Disambiguating Identity Web References using Web 2.0 Data and Semantics

Matthew Rowe[a,*], Fabio Ciravegna,

[a] *The OAK Group, Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK*

Abstract

As web users disseminate more of their personal information on the web, the possibility of these users becoming victims of lateral surveillance and identity theft increases. Therefore web resources containing this personal information, which we refer to as *identity web references* must be found and disambiguated to produce a unary set of web resources which refer to a given person. Such is the scale of the web that forcing web users to monitor their *identity web references* is not feasible, therefore automated approaches are required. However automated approaches require background knowledge about the person whose *identity web references* are to be disambiguated. Within this paper we present a detailed approach to monitor the web presence of a given individual by obtaining background knowledge from Web 2.0 platforms to support automated disambiguation processes. We present a methodology for generating this background knowledge by exporting data from multiple Web 2.0 platforms as RDF data models and combining these models together for use as seed data. We present two disambiguation techniques; the first using a semi-supervised machine learning technique known as *Self-training* and the second using a graph-based technique known as *Random Walks*, we explain how the semantics of data supports the intrinsic functionalities of these techniques. We compare the performance of our presented disambiguation techniques against several baseline measures including human processing of the same data. We achieve an average precision level of 0.935 for *Self-training* and an average f-measure level of 0.705 for *Random Walks* in both cases outperforming several baselines measures.

Key words: Semantic Web, Web 2.0, Identity Disambiguation, Machine Learning, Graphs

1. Introduction

The en masse uptake of Web 2.0 sites has altered the role of web users from information consumers to content contributors: Sites such as Wikipedia[1] and Yahoo Answers[2] have provided the functionality to allow web users to produce content, and organisatheir presence.

This elevation has also produced several notable by-products: through standard web search engines such as Google[3] and Yahoo[4], it is now possible to find information about the majority of people online, whether those people wish to be visible in such a way or not. This has lead to several distressing web practices becoming increasingly popular. One such practice is lateral surveillance [1], the action of finding and surveying a person without their knowledge. This has become a widespread practice among employers who are keen to vet potential employees prior to offering them a job and also among people vetting prospective dates. Another web practice on the rise is online identity theft, which currently costs the UK economy £1.2 billion per annum[5]. As web users share more of their personal information within the public domain, the risk of that information being stolen and reused increases.

To assess whether a person is at risk from identity theft or susceptible to lateral surveillance, web resources (web pages, data feeds, etc) containing the person's personal information must be found and the information contained within those resources assessed for risk. For example the UK government guidelines stipulates that the visibility of the person's name, their address and email address could lead to identity theft [6]. The step of finding the correct web resources (herefter *identity web references*) is what this paper focuses on. Finding includes the actual retrieval of Web references pointing to a specific person name (e.g. by retieving Web pages via a standard search engine) and the decision whether the resource actually refers to the person we are concentrating on or to a person with an identical or similar name. We refer to this decision as *disambiguation*. Due to the immense scale of the web and the large amount of references that can be returned for a specific name (e.g. Google returns 4,750,000 for John Smith only), human disambiguation is largely unfeasible, therefore automated approaches are required. Such automated approaches must rely on background knowledge (seed data) about the person whose information is to be found. Producing this seed data manually for one person (let alone for multiple people) can be costly and time consuming, and limited by access to resources.

The central contribution in this paper is an approach to identify and disambiguate *identity web references* for a specified (set of) person(s). The method is completely automatic and does not require the manual construction of any seed data, as the seed data used is automatically extracted and integrated by Web 2.0 resources (e.g. social sites). In particular our paper proposes:

– Techniques to extract data from Web 2.0 platforms, turn this data into RDF data models and combine them into a single Linked Data [21] social graph which can then be used as seed data for automated disambiguation;
– Two novel techniques to automatically disambiguate *identity web references* each supported by Semantic Web technologies and machine learning;
– A detailed evaluation of said techniques using a large-scale dataset, comparing the resulting accuracy to several baseline measures (including human processing of the same data); we show how our techniques largely outperform both humans and state of the art methods in terms of accuracy.

Moreover, we show empirical evidence that data from Web 2.0 platforms is synonymous with real identity information. We present a detailed methodology for extracting a given person's data from multiple Web 2.0 platforms as RDF data models thereby attributing machine-readable semantics to social data, .

Our first disambiguation technique *Self-training* employs a semi-supervised machine learning strategy to iteratively train and improve a machine learning classifier. We model machine learning instances as RDF models and the features of those instances as RDF instances from within the models. This permits the novel variation of feature similarity measures used by the classifier between different RDF graph similarity techniques. Our second technique employs a graph-based disambiguation technique known as *Random Walks*. We construct a graph space by utilising the graph structure of RDF data models, given that our seed data and the set of web resources to be disambiguated are represented using RDF. We are able to weight edges within the graph space according to their semantic constraints, given that edges within the graph space are created from predicates within triples.

We have structured this paper as follows: Section 2 presents current state of the art approaches for monitoring personal information online and identity tracking. Section 3 details the requirements gleaned from limitations in the current state of art, which

[3] http://www.google.com
[4] http://www.yahoo.com
[5] http://www.identitytheft.org.uk/faqs.asp
[6] http://www.getsafeonline.org/nqcontent.cfm?a id=1132

an approach to monitor personal information must fulfil. Section 4 presents the proposed approach to discover personal information on the web aided with the use of Web 2.0 data. Section 5, 6 and 7 present the various stages in the approach: section 5 presents the techniques used to compile seed data. Section 6 describes how we gather web resources to be disambiguated and generate metadata models describing the underlying knowledge structure of those resources. Section 7 presents two complimentary automated disambiguation techniques. Section 8 investigates the performance of these techniques using a large-scale evaluation with a dataset compiled from real-world data. Section 9 presents the conclusions drawn from our work and plans for future work.

2. Related Works

Approaches designed to monitor personal information online fall into two distinct categories: *unsupervised* - which collect web resources, observe common features between those resources and group web resources by their common features and *semi-supervised* - which obtain background knowledge about a given person and use this information to aid the decision process. We now present current work which contains examples of such approaches.

2.1. *Unsupervised Monitoring Approaches*

The unsupervised approach presented in [32] searches for web pages using a person's name, and then clusters web pages based on their common features, which in this case is the presence of person names. Another unsupervised approach from [17] gathers a set of web pages and creates features for web pages by pairing a persons name within the page together with a relevant concept derived from the page (e.g *Matthew Rowe/PhD Student*). Using these pairings a maximum entropy model is learnt which predicts the likelihood that two pairings refer to the same person. Web pages are then clustered based on the likelihood measures of pairings within separate web pages.

Comparable work in [44] provides a similar approach by aligning topics with person names within a web page. Generative models are used to provide the topic to be paired with the person name and web pages are clustered by common name and topic pairs similar to [17]. Features within web pages are represented by named entities in the approach described in [27]. Using these named entities a term frequency-inverse document frequency (TF-IDF) score for each named entity within the web page is calculated. Web pages are then clustered together by common named entities and TF-IDF scores for those entities above a certain threshold.

An unsupervised monitoring approach described in [50] extracts features from web pages including lexical information, linguistic information and personal information. A basic agglomerative (bottom-up) clustering strategy is then applied to a set of gathered web pages, resulting in several clusters where each cluster contains web pages citing a given person. By varying the combinations of features (lexical, lexical+linguistic, lexical+linguistic+personal) [50] demonstrates the variation in clustering accuracy when applied to the same set of web pages. One of major limitations of unsupervised approaches is their unfocussed nature and the need to discover how many different namesakes (different people sharing the same name)are referred to in the set of resources. Moreover, if the task is to find information about a specific person then large amounts of irrelevant information are processed and compared.

2.2. *Semi-supervised Monitoring Approaches*

Background knowledge, or seed data, is supplied to a semi-supervised monitoring approach described in [3] as a list of names representing the social network of a person. Link structures of web pages are then used as features to group web pages into two clusters; relevant which cite the given person and non-relevant which do not. The intuition behind using link structures is that web pages which cite a specific person are likely to be connected, maybe not directly but via transitive paths. Monitoring personal information involves finding and disambiguating those web resources that contain personal information of a given person from those that do not. This disambiguation process is a binary classification problem where web resources must be labelled by the relevant class label. Classifying web resources in such a manner has been investigated in work by [51] which gathers web pages and then labels those pages as belonging to a given class based on supplied seed data, where the seed data is provided as labelled training examples from which a machine learning classifier can be learnt and applied to unlabelled web pages. A similar approach is presented in [18] which, by using labelled seed data, is able to learn a classifier and apply this to a corpus of text documents.

Seed data is essential for semi-supervised approaches in order to focus the monitoring process, however obtaining such data is a difficult process, limited by access to resources, as highlighted in [3]. In several approaches [51] [49] [18] seed data is previously known and supplied with no explanation of how it was obtained and from what sources. One of the limitations of semi-supervised approaches to monitor personal information is this reliance on seed data. If the seed data does not cover the web resources to be analysed then the accuracy of the approach may be limited. Therefore semi-supervised approaches must cope with this lack of coverage, and use a suitable strategy to overcome this limitation.

The growing need to monitor personal information is reflected in the wide range of commercial services now available created to tackle identity theft by tracking personal information of individuals on the World Wide Web. For example Garlik's Data Patrol[7] service searches across various data sources for personal information about a given person by using background information provided by the person when they signed up to the service as seed data (includes biographical information such as the name and address of the person). The Trackur service[8] monitors social media on the web for references and citations with the intended goal of monitoring the reputation of an individual. Similar to Data Patrol, Trackur requires background knowledge about the person whose reputation is to be monitored. Identity Guard[9] is one of the most widely used personal information monitoring services. It functions in a similar vein to Data Patrol and Trackur by using personal information disclosed by the web user to monitor that person's personal information online.

3. Requirements

From the analysis of current state of the art approaches for monitoring personal information on the web we hypothesise that a semi-supervised strategy outperforms an unsupervised approach when finding identity web references of a given person. However, implementing a semi-supervised strategy places emphasis on the need for reliable seed data and for the approach to cope with correctly finding web resources which contain features not present within the seed data. To address these issues the process of monitoring personal information must fulfil the following requirements:

(i) *Full Automation with minimal supervision:* Our decision process within the approach must be able to disambiguate web resources effectively with no intervention following the provision of seed data. In doing so the approach must be fully automated and require no human involvement.

(ii) *Achieve disambiguation accuracy comparable to human processing of data:* Automated approaches are required to alleviate the need for humans to monitor their personal information. Therefore our approach must achieve accuracy levels comparable to human processing.

(iii) *Disambiguate identity web references for individuals with varying levels of web presence:* Our disambiguation techniques must achieve consistent accuracy levels regardless of the web presence of the person whose *identity web references* are to be disambiguated.

(iv) *Outperform unsupervised disambiguation techniques:* Our approach must outperform unsupervised techniques that do not use seed data, thereby demonstrating the benefits of using seed data to aid disambiguation.

(v) *Cope with web resources which do not contain any features present within the seed data:* Although seed data provides background knowledge for disambiguating *identity web references* for a given person, it may not contain the maximum knowledge coverage. In this case disambiguation approaches must be able to cope with this lack of coverage and learn from unseen features to improve their later performance.

