply eliminates pairs that are not likely to match. Candidate pairs are used to formulate a training set and a test set. When generating training and test sets, a heuristic is used to determine whether a particular item is a positive or negative case. This is done to reduce the time it takes to gather our training and test sets. We cannot always accurately determine if items are positive or negative. Therefore only classes that can confidently be classified as negative or positive with this heuristic are labeled, others are not labeled. By doing so we are able to generate a large number of classes (particularly negative classes) without a lot of manual work. A Confusion matrix is generated for the test set and we record accuracy, recall and precision. Our classifier can then be used to test pairs of FOAF instances.

Data Sources

Our data sources include both FOAF instances retrieved based upon a list of URLs extracted from Swoogle (Ding *et al.* 2004), a set of FOAF instances based on a web crawler crawling a set of URLs of sites based on the following list [http://esw.w3.org/topic/FoafSites], FOAF instances parsed from the Billion Triple Challenge 2009 smaller test dataset (http://vmlion25.deri.ie/index.html) and other social networking web sites that produce FOAF data. We are currently using a small subset of data to perform initial evaluation of the system, with plans to use larger data sets as our system matures. When retrieving documents based on the Swoogle (Ding *et al.* 2004) listing, an attempt is made to retrieve the latest version of the document and if the latest version is no longer accessible we retrieve the cached version from the Swoogle database. Our data set currently contains over 2 million FOAF instances and our current work is based upon 60,000 FOAF instances.

Documents are retrieved from both blog and non-blog web sites. We do not distinguish these documents as was done in previous work (Ding *et al.* 2005) because this property has little effect on our work. Duplication of documents is handled during the feature generation phase.

The criteria we use to determine that a document is a valid FOAF document include the following:

- The document must be a valid RDF document. If the document is not valid, it will fail to be parsed by the RDF parser and therefore not accepted.
- The document must use the FOAF namespace.
- The document must contain at least one foaf:Person class instance.

Many FOAF documents contain multiple FOAF profiles. In some cases, a document is intended as a profile for a single individual but it includes foaf:knows links to other foaf:Person instances included in the document. In other cases, there is not a *primary* foaf:Person instance, as might be the case for a document about a paper that includes multiple authors, each represented as a foaf:Person. We currently focus on profiles that include a single individual which may or may not include foaf:knows.

Feature Set

Property-specific features include the following types:

1. Inverse functional properties
2. Simple distance measures of properties common to both instances
3. More complex distance measures, which might include unpacking semantic information (e.g., the geographical distance between to geotags) and resolving entity mentions (e.g., Baltimore) to linked data nodes
4. Partial analysis of the graphs centered on the instances, such as the immediate (one-hop) social networks formed by foaf:knows properties

Simple distance metrics were calculated using the *levinshtein distance* (Levenshtein 1966) method however future work will support a configurable way to specify which distance method the user would prefer to use.

FOAF's InverseFunctional Properties
foaf:mbox_sha1sum
foaf:mbox
foaf:jabberID
foaf:msnChatID
foaf:yahooChatID
foaf:weblog
foaf:aimChatID
foaf:homepage
foaf:icqChatID
foaf:isPrimaryTopicOf
foaf:openid
foaf:sha1

Table 2: FOAF's inverse functional properties provide strong evidence for co-reference

Inverse Functional Property. Inverse functional properties are useful in that properties which are defined as inverse functional should provide some degree of certainty that if we match a property among two FOAF instances that is an inverse functional property, the two instances could represent the same person given that the property relates to uniquely defining the person. Though the specification (Brickley & Miller 2003) states that for a FOAF instance a foaf:mbox should not be used to represent two different individuals, it is not unlikely that two individuals could share an email address. Previous work (Shi *et al.* 2008; Hogan, Harth, & Decker 2007; http://blog.aidanhogan.com/2008/10/god-entity.html 2008) also provide examples of how these properties can be misused. Of our 60,000 FOAF instance 550 did not contain an inverse functional property. We also made observations where certain inverse functional properties contained duplicate data. Therefore we chose not to rely solely on this as a definitive match indicator. An inverse functional property feature is used as an additional feature that increases the likelihood that a match between FOAF instances could represent the same person.

Simple Property Matching Distance. A simple property match is when a single property matches within the two FOAF instances being evaluated. For example, foaf:name matches in both instances.

