Soc. Netw. Anal. Min. (2014) 4:229
DOI 10.1007/s13278-014-0229-x

ORIGINAL ARTICLE

In users we trust: towards social user interactions based Trust Assertions for the Social Semantic Web

Owen Sacco · John G. Breslin

Received: 18 July 2013 / Revised: 19 February 2014 / Accepted: 1 August 2014 / Published online: 19 August 2014

Abstract Current approaches for asserting trust focus on propagating *known* trust values amongst peers in a trusted network and do not provide measures for asserting a trust value from user interactions between peers. Moreover, Social Web applications assume that all users connected in one's social graph share the same level of trust. These platforms do not provide any means to capture and differentiate trust levels amongst one's connected peers. This makes it difficult for users to decide whom to trust and they have to rely on limited knowledge—their own or recommended by other peers. In this work, we present a trust framework for automatically asserting subjective trust values for connected users based on social user interactions—actions that provide users to interact with one another within a Social Network. In this work, we describe our model how we assert trust from these interactions. We also present our Trust Manager that automatically asserts user's subjective trust values based on our model. These trust values help users to decide who is more trustworthy and with whom they can share their personal information.

Keywords Trust · Privacy · Social-based trust · Social semantic web

This work is funded by the Science Foundation Ireland [Grant SFI/08/CE/I1380 (Líon 2)] and by an IRCSET scholarship co-funded by Cisco systems.

O. Sacco (✉) · J. G. Breslin
Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland
e-mail: owen.sacco@deri.org

J. G. Breslin
e-mail: john.breslin@nuigalway.ie

1 Introduction

Most online Web users have become dependant on Social Networks where they continuously read and share content with their connected peers (Goel et al. 2012). However, these Social Web applications do not provide any information about the trustworthiness of their peers who is accessing the user's personal information. Moreover, these online social platforms also assume that all connected peers in one's social graph share the same level of trust. In real life, we do not trust everybody in the same way and subconsciously, we continuously make trust judgements about other people.

Thus, it is up to the user to decide who is trustworthy and users have to rely on their intuition to arrive to a trust judgement.

Trust in real life is developed over time whilst people interact with each other. Therefore, this work focuses on automatically asserting trust from online social interactions amongst users within Social Networks.

In our previous work (Sacco and Breslin 2013) we have conducted a user study that analysed which online interactions users utilise the most and whether from these interactions we could infer trust. 178 participated in this user survey and our study shows that we can capture trust from the number of: (1) sharing or tweeting of content from external sources; (2) re-sharing or re-tweeting content within the Social Network platform; (3) "like" or "+1" or "favourite" content within the Social Web application; (4) comments or replies on content within the Social Network; and (5) tags or mentions within the Social Network platform.

Table 1 illustrates the average and standard deviation of the participants perception of trust in the outlined social user interactions within these Social Networks. It can be

Table 1 Average and standard deviation of the participants trust perception

Social user interaction	Average trust perception (%)	Standard deviation (%)
Sharing/tweeting	41.67	10.61
Re-sharing/ retweeting	53.33	14.57
Like/+1/favourite	47	11.53
Comment/replies	39	16.09
Tags/mentions	45.67	14.57

noted that re-sharing or retweeting is considered as the most user interaction type that captures trust. This is followed by liking, +1 or favouring content; tags or mentions and sharing. Comments or replies are the user interaction types that users perceive as the least activity to capture trust. This is because comments or replies might contain content that represents distrust.

These results show that by using these interactions, trust values can be asserted. These trust values denote the user's personal trust in their connected peers. With these values, users can know who they can trust most.

Moreover, our results, as illustrated in Fig. 1, also show that users use Facebook and Twitter the most to interact with other peers using these social interactions.

This paper therefore demonstrates how to assert trust values from these online interactions. We have developed a Trust Manager that extracts information from Social Networks, mainly from Facebook and Twitter, and automatically asserts trust from this information. The information is first transformed in RDF and annotated with well defined concepts. Structuring the extracted social data in standardised formats eases the process for asserting trust. The asserted trust values will then assist users to make better trust judgements.

This paper is structured as follows: Sect. 2 provides an overview of current work on trust. Section 3 defines our meaning of trust and outlines our use case of our work. Section 4 provides a high level overview of the architecture of our trust manager. Section 5 explains in detail what information can be extracted from Facebook and Twitter for each user interaction feature. Moreover, this section also explains in detail how the user's subjective trust values are asserted from this information. Section 6 outlines how we derive a trust value for requesters from the trust values asserted using the social user interactions. In Sect. 7 we explain how we have extended our Trust Assertion Ontology (TAO) to describe the user's subjective trust values for each user interaction which are stored for later retrieval. Section 8 provides some screen shots of our trust manager to illustrate how these trust values are displayed to the user on which he/she can decide whether or not to share his/her personal information. Section 9 provides details of our experiment and evaluation of our system and Sect. 10 concludes the paper.

2 Related work

Most current work on trust in Social Networks focus on users giving rating scores to other peers. The authors in Golbeck

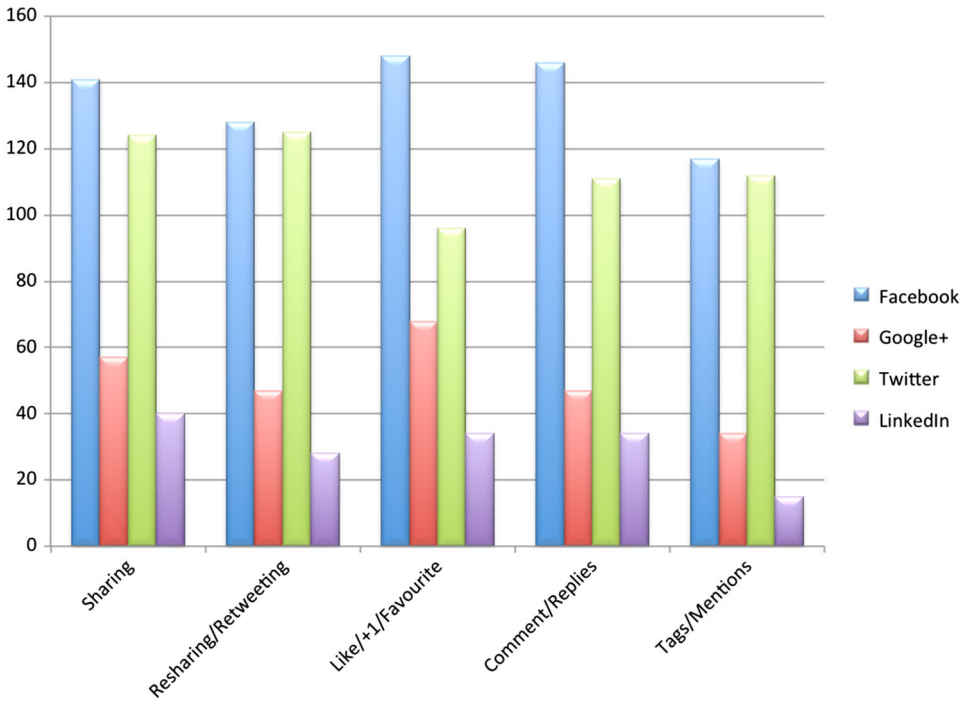

Fig. 1 Overall participant's activity of social user interactions

(2006) focus on recommending films based on trust ratings from Social Networks. Although the authors use social trust, the trust ratings are manually asserted and they focus on a single Social Network. In Golbeck and Hendler (2004) the authors focus on inferring trust and reputation in Social Networks. The trust ratings are assumed to be entered by users which represent trust values about other users to whom they are connected. The authors in Golbeck and Hendler (2006) focus on inferring trust in Social Networks from relationships, however they assume that the user gives a rating to other users they are connected to.

The authors in Golbeck (2009) propose a method for asserting trust amongst users based on profile similarity. The authors examine that users trust other users based on how similar they are with each other. Although they provide beneficial results showing that users trust others who are more similar to them, they do not assert trust on the similarities within profiles. The authors Ziegler and Golbeck (2007) also propose a profile similarity approach whereby they also try to assess similarity based on the trust decisions rather than on profile attributes.