Our semi-supervised approach relies on seed data to support the process of monitoring personal information online, therefore the seed data we use must fulfil the following requirements:

(i) *Produce seed data with minimal cost:* Manually constructing seed data is a time consuming and costly process. Therefore techniques are needed which overcome these issues.

(ii) *Generate reliable seed data:* The seed data used by the approach must contain accurate information about the person whose personal information is to be found. Moreover, inaccuracies

[7] http://www.garlik.com/dpindividuals.php
[8] http://www.trackur.com/
[9] http://www.identityguard.com

in this data will effect the overall accuracy of the approach

4. Overview of Approach to Disambiguation

The proposed approach is shown in Fig. 1; it is divided into three distinct stages: the first stage generates RDF models from distributed Web 2.0 platforms describing a fragment of a given person's digital identity contained within a specific platform. By reconciling these digital identity fragments from multiple platforms we combine distinct RDF models into a single Linked Data social graph to be used as seed data. The second stage of the approach gathers web resources from searching the Semantic Web and the World Wide Web by querying a given person's name. We create metadata models for these resources by generating an RDF model describing the knowledge structure of the resource. The third stage of our approach uses the seed data, modelled as a Linked Data social graph, and analyses the web resources to disambiguate the *identity web references* of a given person. In the next three sections we describe the three stages of the approach in detail.

5. Generating Seed Data

In this section we describe our approach for the automatic collection of seed data with no user intervention. We claim that Web 2.0 data constitutes an excellent starting point for the generation of seed data for the disambiguation of *identity web references*. This is because a) it is plentiful and 2) it is representative of the real identity of people (and hence it provides an excellent basis to discriminate identities of homonyms). Sociological studies which analyse the intrinsic nature of Web 2.0 platforms describe users of such platforms as maintaining an identity which is comparable to their real identity. For instance work by [23] states that Web 2.0 platforms are primarily used to maintain offline relationships and similar work by [16] supports this claim by stating that web users use such platforms to reinforce established relationships.

We performed further investigation of this claim by studying 50 users of Facebook[10] from the University of Sheffield[11]. Each participant listed their real world (offline) social network consisting of those

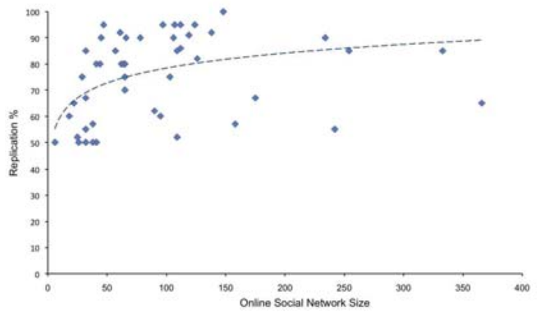


Fig. 2. Results from user study measuring extent to which offline social networks are replicated in an online environment

individuals that they socialised with the most. This network was then compared against their online social network in order to analyse the extent to which the networks were similar. The findings from the study indicated that between 50% and 100% of participants' offline social networks were replicated within the Web 2.0 platform (as shown in Fig. 2), producing an average of 77%. This indicates a strong correlation between the digital identity that is portrayed on a given Web 2.0 platform and the real identities of such web users. To assess the effectiveness of Web 2.0 data for disambiguation purposes, we compiled a dataset of web resources from querying several search engines (Google, Yahoo and Sindice[12]) using distinct person names. By manually analysing the query results we found that on average 5 distinct people appeared within each web resource and that in the majority of cases these people knew one another (i.e. worked together, part of the same sports club). These findings suggest that automated disambiguation techniques would benefit from seed data which describes the people that a given person knows. Therefore Web 2.0 platforms provide an ideal source for such data given their social nature, and as we have demonstrated, the social networks maintained on such platforms.

However, a person's identity is often fragmented and distributed over multiples platforms, where each profile describes a different facet of the persons online identity. It is therefore needed to aggregate this fragmented information together in order to get a more comprehensive set of data for each individual. In the rest of this section we present our approach to generate seed data (shown in Fig. 3) using a two stage process. Firstly we export individual social

[10] http://www.facebook.com
[11] http://www.shef.ac.uk
[12] http://www.sindice.com

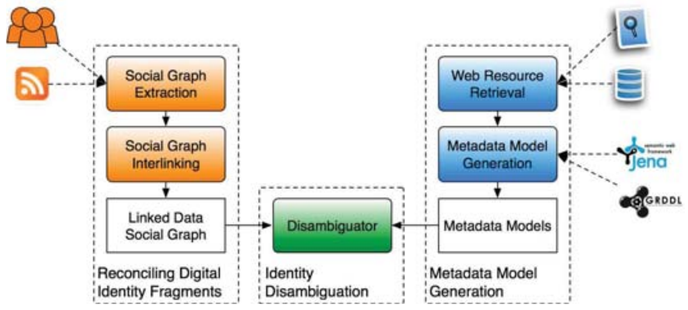


Fig. 1. Overview of our proposed approach to disambiguate *identity web references* using Web 2.0 data

graphs, as RDF models, from distributed Web 2.0 platforms, and secondly we interlink these individual RDF models together.

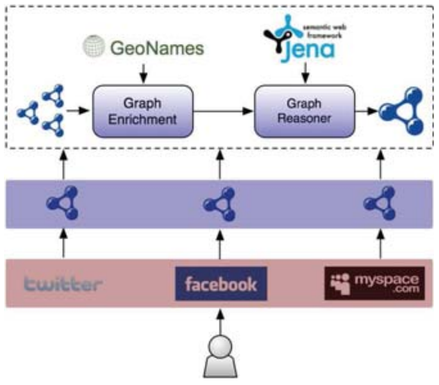


Fig. 3. Generating machine-readable seed data from Web 2.0 platforms

5.1. *Exporting Individual Social Graphs*

Social graphs are exported from individual Web 2.0 platforms as RDF models by using a specific exporter for each platform. Such platforms allow information to be accessed via their API, however this information is often returned in a proprietary format such as XML based on the platform's XML schema. Therefore this response must be leveraged to RDF using consistent ontologies to enable social graphs from multiple platforms to be combined together. We now present several example exporters which generate RDF models from Web 2.0 platforms and describe the internal knowledge structure of the generated models.

5.1.1. *Twitter Exporter*

Our first exporter[13] produces a social graph from the Microblogging platform Twitter[14]. The Exporter requests information from Twitter via the service's API and receives an XML response. We lift this response to RDF by mapping elements in the response to concepts from the FOAF [6] and GeoNames[15] ontologies. The XML contains biographical information and social network information of the Twitter user. We create an instance of *foaf:Person* for the person that the social graph belongs to, which we refer to as the *social graph owner*, and add information to the instance using the *foaf:name* property for the name of the person and *foaf:homepage* for the person's web site. We also attribute a geographical location to the person by creating an instance of *geo:Feature* and relating this to the person using *foaf:based_near*. We assign each instance of *foaf:Person* a URI in the RDF model using a hash URI [40] from the public display name used on Twitter, this provides a reference to the person instance within the exported social graph. For each Twitter user that the person has a relationship with and follows on Twitter we also create an instance of *foaf:Person* and assign the same information as mentioned above to the instance. We use the *foaf:knows* relation to form relationships within the social graph between the owner and each person he/she follows on Twitter.

5.1.2. *Facebook Exporter*

Facebook is a *walled garden* social networking platform, which restricts information access to only

[13] http://ext.dcs.shef.ac.uk/∼u0057/SocialGraphAggregator/twitter.php
[14] http://www.twitter.com
[15] http://www.geonames.org/ontology/

users of the platform. Our exporter[16] works by handling authenticated queries to the platform by first logging in the person whose social graph we wish to export. Once successful we then query the platform's API and handle the XML response in a similar manner to the Twitter exporter. We use the same process as described above by creating an instance of *foaf:Person* for the *social graph owner* and assigning the relevant biographical information to that instance, along with the social network of the person on Facebook. We assign a hash URIs to *foaf:Person* instances using the user identification of the person on Facebook, thereby providing referenceable resources within the generated social graph.

5.1.3. *OpenSocial Exporter*

Several social web platforms and services implement a set of common standards specified within the OpenSocial API specification[17]. By developing an exportation technique conforming to data structures within the OpenSocial API specification, homogenous exporters can be created for platforms which adhere to the same specification (Bebo[18], Orkut[19], Hi5[20]). We implemented an exporter for the social networking site MySpace[21] - which models data access and structures according to the OpenSocial API - using a similar technique to the Twitter and Facebook exporters. Authentication is performed using OAuth[22] prior to granting data access, from querying the API the response is returned as XML which we convert to RDF, describing biographical and social network information using the FOAF ontology and location information using the Geonames ontology.

5.1.4. *Graph Enrichment*

Social graphs exported as RDF models from distributed Web 2.0 platforms contain anonymous resources within the models. Therefore we perform graph enrichment to assign URIs to these anonymous resources. We created instantiations of two classes within exported RDF models; *foaf:Person* and *geo:Feature*. Instances of the former have an assigned URI, whereas the latter do not. Therefore we query the Geonames Web Service[23] (which accesses the Geonames dataset) using the *geo:name* and *geo:inCountry* properties assigned to the *geo:Feature* instance for a URI for the location and add this to the instance. Fig. 4 shows a portion of an example social graph exported and enriched from Twitter.

5.2. *Integrating Distributed Social Graphs*

Using our exporters we are able to generate RDF models describing the digital identity of a given person fragmented across Web 2.0 platforms. We must now combine these social graphs together to produce seed data for disambiguation techniques. Our process of combining the graphs matches *foaf:Person* instances in separate RDF models that refer to the same person, and provide an *owl:sameAs* link between these instances.

Existing work within the field of reference reconciliation has explored the use of training a classifier to detect object matches [13]. References are reconciled in [14] by using contextual information and past reconciliation decisions in the form of a dependency graph. This allows reconciliation of references belonging to different classes. Techniques for the reconciliation of references are presented in [39] which combine both numerical and logical (based on the semantics of distinct data items) approaches . In such work, and in our work also, matching instances using only the *foaf:name* assigned to an instance of *foaf:Person* is unreliable due to person name ambiguity. Therefore we use a strategy called graph reasoning to detect equivalent instances in distinct social graphs which exploits contextual information of the instances.

