Interlinking Distributed Social Graphs*

Matthew Rowe
OAK Group
Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello Street
S1 4DP Sheffield, United Kingdom
m.rowe@dcs.shef.ac.uk

ABSTRACT

The rise in use of the social web has forced web users to duplicate their identity in fragmented information spaces. Commonly these spaces contain rich identity representations hidden within walled garden data silos. This paper presents work to export social graphs from such data silos as RDF datasets, and provide linkage between these social graphs according to a graph matching paradigm. Our work contributes to the linked data movement by providing a decentralised social graph containing linked data describing fragmented identity components.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: General

General Terms

Linked Data

Keywords

Semantic Web, Social Web, Linked Data, RDF, FOAF

1. INTRODUCTION

The social web has allowed web users to replicate their offline lives and actions in an online environment. Sharing photos, messaging friends and networking to build relationships both socially and professionally has drawn users into using social web platforms to organise their lives. This has brought about the generation of rich social data attached to individual web users, describing their online identity. Properties of this identiy include biographical information; name, address, date of birth, and social infromation; relationships, contacts and affiliations. The modern web user has a fragmented identity distributed over multiples platforms and services. Each service contains a different social graph, aggregating each graph would generate a clear description of the person, who their contacts are with useful contact information, and an identity reference point through a URI.

Work by the linked data[1] community has linked together existing accessible social web data sources (Flickr exporter, DBPedia) and published the linked content. Despite social web platforms and services offering advanced functionalities to enhance a person's online identity, exporting the internal social graph is not supported therefore inhibiting access to useful datasets. Such sites are described as 'data silos', where data is hidden within a walled garden. Exported and aggregated social graphs from different services could be used when signing up to a new web service to import the user's existing social network, trust networks could be created based on transitive relationships in the social network, and recommendation systems could retrieve suggestions based on the imported social graph (based on the actions of social network members).

This aggregation also creates a dentralised description of a person's online identity. The aggregation contains linked data which can be referenced for additional identity information from distributed data silos. Thus exporting information from closed data services and opening up information for reuse [7].

In this paper we present our contribution to the linked data project by describing an approach to export social data in a semantic graph format using RDF from a range of social web services, and aggregate the generated RDF by providing links between the exported datasets. The former component relies on mapping XML schemas to the appropriate ontology, whereas the latter is a graph matching problem providing links between person instances in separate graphs that are found to refer to the real world person or entity.

Deciding when a link should be created is an issue due to the non-existence of global unique identifiers for people, using simply the name of a person is problematic due to name ambiguity. Therefore we utilise as much additional semantic information as possible to aid the linking process. We present our own approach to matching person instances using low-level reasoning, and evaluate its success against two formal graph matching functions.

2. SOCIAL WEB LINKED DATA INTIATIVES

LDOW2009, April 20, 2009, Madrid, Spain.

[1] http://linkeddata.org/

Representing social data found within social web services and platforms has been investigated by the SIOC (Semantically-Interlinked Online Communities) project[2]. Work by Bojars et al presented in [1] demonstrates how social web platforms such as blogging services and social bookmarking platforms can express their content using the SIOC specification, providing semantic formalisations for authors, and their generated content. SIOC extends the FOAF (Friend of a Friend) specification [3] designed to describe personal data consisting of biographical information and express relationships with other people in a social network. FOAF is well suited to our work, as each social web platform is built on the social model of making relationships and linking people together.

Although such formalisations exist it is essential to export information from data silos into the suitable format. Work by Alexandre Passant presented in [9] describes how information is exported from Flickr[3], the photo sharing platform, as RDF using the SIOC and FOAF specifications. Similarly work has also been carried out to produce an exporter[4] of RDF using FOAF from the microblogging site Twitter[5] , although the exportation of information is somewhat limited by not extracting any geographical or additional biographical information for social network members. Recent work by QDos[6] has created a FOAF Builder application[7] capable of providing linked data representations between a person's interests with the DBPedia dataset. However, in the context of this paper it fails to aggregate social graphs from data silos, instead pointing links to the existence of accounts within those silos for a given person.