The authors in Golbeck et al. (2003) propose the "Web of Trust" in a Social aspect where users give ratings to each other and based on the links amongst users, a "Web of Trust" is formed. However, similar to other work, they assume that users manually give a trust ranking.

The authors in Kim and Song (2011) propose a method to propagate trust in Social Networks but they also assume that the trust value is provided. Similarly, the authors in Guha et al. (2004) propose algorithms to propagate trust and distrust, however they also assume that the trust value amongst nodes is provided.

The authors in Kim et al. (2008) present a framework to derive a degree of trust for users. Their framework is based on deriving a trust value from user's ratings and user's expertise. However, the framework is not suitable for capturing and deriving trust degrees from social user interactions in Social Networks as proposed in our work. The authors in Liu et al. (2008) also propose a model for predicting trust values. However, their work also focus on using user's ratings unlike our work which focuses on analysing how trust can be captured from user interactions in Social Networks.

The authors in Artz and Gil (2007) provide a comprehensive study that cover many literature on trust. However, most of the work assumes that the users allocate manually their trust values.

3 Motivations

Trust values are personalised and subjective since every user makes trust judgements differently from others. This is because each user has different perceptions of trust based on different past experiences, psychological factors, opinions influenced by other users, other social factors and so forth. In this section we define our meaning of trust and we also outline our use case of our work.

3.1 Defining trust

Most current work on trust differ when defining the meaning of trust. The authors in Mui et al. (2002) define trust as a subjective expectation one has about another's future behaviour based on past encounters. The authors in Grandison and Sloman (2000) define trust as a belief in the user's competence to act within a specified context. However, the authors in Olmedilla et al. (2005) state that trust depends on the actions themselves rather than on the competences.

In our work, trust depends on a person's subjective belief at that point in time s/he is sharing information that another person will act responsibly and will not misuse the information. Moreover, we assume that trust is asymmetric and users do not trust each other in the same way. Therefore, our meaning of trust is defined as:

"Trust of a party A to a party B for personal information X is the measure belief of A in that B behaves dependably for a specified period within a specified context (in relation to personal information X)".

3.2 Scenario

Consider a Social Network where Alice, Bob and John are subscribers and they are all connected with one another as co-workers. John shares with the "co-workers" users list non-personal related information. However, he trusts Alice more than other co-workers since they both have similar interests in common. Bob wants to know John's personal mobile phone number but he does not have access. Although John trusts Alice with this information, even Alice cannot access John's mobile phone number since she is in the "co-workers" user list. Despite John can manually set Alice to view his contact details, John is connected to a large number of users in this Social Network that makes it a tedious task to specify precisely who can access which personal information.

John therefore requires a system that exports and aggregates information from various Social Web applications into a Social Semantic Web platform; consisting of aggregated personal information which is annotated with contextual meaning and formatted in RDF. The system will use this aggregated content to assert trust—such as based on how similar Alice's profile is to John's profile. John would set a threshold value for each of his personal

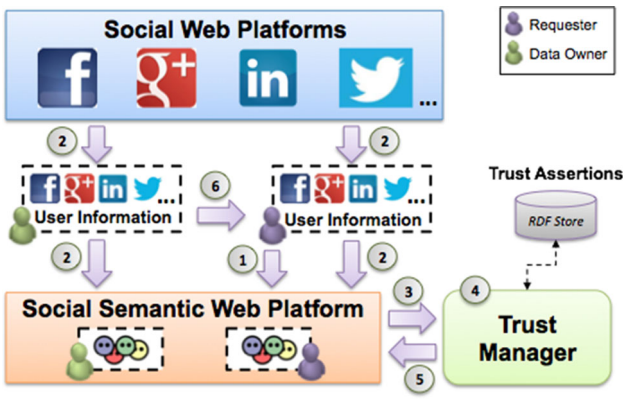

Fig. 2 Asserting user's subjective trust value for a requester

information that one's trust value must satisfy. This trust value is then used to filter information and share it only with trusted peers.

This scenario reflects what our work focuses on—the user wants to assert the subjective trust value of a requester in order to assess whether the requester is trustworthy and will not misuse the user's personal information. The sequence of asserting user's subjective trust values for requesters is illustrated in Fig. 2, and consists of: (1) the requester logs into the Social Semantic Web platform and requests access to a particular user's personal information. (2) The data owner and requester's personal information, together with their interactions are extracted, aggregated, curated and transformed in RDF[1] from the Social Web platforms into the Social Semantic Web platform. (3) A request for the data owner's personal information is sent to the Trust Manager to check whether the requester is trustworthy to be granted access to the information. (4) The Trust Manager asserts the user's subjective trust value for the requester based on the aggregated information of the user and of the requester. The Trust Manager first checks the trust assertions in the trust assertion store and calculates only the subjective trust values which need to be updated. If the subjective trust values are not present in the store, then all the trust values have to be computed. (5) The user's aggregated subjective trust value for the requester is sent back to the Social Semantic Web platform. (6) The requester is granted access only to the information which s/he is trustworthy for.

4 Trust manager architecture

The Trust Manager is a Web application which automatically asserts user's subjective trust values from online Social Web information. As illustrated in Fig. 3, it includes a *Trust Assertion Controller*, *Semantic Web components* and *Trust Assertion components*.

4.1 Trust assertion controller

The *Trust Assertion Controller* (TAC) is responsible for calling and handling the various components within the Trust Manager. Whenever a request is received to assert user's subjective trust values, the TAC calls the *Semantic Web Components* to extract and structure the social data. Once the data is stored in the RDF triple store, the TAC then calls the *Trust Assertion Components* to infer trust from the social semantic data. The user's subjective trust values are stored within the trust assertion store for later retrieval.

Each time the TAC needs to assert user's subjective trust values, the trust assertion store is first queried to check whether trust values already exist. If there are trust values stored for that particular entity which trust is being asserted for, then the TAC checks whether there were any changes to the entity's data. If there were no changes in the data, then the stored trust values are used. If there were any changes to the data, then only the updated information is used to assert a trust value. This asserted value and the stored trust value are then averaged to come up with a new subjective trust value. The user's subjective trust values are then sent to the Social Semantic Web platform to assist the user in his/her trust decision.

4.2 Semantic web components

The *Semantic Web Components* are responsible for handling, structuring and storing the social data. The components include (1) extracting social data; (2) semantic transformation of the data into RDF; (3) semantic matching of concepts and integration of the data; and (4) serializing and storing the RDF data. The RDF social data are stored in an RDF triple store which are used by the *Trust Assertion* components and the Social Semantic Web platform.

4.2.1 Social web data extractor

Social data is extracted from Facebook and from Twitter since our previous results showed that users utilise these platforms the most for interacting with their peers (Sacco and Breslin 2013). The Facebook APIs and the Twitter APIs are used for extracting information from these platforms respectively.

The Facebook Graph API[2] provides endpoints to retrieve Facebook objects from single HTTP calls or joint

[1] If this information is already in RDF, then only the new information is extracted and structured in RDF.

[2] Facebook Graph API—https://developers.facebook.com/docs/reference/api/.