5.2.1. *Graph Reasoning*

Our graph reasoning technique uses lightweight inference rules to detect equivalent *foaf:Person* instances. Given that we have two social graphs to detect equivalent instances within (G and H), we iteratively move through the *foaf:Person* instances within G and compare each instance with each *foaf:Person* instance in H. We first verify that the names of the two instances are similar using the Levenstein [8] string similarity measure to allow for

[16] http://ext.dcs.shef.ac.uk/~u0057/FoafGenerator
[17] http://www.google.com/opensocial
[18] http://www.bebo.com
[19] http://www.orkut.com
[20] http://hi5.com
[21] http://www.myspace.com
[22] http://www.oauth.net

[23] http://www.geonames.org/export/

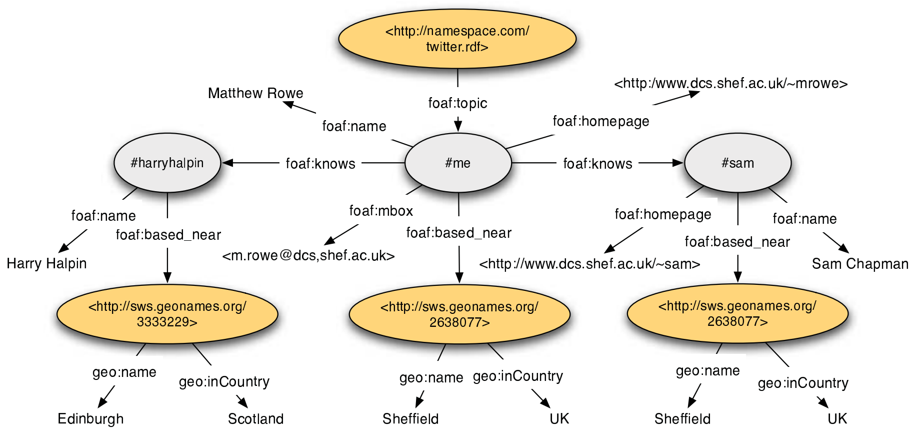


Fig. 4. Portion of an exported Twitter Social Graph

slight variations. If the names are sufficiently similar (i.e. the similarity exceeds a predefined threshold) then we compare additional information within the instance descriptions as follows:

5.2.1.1. *Unique Identifiers* We compare unique identifiers associated with each instance. Given the use of FOAF to describe the semantics of the information within each social graph we compare properties typed as *owl:inverseFunctionalProperty* such as *foaf:homepage*, and *foaf:mbox* from either instance [42]. Such is the strength of these properties that should information associated with both instances *foaf:Person* match then we class the instances as referring to the same real world person. We define an inference rule to detect such a match, using Jena rule based syntax[24], where the antecedent of the rule contains statements which must be matched within the instance knowledge structure for the consequent, which is also a statement, to be inferred:

(?x rdf:type foaf:Person), (?y rdf:type foaf:Person), (?x foaf:mbox ?email), (?x foaf:mbox ?email)
$\rightarrow$ *(?x owl:sameAs ?y)*

5.2.1.2. *Geographical Reasoning* If unique identifiers are unavailable in either subgraph then additional information is analysed. Exported social graphs include location data within *foaf:Person* instance descriptions. First we compare the URIs of location resources associated with the *foaf:Person* instances:

(?x rdf:type foaf:Person), (?y rdf:type foaf:Person), (?x foaf:based_near ?geo), (?y foaf:based_near ?geo)
$\rightarrow$ *(?x owl:sameAs ?y)*

If the URIs are distinct then we assess the semantics of the relation between the locations. In order to derive this relation we utilise open linked datasets containing structured machine-readable data which explicitly defines relations between locations and their semantics. In this instance we query DBPedia[25] through an available SPARQL endpoint[26] using the following query:

```
SELECT ?relation WHERE {
  <http://dbpedia.org/resource/location1>
    ?relation
      <http://dbpedia.org/resource/location2>
}
```

The query returns a relation from DBPedia depicting the association between the locations should one exist, and the semantics of this association. For instance when we use the query to ask for the relation between *Crookes* and *Sheffield* the relation *dbprop:metropolitanBorough* is returned. If a relation exists between the locations then we class the *foaf:Person* instances as being equivalent. This technique also covers abbreviations of location names (e.g. *NYC*) as the relation returned by DBPedia in that case is *dbprop:redirect.* thereby defining the two locations as being equivalent.

[24] http://jena.sourceforge.net/inference/#rules

[25] http://dbpedia.org
[26] http://dbpedia.org/sparql

5.2.2. *Producing Linked Data*

Our graph reasoning method produces a set of matched *foaf:Person* instances. We construct a new linked data social graph by creating a new instance of *foaf:Person* for every instance in each of the compared social graphs. For those that have been matched in separate graphs we create one instance. We maintain the format of the models exported from each of the Web 2.0 platforms by creating one instance of *foaf:person* for the social graph owner and then linking each person in the graph owners social network using the *foaf:knows* relation. For each *foaf:Person* instance in the Linked Data social graph we provide a HTTP URI for additional instance data by providing a reference to the resource using the *owl:sameAs* predicate. An instance of *foaf:Person* which resides in two social graphs is defined as follows:

```
<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#sam"/>
</foaf:Person>
```

Only the *foaf:name* property and a URI are included in the instance description. Hash values are used for identifiers according to guidelines described in [40] due to the relatively small size of the datasets being considered for linkage. Where possible the identifiers are reused from the available social graphs as such identifiers, if user selected, commonly resemble the handle of the user. This social graph now provides the seed data to be used for automated disambiguation approaches.

6. Generating Metadata Models from Web Resources

6.1. *Web Resource Retrieval*

We define a web resource as any document on the web which is accessible by a URI such as web pages or data feeds. We wish to find all the web resources residing on the web which are *identity web references* of a given person. First we must collect a set of web resources which might refer to the person so that we can then use disambiguation techniques to find the resources within the set that are *identity web references*. Therefore we query the World Wide Web search engines Google and Yahoo and the Semantic Web search engines Swoogle[27], Watson[28] and Sindice using the name of the person whose personal information we wish to find. We collect the set of resources returned by those queries.

6.2. *Generating RDF Models from Web Resources*

To enable automated processing of information within the retrieved resources we generate metadata models representing the resources' underlying knowledge using common semantics. Web resources such as ontologies and RDF documents already contain machine-readable content therefore no further processing is required. Instead we must turn our attention to handling XHTML documents which contain implicit knowledge structures and HTML documents which contain no metadata descriptions. In each case we are interested in observing information within the resources which is consistent with our seed data, and therefore describes people.

6.2.1. *Generating RDF Models from XHTML Pages*

Methodologies exist for generating RDF models from XHTML pages containing lowercase semantics. Technologies such as Gleaning Resource Descriptions from Dialects of Language (GRDDL) [48] allow RDF models to be gleaned from web pages using an XSL transformation [28] specified within the header of the page. This transformation lifts the XHTML structure to an RDF model by providing references to concepts from ontologies specified within the transformation. Therefore when a given web resource contains Microformats [29] or RDFa [20] then we use the XSL transformation specified within the header of the resource together with GRDDL to glean an RDF model describing the knowledge structure of the resource. If the resource does not contain an XSL transformation declared within the header then we apply external transformations for the hCard Microformat[29], the XFN Microformat[30] and RDFa[31] to glean an RDF model from the resource.

[27] http://swoogle.umbc.edu/
[28] http://watson.kmi.open.ac.uk/WatsonWUI/
[29] http://www.w3.org/2006/vcard/hcard2rdf.xsl
[30] http://www.w3.org/2003/12/rdf-in-xhtml-xslts/grokXFN.xsl
[31] http://ns.inria.fr/grddl/rdfa/2008/09/03/RDFa2RDFXML.xsl

6.2.2. *Generating RDF Models from HTML Pages*

The majority of resources on the web are encoded using semi-structured HTML, which contain information that automated processes are unable to interpret. Therefore we must construct an RDF model from scratch using information that we observe within the page: First we construct a simple gazetteer based Named Entity Recogniser from a list of person names, which matches the occurrence of a person's name within the web page. When a match occurs an instance of *foaf:Person* is created and the matched person name is attributed to the instance using the *foaf:name* property. It is common for a person's name to appear on a web page with additional information about that person appearing in the vicinity of the name.

For instance a web page containing a list of researchers or employees often has the name of a person together with their contact details. Therefore when a person name is found, we use a window function surrounding the name to find contextual information about the person using regular expressions for the email and homepage of the person, and a geographical place name gazetteer for the location of the person. We then add this data to the instance description using the relevant semantics.

6.3. *Normalising RDF Models to Consistent Semantics*

Ontologies and gleaned RDF models suffer from the problem of heterogeneous semantics describing homogeneous data. Due to the variety of digital identity ontologies it is common for equivalent data in disparate locations to be described using distinct ontologies. For instance an XHTML page may contain personal information described semantically using the *hCard* microformat whereas a published RDF document may contain the same information described using the FOAF ontology. In order for our disambiguation techniques to function the RDF models of the web resources to be disambiguated must have semantics which are consistent with the seed data. Given that our seed data uses the FOAF and GeoNames ontologies to describe the semantics of the data obtained from Web 2.0 platforms we must normalise the RDF models to use the same ontologies thus achieving consistent semantics.

We normalise RDF models to consistent semantics using transfomation rules automatically generated from the Social Identity Schema Mapping (SISM) vocabulary [37]. SISM contains mappings between six commonly used digital identity ontologies: FOAF, the GeoNames ontology, the ontology for VCards [22], the XFN ontology [11], and the Nepomuk ontologies; Personal Information Model Ontology [41], and the Nepomuk Contact Ontology.

SISM contains mapping constructs from the Web Ontology Language (OWL) [43] and the Simple Knowledge Organisation System (SKOS) [26] to cover the range of mappings needed to relate concepts from disparate digital identity ontologies. Mapped concepts within SISM are expressed as instances of *skos:Concept*, this allows the use of 5 mapping predicates between these concepts: *owl:equivalentClass*, *owl:equivalentProperty*, *skos:narrower*, *skos:broader* and *skos:related*. In order to normalise a given RDF model to a consistent set of semantics we consult SISM and automatically generate transformation rules according to the semantics of mappings.

If the RDF model contains concepts which are mapped to the required ontology using using *owl:equivalentProperty*, *owl:equivalentClass* or *skos:broader* then we are able to directly alter the triples containing those concepts. This is due to the semantics of the mappings facilitating a basic transformation. For instance if an instance of *pimo:Person* exists within a given RDF model and we wish to transform the knowledge structure to use the FOAF ontology we consult SISM and find the mapping triple: (*pimo:Person owl:equivalentClass pimo:Person*). Based on the semantics of the mapping (*owl:equivalentClass*) we are able to infer the following rule which is directly applied to the RDF model to transform the class definition and remove the previous triple (denoted by *remove(i)* where *i* is the triple index in the antecedent based on syntax from [35]):

(?x rdf:type pimo:Person) $\rightarrow$ *remove(0)*, *(?x rdf:type foaf:Person)*

In certain cases however concepts are mapped together using *skos:related* thereby indicating an association between the concepts which is weaker than equivalence. Transforming between these concepts may result in information loss. For instance consider the example in Fig 5 where an RDF model contains an instance of *vcard:VCard* describing a person's contact details using the VCard ontology. We wish to convert the gleaned model to use the FOAF and GeoNames ontologies in order to provide

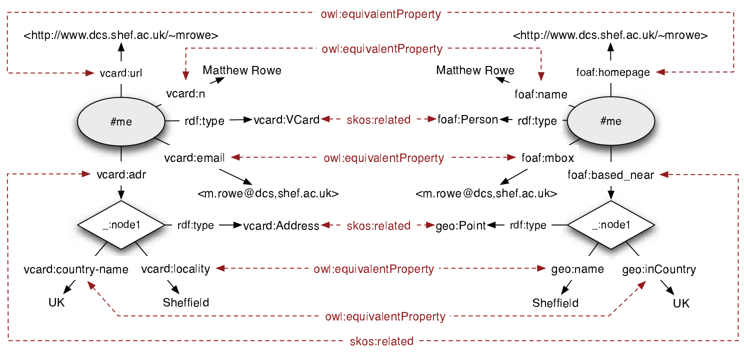


Fig. 5. Transforming an instance within a gleaned RDF model from using the VCard ontology to use the FOAF and GeoNames ontologies

consistent semantics with the Linked Data social graph. To convert the *vcard:VCard* instance we consult the mapping vocabulary (SISM) and find that the concept is mapped to *foaf:Person* using *skos:related*. The model is checked to ensure that a transformation between the concepts would not result in information loss - this is done by checking the predicates used within the instance description and their mappings to the required ontology, a detailed explanation of this procedure falls outside of the scope of this paper. Once the model has been checked and the transformation has been approved then the a rule can be applied to alter the concept definition.