Social data extraction was described by Halpin in [6] as exporting information into RDF using GRDDL[8] from XHTML, providing social markup existed. The markup had to be mapped to a well defined vocabulary such as FOAF or SIOC depending on the information used. I.e., the XFN Microformat[9] would be mapped to FOAF. In our previous work [11] we describe our methodology for exporting social data from the social networking site Facebook[10], producing RDF according to the FOAF specification. We extend this work in the context of this paper.

3. REQUIREMENTS

Our approach to aggregating social graphs is split into two distinct stages: The first stage generates RDF using the necessary ontologies from various social web services, and the second stage then aggregates these social graphs. RDF provides a useful formalisation to describe information within each data silo, capturing biographical information and social network information. Where possible our approach must provide links to social network members in separate networks that are the same person. Based on these functionalities we have defined four requirements that our approach must fulfill:

1. Export social data contained within data silos into the same semantic form.
2. Link person instances from separate social networks referring to the same real world person.
3. Maximise the number of correct links while minimising the number of incorrect links.
4. Publish a decentralised linked social graph.

4. SOCIAL GRAPH EXPORTATION

Exporting social graphs from walled garden data silos commonly involves the trivial task of mapping XML schemas offered by the web service to a semantic specification. We extend our previous work in [11] by incorporating OpenID[11] to address person resolution, and enable information linkage. At a low-level this involves the requirement of an OpenID resource for the social graph owner, which is then assigned to the `foaf:Person` instance in the graph using the `foaf:openid` relation. For exporting social graphs, we use the FOAF specification to describe available social information in a semantic format. In the remainder of this section we present several exporters developed to extract RDF from several web platforms[12].

4.1 Social Networking Sites

We have created social graph exporters for two social networking sites. The first exporter extracts social data from Facebook by mapping the returned XML response to the FOAF ontology thus capturing the identity information, and the social network consisting of instances of `foaf:Person` linked by the `foaf:knows` property to provide relationship ties. The FOAF ontology is well suited to capturing social information due its extensive expression and definition of identity information. Thus we consider XML schemas used by social web services to be subsets of the FOAF specification. We found that there were no properties that we could not map from the XML schema to concepts from the FOAF ontology.

Geographical information including city and country is formliased as an instance of `geo:Feature` by assigning the person's city and country to the `geo:name` and `geo:inCountry` properties from the Geonames ontology[13]. We chose the Geonames ontology due to its adoption by the Semantic Web community as a standard for describing geographical concepts. Each social network member is assigned a URI using the user identification number from the service, unfortunately the exportation of email addresses and web sites is not allowed which would serve as a useful dereferencing point. The following is a snippet of the RDF exported from Facebook.

```
<foaf:Person rdf:ID="#me">
  <foaf:name>Matthew Rowe</foaf:name>
  <foaf:givenname>Matthew</foaf:givenname>
```

[2]http://sioc-project.org/
[3]http://www.flickr.com
[4]http://tools.opiumfield.com/twitter/mattroweshow/rdf
[5]http://www.twitter.com
[6]http://qdos.com/
[7]http://foafbuilder.qdos.com/
[8]http://www.w3.org/2004/01/rdxh/spec
[9]http://www.gmpg.org/xfn/
[10]http://www.facebook.com
[11]http://openid.net
[12]http://ext.dcs.shef.ac.uk/~u0057/SocialGraphAggregator/
[13]http://www.geonames.org/ontology/

```
  <foaf:family_name>Rowe</foaf:family_name>
  <foaf:openid rdf:resource="http://matthewroweshow.blogspot.com/"/>
  <foaf:knows>
    <foaf:Person rdf:about="#617555567">
      <foaf:name>Sam Chapman</foaf:name>
      <foaf:givenname>Sam</foaf:givenname>
      <foaf:family_name>Chapman</foaf:family_name>
      <foaf:img
        rdf:resource="http://profile.ak.facebook.com/617555567.jpg"/>
      <foaf:holdsAccount>
        <foaf:OnlineAccount>
          <foaf:accountServiceHomepage
            rdf:resource="http://www.facebook.com/" />
          <foaf:accountName>617555567</foaf:accountName>
        </foaf:OnlineAccount>
      </foaf:holdsAccount>
      <foaf:based_near>
        <geo:Feature>
          <geo:name>Sheffield</geo:name>
          <geo:inCountry>United Kingdom</geo:inCountry>
        </geo:Feature>
      </foaf:based_near>
    </foaf:Person>
  </foaf:knows>
  <foaf:knows>
    ...
  </foaf:knows>
</foaf:Person>
```