Partial Property Matching Distance. In some cases, a property has a subpart which represents uniqueness that can be used as a distinguishing string to be matched to a subpart of the same property in a different instance. For example, part of the foaf:weblog property offers a partial match.

Cross-Property Matching Distance. In some cases, either a full property or a subpart of a property can be used to match a different property in another FOAF instance. In some of our gathered FOAF instances we discovered properties that were commonly cross-matched. For example, a

```
<foaf:Person rdf:ID="me">
<foaf:name>Daniel Krech</foaf:name>
<foaf:nick>eikeon</foaf:nick>
<foaf:mbox_sha1sum>5a8...d022</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="http://eikeon.com/"/>
</foaf:Person>

<foaf:Person rdf:ID="me">
<foaf:name>Daniel Krech</foaf:name>
<foaf:mbox rdf:resource="mailto:eikeon@eikeon.com"/>
<foaf:nick>eikeon</foaf:nick>
</foaf:Person>

Sources:
http://www.advogato.org/person/eikeon/foaf.rdf
http://eikeon.com/foaf.rdf
```

Figure 1: Simple property match

```
<foaf:Person>
<name>Richard Stallman</name>
<homepage rdf:resource="http://www.gnu.org/"/>
<weblog rdf:resource="http://identi.ca/rms"/>
</foaf:Person>

<foaf:Person>
<foaf:name>Richard Stallman</foaf:name>
<foaf:nick>rms</foaf:nick>
<foaf:mbox_sha1sum>685...ecd</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="http://www.gnu.org/"/>
<foaf:weblog rdf:resource="http://.../person/rms/..."/>
</foaf:Person>

Sources:
http://www.advogato.org/person/rms/foaf.rdf
http://identi.ca/rms/foaf
```

Figure 2: The partial match between the weblog values is evidence for co-reference.

foaf:name string part would correspond to a foaf:nick.

Candidate Designation

The heuristic for determining candidates is used to reduce the set of URIs to be processed. Candidates are generated based on a simple heuristic that is used to remove pairs that are unlikely to match. We calculate a simple score based on string matches of common properties and use a threshold as a way to control the filter for candidates. Properties are associated with a weight given how likely they are to indicate a match. The weight is based upon how likely the property will contain a value that is unique and therefore providing stronger evidence that the profiles are co-referent if matched. If the property is an IFP then the weight would likely be higher than an non-IFP property. For instance, foaf:firstname is weighted lower than foaf:mbox. For each possible pair of FOAF instances, an exact match is attempted for each property. If the exact match returns a false match

```
<foaf:Person>
<mbox_sha1sum>ed...a3</mbox_sha1sum>
<name>Nitro Velvet</name>
<weblog rdf:resource="http://identi.ca/nitrovelvet"/>
</foaf:Person>

<foaf:Person>
<foaf:name> </foaf:name>
<foaf:nick>nitro</foaf:nick>
<foaf:homepage rdf:resource=""/>
</foaf:Person>

Sources:
http://identi.ca/nitrovelvet/foaf
http://robots.net/person/nitro/foaf.rdf
```

Figure 3: A More complex value match

then a partial match is attempted. If the weighted average of all the properties is below a set threshold then a cross-property match is attempted for all the properties. By performing this step we can significantly reduce the number of potential matches per url which has improved total running time. This step also produces a richer set of candidate pairs as we eliminate the cases which are confidently not co-referent.

Generating the Training and Test Set

We use this filtered subset of candidates to generate both training sets and test sets. A distance measure is calculated for each pair of properties. If a property is cross-matchable then we cross-match the property with other cross-matchable properties. For each distance measure we also use the average size of the property value as an additional feature. In addition, the fact that a property is an IFP is also used as a feature. If the profile contains foaf:knows properties then the graphs are used to generate additional features. For specific strings, such as names and urls, transformations are used and aliases are retrieved and used as additional features.

In an attempt to reduce the time it takes to create training and test sets that are labeled as positive or negative, we use a simple heuristic as we process each feature. This heuristic produces a likelihood value of the pair representing a positive or negative case. This does not provide a way to classify our pairs but a rough guide that reduces our manual efforts to generate training and test sets for our classifier. In addition to this process, when parsing pairs that are 'known' to be negative or positive, a negative and positive test pair table is populated. We build our positive test pair table by first exporting user FOAF profiles from a Yahoo social networking site [http://www.mybloglog.com]. We then parse this FOAF information and find the user's Twitter account. We then use [http://semantictweet.com/], a FOAF generator for Twitter profiles, to get a second FOAF profile for that individual.