An example mapping within SISM using *skos:related* associates the concept *foaf:based_near* with the concept *vcard:adr*. In this instance we have defined the two concepts as being similar but not completely equivalent. Therefore prior to applying a generated rule from this mapping we verify that no information is lost when transforming the RDF model. The generated rule from this mapping is as follows:

(?x vcard:adr ?geo) → *remove(0), (?x foaf:based_near ?geo)*

From applying the rules we are able to transform any RDF model containing digital identity information to use the FOAF and GeoNames ontologies. This allows instances within the models to be normalised to contain the same semantics as the seed data, thus aiding information interpretation in the following disambiguation techniques

7. Identity Disambiguation

At this stage in our approach we have gathered seed data in the form of a Linked Data social graph describing a given person and a set of web resources, each represented as an RDF model. Our disambiguation task is a binary classification problem (i.e. Does this web page contain information about person X or not?), therefore we must use the seed data to partition the set of web resources into *positive* and *negative* sets thereby disambiguating *identity web references*. Two disambiguation techniques are now presented.

7.1. *Disambiguating using a Semi-supervised Machine Learning Technique*

Semi-supervised machine learning techniques utilise training sets to initially construct a naive classifier, and then support the learning process of retraining the classifier. Several techniques exist within the literature including expectation-maximization [34], which re-estimates parameters of a classification model and then tunes the model based on these expected parameters, co-training [5], which trains multiple classifiers simultaneously and updates the training sets of each classifier according to a separate classifier's predictions, and self-training [18] which learns an initial classifier from a limited set of seed data and iteratively retrains the classifier using the strongest classification instances. Given our limited amount of seed data, only a single Linked Data social graph, our first disambiguation technique employs a self-training strategy to iteratively learn and improve a machine

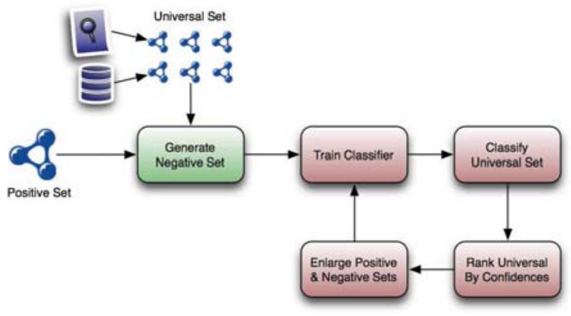


Fig. 6. Self-training a machine learning classifier to disambiguate *identity web references*

learning classifier.

By using self-training we have been able to investigate the effects in disambiguation accuracy based on two variations: a) Different machine learning classifiers, and b) Different feature similarity measures. Machine learning classifiers must be given labelled instances from which they are able to learn. These instances are represented as a binary feature vector (i.e. $\overrightarrow{x}$), where the classifier learns weights for each feature. By varying the feature similarity measure and the classifier we are able to observe the effects in performance of different permutations of classifier and feature similarity measure. Our self-training technique is shown in Fig. 6 and consists of two steps: the first step gathers binary seed data, both *positive* and *negative* to be used for learning a classifier, given that the initial seed data is only *positive*.

The second step of the approach uses the *positive* and *negative* sets as training data. Once the classifier has been trained, we apply the classifier to the set of web resources (we refer to this as the *universal* set) in order to derive labels. We then enlarge the seed data with the labelled instances exhibiting the strongest confidence scores. We repeat this process until the *universal* set is empty by retraining the classifier using the new seed data, classifying instances from the *universal* set and enlarging the training sets. As Fig. 7 shows, the resulting *positive* and *negative* sets are enlarged with all instances from the *universal* set, thereby generating classifications. We deal with three data sets within this technique the *positive* set, the *negative* set and the *universal* set, which we will now refer to as P, N and U respectively.

7.1.1. *Generating Binary Seed Data*

A learned classifier must be able to label a web resource as citing a given person or not. Therefore

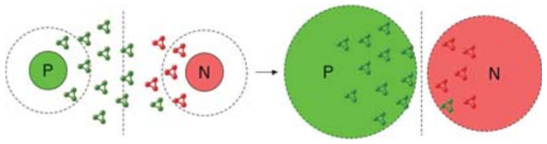


Fig. 7. Initial training sets iteratively enlarged using resources with the strongest confidence scores

we must populate N to accompany data within P. We initially populate P with the Linked Data social graph. Using the semantics of information within this graph, any resources related to the *foaf:Person* instance of the graph owner using *foaf:homepage*, *foaf:workplaceHomepage* or *foaf:weblog* are assumed to contain personal information describing the person. Therefore we generate an RDF model from the linked resource and add to this to P, thereby adding additional instances to P from which a classifier can learn.

We add instances to N using a similar strategy to the technique from [51] by compiling a list of positive features from instances in P. All instances from U are then filtered out that contain any positive features from the list. The intuition behind this technique, according to [51], is that the resulting instances will contain no positive features and are therefore more likely to be negative. Due to the relatively small size of our positive data this strategy initially resulted in false negatives being added to N, additionally the relatively size of our positive training data meant that statistical distribution methods such as those described in [18] were not feasible. Instead we place constraints on the size of the negative set by forcing its size to be approximately the same as the positive set, both in the number of instances and the number of features within those instances. The intuition being that the hypothesis of the trained classifier will correct itself and improve over time thereby overcoming the bias of false negatives (we observe this empirically when evaluating this method). If N is too large initially then this could unfairly bias the initial classifier from which it will never recover.

7.1.2. *Learning to Classify Identity Web References*
Instances are supplied to machine learning classifiers as RDF models, the classifier uses features from those instances to construct a classification model. Therefore we now explain how features are extracted from the RDF models, what different feature similarity measures are used, and how we map the RDF models into a machine learning dataset:

7.1.2.1. *Feature Extraction* We use seed data extracted from Web 2.0 platforms, given that such data contains social network information, under the intuition that a person will appear within a web resource with individuals that he/she knows. Therefore we use RDF instances as features from the RDF models within P, N and U. We generate these features by extracting subgraphs from the RDF model using a recursive algorithm called "*Concise Bounded Description*" [19]. This algorithm functions by "*extracting a subgraph around an RDF resource*", thus returning the immediate properties of a resource as its description and therefore the feature to use. This not only produces instances of *foaf:Person*, but also other types of instances that are present within the RDF models (e.g. *geo:Feature*).

7.1.2.2. *Feature Similarity Measures* As previously mentioned, we are able to vary the feature similarity measure used by a machine learning classifier. Features are represented by *foaf:Person* instances within RDF models of web resources. This permits the variation of the feature similarity measures the following three RDF graph comparison techniques. It worth noting that we use the Levenshtein string similarity metric when literals are compared - to overcome misspellings - as part of the following RDF graph comparison techniques.

Jaccard Similarity: The Jaccard similarity of graphs as described in [7] looks at how similar two graphs are based on the number of edits required to transform one graph into another. Jaccard similarity requires the learning of a given threshold, which when exceeded classes two graphs as equivalent, thus declaring two RDF instances as representing the same person. The measure is defined as follows: let V_g denote the vertices in graph g corresponding to resources and literals in the RDF instance description, and E_g denote the set of edges in graph g corresponding to properties and relations. Graphs g and h are equivalent if $jaccsim(g,h) > \sigma$ where the Jaccard similarity is defined as:

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

Inverse Functional Property Matching One of the key advantages of modelling information semantically is the expressivity of certain properties and relations, such as the use of inverse functional properties. According to the FOAF ontology, properties such an *foaf:mbox* and *foaf:homepage* are typed as *owl:InverseFunctionalProperty* and depict a one-to-one relation between a *foaf:Person* instance and the attributed resource. Therefore we use a similar technique to our graph reasoning method when interlinking social graphs by defining *foaf:Person* instances as being equivalent when inverse functional properties match from both instances, this is also known as "*Data Smushing*" [42].

RDF Entailment It is common for RDF instances to vary size depending on available knowledge. Imagine there are two RDF instances g and h describing the same person: g contains geographical information along with the person's name, email address and web site, whereas h contains the same data without geographical information. One can infer that both g and h refer to the same person, and therefore the tripleset of h is a subset of the tripleset of g. According to Pat Hayes' Interpolation Lemma[32] and work in [30], one can say that h entails g ($h \models g$) if every model of h is also a model of g. Therefore we define two RDF graphs as equivalent if one graph entails the other.

7.1.2.3. *Mapping RDF models into a Machine Learning Dataset* In order to use RDF models as instances from which machine classifiers can learn we must represent these models as binary feature vectors where each feature represents a distinct instance of *foaf:Person*. Therefore we map the RDF model for each web resource into a dataset for learning using the following three steps:

(i) One RDF subgraph (instance) per person is extracted from the RDF model of the web

[32] http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#entail

resource using the previously described technique.

(ii) Each extracted RDF subgraph is flattened, representing it as a set of RDF triples, to enable comparison using the feature similarity measures. The metrics work at the statement level by comparing the triplesets of each subgraph and deriving a match.

(iii) Machine learning instances are constructed as binary features vectors using a feature indexer (i.e. $\overrightarrow{x_i}$ where the index i denotes a given feature). To compile a new feature vector representation of a machine learning instance, indexes are derived for the features of the given web resource by comparing those features with existing features within the indexer. If a feature is matched then the relevant index is returned, otherwise a new index is created. Given that the features are represented as RDF subgraphs, the feature matching technique can be varied thereby effecting the indexes within the feature vectors and the resultant dataset.

By modelling dataset instances as RDF models, and instance features as *foaf:Person* instance descriptions in the RDF model we allow for the variation of feature similarity measure. Within this disambiguation technique we compare three machine learning classifiers; Naive Bayes [34], Support Vector Machines [15] and Perceptron [4]. Naive Bayes derives the conditional probabilities from labelled instances and their features to build the initial generative model. SVM and Perceptron construct their discriminative hyperplanes within the feature space via optimization based on the labelled instances within the dataset. By mapping the RDF models of each web resource into such a dataset, the internal functionality of the machine learning classifiers is not altered. Instead we use the classifiers merely as a "blackbox" by inputting the datasets for training and testing and returning the classification labels for instances within U along with the classification confidences of those labels.