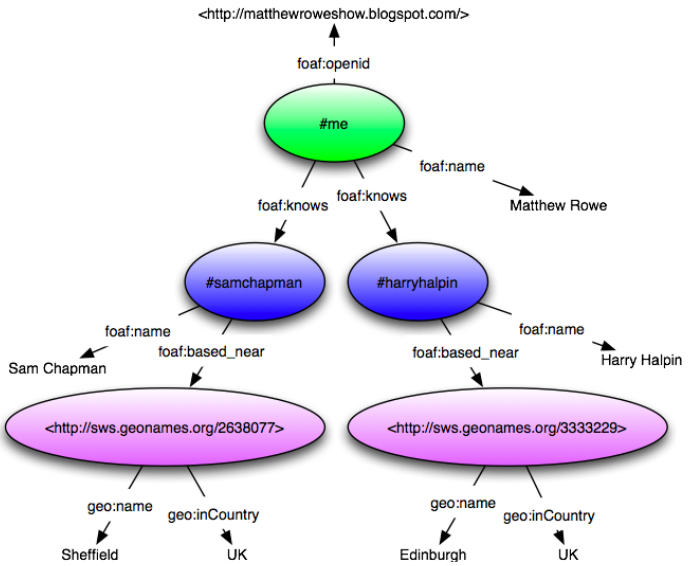

Figure 1: Twitter Social Graph

We applied the same approach when exporting social graphs from the social networking site MySpace[14]. As MySpace follows the OpenSocial[15] specification, our exportation tool can be adapted to easily extract social graph information from several other social networking platforms supporting the same specification (Bebo, Orkut, Hi5). As the OpenSocial specification and MySpace's data accessibility are still in development at the time of writing this paper and conducting our work, we were unable to extract rich social network data. This reduced the exportation process to only the person name of each social network member described using `foaf:name` and the user identification number from the site which was employed as a URI, each is assigned to an instance of `foaf:Person` which we bind to the social graph owner using the `foaf:knows` property. The ability to only extract person names and a URI meant that we were unable to export geographical information.

4.2 Microblogging Platforms

For exporting RDF from the micro-blogging site Twitter we followed the same methodology as the exportation of social graphs from the previous social networking sites. Twitter also offers an XML response (without the required authentication) enabling a mapping to be made between the XML schema to concepts from the FOAF ontology. As Twitter allows access to geographical information we model this data as an instance of `geo:Feature`, and assign the city and country to the `geo:name` and `geo:inCountry` properties respectively. Each social network member is described using an instance of `foaf:Person`, and the `geo:Feature` instance is assigned using the `foaf:based_near` property. We also used the display name used by each member of the social network as their URI. Figure 1 shows an example social graph exported from Twitter.

4.3 Graph Enrichment

Following the exportation of social graphs from their hosting service, it is essential to enrich the graph where possible. The first stage of the exportation process only creates a geographical reference using the place name which can cause problems with ambguity. Therefore we enrich this representation by resolving the place name with unique identifiers. To perform this process we query the Geonames Web Service[16] (which accesses the Geonames dataset) using the `geo:name` and `geo:inCountry` properties assigned to the `geo:Feature` instance for each social network member. This returns a list of possible URIs for the location. We select the most relevant URI from the list and then assign this to the `geo:Feature` instance, and the latitude and longitude of the location which are assigned using the `geol:lat` and `geol:long` properties from the Geolocations vocabulary [2]. This additional enrichment offers extra information for person resolution in the graph space, essential for the following social graph aggregation procedures.

5. SOCIAL GRAPH AGGREGATION

Aggregating social graphs identifies matching `foaf:Person` instances in separate graphs and provides links between these instances using the `owl:sameAs` property. The main challenge we face is deciding if two instances of `foaf:Person` with the same name in different social networks refer to the same real world person or entity. To make this decision we formalise a graph from each instance of `foaf:Person` and all outgoing properties and relations in the social network: This graph contains biographical information about the person, which can then be compared using graph matching techniques to derive a similarity measure and therefore the possibility of a match.