We will continue to build this table of positive profiles by using this approach with other social networking sites that produce FOAF data.

Classification

We used SVMLight (Joachims 1999) with a linear kernel and standard parameters to build our model for classification. Our system enables one to perform a training process that will generate a training file and call upon SVMLight to generate a model. The model can then be used to perform predictions for both our test sets and any pair of FOAF instances that wish to be matched. We return either the value returned by the classifier or a true/false value depending upon how the system is called upon.

Initial Results

Our current work includes one training set of 500 cases (250 positive cases and 250 negative cases) and two different test sets. One of our test sets represents cases that should be 'easy' to classify (1000 cases - 500 positive cases and 500 negative cases) and our second set represents harder to classify cases (50 cases). Our next set of tests will include a substantially larger set of examples from our data sets.

Calculated Measures. After running our classifier, a confusion matrix is generated based on calculated precision, recall, and accuracy of the test set and the predicted values.

The following measures were calculated for our current test sets:

Test	Precision	Recall	Accuracy
1000 easy cases	97.5%	100%	98.7%
50 hard cases	73.33%	75.86%	63.41%

These measures are based on preliminary tests and really provide us with a foundation for our future work. Our framework enables us to run various sized tests and change features accordingly by examining the calculated measures for each test generated by our analyzer.

Evaluation

When building our training and test set, we automatically determine if an item is a positive or negative test case. We use this technique as a way to reduce the time it takes to build the training and test sets. However, we clearly cannot accurately label all of our cases but we can fairly confidently label extreme negative and extreme positive cases. For instance if two FOAF profiles have exactly the same foaf:mbox or exactly the same foaf:mbox_sha1sum we can safely make this a positive case. What we have found is we can generate many negative test cases but far fewer positive cases. To address this issue we now generate a positive training and test table based upon known positive profiles.

When we determine property matches among strings, we do not currently distinguish common and non-common strings. This actually can decrease the accuracy of our classifier. Our future work looks to address this problem.

Future Work

We are currently improving our training/test set and adding additional features for our classification. Our initial work used the basic features of the FOAF vocabulary. Based on what we have learned, more information can be obtained by using properties from the other categories of the FOAF vocabulary, by using synonyms and word expansions and by including common misspellings. The methodology we use to determine if a pair of FOAF documents is co-referent can be applied to other domains. Future work will also apply our methodology to other domains.

Future work will include exploiting additional properties within the instance that are not of the FOAF vocabulary and using these properties to provide additional evidence as to whether a pair of FOAF instances represent the same individual or not. A portion of our collected FOAF documents had non-FOAF vocabularies that offered additional information such as 'author'. By exploiting these additional properties, we could increase accuracy particularly when a FOAF property is absent and the non-FOAF property offers the same meaning.

Many properties asserted about FOAF instances have string values that refer to entities. Examples from the core FOAF vocabulary are foaf:Organization and foaf:fundedBy. We would like to recognize that two strings refer to the same entity when their values are different but known aliases or alternate names. Luckily, for many entities, it is easy to generate lists of known aliases drawing on resources such as Gazetteers, Wikipedia and Freebase. We have developed lists of known aliases for organizations and places from data extracted from Wikipedia and Freebase. The data includes set of aliases for about 270K places and 50K organizations. For example, the alias set for UMBC includes variations on the Universities name due acronyms, abbreviations, and punctuation.

In general, a given string can be a member of several alias sets. This is especially true for acronyms and abbreviations. The current system does not yet exploit these lists but we plan to do so in the next version, probably as an additional string matching metric, e.g., the two instances have a property whose values differ but are in members of a known set of aliases.

As mentioned above, owl:sameAs can be used to 'smush' FOAF profiles. In addition, if there is a Functional Property (Brickley & Miller 2003) present we could make use of this information to aide in our determination as to whether two instances are co-referent. If a Functional Property is present in both profiles and the profiles are said to be representing two different individuals then the value of that property must also be different.

We can also use a 'smushed' instance as an input into our system to provide futher evidence as to whether that newly 'smushed' instance is co-referent with others in the database. This is an interesting enhancement that we plan to explore in our future work.

The FOAF co-referent problem described here is also a common problem in non-FOAF domains. Our approach can be abstracted and applied to other domains. In particular, instance matching (Ferrara *et al.* 2008) among ontologies is a domain that could benefit from such a co-referent solution. Entity clustering (Artiles, Gonzalo, & Sekine 2009) is also another domain which could benefit from our co-referent solution.