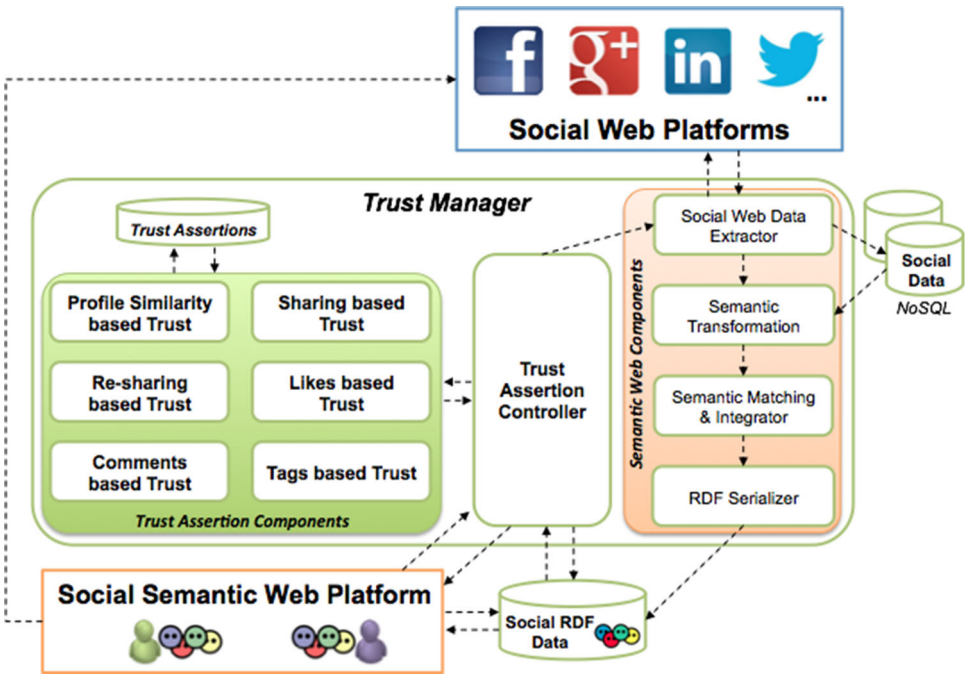

Fig. 3 Trust manager architecture

queries to multiple graphs in a single call. The Twitter Rest API v1.1[3] provides several endpoints that make it easy to retrieve information from Twitter. Section 5 explains in detail what information is extracted from Facebook and Twitter.

4.2.2 Semantic transformation

The extracted content from Facebook and Twitter is then transformed in RDF using various vocabularies such as Friend-of-a-Friend (FOAF)[4] for describing basic personal information, the Relationship Ontology[5] for describing relationship types with other users, the Description-of-a-Career (DOAC)[6] for describing career related information and Semantically Interlinked Online Communities (SIOC)[7] for describing activities within the Social Web platform such as sharing of microblog posts.

Tools such as AlchemyAPI[8] and Zemanta API[9] are used since they provide natural language processing for string matching and also it provides named entity recognition for identifying entities. It also resolves disambiguation of entities by analysing the context of the sentence and annotates entities with URIs from the Linked Open Data Cloud[10].

4.2.3 Semantic matching and integrator

The transformed RDF data from Facebook and Twitter is then integrated and aggregated to provide complete social information including user profiles, social interactions, shared content and so forth from different platforms. When aggregating social information across various systems, it provides more complete information about the user which can be used to assert more fine-grained subjective trust values. However, this poses several challenges when matching equivalent entities together such as when trying to identify a user on Facebook is the same user on Twitter. We therefore use algorithms similar to Shvaiko et al. (2009) and Giunchiglia et al. (2004) to semantically match entities together.

4.2.4 RDF serializer.

Once the information is completely transformed into RDF, it is then stored in a triple store. The ARC2[11] library is used since it provides several parsers, serializers and also provides storing RDF into datastores.

[3] Twitter Rest API—https://dev.twitter.com/docs/api/1.1.
[4] FOAF—http://www.foaf-project.org.
[5] Relationship—http://vocab.org/relationship/.html.
[6] DOAC—http://ramonantonio.net/doac/0.1/.
[7] SIOC—http://sioc-project.org/.
[8] Alchemy—http://www.alchemyapi.com/api/entity/.
[9] Zemanta API—http://www.zemanta.com/.
[10] Linked Open Data Cloud—http://lod-cloud.net/.
[11] ARC2—http://arc.semsol.org/.

4.3 Trust assertion components

The *Trust Assertion Components* are responsible for asserting the user's subjective trust values from the social semantic data. The components include: (1) Profile Similarity based Trust; (2) Sharing based Trust; (3) Re-sharing based Trust; (4) Likes based Trust; (5) Comments based Trust; and (6) Tags based Trust. These components are explained in more detail in Sect. 5.

5 Social user interactions based Trust

It is the norm that in real life trust is evolved over time based on interactions with other peers. The more one interacts with a person, the more trust is built. On this notion, our work focuses on asserting trust based on the frequency of these user interactions.

When one interacts with a connected peer, the level of trust is not the same when one interacts with another peer. This is because trust is very subjective and it depends on various social factors including the perception of the user has towards that person with whom the user is interacting with. Therefore, the trust that the user has about another person has to be taken into consideration in addition to asserting trust from social interactions. In our work, we take into consideration how similar the user is with another person because other work on trust, such as Golbeck (2009), have shown that "the more similar two people were, the greater the trust between them". Hence, we use profile similarity trust to weight the interactions one has with other users.

The trust values are modelled similar to the trust model in Marsh (1994) and Hartig (2009). Subjective trust values are represented in the range of $[-1,1]$ where the range boundaries define: a subjective trust value of 1 represents absolute trust, -1 represents absolute distrust, and values in between the range define subjective trust values of trust or distrust. The subjective trust value 0 represents either uncertainty or unknown due to a lack of information that the trust value could not be asserted. Positive values less than 1 still represent trust but it represents that there is an element of uncertainty or unknown information rather than absolute trust. This also applies to negative values that represent distrust.

5.1 Profile similarity

Profile Similarity based trust represents a trust value based on how two users are similar to each other. This value is used to weight the number of interactions one has with other users. This trust value is asymetric—meaning that the trust is not identical for both users since users do not trust each other in the same way.

Profile similarity trust is asserted by the *Profile Similarity based Trust component*. This component uses the FOAF profiles extracted and transformed from the Social Web applications stored in the RDF datastores.

The profile attributes for both the user and the requester that can be extracted from the Facebook and Twitter APIs and mapped to RDF are categorised as *basic information*, *other information* and *knows information*. The basic information is unique for each user and will not be considered when asserting the user's subjective trust value for profile similarity. The other information consists of information that can be found in other user's profiles as well which include the user's interests. The knows information contains the peers with whom the user is connected who can also be connected to other users. Both the other information and the knows information are taken into consideration when asserting the user's subjective trust value for profile similarity.

The *basic information* which can be extracted from Facebook are the following fields: (1) id; (2) full name; (3) first name, middle name and last name; (4) gender; (5) link—the user's Facebook profile URL; (6) username; (7) bio; (8) birthday; (9) cover—the user's cover photo; (10) email; (11) picture—the user's profile picture; (12) relationship status—the user's relationship status; (13) significant other—with whom the user is in a relationship with; and (14) website—the user's personal website.

The *other information* which can be extracted from Facebook are the following fields: (1) languages; (2) education; (3) hometown; (4) quotes—the user's favourite quotes; (5) interested in—the genders the user is interested in; (6) location—the city where the user is currently living; (7) political—the user's political views; (8) religion - the user's religious views; (ix) work—the user's work history; (10) books—the books which the user is interested in; (11) interests—the user's interests; (12) movies—the movies which the user is interested in; (13) music—the music which the user is interested in; (14) television—the television programmes and series which the user is interested in; (15) games—the games which the user is interested in; (16) achievements—the achievements which the user accomplished in games; (17) scores—the current scores which the user achieved in games; (18) activities—the user's activities; (19) events—the events which the user has attended or will attend; (20) locations—the locations which a user has been to; (21) checkins—the places where the user has checked into; (22) notes—the user's notes; (23) questions—the user questions; (24) pokes—the user's pokes; and (25) groups—the groups in which the user is a member of.

The *knows information* which can be extracted from Facebook are the following fields: (1) family—the user's close relatives; (2) friends - the user's connected peers; and (3) friendlists.

Facebook provides a field called `mutual friends` which retrieves those friends who are both friends with one user and with another. Therefore, the `mutual friends` field can be used without having to compare the friends lists of both users.

Twitter does not provide users to store any profile data except for an e-mail address and a personal website. However, Twitter does contain a list of those who the user is following (known as friends) and a list of those that are followers of the users. The Twitter API provides these as `friends` that denote those who the user is following and `followers` those who are following the user. These friends could be matched to see whether the requester and the user have similar friends (i.e. following users) and followers in common.