Formally we define the person graph as $G = \langle V, E \rangle$ where V denotes all the nodes within the graph represented as resources and literals extracted from the `foaf:Person` instance in the social network, and E denotes the edges con-

[14]http://www.myspace.com
[15]http://code.google.com/apis/opensocial/
[16]http://www.geonames.org/export/

necting those nodes represented as semantic relations and properties. Therefore the social network found within a given FOAF file contains a set of person graphs, each describing a different real world entity. Imagine we have two FOAF files $F_1$and F_2 where $g \in F_1$and $h \in F_2$ describes all the graphs within F_1 and F_2 respectively, and a similiarity function $sim(g, h)$ that measures the similarity of the two graphs: Our task is to find the graphs in both F_1 and F_2 that achieves the maximum similarity measure whilst exceeding a given threshold, and therefore provide a match. This reduces the aggregation problem to a graph matching task; to investigate this procedure we now present three alternative methods for computing graph similarity. We evaluate and compare the success of each method later.

5.1 Node/Edge Overlap

Graph matching using node and edge overlap as described in [8] utilises the Jaccard distance [4] between two graphs to derive a similarity measure, the intuition behind this method being that the fewer edits required to transform one graph into another, then the more similar the graphs are. Essentially this method counts the number of edit operations to perform the transform, which are then normalised by summing the node and edge counts from each graph. Therefore the edit distance is described as:

$$sim(g, h) = 1 - 2\frac{|V_g \cup V_h| + |E_g \cup E_h|}{|V_g| + |V_h| + |E_g| + |E_h|}$$

When generating the intersection of the node set from g and h we used the Levenstein string similarity measure [5] to derive term similarity. If the similarity measure is above a predefined threshold then the nodes are classed as equivalent. String matching is not required for finding the intersection between the edge sets due to their semantic types being formally defined, this reduces the comparison to a trivial matter of binary comparison of objects. It is important to note that for consistency in our work, we used the same Levenstein string similarity measure when comparing literals in each graph matching method.

5.2 Node Mapping

Work by [10] provides a method to match graphs by providing possible mappings between nodes in each graph. In a similar approach to our work, semantic triples are modeled as edges in a graph describing a link between an object and a subject by a predicate taking the form (s, p, o). This approach derives similarity measures between every possible combination of object nodes (with an outgoing edge) and also between every combination of subject nodes (with an incoming edge) in separate graphs. Thus creating a list of all possible node mapping combinations between two graphs along with the similarity measure of the two nodes. The set of node mappings between two graphs is chosen that maximises the cumulative similarity score. Therefore the similarity between two graphs is defined as:

$$sim(g, h) = \frac{\sum_{i=1}^{n} max(strsim(s_g^i, s_h)) + \sum_{j=1}^{n} max(strsim(o_g^j, o_h))}{\#mappings}$$

The application of the approach in [10] is to detect links between music datasets for artists, and records. In the context of our work we follow a similar line of application by attempting to provide links between members of different social networks, essentially held in separate datasets. Therefore we can adapt this approach for our work by comparing person graphs, and deriving matches based on the best possible node mappings according to the cumulative similarity score: This score like node/edge overlap must exceed a predefined threshold for a match to be valid.

5.3 Graph Reasoning

Due to the semantic structure of the graphs we are comparing it is possible to utilise semantic metadata to detect a positive match using some basic low-level reasoning: Imagine we have two graphs that we wish to compare: We extract the string literal from both graphs connected from the `foaf:Person` instance using the `foaf:name` property, thereby returning the person name of each graph. We compare the names using the Levenstein string metric to derive a match. If the names match we then move on to comparing other properties from each graph to confirm that the `foaf:Person` instances are in fact referring to the same real world object.

5.3.1 Unique Identifiers

Unique identifiers, where available, can be exported from social web services and defined using the `foaf:homepage`, `foaf:mbox` and `foaf:phone` properties for the website, email address and telephone number respectively within the social graph. We find the edges in each graph that point to such unique identifiers, and compare them. Our intuition is that a match would provide suffficient confidence to confirm the link between `foaf:Person` instances in both social networks. However, should an edge only exist in one graph there is not sufficient knowledge to confirm the link, therefore we move on to analysing further semantic information defined in the graph space.