7.1.2.4. *Self-training* The self-training strategy (as Algorithm 1 explains) begins by first using the seed data to train a classifier, at this stage the size of the sets are relatively small. Instances from U are classified using the initially trained classifier and the instances are ranked based on their classification confidence. The top k positive instances and the top k negative instances from U are chosen according to rankings, and P and N are enlarged with those instances respectively. The strongest k positives and strongest k negatives are also removed from U, thereby decreasing the size of the universal set. We repeat this process by retraining the classifier and reapplying it to the universal set until the universal set becomes empty. The intuition behind this strategy is that the trained classifier will improve over time as the hypothesis of the classifier gradually moves to a stable state and the classification confidence of instances begin to converge.

Algorithm 1 ItEnl(P,N,U) : Iterative self-training based on enlargement of training data according to classification rankings. P is the positive set, N is the negative set and U is the universal set

```
Input: $P$, $N$ and $U$
Output: $P$ and $N$
 1: while $U \neq \emptyset$ do
 2:     Train $\Psi$ using $P$ and $N$
 3:     Classify $U$ using $\Psi$
 4:     Rank $U$ based on classification confidences
 5:     Enlarge training sets by strongest $k$ positive ($U_k^P$) and
        negative ($U_k^N$) instances
 6:         $P = P \cup U_k^P$
 7:         $U = U - U_k^P$
 8:         $N = N \cup U_k^N$
 9:         $U = U - U_k^N$
10: end while
11: return  $P$ and $N$
```

According to [46] and [49] ranking instances based on their classification confidences and then enlarging the relevant training sets with the top k instances yields tradeoffs between resulting accuracy and optimized performance. Work in [49] states that lowering k achieves high accuracy whereas increasing k speeds up the process. Therefore we must set k to a suitable value given the domain of application. We explain within the experimental setup our specified value and the decisions behind this.

7.2. *Disambiguating using a Graph-Based Technique*

State of the art graph-based disambiguation techniques compile a graph space from instances to be disambiguated and their features, and then exploit the graph space to support the disambiguation process. For instance [10] addresses the general problem of entity disambiguation by constructing the graph space from declarative links between entities extracted from the Linked Data web, and employ

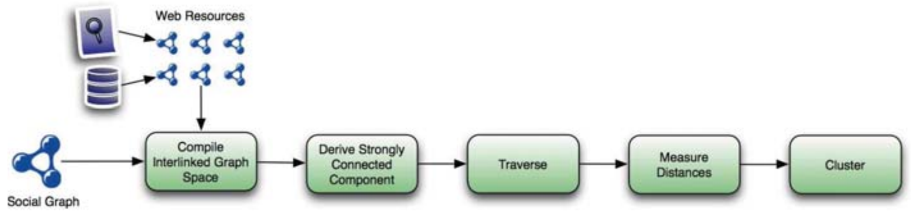

Web Resources
Social Graph
Compile Interlinked Graph Space
Derive Strongly Connected Component
Traverse
Measure Distances
Cluster

Fig. 8. Using Random Walks to disambiguate *identity web references*

the links to infer relationships between entities using a probabilistic model. [2] uses a partially labelled graph space as input to a machine learning classifier, which then predicts labels for the unlabelled nodes. *Relaxation labeling* is then used to correct labelled nodes within the graph space based on the labels of their neighbouring nodes. [25] constructs a graph space from various features (named entities, hyperlinks, and lexical tokens) from a given set of web pages and performs Random Walks through the graph space, measuring distances between pages and clustering those that are closest together.

Our second disambiguation technique uses a similar Random Walks approach to [25]. This technique is well suited to our problem setting given that the seed data and the set of web resources to be disambiguate are modelled using RDF which provides a graph structure consisting of interlinked triples. Our graph-based disambiguation technique is divided into five steps (see Fig. 8). We first construct an interlinked graph space from the seed data and the set of web resources. The second step derives strongly connected components within the graph space. The third step assigns weights to edges and constructs a random walks model to traverse the graph space. The fourth step measures distances within the graph space, and the fifth step clusters the web resources according to those distances. Clustering aims to partition the web resources into two groups: *positive* and *negative*. We now explain each of these steps in greater detail.

7.2.1. *Building an Interlinked Graph Space*

We construct the graph space by initially adding the seed data (the Linked Data social graph), and iteratively with the RDF model of each web resource to be disambiguated using an indexing paradigm. Nodes are represented in the graph space by resources and literals from the RDF models and edges are represented in the graph by properties and relations. The indexing paradigm functions by comparing the resources and literals from an RDF model which are yet to be integrated into the graph space with all the resources and literals currently in the graph space. If a match occurs, for example if an RDF model contains a person's name which already resides in the graph space, then the target node of the edge in the RDF model is replaced with the matched node in the graph space, thereby providing a link. It is common for literals to contain misspellings therefore the Levenshtein string similarity metric is used. When comparing literals, if the derived measure is above an assigned threshold then a match is confirmed and the RDF model is integrated into the graph space.

Resources within the RDF models, denoted by URIs, are compared for strict equivalent (e.g. geographical locations with a URI). Each of the web resource models and the social graph have a document namespace URI which we refer to as the *web resource root node* and the *social graph root node* respectively (as shown in Fig. 9). This strategy compiles a graph space which is rich in features thereby allowing web resources to be linked together by many common literals and resources, this provides a multitude of paths through which traversals can be made.

7.2.2. *Identifying Strongly Connected Components*

It is possible for a compiled graph space to contain islands of connected nodes called strongly connected components. These islands are created due to the RDF models integrated into the graph space only having common literals and resources with a limited number of other web resource models. This inhibits the ability of a walker from traversing across the full diameter of the graph space in that the walker cannot *island-hop*, therefore individual strongly connected components must be identified. We identify these components using Tarjan's algorithm [45] by starting at the *social graph root node* within the graph space and performing a depth first search, returning all connected nodes and edges from the starting node and therefore the strongly connected component which the *social graph root node* resides in.

By only considering this component for traversal information is removed that is irrelevant to the person whose personal information we wish to find. For instance, the social graph contains seed data describing literals and resources which web resources should also contain if they are correct *identity web references*. Even if certain web resources do not exhibit any of those features, they may share features with other web resources which in turn share features with the social graph (thereby providing transitive linking). Therefore, we assume that any resource which is not linked to this island of nodes does not contain information we wish to find.

7.2.3. *Traversing the Graph Space*

Once we have the strongly connected component we traverse this graph using Random Walks [31], moving from a given node to a random adjacent node, and then randomly to any adjacent node of that node, thus forming a stochastic transition process. Edges must first be weighted within the graph space to effect the transition probabilities of the random walker.

7.2.3.1. *Weighting Edges*

We apply a weighting strategy similar to the technique presented in [32]: given that the edges denote properties and relations from an ontology, we able to weight edges which denote unique links as carrying *more* weight. These are edges that represent concepts typed as *owl:inverseFunctionalProperty* or *owl:functionalProperty* (e.g. *foaf:mbox*, *foaf:homepage*, and *foaf:weblog*). We denote the set of edges within the graph as E and the set of nodes as N. The set of edge labels is denoted as $L = \{l_1, l_2, \cdots, l_n\}$ where l_i denotes an edge label corresponding to a distinct concept from an ontology. Therefore a weighting function is defined that returns the weight associated with a given edge (and therefore a distinct ontology concept) $w(l_i)$. The weights are distributed using the same strategy from [25] as follows:

$$\sum_{l_i \in L} w(l_i) = 1.$$

7.2.3.2. *Random Walks*

We use the Random walks model to return the probability of moving from one node in the graph space to another using a given number of steps. Essentially this walk forms a stochastic process and is defined according to a first-order Markov chain [33] where future states are independent of previous states. To compute these transition probabilities we initially create a local similarity matrix known as the adjacency matrix A. A contains the similarity between nodes i and j and is computed by normalising the weight of the edge between i and j with how many edges of distinct types leave i (the *degree* of node i). A is formally defined according to work by [38] as follows:

$$a_{ij} = \begin{cases} \sum_{l_k \in L} \frac{w(l_k)}{|(i,.) \in E : l(i,.) = l_k|}, & (i,j) \in E \\ 0, & otherwise \end{cases}$$

We then construct the diagonal degree matrix as:

$$D = \sum_k A_{ik}$$

Using the Markov chain model of a random walk we wish to derive the probability of moving from node i to node j in t steps. Therefore we create a transition probability matrix using $P = D^{-1}A$ for one step, and $P^t = (D^{-1}A)^t$ for t steps.

7.2.4. *Measuring distances within the Graph Space*

7.2.4.1. *Commute Time*

We wish to measure the distances between root nodes within the graph space so that we may then cluster nodes which are closer together with one another. One such distance measure is Average First-Passage Time [38] which calculates the number of steps taken when performing a random walk through the graph space from one node to another, in essence this returns the hitting time for the traversal:

$$m(k|i) = \begin{cases} 1 + \sum_{j=1, j \neq k}^{n} p_{ij} m(k|j), & i \neq k \\ 0 \end{cases}$$

This measure only returns the number of steps taken from leaving a given node and reaching another. Therefore by calculating the Commute Time it is possible to derive the round trip cost of leaving a given node, reaching another node and then returning again to the previous node:

$$n(i|k) = m(k|i) + m(i|k)$$

Both $m(k|i)$ and $m(i|k)$ will return different values (given that P^t is not symmetric), therefore by summing their traversal costs a symmetric matrix is produced for the commute time distances N where $n_{ij} = n_{ji}$. When deriving the average commute

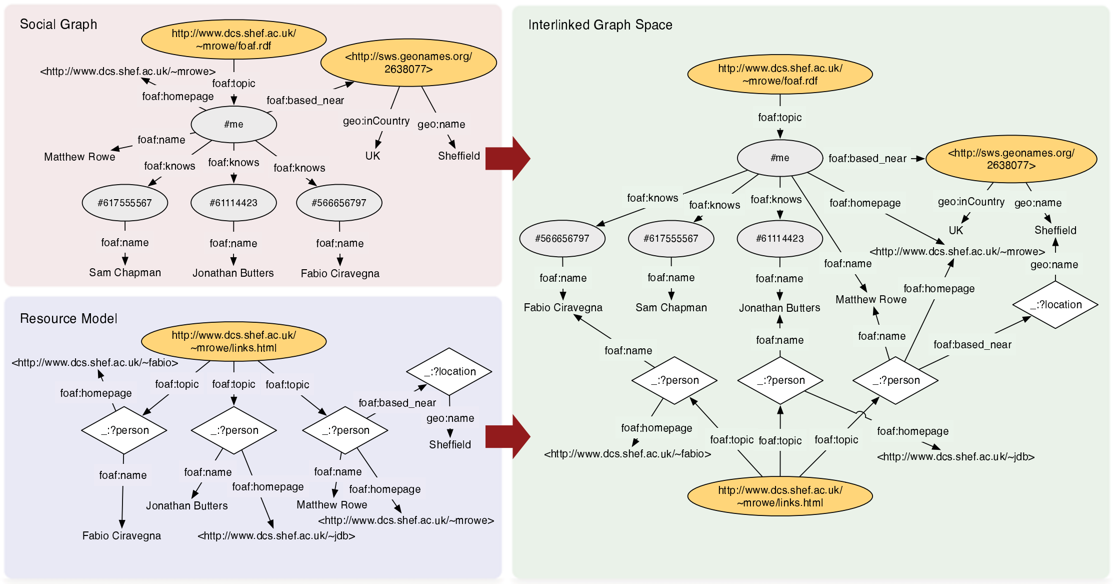


Fig. 9. Snippet of a compiled graph space. Anonymous nodes are represented using blank node notations *_:?person*. A complete graph space would be too large to display graphically in a legible manner.

time we must set the random walks model to a suitable number of steps and therefore set the size of t, (P^t). If t is too high then the random walker covers a larger portion of the graph, reducing the distinction between locally connected nodes. By setting t to a lower value, the random walker focuses on moving along shorter paths, preserving the clarity of locally connected nodes. As the number of paths between nodes increases, where these paths are short in length, then the commute time decreases [36]. Therefore in order to apply commute time effectively as a clustering distance measure we set $t = 2$ thereby focussing on short paths within the graph space.