We assert trust by observing the similarity between two user profiles that have interacted with each other by comparing the distinct attributes that are common in both profiles. For each matched distinct attribute a value of 1 is assigned and the sum of matched attributes is calculated once all attributes are compared. The sum of matched attributes is averaged by the total number of distinct attributes found in the profile of whom is making the trust assessment (known as the assessor). Thus, the subjective trust value for profile similarity represents the relationship between the sum of matched distinct profile attributes between the assessor's profile and the requester's profile, and the total sum of all the assessor's distinct attributes within his/her profile. This calculation is represented with the following formula:

$$\tau = \frac{\sum_{i=1}^{n} m_i}{\sum_{i=1}^{n} a_i} \tag{1}$$

where τ denotes profile similarity subjective trust value, m denotes the matched distinct profile attributes between the user's profile and the requester's profile, and a denotes the total of all the user's distinct profile attributes.

Definition 1: profile similarity-based trust. Let PST be the subjective trust value for profile similarity, P a peer identified by a URI, PF a peer's FOAF profile, PAT a peer's profile attribute, A an assessor identified by a URI, AF an assessor's FOAF profile and AAT an assessor's profile attribute. Let $Profile(PF, P)$ or $Profile(AF, A)$ mean that PF is the profile of P or AF is the profile of A, $Contain(PAT, PF)$ or $Contain(AAT, AF)$ mean that PAT is within profile PF or AAT is within profile AF, $Match(PAT, AAT)$ mean that PAT is matched with AAT, $AssertedBy(P, A)$ mean that P is asserted by A and $AssignTrust(PST, P)$ mean that P is assigned PST, where $PST \in [-1, 1]$. Thus, Profile Similarity-based trust is defined:

$$\begin{aligned} &\forall AAT(Profile(PF, P) \wedge Profile(AF, A) \\ &\wedge Contain(PAT, PF) \wedge Match(PAT, AAT) \\ &\wedge AssertedBy(P, A)) \Rightarrow AssignTrust(PST, P) \end{aligned} \tag{2}$$

5.2 Sharing and tweeting

Sharing and tweeting based trust represents a trust value based on the number of shares or tweets between the user and requester. These contain content shared from other websites. This trust value is asserted by the *Sharing based Trust component*.

The external shared content are extracted from the Facebook API using the `links` object from both the user's and requester's account independently. The `links` contains information about the shared content including the original URL link of the shared content, the name of the content, a message written by whom is sharing, who shared the link and so forth. Since a user shares content with all the peers in his/her social graph, then all shares are taken into consideration if the requester is a connection with the user. Moreover, this also applies to the external content shared by the requester, i.e. external shared content by the requester is taken into consideration if the user is a connection with the requester.

Facebook does not differentiate between a share and a re-share in a user's `links` object. Hence, all the `links` objects returned are all considered as shares of that user. Only the original post would contain the information that it has been re-shared by a particular person. Therefore, the connection between a share and a re-share is a direct connection described within the original shared post's object. However, since in our use-case both the sharing and re-sharing of content are used to assert the requester, then this does not effect the trust assertion algorithm and all re-shares from Facebook will be treated as shares.

The direct shares from Twitter are retrieved from the Twitter API by calling the `statuses/user_timeline` endpoint from both the user's and requester's account. When requesting the Twitter API for the user's tweets, the parameter `include_rts` is set to false in order not to retrieve any re-tweets. Each tweet is then analysed to take into account only those tweets that contain `URLs` in the `entities` object. Moreover, these tweets are also analysed to take into account only those tweets that do not have any mentions within the tweet. The followers of the user are also retrieved to verify that the requester is following the user. This identifies that the requester had retrieved the externally shared content within his/her timeline. The tweets from the requester's timeline are also retrieved by analysing the tweets whether they contain any

URLs and that they do not have any direct mentions in the tweets. Also, the requester's followers are retrieved to identify that the user is following the requester. If the requester is not following the user or the user is not following the requester, then those tweets are not taken into consideration when calculating trust for the requester.

After all shares and tweets are extracted, a value of 1 is assigned to each share and tweet. Since not all peers are trusted in the same way for the same content, the profile similarity based trust metric is taken into consideration to weight this subjective trust value based on sharing or tweeting. Thus, the subjective trust value for sharing or tweeting is calculated as the weighted average of all the externally shared and tweeted content between the user and the requester weighted by the trust of the requester. This is represented as follows:

$$\bar{\tau} = \frac{\sum_{i=1}^{n} w_i s_i}{\sum_{i=1}^{n} w_i} \tag{3}$$

where $\bar{\tau}$ denotes the user's subjective trust value of shares and tweets-based trust value, w denotes the user's trust value of the requester within the user's social graph and s denotes the number of shares and tweets between the user and requester.

Definition 2: sharing and tweeting-based trust. Let *STT* be the subjective trust value for sharing and tweeting the content C, P a peer identified by a URI, *PST* a profile similarity trust value and A an assessor identified by a URI. Let $TrustValue(PST, P)$ mean that *PST* is the trust value of P, $AssertedBy(PST, A)$ mean that *PST* is asserted by A, $SharedBy(C, A)$ mean that C is shared by A, $TweetedBy(C, A)$ mean that C is tweeted by A, $SharedBy(C, P)$ mean that C is shared by P, $TweetedBy(C, P)$ mean that C is tweeted by P, $AssertedBy(C, A)$ mean that C is asserted by A and $AssignTrust(STT, C)$ mean that C is assigned *STT*, where $STT \in [-1, 1]$. Thus, Sharing and Tweeting-based trust is defined:

$$\begin{aligned} \forall C(&TrustValue(PST, P) \wedge AssertedBy(PST, A) \\ &\wedge (SharedBy(C, A) \vee TweetedBy(C, A)) \\ &\wedge (SharedBy(C, P) \vee TweetedBy(C, P)) \\ &\wedge AssertedBy(C, A)) \Rightarrow AssignTrust(STT, C) \end{aligned} \tag{4}$$

5.3 Re-sharing and retweeting

Re-sharing and retweeting based trust represents the trust value based on the number of re-shares or retweets of a particular content. This trust value is asserted by the *Re-sharing based Trust component*. Similar to sharing and tweeting, the re-shares and retweets of the user and the requester are taken into consideration. This denotes that the user and requester interacted with each other through re-sharing and retweeting content.

As mentioned earlier, Facebook does not differentiate between shares and re-shares. Therefore, re-shares are considered as shares. This does not effect the trust assertion algorithm since both shares and re-shares are ultimately used to assert the user's subjective trust value for the requester.

Whereas in Twitter, a tweet and a re-tweet are identified by the prefix *"RT"* within tweets which denote re-tweets. Re-tweets are extracted from the Twitter API from the user's and requester's account by calling the `statuses/user_timeline` endpoint and by setting the parameter `include_rts` to true. Once all the re-tweets are extracted, the user's followers are extracted to check whether the requester is following the user. Moreover, the requester's followers are also extracted to check whether the user is following the requester. If the user or the requester is not a follower, then those re-tweets are not taken into account when calculating trust based on this metric.

For each re-share and retweet by the user or requester is assigned a value of 1. Moreover, each re-share and retweet is weighted by the assessor's (i.e. user) trust value of the requester. Hence, the subjective trust value for re-sharing or retweeting is calculated as the weighted average of all the re-shares and retweets weighted by the trust of the requester. The trust of the requester is asserted using the profile similarity based trust method. This is represented as follows:

$$\bar{\tau} = \frac{\sum_{i=1}^{n} w_i r_i}{\sum_{i=1}^{n} w_i} \tag{5}$$

where $\bar{\tau}$ denotes the user's subjective trust value of re-tweets based trust value, w denotes the user's trust value of the requester within the user's social graph and r denotes the number of re-tweets between the user and the requester.