5.3.2 Geographical Reasoning

When deciding if two people from different social networks refer to the same real world entity we rely on geographical information as another useful information source for reaching a match decision. We follow the intuition that the owner of several social networks would not be friends with two or more people who share the same name and live in the same place. For example, a person named "Matthew Rowe" is friends with "Sam Chapman" on Facebook, and friends with "Sam Chapman" on Twitter. It is likely that "Sam Chapman" refers to the same person in each social network. On Facebook "Sam Chapman" is described as living in "Sheffield" and on Twitter "Sam Chapman" is also described as living in "Sheffield". Therefore we believe such additional information is sufficient to confirm that both instances of "Sam Chapman" are the same person, and should therefore be linked.

We extract the `geo:Feature` class attached to each instance of `foaf:Person` by the `foaf:based_near` property. In the previous section, we added additional geographical edges to the `geo:Feature` instance to describe the latitude and longitude of the location, together with a URI obtained from the geonames web service. Given that we wish to compare instances of `foaf:Person` sharing the same name, we

first compare the location URI assigned to the `geo:Feature` class. Should the URIs match then we confirm that both `foaf:Person` instances refer to the same real world entity.

However, if the URIs are different we compare the geographical proximity of the locations. Our reasoning behind this comparison is that people will divulge more sensitive information in different social networks, for example in a walled garden social networking site the user feels safer, averting prying eyes and would therefore state what suburb they reside in. Conversely, on a micro-blogging platform the user may only define the city they reside in. One method is to derive the geographical distance using the latitude and longitude described by the `geo:lat` and `geo:long` properties, and calculate the distance between the two points using the Haversine formula from [13]. Should the derived distance be less than a predefined threshold then the person instances are deemed to be the same. However, following several experiments we found such a method to be prone to making incorrect matches in rural areas. Therefore decided to use the semantics of the location to better effect:

We analyse the place names to derive a relation between the locations and discover the semantics of that relation, if one exists. For example, a person named "Matthew Rowe" may reside in "Crookes" whereas another person, also named "Matthew Rowe" may reside in "Sheffield". We derive the locality of Crookes to analyse if there exists a relation to Sheffield. To do this we query DBPedia[17] using the following SPARQL query:

```
SELECT ?city WHERE {
  <http://dbpedia.org/resource/Crookes>
    <http://www.w3.org/2004/02/skos/core#subject>
      ?districts .
  ?districts
    <http://www.w3.org/2004/02/skos/core#prefLabel>
      ?city
}
```

This query returns the literal "Districts_of_Sheffield" describing the label for the DBPedia category containing all districts in Sheffield. The `geo:name` property can then be matched against this literal to confirm that the location "Crookes" is a district of "Sheffield" and therefore the graphs should be linked as they refer to the same real world entity. We define the similarity function using the following pseudocode:

$sim(g, h)$:
  Get person name n_g from g using `foaf:name`
  Get person name n_h from h using `foaf:name`
  If $strsim(n_g, n_h) > threshold$

    Get $mbox_g$ from g using `foaf:mbox`
    Get $mbox_h$ from h using `foaf:mbox`
    If $mbox_g$=$mbox_h$
      return match

    Get $homepage_g$ from g using `foaf:homepage`
    Get $homepage_h$ from h using `foaf:homepage`
    If $homepage_g$=$homepage_h$
      return match

    Get location URI $geoid_g$from g assigned to `geo:Feature`
    Get location URI $geoid_h$from h assigned to `geo:Feature`
    If $geoid_g = geoid_h$
      return $match$
    Else If $geoid_g = null$ and $geoid_h = null$
      return $maybematch$
    Else
      Get city name c_g from g using `geo:name`
      Get city name c_h from h using `geo:name`
      return $checkSuburb(c_g, c_h)$
  Else
    return $nomatch$

$checkSuburb(c_g, c_h)$:
  Get districts_of_city label l_g for c_g
  If $strsim(l_g, c_h)$
    return $match$
  Else
    return $maybematch$

As we demonstrate above, the $sim()$ function returns three classes of match: $match$, $maybematch$, and $nomatch$. If the `foaf:name` and `geo:Feature` URI match in each graph then we are fairly confident that the `foaf:Person` instances refer to the same real world entity, we also believe that the same level of confidence should be attached to matching a suburb using the the $checksuburb()$ function. In both cases we return $match$. However, we are less confident when location infomation is not available for either person, likewise if we cannot discover if either `foaf:Person` instance's location is somehow related, therefore we only return a $maybematch$. Only when the `foaf:name` properties do not match are we confident that the `foaf:Person` instances refer to different people.