7.2.4.2. *Optimum Transitions* The variation in the number of steps that the random walker takes (denoted by t) can be exploited to measure the transition probabilities. In [32] walks are made through the graph space until a desired node is reached or when t exceeds 50 and [9] alters the size of t to effect information retrieval performance. It is possible to measure distances between nodes within the graph space by deriving the optimum number of steps needed to traverse from a given node to another node. We find the optimum number of steps between two nodes by incrementing t until the transition probability has peaked for the first time. Therefore we iteratively increase the matrix power of the transition probability matrix P^t until the probability of moving from node i to node j has peaked (as shown in Algorithm 2) and store the number of steps within the matrix O.

Algorithm 2 OptTrans(P) : Derives the Optimum Transitions matrix O from incrementing the matrix power of P

```
Input: $P$
Output: $O$
 1: $O_{|P|} = 0$ {Initalise the Optimum Transitions Matrix as
    the zero matrix}
 2: for $i = 0$ to $|P|$ do
 3:   for $j = 0$ to $|P|$ do
 4:     for $t = 1$ to $\infty$ do
 5:       if $(p^t_{ij} \geq p^{t-1}_{ij})$ and $(i \neq j)$ then
 6:         $o_{ij} = t$
 7:       else
 8:         break
 9:       end if
10:     end for
11:   end for
12: end for
13: return $O$
```

7.2.5. *Clustering Web Resources*

Using the above distance measures we use pairwise agglomerative clustering [50] to produce the two clusters (*positive* and *negative*). Our clustering strategy begins by placing every *web resource root* *node* and the *social graph root node* within the graph space in an individual cluster. Clusters are then compared to measure their average distance. If this is below a given threshold then the clusters are merged together. We compare and merge clusters until no more clusters can be merged; at this point we select the cluster containing the social graph as the *positive* cluster. Clusters are compared as follows:

$$average\_dist(c_1, c_2) = \frac{\sum_{r_i \in c_1} \sum_{r_j \in c_2} dist(r_i, r_j)}{|c_1||c_2|}$$

Where $dist(r_i, r_j)$ is defined for Commute Time Distance as:

$$dist(r_i, r_j) = \frac{n_{r_i r_j}}{max(N)}$$

And for Optimum Transitions as:

$$dist(r_i, r_j) = \frac{o_{r_i r_j}}{max(O)}$$

In order for two clusters to be merged together $average\_dist(c_1, c_2)$ must be less than a given threshold σ where $0 \leq \sigma \leq 1$.The value of the threshold is proportional to the maximum commute time from N, when clustering according to commute time distances, and proportional to the maximum number of transitions from O, when clustering using optimum transitions. We assume that the strongly connected component containing the *social graph root node* will also contain web resources which do not cite the person we are interested in, given that such resources may contain a low number of features in common with the social graph. This clustering strategy therefore dictates that a value for σ must be derived by tuning on known data, we discuss this in the following section.

8. Evaluation

In order to assess the performance of each of the presented disambiguation techniques a thorough evaluation was conducted. This section describes the experimental setup followed by the results from the evaluation.

8.1. *Experiment Design*

8.1.1. *Evaluation Measures*

We are concerned with assessing each of our presented disambiguation techniques' ability to disambiguate web resources which contain personal information about a given person. Therefore we use the information retrieval metrics; recall, precision and f-measure [47]. A denotes the set of web resources which contain personal information and B denotes the set of retrieved web resources therefore $precision = |A \cap B|/|A|$ and $recall = |A \cap B|/|B|$. F-measure provides the harmonic mean of both precision and recall as follows:

$$f\text{-}measure = \frac{2 \times precision \times recall}{precision + recall}$$

8.1.2. *Dataset*

The dataset was compiled using 50 members of the Semantic Web and Web 2.0 communities as participants. For each participant seed data was produced using our previously described method in the form of a Linked Data social graph. A set of web resources to be disambiguated were gathered using our approach by querying the World Wide Web and the Semantic Web. The first 100 results were gathered from each query and cached for the experiments. As expected there were many duplicate web resources returned from each search engine (i.e. equivalent publication and homepages) and several web resources were also noise (i.e. Web resources which require secure access) and were removed. Therefore the resulting dataset contained approximately 17300 web resources with 346 web resources to be disambiguated for each participant.

We then created a gold standard by manually labelling each of the web resources as either citing the given person or not. It is worth noting that the dataset contains participants who have a varying level of web presence. Each participant's web presence level is measured as the proportion of web resources within the total set that are *identity web references*, this is given as a percentage. For instance if participant A appears in 50 of the 350 web resources within their total set, then the web presence level of that participant is 14%. This web presence level allows each of the disambiguation technique to be assessed based on its ability to handle and process varying amounts of information.

8.1.3. *Selection of Baseline Measures*

In order to provide contextual assessment of each of the presented disambiguation techniques three baseline measures have been used:

8.1.3.1. *Baseline 1: Person Name Occurrence* Our first baseline technique utilises the presence of the participant's name within a web resource as being a positive indicator of the presence of personal information thereby yielding low levels of precision but high recall levels. It offers a useful baseline against which our approaches can be compared and highlights the need for background knowledge for each person.

8.1.3.2. *Baseline 2: Hierarchical Clustering using Person Names* Our second baseline technique uses latent semantic analysis via hierarchical clustering [32]. This is our baseline unsupervised approach which does not use seed data, instead the approach relies on a feature comparison technique where web resources are clustered based on the presence of person names within resources. This method produces k clusters where each cluster contains web resources containing personal information of a given person. We then select the clusters which is relevant to our participant for evaluation.

8.1.3.3. *Baseline 3: Human Processing* Our third baseline measure compares the presented disambiguation techniques against human processing of the same data. To provide this baseline measure a group of 15 people, denoted as raters, were set the task of manually processing the dataset. Three different raters disambiguate web resources for each evaluation participant, thereby providing three distinct sets of results. We then used interrater agreement to find the agreement of results between the raters [24] by using one rater's results as the gold standard and another rater's results as the test subject. We then compute the average of the three separate rater agreement measures for each participant.

8.1.4. *Experimental Setup*

8.1.4.1. *Self-training* We assess the performance of nine distinct permutations of machine learning classifier and feature similarity measure, (given that we employ three classifiers; Naive Bayes, Perceptron and SVM and three measures; Jaccard similarity, Inverse Functional Property matching and RDF entailment). For each of the 50 participants we test nine permutations, resulting in 450 experiments to be run for this disambiguation technique. We set self-training to enlarge the training data with the top k strongest classified instances where $k = 1$. We set k at the lowest value possible under the assumption that accuracy of the derived classifications is paramount, and therefore the strongest example should be selected to enlarge the training sets with. We set the similarity threshold for Jaccard similarity to 1 to detect strict graph equivalence when comparing features. We employed the machine learning framework Aleph[33] to perform self-training of the framework's built in classifier algorithms.

8.1.4.2. *Random Walks* Random Walks uses agglomerative clustering to cluster web resources closest to the social graph. The clustering threshold was derived by tuning σ using 10% of the dataset and was set to 0.3 for commute time distances and 0.55 for optimum transitions.

8.2. *Experimental Results*

8.2.1. *Self-Training*

Table 1 presents the results derived from evaluating self-training using different permutations of machine learning classifier and feature similarity measure. The results indicate that the most precise similarity measure used by each classifier was Jaccard similarity. However, this measure also yielded the poorest scores for recall, largely due to its strict behaviour when detecting a match (by not allowing for slight variations between comparable features). The results demonstrate that RDF entailment performs consistently well when combined with different classifiers. We believe that this is due to the large variance in the level of personal information displayed on web pages. For example, there were several cases where a person would display their name and web address with their e-mail encoded to restrict spam from screen-scrapers. The same person would then appear on another web page with the same information yet without the encoded e-mail address. RDF entailment is able to derive equivalence between the instances, thereby detecting a feature match.

The combination of SVM and RDF entailment yielded the highest f-measure score. As expected, SVM outperforms Perceptron for all similarity measures studied, given that Perceptron induces a single vector linear model for explaining the data, whereas SVM uses several support vectors leading to a better optimization of the induced model. The results present the most precise combination as the use of Perceptron with Jaccard similarity. The use of a single vector to classify instances within the feature space offers a precise approach and coupling this with the Jaccard similarity achieves a high level of

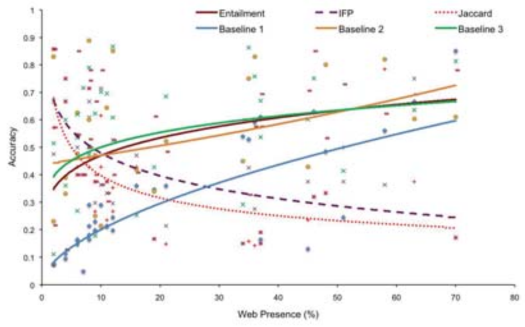
[33] http://aleph-ml.sourceforge.net/

Table 1
Accuracy of Different Permutations of Machine Learning Classifier and Feature Similarity Measure with respect to Baseline Techniques

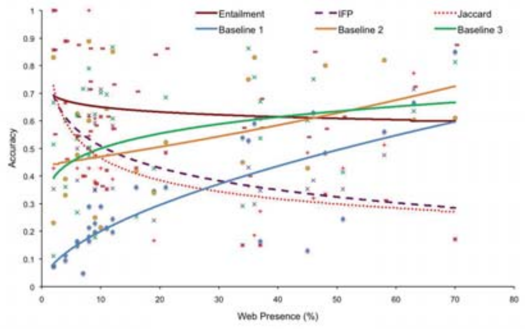
		Precision	Recall	F-Measure
Perceptron	Entailment	0.645	0.636	0.540
	IFP	0.865	0.364	0.512
	Jaccard	0.935	0.318	0.474
SVM	Entailment	0.733	0.670	0.642
	IFP	0.811	0.412	0.546
	Jaccard	0.912	0.384	0.540
Naive Bayes	Entailment	0.684	0.665	0.606
	IFP	0.786	0.418	0.545
	Jaccard	0.906	0.389	0.545
	Baseline 1	0.191	0.998	0.294
	Baseline 2	0.648	0.592	0.556
	Baseline 3	0.765	0.725	0.719

precision. Overall the results indicate that the use of self-training is a very precise disambiguation technique outperforming all of the baselines in terms of precision, however its precise nature leads to poor levels of recall. In terms of f-measure permutations of a given classifier with Entailment outperform the baseline unsupervised technique, but do not perform as well as human processing. To provide a more in depth analysis of this technique we now discuss the performance of individual classifiers.