Definition 3: re-sharing and retweeting-based trust. Let *RTT* be the subjective trust value for re-sharing and retweeting the content C, P a peer identified by a URI, *PST* a profile similarity trust value and A an assessor identified by a URI. Let $TrustValue(PST, P)$ mean that *PST* is the trust value of P, $AssertedBy(PST, A)$ mean that *PST* is asserted by A, $ResharedBy(C, A)$ mean that C is re-shared by A, $RetweetedBy(C, A)$ mean that C is retweeted by A, $ResharedBy(C, P)$ mean that C is re-shared by P, $RetweetedBy(C, P)$ mean that C is retweeted by P, $AssertedBy(C, A)$ mean that C is asserted by A and $AssignTrust(RTT, C)$ mean that C is assigned *RTT*, where $RTT \in [-1, 1]$. Thus, Re-sharing and Retweeting-based trust is defined:

$$\begin{aligned}\forall C(&TrustValue(PST,P)\wedge AssertedBy(PST,A)\\&\wedge(ResharedBy(C,A)\vee RetweetedBy(C,A))\\&\wedge(ResharedBy(C,P)\vee RetweetedBy(C,P))\\&\wedge AssertedBy(C,A))\Rightarrow AssignTrust(RTT,C)\end{aligned}\tag{6}$$

5.4 Likes and favourites

The *Likes and Favourites based trust* represents the trust value based on the number of likes and favourites which the user and/or requester give to any content shared between them within the Social Web application. This trust value is asserted by the *Likes based Trust component*.

The Facebook API provides a call to the `likes` object which can be called directly for the user's or requester's account. Once all the `likes` objects are retrieved both from the user and from the requester, these can be matched to see whether they are all related to either the user or the requester.

However, the `likes` object contains only the likes to pages which a user has liked and not all the likes of the user. In order to get all the likes, rather than using the Facebook API, a query can be made using the Facebook's Query Language (FQL). A query is sent to the `like` table to retrieve all the `like` objects of the user. The objects include photos, albums, events, groups, notes, links, videos, applications, statuses, check-ins, reviews, comments and posts. In order to retrieve the likes of the requester, the query first must check that the requester's ID is a friend of the user. If this is true, then all the object IDs, object types and the post ID are retrieved. The query to retrieve the requester's likes is the following:

```
SELECT object_id,object_type,post_id
FROM like
WHERE user_id IN
  (SELECT uid2 FROM friend WHERE uid1=me())
```

The objects can be retrieved from their respective tables depending on the object type. Moreover, the posts can be retrieved from the `stream` table. Once the content is retrieved, the objects and content are matched to retrieve only the likes related to the user and requester.

In Twitter, favourites can be retrieved for a particular user ID by calling `GET favorites/list` from the Twitter API. The user's ID or the requester's ID must be provided in this call. However, only the first 20 favourites could be retrieved from Twitter which makes it restrictive to assert trust on this metric. Moreover, the requester's favorites are taken into consideration only if the user is following the requester and the user's favorites are taken into consideration only if the requester is following the user.

Each like and favourite is assigned a value of 1. Moreover, each like and favourite by a requester are weighted by the trust value of the requester asserted using the profile similarity based trust method. Hence, the subjective trust value for likes and favourites is calculated as the weighted average of all the likes and favourites related to the user and requester weighted by the trust value of the requester. This is represented as follows:

$$\bar{\tau}=\frac{\sum_{i=1}^{n} w_i l_i}{\sum_{i=1}^{n} w_i}\tag{7}$$

where $\bar{\tau}$ denotes the user's subjective trust value of likes and favourites based trust value, w denotes the user's trust value of the requester within the user's social graph and l denotes the number of likes and favourites related to the user and requester.

Definition 4: likes and favourites-based trust. Let LFT be the subjective trust value for likes and favourites of content C, P a peer identified by a URI, PST a profile similarity trust value and A an assessor identified by a URI. Let $TrustValue(PST,P)$ mean that PST is the trust value of P, $AssertedBy(PST,A)$ mean that PST is asserted by A, $LikedBy(C,A)$ mean that C is liked by A, $FavouriteBy(C,A)$ mean that C is favourite by A, $LikedBy(C,P)$ mean that C is liked by P, $FavouriteBy(C,P)$ mean that C is favourite by P, $AssertedBy(C,A)$ mean that C is asserted by A and $AssignTrust(LFT,C)$ mean that C is assigned LFT, where $LFT\in[-1,1]$. Thus, Likes and Favourites-based trust is defined:

$$\begin{aligned}\forall C(&TrustValue(PST,P)\wedge AssertedBy(PST,A)\\&\wedge(LikedBy(C,A)\vee FavouritedBy(C,A))\\&\wedge(LikedBy(C,P)\vee FavouritedBy(C,P))\\&\wedge AssertedBy(C,A))\Rightarrow AssignTrust(LFT,C)\end{aligned}\tag{8}$$

5.5 Comments and replies

The *Comments and Replies-based trust* represents the trust value based on the number of comments and replies for requesters based on comments and replies exchanged between users and requesters. This value is asserted by the *Comments-based Trust component*.

In order to retrieve the comments of the user and the requester from the Facebook API, multiple calls have to be made using the `comments` filter from the `feed` and `post` objects. Therefore, we use the Facebook Query Language (FQL) to retrieve the comments directly from the tables since it is easier to handle in this way. The comments are retrieved by using the following query:

```
SELECT post_id, actor_id
FROM stream
WHERE filter_key in
  (SELECT filter_key FROM stream_filter WHERE uid=me())
```

– instead of `me()`, this is replaced by the requester ID when retrieving the requester's posts. With the post IDs, we get all comments from the `comment` table and we then match the `fromid` with either the user or the requester. The `fromid` cannot be used in queries since it is non-indexable, otherwise it would have been easier to query using the `fromid`. The number of matched comments is taken into consideration when asserting trust using this subjective trust method; taking into account that the requester is within the user's social graph.

In Twitter it is much easier as replies are in the form of direct messages using the `@` prefix. The Twitter API provides calls to retrieve the direct messages by using `GET direct_messages` and also to retrieve the sent direct messages by a user, by using `GET direct_messages/sent`. Hence, we retrieve the user's and requester's sent and received direct messages by using these `GET` methods. We match the sender or receiver with the user and/or requester. The number of matched messages is taken into consideration when asserting trust for this method. Moreover, whether the the user and requester are following each other is also taken into account.

For each comment and reply by a user and/or a requester, a value of 1 is assigned. Moreover, this value is weighted by the trust value of the requester—using the profile similarity-based trust metric. Hence, the subjective trust value for comments and replies is calculated as the weighted average of all the comments and replies related to the user and requester weighted by the trust value of the requester. This is represented as follows:

$$\bar{\tau} = \frac{\sum_{i=1}^{n} w_i c_i}{\sum_{i=1}^{n} w_i} \tag{9}$$

where $\bar{\tau}$ denotes the user's subjective trust value of comments and replies based trust value, w denotes the user's trust value of a third party user within the user's social graph and c denotes the number of comments and replies of the third party user which are related to the requester being judged.

Definition 5: comments and replies-based trust. Let *CRT* be the subjective trust value for comments and replies of content C, P a peer identified by a URI, *PST* a profile similarity trust value and A an assessor identified by a URI.

Let $TrustValue(PST, P)$ mean that *PST* is the trust value of P, $AssertedBy(PST, A)$ mean that *PST* is asserted by A, $CommentBy(C, A)$ mean that C is commented by A, $ReplyBy(C, A)$ mean that C is replied by A, $CommentBy(C, P)$ mean that C is commented by P, $ReplyBy(C, P)$ mean that C is replied by P, $AssertedBy(C, A)$ mean that C is asserted by A and $AssignTrust(CRT, C)$ mean that C is assigned *CRT*, where $CRT \in [-1, 1]$. Thus, Comments and Replies-based trust is defined:

$$\begin{aligned} \forall C(&TrustValue(PST, P) \wedge AssertedBy(PST, A) \\ &\wedge (CommentBy(C, A) \vee ReplyBy(C, A)) \\ &\wedge (CommentBy(C, P) \vee ReplyBy(C, P)) \\ &\wedge AssertedBy(C, A)) \Rightarrow AssignTrust(CRT, C) \end{aligned} \tag{10}$$

5.6 Tags and mentions

The Tags and Mentions-based trust represents the trust value based on the tags and mentions by the user and requester. The perception of trust whilst using this interaction is twofold: (1) when an assessor (i.e. user) is tagged or mentioned by a requester; and (2) when a requester is tagged or mentioned by an assessor (i.e. user).

This value is asserted by the *Tags-based Trust component*.