6. PRODUCING LINKED DATA

Following the previous section we now have a set of matched graphs where each graph refers to the same real world entity or person. We produce links between these representations in the form of a new RDF graph describing the aggregated social graph content. We do not wish to duplicate information contained within each exported social graph, but instead provide links to this information for later reuse (we explain our reasoning behind this decision in the preceding subsection). For biographical information we aggregate all available properties from each social graph to generate a complete identity representation. For example the Facebook social graph contains identity information such as name, and data of birth, whereas the Twitter and MySpace social graphs contain the homepage, aggregating this contain defragments this person identity to generate a complete profile.

A new social network is created containing the aggregation of individual social networks from each social graph, matched instances of `foaf:Person` are merged to create a new instance as follows:

[17] http://dbpedia.org

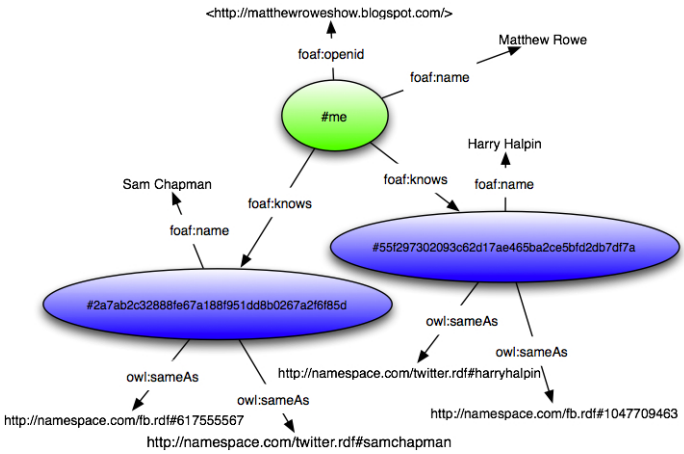

Figure 2: Linked Social Graphs

```
<foaf:knows>
  <foaf:Person rdf:about="#samchapman">
    <foaf:name>Sam Chapman</foaf:name>
    <owl:sameAs
      rdf:about="http://namespace.com/fb.rdf#617555567"/>
    <owl:sameAs
      rdf:about="http://namespace.com/twitter.rdf#samchapman"/>
  </foaf:Person>
</foaf:knows>
```

For this instance we only include the `foaf:name` property and an identifier. We use hash values for identifiers according to guidelines described in [12] due to the relatively small size of the datasets being considered for linkage. Where possible we reuse the identifiers from the available social graphs. As figure 2 demonstrates we reuse the identifier from the Twitter social graph for the merged `foaf:Person` instances. For `foaf:Person` instances that contain no aggregated content (I.e., only appear in the Facebook social graph), we simply reuse the identifier from the accompanying social graph (eg. User identification number). For each merged instance of `foaf:Person` we include a reference using the `owl:sameAs` property to the resource containing the instance.

6.1 Social Graph Control

By providing linked data representations for each instance of `foaf:Person` in the aggregated social graph we attempt to minimise the duplication of personal information while offering decentralisation through linked data. This minimisation is an essential component of the defragmentation of identity information. It also passes responsibility for data access to separate locations and therefore separate access policies. For example, if a person may wish for their Facebook social graph to remain separate and only accessible by people they know and trust then access responsibility is delegated to the hosting service. Work by Yeung et al presented in [7] presents a case for the control of social data using trusted hosting services, thereby delegating access responsibility to the hosting party. Recent advancements in access to social data now allows authentication to be controlled by such services such as OAuth[18] and recently FOAF+SSL[19]. The latter option being particularly well suited to this setting where users are allowed access to a social graph depending on their authenticated FOAF file containing a trusted person that matches the requested social graph.