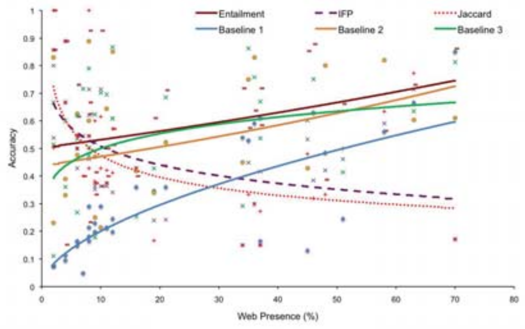
8.2.1.1. *Perceptron* In order to analyse the relationship between the performance of a given disambiguation technique and web presence levels, we use best fitting lines of regression (as shown in Fig. 10). We select the regression model that maximises the coefficient of determination [12] within the statistical model of the data, this can either explain a linear or nonlinear relationship between the accuracy of the disambiguation technique and web presence levels. As Fig. 10(a) indicates Perceptron combined with Jaccard and IFP performs well for participants with low web presence levels. At such levels these permutations outperform the baseline techniques and Perceptron with Entailment. As web presence levels increase Perceptron with Entailment improves in accuracy whereas the two alternative permutations involving Perceptron actually reduce in accuracy resulting in poorer performance than the baseline techniques.



(a) Perceptron



(b) SVM



(c) Naive Bayes

Fig. 10. Accuracy in terms of f-measure of self-trained machine learning classifiers with respect to web presence levels

8.2.1.2. *SVM* Like Perceptron permutations, SVM combined with all similarity measures performs well for low web presence levels outperforming the baseline measures (as Fig. 10(b) shows). As web presence increases however accuracy levels of these permutations reduces. Only SVM combined with Entailment indicates a less drastic reduction in performance with only a slight reduction as web presence increases with the regression curve tending towards a linear model.

8.2.1.3. *Naive Bayes* Combining precision and recall into f-measure values (see Fig. 10(c)) indicates that Naive Bayes with Entailment improves as the web presence level increases and is therefore more suited for participants with higher levels of web presence, while Naive Bayes and IFP and Naive Bayes and Jaccard are more appropriate permutations for low-levels of web presence. Moreover, this suggests that entailment is a better suited feature similarity measure for individuals with higher web presence levels.

8.2.1.4. *Iterative Improvement of Classifiers* We earlier stipulated that self-training must improve the performance of a classifier in order to overcome the possible bias of false negatives within the seed data. To observe the effects of Web 2.0 data and the improvement in performance when applying self-training to each permutation of classifier and feature similarity measure, we analysed the performance of the classifier at each iteration step. This was performed for each permutation for each participant, thereby providing a detailed insight into the effects of self-training.

The performance of Perceptron with IFP and Jaccard improves through training, while Perceptron with Entailment worsens. This was due to misclassified instances early within the training process leading to an overall reduction in accuracy. The decision boundary created by Perceptron with IFP (as shown in Fig. 11(a) reaches a stable state very early within the training cycle due to the strict nature of instance matching via inverse functional properties. The decision boundaries constructed using Perceptron with Jaccard and Entailment take longer to reach a stable state as the hyperplane fluctuates between positions within the feature space.

SVM experiences large fluctuations prior to establishing a stable hyperplane within the feature space when combined with Jaccard and Entailment. As retraining is performed the margins running parallel to the hyperplane are tweaked due to new support vectors being discovered within the feature space. This variance is similar to the behaviour of combinations of the same feature similarity measure with Perceptron (given that both construct separating hyperplanes via discriminative models). Another similarity with Perceptron is the relatively small accuracy fluctuation when combining SVM with IFP.

As shown in Fig. 11(c) Naive Bayes also experiences early fluctuations in classification accuracy for all permutations with feature similarity measures, however these variations are not as great as those experienced by SVM and Perceptron. Unlike the previous two classifiers, a hypothesis is found relatively early on within the training process for combinations of Naive Bayes with Jaccard and Entailment. Observing the effects of training in Fig. 11(c) indicates that the initial hypothesis compiled by a Naive Bayes classifier is close to the final retrained classifier for all permutations, while the classifier does improve it is not a significant amount.

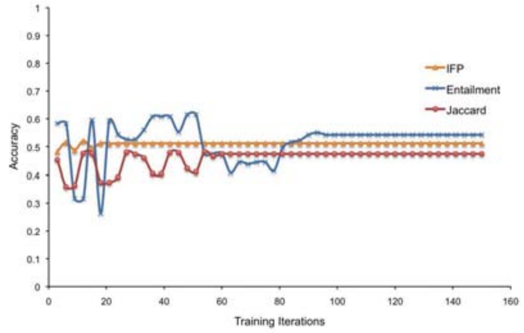
8.2.2. *Random Walks*

Our second disambiguation technique; Random Walks, outperforms two of the baseline techniques in terms of precision and overall f-measure. As the results in Table 2 show clustering using both commute time and optimum transitions produces high levels of recall outperforming human processing. This indicates that a graph-based technique using Web 2.0 data is better suited for finding the maximum number of web resources that contain personal information of a given individual whilst preserving satisfactory levels of precision.

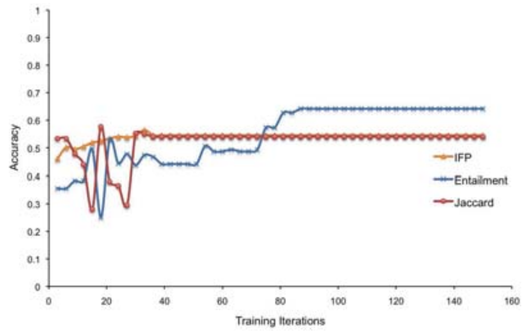
Table 2
Accuracy of Commute Time and Optimum Transitions with respect to Baseline Techniques

	Precision	Recall	F-Measure
Commute Time	0.707	0.798	0.705
Optimum Transitions	0.659	0.805	0.684
Baseline 1	0.191	0.998	0.294
Baseline 2	0.648	0.592	0.556
Baseline 3	0.765	0.725	0.719

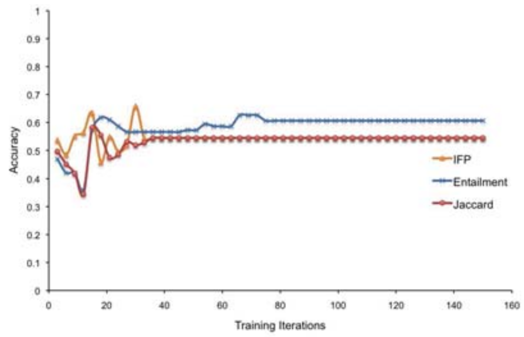
As Fig. 12 demonstrates both commute time and optimum transitions improves in terms of precision and recall as web presence levels increase. Moreover in terms of recall the relationship between accuracy and web presence is linear, thereby indicating a greater increase in performance for optimum transitions. The results also indicate that clustering using commute time is a more precise method than utilising optimum transitions. This could be attributed to the round trip cost utilised by commute time, whereas optimum transitions only considers the number of steps to reach a node within the graph space as a proximity measure. Conversely clustering with optimum transitions returns more *identity web references* to a given person compared to clustering by commute time.



(a) Perceptron



(b) SVM



(c) Naive Bayes

Fig. 11. Effects on f-measure levels when applying self-training to permutations of machine learning classifiers with different feature similarity measures

As stipulated in our earlier requirements the accuracy of automated disambiguation techniques must benefit from the use of seed data. The use of Random Walks as a disambiguation technique indicates that this requirement has been fulfilled. Additionally the results demonstrate that this technique outperforms both an unsupervised technique and human processing in terms of precision, recall and f-measure

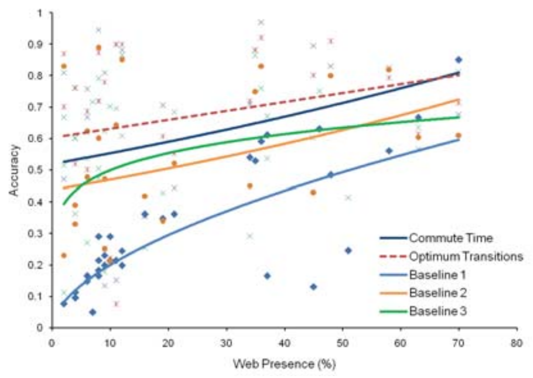


Fig. 12. Accuracy of Random Walks with clustering using Commute Time and Optimum Transitions in terms of f-measure with respect to baseline techniques

9. Conclusions

In this paper we have presented an accurate and high-performance approach to disambiguate *identity web references* through the use of Web 2.0 data. For disambiguation techniques to provide accurate results we claim that seed data supplied to such techniques must be representative of real identity information. Our study of users of the Web 2.0 platform Facebook supports this claim by demonstrating the great extent to which offline relationships are now an intrinsic part of such platforms. Moreover the evaluation of our disambiguation techniques demonstrates that the use of Web 2.0 data provides accurate results.

We have shown that monitoring of personal information online can be performed using automated techniques thereby alleviating the need for human processing. We hypothesised that a semi-supervised approach would outperform a state of the art unsupervised technique and have shown through a detailed evaluation that both Self-training and Random Walks achieve this.

Data semantics plays a crucial role in our approach. It facilitates information integration when reconciling digital identity fragments for use as seed data by attributing semantics to data extracted from Web 2.0 platforms. This provides a common interpretation of what the data is about, thus supporting inferencing. We describe the underlying knowledge structures of web resources by generating metadata models from those resources which contains concepts from web accessible ontologies, thus providing interpretable semantics.

Using semantics supports disambiguation techniques, for instance our use of Self-training models machine learning instances as RDF models and features as RDF instances from the models. We are then able to vary the feature similarity measure used by the classifier between various RDF instance comparison techniques. When using Random Walks we capitalise on the graph structure of RDF to compile a graph space and exploit the graph space to disambiguate *identity web references*. We are able to weight edges within the space based on the semantics of the ontology concept that they represent.

The requirements we stipulated previously have been fulfilled as follows:

(i) *Bootstrap the disambiguation process with minimal supervision:* Our approach only requires seed data at the start of the process. Once this data has been supplied the approach requires no intervention.

(ii) *Cope with web resources which do not contain any features present within the seed data:* We have empirically observed the improvement in performance of the self-trained classifiers thereby demonstrating learning capabilities from minimal seed data. Random walks over an interlinked graph space allowed web resources to be indirectly linked to the social graph, thus enabling the detection of *identity web references* via transitive relations, the effectiveness of this strategy is demonstrated in the resultant high accuracy levels.

(iii) *Outperform unsupervised disambiguation techniques & Achieve disambiguation accuracy comparable to human processing of data:* The evaluation of our disambiguation techniques resulted in accuracy levels which were greater than an unsupervised technique and outperformed human processing in terms of precision when using Self-training and recall when using Random Walks.

(iv) *Disambiguate identity web references for individuals with varying levels of web presence:* The dataset used for evaluation contained individuals exhibiting a range of web presence levels. By evaluating our methods using this dataset we have shown that our disambiguation techniques function for individuals with all web presence levels, which is essential when considering the rise in participation of web users in a virtual environment.