In Facebook, there are various ways how users tag others and how users are tagged. Users are tagged for posts (including locations, checkins etc.), pictures and videos. The Facebook API provides the `tagged` object for users tagged in posts; the `photos` object for the photos in which the user is tagged in and the `videos` object for those videos in which the user is tagged in. These objects also correspond to the FQL tables: `stream_tag` table for posts in which users are tagged in; `video_tag` for videos in which the user is tagged in and `photo_tag` for the photos in which the usrs are tagged in. Queries such as:

```
SELECT post_id,actor_id
FROM stream_tag
WHERE target_id=me()
```

retrieves all the posts where the user was tagged in and by whom the user was tagged. We retrieve all the tags about the user and/or requester and tags tagged by the user/and or requester. We then match these results to the user and requester in order to know how many times the requester tagged the user and how many times the requester was tagged by the user. Hence, we take these counts to assert the user's subjective trust value of the requester.

In Twitter, retrieving mentions are done with this call: `GET statuses/mentions_timeline` which retrieves all the tweets for a user's `@screen_name`. Therefore, we retrieve all the tweets containing the mentions from both the user's and requester's account. The tweets are than checked to match whether the tweets were posted by either the user or requester. The number of mentions are used to assert the subjective trust value for the requester.

For each tag and mention, a value of 1 is assigned. Moreover, this value is weighted by the trust value of the requester, using the profile similarity-based trust metric. Hence, the subjective trust value for tags and mentions is calculated as the weighted average of all the tags and mentions related to the user and/or requester weighted by the trust value of the requester. This is represented as follows:

$$\bar{\tau} = \frac{\sum_{i=1}^{n} w_i g_i}{\sum_{i=1}^{n} w_i} \tag{11}$$

where $\bar{\tau}$ denotes the user's subjective trust value of comments and replies based trust value, w denotes the user's trust value of a requester within the user's social graph and g denotes the number of tags and mentions by the user and/or requester that have tagged and mentioned the user and/or requester.

Definition 6: tags and mentions-based trust. Let TMT be the subjective trust value for tags and mentions of content C, P a peer identified by a URI, PST a profile similarity trust value and A an assessor identified by a URI. Let $TrustValue(PST, P)$ mean that PST is the trust value of P, $AssertedBy(PST, A)$ mean that PST is asserted by A, $TaggedBy(C, P)$ mean that C is tagged by P, $TaggedIn(A, C)$ mean that A is tagged in C, $TaggedBy(C, A)$ mean that C is tagged by A, $TaggedIn(P, C)$ mean that P is tagged in C, $Mentioned\ By(C, P)$ mean that C is mentioned by P, $MentionedIn(\ A, C)$ mean that A is mentioned in C, $MentionedBy(C, A)$ mean that C is mentioned by A, $MentionedIn(P, C)$ mean that P is mentioned in C, $AssertedBy(C, A)$ mean that C is asserted by A and $AssignTrust(TMT, C)$ mean that C is assigned TMT, where $TMT \in [-1, 1]$. Thus, Comments and Replies-based trust is defined:

$$\begin{aligned} &\forall C(TrustValue(PST, P) \wedge AssertedBy(PST, A) \\ &\quad \wedge((TaggedBy(C, P) \wedge TaggedIn(A, C)) \\ &\quad \vee(TaggedBy(C, A) \wedge TaggedIn(P, C)) \\ &\quad \vee(MentionedBy(C, P) \wedge MentionedIn(A, C)) \\ &\quad \vee(MentionedBy(C, A) \wedge MentionedIn(P, C))) \\ &\quad \wedge AssertedBy(C, A)) \Rightarrow AssignTrust(TMT, C) \end{aligned} \tag{12}$$

6 Assessing trust for requesters

The Social User Interactions based Trust explained in section 5 are used to assert the trustworthiness of requesters. In order to assign a fine-grained user's subjective trust value to a requester, we calculate an average of all the subjective trust values of a requester from each social interaction assigned by the user. This calculation is represented by the following formula:

$$\tau = \frac{1}{n} \sum_{i=1}^{n} si_i \tag{13}$$

where τ denotes the aggregated subjective trust value and si a subjective trust value asserted based on a social user interaction.

Definition 7: requestor subjective trust. Let RST be the requester's subjective trust value, R a requester identified by a URI, U a user identified by a URI, PST be the subjective trust value for profile similarity, STT be the subjective trust value for sharing and tweeting, RTT be the subjective trust value for re-sharing and retweeting, LFT be the subjective trust value for likes and favourites, CRT be the subjective trust value for comments and replies and TMT be the subjective trust value for tags and mentions. Let $Assigned(PST, R)$ mean that PST is assigned to R, $Assigned(STT, R)$ mean that STT is assigned to R, $Assigned(RTT, R)$ mean that RTT is assigned to R, $Assigned(LFT, R)$ mean that LFT is assigned to R, $Assigned(CRT, R)$ mean that CRT is assigned to R, $Assigned(TMT, R)$ mean that TMT is assigned to R $AssertedBy(R, U)$ mean that R is asserted by U and $AssignTrust(RST, R)$ mean that R is assigned RST, where $RST \in [-1, 1]$. Thus, the Requester's Subjective Trust is defined:

$$\begin{aligned} &Assigned(PST, R) \wedge Assigned(STT, R) \\ &\wedge Assigned(RTT, R) \wedge Assigned(LFT, R) \\ &\wedge Assigned(CRT, R) \wedge Assigned(TMT, R) \\ &\wedge AssertedBy(R, U) \Rightarrow AssignTrust(RST, R) \end{aligned} \tag{14}$$

Asserting trust for requesters with no previous interactions with the user will result in zero trust. In this case, the requester's trust can be asserted using our previous work (Sacco et al. 2013). In this work, we had presented a trust model to assert trust for requesters from information in the Social Web. Our model asserts trust based on (1) the requester's identity; (2) profile similarity between the requester and the user; (3) the relationship between the requester and the user; and (4) the reputation of the requester in the "Web of Trust" (i.e. based on other users trust values of the requester).

7 Modelling trust assertions: trust assertion ontology (TAO)

The Trust Assertion Ontology illustrated in Fig. 4—http://vocab.deri.ie/tao#—is a light-weight vocabulary that provides classes and properties to describe user's subjective trust values for requesters (Sacco et al. 2013). The user's subjective trust values are computed and stored in an RDF store. Whenever the user's subjective trust values are required, a new value is computed (if necessary) on the information which was created after the time the pre-computed subjective trust value was asserted. The pre-computed subjective trust value is also taken into account whilst recomputing the new user's subjective trust value. Once the subjective trust values are recomputed and stored, the aggregate subjective trust value is then calculated on which a trust decision is based. However, this is not stored since user interactions are continuously changing and it can be computed using the stored subjective trust values. We have extended this vocabulary with new properties to take into account the user's subjective trust value asserted from user interactions within Social Networks. These new properties are highlighted in Fig. 4 which include the following:

- `tao:hasSharesTrust` specifies the user's subjective trust value based on the number of shares of content.
- `tao:hasResharesTrust` specifies the user's subjective trust value based on the number of reshares of content.
- `tao:hasLikesTrust` specifies the user's subjective trust value based on the number of "likes", "+1s" or "favourites" of content.
- `tao:hasCommentsTrust` specifies the user's subjective trust value based on the number of comments.
- `tao:hasTaggingTrust` specifies the subjective trust value based on the number of tags within content.