Conclusion

We have described an approach that uses supervised machine learning to generate a classifier which can be used to identify co-referent FOAF instances for linking. We currently use the basic set of FOAF properties as features for our classifier and plan to add more FOAF properties as the accuracy of our system improves. Our method is different from past methods (Shi *et al.* 2008; Golbeck & Rothstein 2008; A. & H. 2005) that used inverse functions and string similarity heuristics. We are still early in our work and expect to see more accurate results as we increase the size of our training set and expand our feature sets. We believe that our methodology could be applied to other domains for solving similar problems.

References

A., H., and H., G. 2005. On searching and displaying RDF data from the web. In *Proceedings of Demos and Posters of the 2nd European Semantic Web Conference*.

Artiles, J.; Gonzalo, J.; and Sekine, S. 2009. Weps 2 evaluation campaign: Overview of the web people search clustering task. In *18th International World Wide Web Conference*.

Bechhofer, S.; Harmelen, F.; Hendler, J.; Horrocks, I.; McGuinness, D.; Patel-Schneider, P.; and L.Stein. 2004. Owl web ontology language reference w3c recommendation 10 february 2004. http://www.w3.org/TR/owl-ref/.

Billion triple challenge 2009 dataset.

Brickley, D., and Miller, L. 2003. Foaf vocabulary specification. http://xmlns.com/foaf/0.1/.

Carroll, J.; Dickinson, I.; Dollin, C.; Reynolds, D.; Seaborne, A.; and Wilkinson, K. 2003. The JENA Semantic Web platform: architecture and design. Technical Report Technical Report HPL-2003-146, HP Laboratories.

Ding, L.; Finin, T.; Joshi, A.; Pan, R.; Cost, R. S.; Peng, Y.; Reddivari, P.; Doshi, V. C.; ; and Sachs, J. 2004. Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*.

Ding, L.; Zhou, L.; Finin, T.; and Joshi, A. 2005. How the Semantic Web is Being Used:An Analysis of FOAF Documents. In *Proceedings of the 38th International Conference on System Sciences*.

Ferrara, A.; Lorusso, D.; Montanelli, S.; and Varese, G. 2008. Towards a benchmark for instance matching. In *International Workshop on Ontology Matching, volume 431, 2008*.

Golbeck, J., and Rothstein, M. 2008. Linking social networks on the web with FOAF. In *Proceedings of the 17th International World Wide Web Conference*.

Hogan, A.; Harth, A.; and Decker, S. 2007. Performing object consolidation on the semantic web data graph. In *In Proceedings of I3: Identity, Identifiers, Identification. Workshop at 16th International World Wide Web Conference (WWW2007)*.

2008. The god entity. http://blog.aidanhogan.com/2008/10/god-entity.html. Accessed January 2010.

2010. Foaf-project.org definition of smushing. http://wiki.foaf-project.org/w/Smushing. Accessed January 2010.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*. Springer.

Joachims, T. 1999. SVMLight: Support Vector Machine. University of Dortmund, http://svmlight.joachims.org/.

Joachims, T. 2002. Learning to classify text using support vector machines methods, theory, and algorithms. Springer.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10:707–710.

Mayfield, J.; Alexander, D.; B, B. D.; Eisner, J.; Elsayed, T.; and Finin, T. 2009. Cross-document coreference resolution: A key technology for learning by reading. In *Proceedings of AAAI Spring Symposium on Learning by Reading and Learning to Read*. AAAI.

Price, S.; Rawles, S.; and Flach, P. 2004. Estimating whether partial FOAF descriptions describe the same individual. In *Proceedings of the Workshop on Friend of a Friend, Social Networking and the Semantic Web*.

Shi, L.; Berrueta, D.; Fernandez, S.; Polo, L.; and Fernandez, S. 2008. Smushing rdf instances: are alice and bob the same open source developer? In *Proceedings of the Third Expert Finder workshop on Personal Identification and Collaborations: Knowledge Mediation and Extraction, (PICKME 2008), co-located with the Seventh International Semantic Web Conference*.

Volz, J.; Bizer, C.; Gaedke, M.; and Kobilarov, G. 2009. Silk - a link discovery framework for the web of data. In *Proceedings of the Second Workshop on Linked Data on the Web (LDOW2009)*.