Another motivation behind this design decision was for the allowance of extensibility and addition of new triples into the aggregated social graph. If we imagine that a user has created their aggregated social graph containing exported social data from Facebook and Twitter, meanwhile they have also recently built up a rich social graph of contacts and information on the professional networking site LinkedIn[20]. Linking this new social graph to the existing aggregated graph only requires the addition of new triples to the exisiting sctructure, which in essence would be either a new `foaf:Person` instance linked by the `foaf:knows` property for a new person in the aggregated graph space, or a new `owl:sameAs` relation pointing to the `foaf:Person` instance in the exported social graph from LinkedIn.

7. EXPERIMENTS

7.1 Data Sets

In order to evaluate and compare the success of our graph matching methodology against the two alternative methods we generated three files containing valid RDF according to the FOAF ontology. Each file was obtained from a different web service (Facebook, MySpace, and Twitter) using the exportation methodology described in section 4, and each file holds information related to one web user who has an account with each social web service.

We analyse the success of the three matching methods when attempting to match instances of `foaf:Person` in the different datasets by evaluating for type I errors (false positives) and type II errors (false negatives). Positives indicate a match and therefore where the datasets should be interlinked for that concept, negatives indicate where a match should not take place and therefore no linked data should exist. The optimum method should produce neither of these error types.

Figure 3 demonstrates how each individual dataset used in the experiment contains possible overlaps which should be linked together. These overlaps consist of social network members from each dataset who are the same person. For example, "Sam Chapman" appears in the Facebook dataset, and "Sam Chapman" also appears in the Twitter dataset, therefore a decision must be made whether a link could be established. The lack of an intersection between the Twitter and MySpace datasets is due to the fragmentation of identity information in each data silo. The user in question whose information we extracted from each service, used Twitter for professional purposes, Facebook for both professional and social, and MySpace for music and social purposes. Therefore the music and professional elements should not overlap, thus separating the Twitter and Myspace datasets.

7.2 Results

Tables 1 and 2 show the results obtained from our analysis together with the gold standard indicated in the final column. As we can see from Table 1 node/edge overlap returns

[18]http://oauth.net/
[19]http://esw.w3.org/topic/foaf+ssl
[20]http://www.linkedin.com

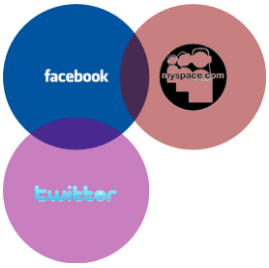

Figure 3: Social Graph Overlap

	N'/E' Overlap	N' Mapping	G' Reasoning	GS'
True Pos	7	8	11	11
True Neg	387	388	389	389
False Pos	2	1	0	0
False Neg	4	3	0	0

Table 1: Matching the Facebook and Twitter Datasets

the poorest results by correctly matching the fewest person graphs and also incorrectly matching the most person graphs. Node mapping performs well but still falsely classifies 3 instances of `foaf:Person` as being no matches. Graph Reasoning outperforms both previous methods by producing the most correct links between the person graphs. The reason for this method's outperformance is due to the large number of triples available in each person graph. Both the Facebook and Twitter datasets contain rich social data that can be exported from each service, namely geographical information that can be used to classify positive matches. In the experiments we permitted links to be created for `foaf:Person` concepts that returned a *maybematch* when performing graph reasoning.

The matching of graphs within the Facebook and MySpace datasets yields interesting results. As Table 2 demonstrates node/edge overlap performed poorly by only finding 2 instances of `foaf:Person` that intersected the datasets. Node mapping derived 10 positive links between `foaf:Person` instances, but incorrectly created 107 links. Graph Reasoning did not classify as many correct links as node mapping yet only falsely generated two links between the datasets. The results from this part of the evaluation throw up some interesting points: When we consider that we wish to minimise the number of false links, then the most naive method; node/edge overlap is the most reliable, however this procedure creates very few correct links. The reason for the poor performance of node mapping (generated 107 incorrect links) and graph reasoning (only generating 5 correct links) is due to the triples available in each dataset. Unlike the Twitter dataset, the MySpace dataset only contains a name property for each instance of `foaf:Person`. This means that the graph matching task has very few triples and therefore a limited graph structure to perform low-level reasoning with, and in the case of node mapping must rely heavily on the string similarity metric to derive the mapping which as the results demonstrates is unreliable.