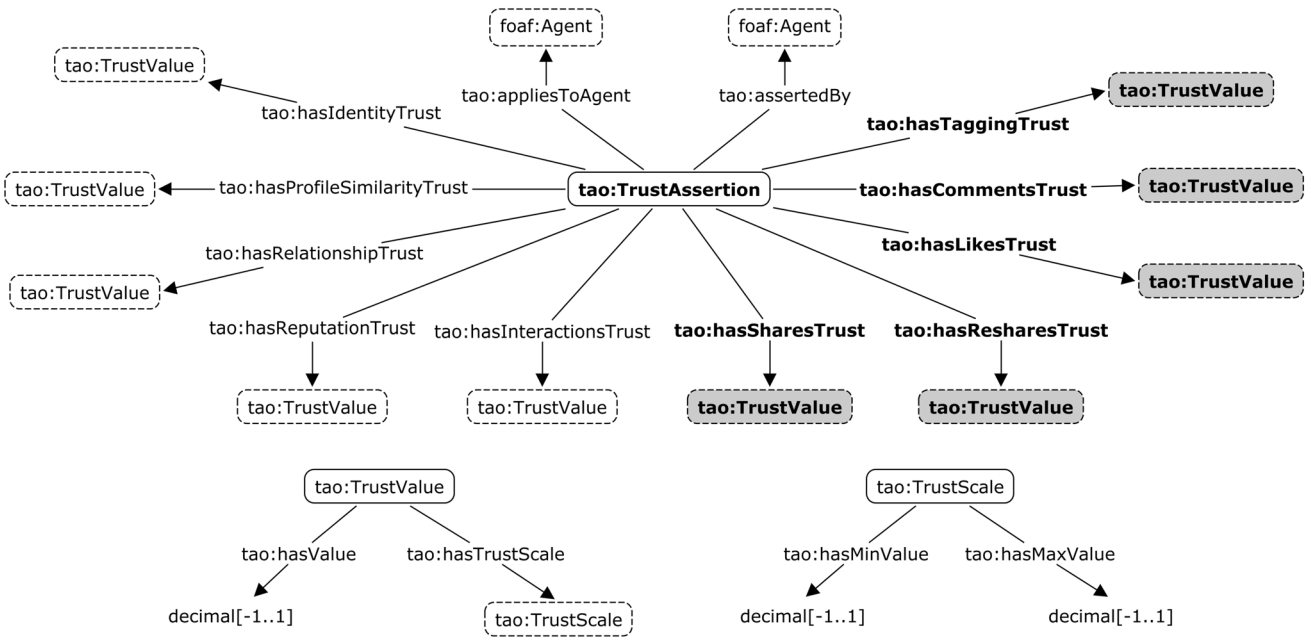

Fig. 4 Trust assertion ontology

```
PREFIX tao: <http://vocab.deri.ie/tao#> .
PREFIX ex: <http://vmuss13.deri.ie/> .

ex:tao1 a tao:TrustAssertion;

tao:assertedBy
  <http://vmuss13.deri.ie/userprofiles/winu#
     me>;
tao:appliesToAgent
  <http://vmuss13.deri.ie/userprofiles/
     terraces#me>;

tao:hasProfileSimilarityTrust [
  tao:hasValue "0.46";
  tao:hasTrustScale taoscale;].

tao:hasSharesTrust [
  tao:hasValue "0.23";
  tao:hasTrustScale taoscale;].

tao:hasResharesTrust [
  tao:hasValue "0.14";
  tao:hasTrustScale taoscale;].

tao:hasLikesTrust [
  tao:hasValue "0.37";
  tao:hasTrustScale taoscale;].

tao:hasCommentsTrust [
  tao:hasValue "0.14";
  tao:hasTrustScale taoscale;].

tao:hasTaggingTrust [
  tao:hasValue "0.28";
  tao:hasTrustScale taoscale;].

ex:taoscale a tao:TrustScale;
tao:hasMaxValue "1.0";
tao:hasMinValue "-1.0".
```

Fig. 5 Describing user's subjective trust assertions using the trust assertion ontology (TAO)

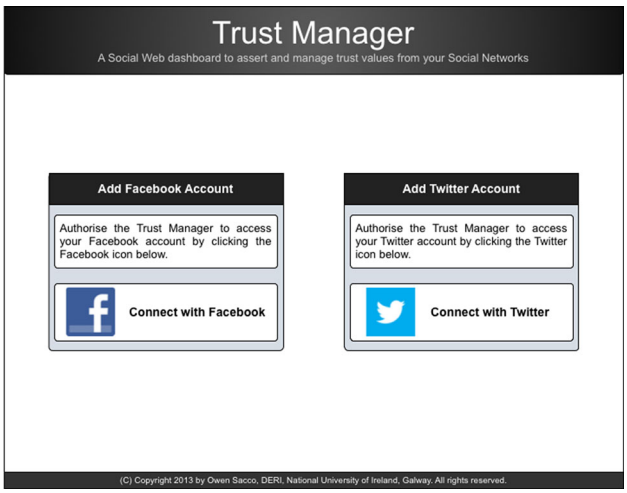

Fig. 6 Trust manager home page

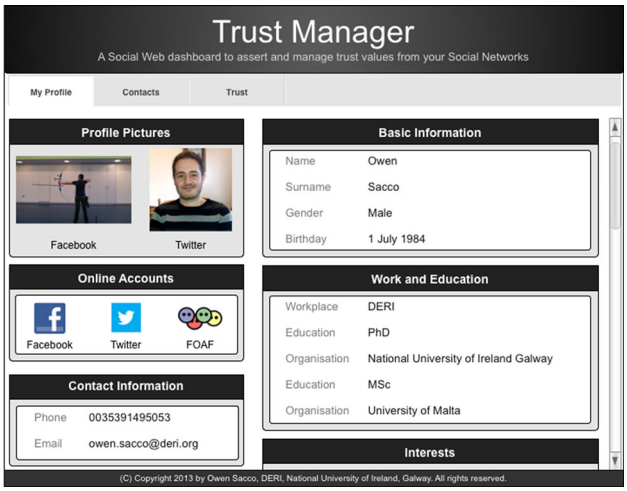

Fig. 7 Trust manager user's profile

An example of describing subjective trust values using this ontology is illustrated in Fig. 5 which depicts a user asserting a subjective trust value for a requester.

8 Trust manager in-use

The Trust Manager application provides users with the trust values of their contacts and also of requesters when requesting any personal information. It also provides other statistical information for each social user interaction used to assert the trust value for each contact. In this section, screen shots of the application are illustrated to explain how this tool is used. Although this tool is used to assert trust values, its purpose is to also demonstrate and encourage Social Web applications to implement trust as a feature in their system; it can also be implemented in Social Web aggregators; or used as an independent tool such as a contact management system; and also in many other scenarios. A REST endpoint could be implemented to provide other applications with users trust values; for example for online payment gateways and recommender systems; whilst preserving users privacy.

The main page of the Trust Manager is illustrated in Fig. 6. The user has to authorise the Trust Manager to access his/her Facebook and/or Twitter account in order to assert trust values of his/her contacts. Once the user authorises either Facebook and/or Twitter, the authorisation page from the respective Social Network appears. The Trust Manager then requests the user to continue once the authorisation phase is successful. The user's profile from Facebook and Twitter are then aggregated and transformed in RDF using common vocabularies as explained in Sect. 4.2.2. The aggregated profile is displayed to the authenticated user as illustrated in Fig. 7 and can be downloaded using the FOAF icon.

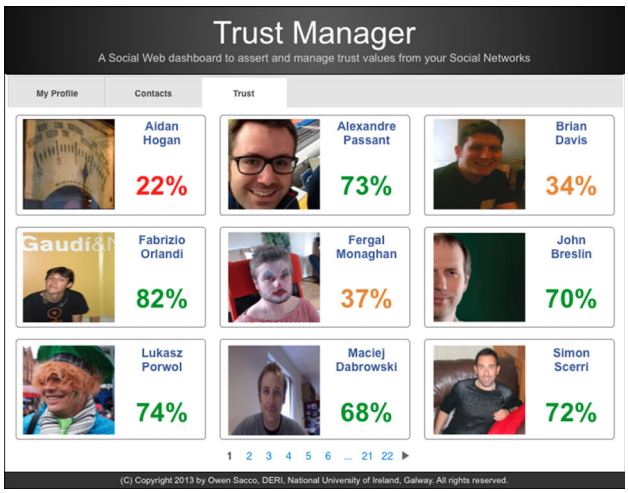

Fig. 8 Trust manager user's contacts with trust values

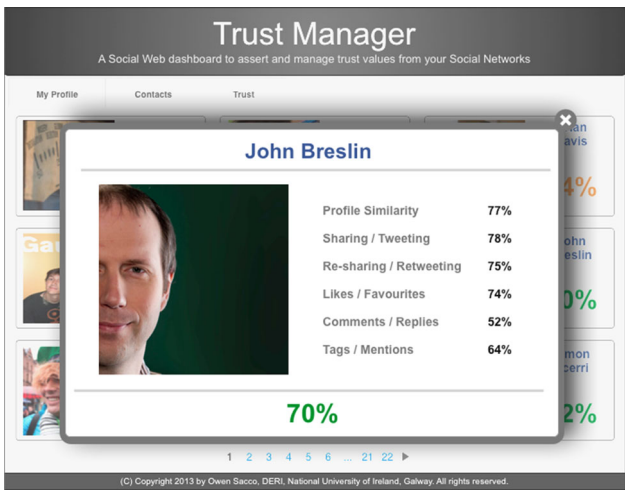

Fig. 9 Trust manager user's contact trust values

Whenever the user clicks on the `Trust` menu, the Trust Manager calculates the trust values for each user's contact. If the user is using the tool for the first time, then the Trust Manager calculates the trust values for each contact using each social user interaction trust metric. These subjective trust values for each social user interaction are described using the TAO vocabulary (as explained in Sect. 7) and stored in an RDF store for later retrieval. If the trust values were previously computed, the Trust Manager checks the last date of when the trust values were computed. The Trust Manager then updates the trust values by using the social user interactions as from this date onwards. Once all trust values are asserted for each contact, then the aggregated score is computed and displayed to the user as illustrated in Fig. 8.