(v) *Produce seed data with minimal cost:* We generate seed data from data residing within Web 2.0 platforms. By leveraging this existing data we are able to produce seed data incurring no production cost.

(vi) *Generate reliable seed data:* Seed data supplied to our disambiguation techniques is reliable given that it has been obtained from Web 2.0 platforms. We have demonstrated that data found on Web 2.0 platforms is representative of real identity and offline social network information, and is therefore representative of a given individual.

The seed data used within our approach is leveraged from only two Web 2.0 platforms. Despite this limited number our results empirically demonstrate how disambiguation techniques perform well when supported by such data. However the used platforms only expose a portion of an individual's digital identity, additional identity fragments could be obtained by leveraging seed data from additional platforms such as the business networking site LinkedIn[34]. Given our presented techniques this is something which we plan to explore in the future, by aggregating data from a range of Web 2.0 platforms and observing the effects on the performance of the presented disambiguation techniques.

The use of data leveraged from Web 2.0 platforms as seed data can limit our approach if the identity which web users construct on such platforms is not consistent with their identity on the WWW. Within the evaluation we observed several cases where this occurred. However in many cases, despite the seed data being minimal - due to the user exposing a limited about of information within their profile - the disambiguation techniques were still able to learn from previously unknown features and achieve accurate results. We found that those individuals who exposed a large amount of identity information on Web 2.0 platforms yielded more accurate results - given the greater feature coverage offered by the seed data. This suggests that paradoxically web users must share more digital identity information on Web 2.0 platforms in order to aid disambiguation techniques, where the privacy of several such platforms is debatable. In the case of Facebook and similar *walled garden* sites identity information is shared under the pretense that it is safe and secure, and given the profile owner's permission, can then be used as seed data.

As a final concluding remark to provide a contextual indication of how at risk our evaluation partici-

[34] http://www.linkedin.com

pants were to identity theft, we analysed the *identity web references* disambiguated using each technique. According to UK government guidelines[35] which stipulates that the visibility of the persons name, their address and email address could lead to identity theft we found that over half of the participants were at risk.

We are currently investigating alternative disambiguation methods, most notably rule induction from RDF instances within the provided seed data. Our future work will compare the performance of precise rule-based disambiguation methods against the presented disambiguation techniques. We also plan to deploy the approach presented within this paper to allow web users to monitor their online presence and compare the performance of the deployed system against several commercial ventures which were discussed within the state of the art of this work - thereby comparing our effort against similar semi-supervised approaches.

10. Acknowledgements

This work was partially supported by X-Media, a project funded by the European Commission as part of the Information Society Technologies (IST) programme under grant number IST-FP6-026978.

References

[1] M. Andrejevic, The discipline of watching: Detection, risk, and lateral surveillance, Critical Studies in Media Communication 23 (5) (2006) 392–407.

[2] R. Angelova, G. Weikum, Graph-based text classification: learn from your neighbors, in: SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, 2006.

[3] R. Bekkerman, A. McCallum, Disambiguating web appearances of people in a social network, in: WWW '05: Proceedings of the 14th international conference on World Wide Web, ACM, New York, NY, USA, 2005.

[4] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[5] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, Morgan Kaufmann Publishers, 1998.

[6] D. Brickley, L. Miller, FOAF vocabulary specification, Tech. rep., FOAF project, published online on May 24th, 2007 at (May 2007).
URL http://xmlns.com/foaf/spec/20070524.html

[35] http://www.getsafeonline.org/nqcontent.cfm?a_id=1132

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

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

[9] N. Craswell, M. Szummer, Random walks on the click graph, in: SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, 2007.

[10] P. Cudré-Mauroux, P. Haghani, M. Jost, K. Aberer, H. D. Meer, idmesh: Graph-based disambiguation of linked data, in: 18th International World Wide Web Conference, 2009.
URL http://www2009.eprints.org/60/

[11] R. Cyganiak, Expressing the xfn microformat in rdf, Tech. rep., Deri (2008).
URL http://vocab.sindice.com/xfn

[12] R. Darlington, Regression and Linear Models, McGraw-Hill, 1990.

[13] A. Doan, Y. Lu, Y. Lee, J. Han, Object matching for information integration: A profiler-based approach, in: In: Proceedings of the IJCAI-03 Workshop on Information Integration on the Web. (2003, 2003.

[14] X. Dong, A. Halevy, J. Madhavan, Reference reconciliation in complex information spaces, in: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, New York, NY, USA, 2005.

[15] S. Dumais, J. Platt, D. Heckerman, M. Sahami, Inductive learning algorithms and representations for text categorization, in: CIKM '98: Proceedings of the seventh international conference on Information and knowledge management, ACM, New York, NY, USA, 1998.

[16] N. B. Ellison, C. Steinfield, C. Lampe, The benefits of facebook friends: Social capital and college students' use of online social network sites, Journal of Computer-Mediated Communication 12 (2007) 1143–1168.

[17] M. B. Fleischman, E. Hovy, Multi-document person name resolution, in: Annual Meeting of the Associtation for Computational Linguistics (ACL), Reference Resolution Workshop, 2004.

[18] G. P. C. Fung, H. Lu, Text classification without negative examples revisit, IEEE Trans. on Knowl. and Data Eng. 18 (1) (2006) 6–20, member-Yu, Jeffrey X. and Fellow-Yu, Philip S.

[19] G. Grimnes, P. Edwards, A. Preece, Instance based clustering of semantic web resources, in: Proceedings of the 5th European Semantic Web Conference, LNCS, Springer-Verlag, 2008.

[20] W. W. Group, Rdfa primer: Bridging the human and data webs (October 2008).
URL http://www.w3.org/TR/xhtml-rdfa-primer/

[21] W. Halb, Y. Raimond, M. Hausenblas, Building linked data for both humans and machines, in: Linked Data on the Web (LDOW2008), 2008.
URL
http://data.semanticweb.org/workshop/LDOW/2008/paper/10

[22] H. Halpin, B. Suda, N. Walsh, An ontology for vcards, Tech. rep., W3C (2007).
URL `http://www.w3.org/2006/vcard/ns`

[23] J. Hart, C. Ridley, F. Taher, C. Sas, A. Dix, Exploring the facebook experience: a new approach to usability, in: NordiCHI '08: Proceedings of the 5th Nordic conference on Human-computer interaction, ACM, New York, NY, USA, 2008.

[24] G. Hripcsak, A. S. Rothschild, Agreement, the f-measure, and reliability in information retrieval, Journal of American Medical Informatics Association 12 (3) (2005) 296–298.

[25] J. Iria, L. Xia, Z. Zhang, Wit: Web people search disambiguation using random walks, in: Proceedings of 4th International Workshop on Semantic Evaluations, Prague, Czech Republic, 2007.

[26] A. Isaac, E. Summers, Skos simple knowledge organization system primer, Tech. rep., W3C (2009).
URL `http://www.w3.org/TR/skos-primer/`

[27] D. V. Kalashnikov, R. Nuray-Turan, S. Mehrotra, Towards breaking the quality curse.: a web-querying approach to web people search., in: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, 2008.

[28] M. Kay, Xsl transformations (xslt) version 2.0 (w3c recommendation 23 january 2007), Tech. rep., W3C (2007).
URL `http://www.w3.org/TR/xslt20/`

[29] R. Khare, Microformats: The next (small) thing on the semantic web?, IEEE Internet Computing 10 (1) (2006) 68–75.

[30] S. Liu, J. Zhang, Retrieving and matching rdf graphs by solving the satisfiability problem, in: WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, IEEE Computer Society, Washington, DC, USA, 2006.

[31] L. Lovasz, Random walks on graphs: A survey (1993).

[32] B. Malin, Unsupervised name disambiguation via social network similarity, in: Workshop on Link Analysis, Counterterrorism, and Security, 2005.

[33] S. P. Meyn, R. L. Tweedie, Markov chains and stochastic stability, Springer–Verlag, 1993.

[34] T. M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.

[35] M. Obitko, V. Marík, Integrating transportation ontologies using semantic web languages, in: Second International Conference on Industrial Applications of Holonic and Multi-Agent Systems, 2005.

[36] H. Qiu, E. R. Hancock, Clustering and embedding using commute times, IEEE Trans. Pattern Analysis. Mach. Intell. 29 (11) (2007) 1873–1890.

[37] M. Rowe, Mapping between digital identity ontologies through sism, in: In Proceedings of the Social Data on the Web Workshop at the International Semantic Web Conference, 2009.

[38] M. Saerens, F. Fouss, L. Yen, P. Dupont, The principal components analysis of a graph, and its relationships to spectral clustering, in: Proceedings of the 15th European Conference on Machine Learning (ECML 2004). Lecture Notes in Artificial Intelligence, Springer-Verlag, 2004.

[39] F. Saïs, N. Pernelle, M.-C. Rousset, Combining a logical and a numerical method for data reconciliation, Journal on Data Semantics XII (2009) 66–94.

[40] L. Sauermann, R. Cyganiak, M. Voelkel, Cool uris for the semantic web, Tech. rep., Deutsches Forschungszentrum fuer Kuenstliche Intelligenz GmbH (2007).
URL
`http://www.w3.org/TR/2007/WD-cooluris-20071217/`

[41] L. Sauermann, L. van Elst, A. Dengel, Pimo - a framework for representing personal information models, in: K. Tochtermann, W. Haas, F. Kappe, A. Scharl, T. Pellegrini, S. Schaffert (eds.), Proceedings of I-MEDIA '07 and I-SEMANTICS '07, 2007.

[42] L. Shi, D. Berrueta, S. Fernandez, L. Polo, S. Ferna?dez, Smushing rdf instances: are alice and bob the same open source developer?, in: ISWC2008 workshop on Personal Identification and Collaborations: Knowledge Mediation and Extraction (PICKME 2008), 2008.

[43] M. Smith, C. Welty, D. McGuinness, OWL web ontology guide. (2004).
URL `http://www.w3.org/TR/owl-guide/`

[44] Y. Song, J. Huang, I. G. Councill, J. Li, C. L. Giles, Efficient topic-based unsupervised name disambiguation, in: JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries, ACM, New York, NY, USA, 2007.

[45] R. Tarjan, Depth-first search and linear graph algorithms, SIAM Journal on Computing 1 (2) (1972) 146–160.

[46] M. Thelen, E. Riloff, A bootstrapping method for learning semantic lexicons using extraction pattern contexts, in: EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Association for Computational Linguistics, Morristown, NJ, USA, 2002.

[47] C. J. van Rijsbergen, Information Retrieval, 2nd ed., Butterworths, London, 1979.

[48] W3C, Gleaning resource descriptions from dialects of languages (GRDDL), W3c recommendation, W3C (September 2007).
URL `http://www.w3.org/TR/grddl/`

[49] G. Wang, Y. Yu, H. Zhu, Pore: Positive-only relation extraction from wikipedia text, in: ISWC/ASWC, 2007.

[50] X. Wan, J. Gao, M. Li, B. Ding, Person resolution in person search results: Webhawk, in: CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management, ACM, New York, NY, USA, 2005.

[51] H. Yu, J. Han, K. C. chuan Chang, Pebl: Web page classification without negative examples, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 70–81.