As mentioned previously when discussing the datasets for use during the experiment, the Twtiter and MySpace datasets do not contain any overlap. When performing the experiments none of the three graph matching methods produced links between these datasets, therefore we decided to omit the results as there was nothing to present.

	N'/E' Overlap	N' Mapping	G' Reasoning	GS'
True Pos	2	10	5	10
True Neg	662	555	660	662
False Pos	0	107	2	0
False Neg	8	0	5	0

Table 2: Matching the Facebook and MySpace Datasets

8. CONCLUSION AND FUTURE WORK

This paper presents our work investigating the exportation of social data described using semantic ontologies, and the linking of this data where possible. Comparing the outcome of this work with the previously detailed requirements it is clear that social data has been exported from data silos in a semantic form, and person instances from separate social networks are linked together where possible. Our method to perform low-level reasoning when matching person graphs yields good results in comparison with similar methods by maximising the number of correctly matched person instances and minimising the number of incorrect matches.

The produced RDF containing linked data describes links between matched person instances. Our decision not to aggregate all biographical information for each social network member is due to the privacy policies that we believe social data must adhere to. Instead, links to the existence of this data are provided. The data contained within the exported social graph data can then be controlled by a separate access policy. This fits in nicely with recent work to address privacy and trust within the Semantic Web community, where technologies such as FOAF+SSL (to control social graph access) and POWDER[21] (to describe social graph properties) are being adopted.

Our future work will include additional social graph exportation tools for other web services, and also the release of the aggregation service. We also plan to test our graph matching approach on additional larger datasets, and hope that existing XML schemas expand to allow additional information to be exported.

9. ACKNOWLEDGEMENTS

We would like to thank both Harry Halpin and Sam Chapman for allowing their information to be presented in this paper as examples. And also Neil Ireson for his meticulous proof reading expertise.

10. REFERENCES

[1] U. Bojars, A. Passant, R. Cyganiak, and J. Breslin. Weaving SIOC into the Web of Linked Data. In *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China*, Apr 2008.

[2] D. Brickley. Basic geo (wgs84 lat/long) vocabulary, January 2003.

[21] http://www.w3.org/TR/2008/WD-powder-dr-20081114/

[3] D. Brickley and L. Miller. FOAF vocabulary specification. Technical report, FOAF project, May 2007. Published online on May 24th, 2007 at.

[4] H. Bunke, P. Dickinson, M. Kraetzl, and W. D. Wallis. *A Graph-Theoretic Approach to Enterprise Network Dynamics*. Birkhauser, 2006.

[5] S. Chapman, B. Norton, and F. Ciravegna. Armadillo: Integrating knowledge for the semantic web. In *Proceedings of the Dagstuhl Seminar in Machine Learning for the Semantic Web*, February 2005.

[6] H. Halpin. Social semantic mashups: Exploring networks using microformats and grddl. In *Proceedings of XML Conference*, 2006.

[7] C. man Au Yeung, I. Liccardi, K. Lu, O. Seneviratne, and T. Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, 2009.

[8] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina. Web graph similarity for anomaly detection (poster). In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1167–1168. ACM, 2008.

[9] A. Passant. :me owl:sameAs flickr:33669349@N00. In *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China*, Apr 2008.

[10] Y. Raimond, C. Sutton, and M. Sandler. Automatic interlinking of music datasets on the semantic web. In *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China*, Apr 2008.

[11] M. Rowe and F. Ciravegna. Getting to me: Exporting semantic social network from facebook. In *Proceedings of the ISWC 2008 Workshop Social Data on the Web (SDOW2008)*, October 2008.

[12] L. Sauermann, R. Cyganiak, and M. Voelkel. Cool uris for the semantic web. Technical report, Deutsches Forschungszentrum fuer Kuenstliche Intelligenz GmbH, 2007.

[13] R. W. Sinott. Virtues of the haversine, 1984.