The user can click on any contact in order to view the details of how the trust value was calculated using each social user interaction metric. These statistics are provided to the user as illustrated in Fig. 9.

Table 2 Average number of information extracted for each profile

Social user interaction types	Average No.
Profile attributes	20
Sharing external content	30
Re-sharing or retweeting content	10
Like, +1 or "favourite" content	55
Comment or Reply	45
Tag or mention other users	65
Tagged or mentioned by other users	55
Friends or followers or followees	550

9 Experiment and evaluation

We carried out our experiments to evaluate the Trust Manager. Our experiment consisted of 500 profiles from one user's social graph from Facebook and 380 accounts from twitter of the same user. Once these accounts were aggregated, 520 distinct user accounts were used in our evaluation. We analysed the graph structure and we identified the connections amongst these users based on mutual connections. Hence, we created social sub graphs for each user based on the information that we collected. The full social graph for each contact (i.e. "friend") was not extracted since the social graph from Facebook cannot be extracted unless authorised by the respective user.

Table 2 illustrates some information about the collected profiles. These figures show the average number of interactions for each social user interaction types. For each connection, the information for each Social User interaction was retrieved. We then calculated the trust values amongst the connections.

We calculated the weighted harmonic mean (F measure) for each individual metric based on the extracted content when $\beta = 1$. We then calculated an average of the scores across all profiles. Table 3 illustrates a comparison of the averages for precision, recall and F_1 score for each Social User interaction. We noticed that when combining all the metrics, this produced better results with an average F_1 score of 74.2 %. Also, since in Social Media precision is regarded more important than recall due to the continuous flow of data; when aggregating scores, this gives a higher precision score. Hence, aggregating all values from these Social User interactions gives a better result for computing trust than using individual scores.

The profile similarity method gives the highest individual score, since the attributes are extracted directly from the profile; unlike the other methods whereby the information is extracted from various content in the Social Network. Moreover, although Facebook does not differentiate content between shared or as re-shared and so re-shares were considered as shares; the scores for the re-

Table 3 Average precision, recall and F_1 score when $\beta = 1$ for each social user interaction

Social user interaction types	P (%)	R (%)	F_1 score (%)
Profile similarity	72.4	73.4	72.9
Sharing	68.7	64.7	66.6
Re-sharing or retweeting	72.2	71.2	71.7
Like, +1 or "favourite"	64.8	67.2	66.0
Comment or reply	62.2	64.2	63.2
Tag or mention	72.3	72.2	72.2
Tagged or mentioned	62.4	64.3	63.3
Aggregated trust value	77.4	71.3	74.2

sharing or retweeting method is higher due to a higher number of retweets published in Twitter. Comment or reply metric gives the lowest F_1 score because Facebook API's is not straight forward to extract all the comments—as explained in Sect. 5.5.

10 Conclusion

In this paper we have demonstrated how Social User Interactions can be used to assert trust. We have developed a Trust Manager that contains the Trust Assertion Controller, the Semantic Web Components and the Trust Assertion Components. The Trust Assertion Controller handles the calls to the Semantic Web Components and to the Trust Assertion Components. The Semantic Web components are responsible for extracting profile and user interactions information from the Social Web and transforming this data into RDF. The Trust Assertion Components are responsible for asserting trust based on profile similarity and user interactions. Our results show that when aggregating each social user interaction based trust metric produces a higher score.

As future work, we will continue to analyse other user interactions that could improve our trust assertion model. Moreover, since in this work we focused on asserting trust at a particular point in time, we will also analyse how trust values decay based on the fluctuation in the number of interactions between users. Furthermore, we will also integrate our Trust Manager with our Privacy Preference Framework (Sacco and Passant 2011a, 2011b; Sacco et al. 2011) to enforce privacy preferences based on these trust assertions.

References

Artz D, Gil Y (2007) A survey of trust in computer science and the semantic web. Science, Services and Agents on the World Wide Web, Web Semantics

Giunchiglia F, Shvaiko P, Yatskevich M, Giunchiglia F (2004) S-match: an algorithm and an implementation of semantic matching. In: Proceedings of ESWS

Goel S, Hofman JM, Sirer MI (2012) Who does what on the web: a large-scale study of browsing behavior. In proceedings of the sixth international conference on weblogs and social media, ICWSM2012

Golbeck J (2006) Generating predictive movie recommendations from trust in social networks. In Trust Management, iTrust'06

Golbeck J (2009) Trust and nuanced profile similarity in online social networks. ACM Transactions on the Web

Golbeck J, Hendler J (2004) Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. In knowledge engineering and knowledge management, EKAW'04

Golbeck J, Hendler J (2006) Inferring binary trust relationships in web-based social networks. ACM transactions on internet technology

Golbeck J, Parsia B, Hendler J (2003) Trust networks on the semantic web, In cooperative intelligent agents

Grandison T, Sloman M. A survey of trust in internet applications. IEEE communications surveys and tutorials

Guha R, Kumar R, Raghavan P, Tomkins A (2004) Propagation of trust and distrust. In proceedings of the 13th international conference on world wide web, WWW'04

Hartig O (2009) Querying trust in rdf data with tsparql. In European semantic web conference, ESWC'09

Kim YA, Le M-T, Lauw H, Lim E-P, Liu H, Srivastava J (2008) Building a web of trust without explicit trust ratings. In data engineering workshop. ICDEW 2008. IEEE 24th international conference on, 2008

Kim YA, Song HS (2011) Strategies for predicting local trust based on trust propagation in social networks. Knowledge-Based Systems

Liu H, Lim E-P, Lauw HW, Le M-T, Sun A, Srivastava J, Kim YA (2008) Predicting trusts among users of online communities: an epinions case study. In proceedings of the 9th ACM conference on electronic commerce, EC '08

Marsh SP (1994) Formalising trust as a computational concept. PhD thesis, University of Stirling

Mui L, Mohtashemi M, Halberstadt A (2002) A computational model of trust and reputation for e-businesses. In Hawaii international conference on system sciences, HICSS '02

Olmedilla D, Rana OF, Matthews B, Nejdl W (2005) Security and trust issues in semantic grids, In semantic grid

Sacco O, Breslin JG (2013) I like—analysing interactions within social networks to assert the trustworthiness of users, sources and content. In ASE/IEEE SocialCom'13

Sacco O, Passant A (2011) A privacy preference manager for the social semantic web. In semantic personalized information management workshop, SPIM'11

Sacco O, Passant A (2011) A privacy preference ontology (PPO) for linked data. In linked data on the web workshop, LDOW'11

Sacco O, Passant A, Decker S (2011) An access control framework for the web of data. In IEEE TrustCom-11

Sacco O, Passant A, Decker S (2013) Fine-grained trust assertions for privacy management in the social semantic web. In IEEE TrustCom-13

Shvaiko P, Giunchiglia F Yatskevich M (2009) Semantic matching with s-match, In semantic web information management

Ziegler C-N, Golbeck J (2007) Investigating interactions of trust and interest similarity. Decision support systems