Web Quality Assessment Model (Technical Report 02/11)

Tomáš Knap
Department of Software Engineering
Charles University in Prague
Czech Republic
tomas.knap@mff.cuni.cz

Andre Freitas
Digital Enterprise Research Institute (DERI)
National University of Ireland
Galway, Ireland
andre.freitas@deri.org

Irena Mlýnková
Department of Software Engineering
Charles University in Prague
Czech Republic
irena.mlynkova@mff.cuni.cz

ABSTRACT

The unprecedented volume of information presented on the Web as Linked Data and the wide variety of web users with distinct situations at their hands pose the practical difficulties when designing a quality assessment (QA) process assessing the quality of information on the Web. To that end, we propose the Web Quality Assessment (WQA) model formalizing such a QA process, which is customizable according to the information consumer's needs – every consumer can define a set of policies, which are then, as part of the QA process, applied to the set of resources requested by him to deduce the score and, hence, the suitability of these resources. To improve the results of such QA process, the QA policies are shared among entities connected with each other in the QA social network to reinforce the number of QA policies applied during the QA process. Since the information consumer cannot use QA policies of an arbitrary entity in the QA social network, we need to define a trust model enabling to express that an entity trusts another entity (there is a trust relation between them) and propose a trust algorithm ReWA capable of deriving trust between two entities not connected with a trust relation. Since the social networks on the Web (1) can be quite huge and (2) full of intentionally or unintentionally malicious entities, the paper compares the accuracy, robustness, complexity, and scalability of the algorithm ReWA with other trust algorithms. As a result, we provide developers a solid social trust network-based model for assessing the quality of information on the Web.

1. INTRODUCTION

The advent of Linked Data [9] in the recent years accelerates the evolution of the Web into a giant information space where the unprecedented volume of resources will offer to the information consumer a level of information integration and aggregation that has up to now not been possible. Consumers can now 'mashup' and readily integrate information for use in a myriad of alternative end uses. Indiscriminate addition of information can, however, come with inherent problems such as the provision of poor quality, inaccurate, irrelevant or fraudulent information. All will come with an associate cost which will ultimately affect decision making, system usage and uptake.

Therefore, the ability to assess the *information quality* (*IQ*) presents one of the most important aspects of the information integration on the Web and will play a fundamental role in the continued adoption of Linked Data principles [43]. IQ is usually described in different works by a series of *IQ dimensions* which represent a set of desirable characteristics for an information resource [3, 49, 6, 43]. Wang & Strong [49] present an extensive survey of IQ dimensions, based on the results of the questionnaire given to the panel of human subjects; papers [3, 2] cover IQ dimensions for the Web. In this paper, we neither intend to present our set of IQ dimensions for the Web, nor discuss the current sets of IQ dimensions; we merely introduce the universal way how the quality of unprecedented volume of resources on the Web can be assessed and mediated to the information consumer.

According to traditional practices on the Web, the information consumers "make judgments about which sources to rely on based on prior knowledge about a source's perceived reputation, or past personal experience about its quality relative to other alternative sources" [21]. This approach is impractical, (1) having scalability issues with the increasing information available on the Web and (2) eliminating a priori any kind of non-human agents, where only the automated judgments are feasable.

Naumann and Rolker [43] argue that assessing IQ is rightly considered difficult, because IQ dimensions are "often of subjective nature and can therefore not be accessed automatically". Knight and Burn [3] argue similarly that the perception of IQ is highly dependent on the fitness for use being relative to the specific task that entities have at their hands. Furthermore, the Web is emerging as a global information space where the documents and data can be reused, aggregated and interconnected in new and unexpected ways, thus, we have to cope with the data with the beforehand unknown structure.

To that end, we define a generic social trust network-based *Web Quality Assessment model* (*WQA model*), where customizable sets of *QA policies* drive the *QA process* assessing the quality of resources based on a set of IQ dimensions preferred by the information consumer. As a result, each resource considered by the WQA model is associated with the *QA score*, the primary indicator of the quality of the resource. Each QA policy belongs to a *QA analysis* – an implementation of one or more IQ dimensions. The set of QA analyses and QA policies determining the QA process is hold in the *QA profile* of an information consumer.

The idea of policies and rules is not novel and was successfully used by variety of access control mechanisms [34, 42, 13]. What is novel in the WQA model is the concept of *QA social networks* representing a social network [44, 40] an information consumer is part of, where every entity can (1) specify his QA profiles and (2) express how much he trusts his acquaintances. Based on that, the QA policies can be shared among entities connected with each other in the QA social network and trusting each other above the certain threshold to reinforce the number of QA policies applied during the QA process. Since it has been widely documented that social networks have the properties of small world networks, where the average distances between entities in the network are small and connectance of entities is high [44, 40], there is a chance of obtaining substantial amount of QA policies from other entities.

To enable this, we have to define a trust model explaining what we mean under *trusting another entity* and enabling to express that an entity trusts another entity; such an expression is then represented as an edge – trust relation – in the QA social network weighted by the amount of trust. Furthermore, we need to propose a trust algorithm capable of deriving trust between two entities not connected with a trust relation, so that we can use QA profiles of that entity, if the entity is sufficiently trusted.

Since the social networks on the Web (1) can be quite huge and (2) full of intensionally or unintentionally malicious entities, we have to carefully select the suitable trust algorithm deriving the trust between arbitrary two entities in the QA social network. Specially, we have to exploit the accuracy, robustness, complexity, and scalability of that algorithm.

To wrap the introduction up, our main contribution is threefold:

- A proposal of the generic Web Quality Assessment (WQA) model for assessing the quality of information on the Web. By incorporating the provenance IQ dimension to the WQA model, we partially verify and demonstrate the model's feasibility.
- A proposal of the trust model (including the concept of trust dynamics) and trust algorithm ReWA used to derive trust between arbitrary two entities in the QA social network of the WQA model.
- Comparison of accuracy, robustness, complexity, and scalability of the ReWA trust algorithm with other trust algorithms.

The paper is organized as follows. Section 2 introduces the financial motivational scenario defining the *provenance QA criteria* on the quality of consumed data. Section 3 defines the concept of trust in social trust networks, which are then used in Section 4 in the definition of the WQA model and the QA process itself. Section 5 presents the selected trust algorithms which are going to be compared; consequently, Section 6 evaluates the presented trust algorithms w.r.t their accuracy, robustness, complexity, and scalability and discuss their suitability in the QA process of the WQA model. Section 7 reviews related work and the paper is rounded off with a conclusion and discussion of future work in Section 8.

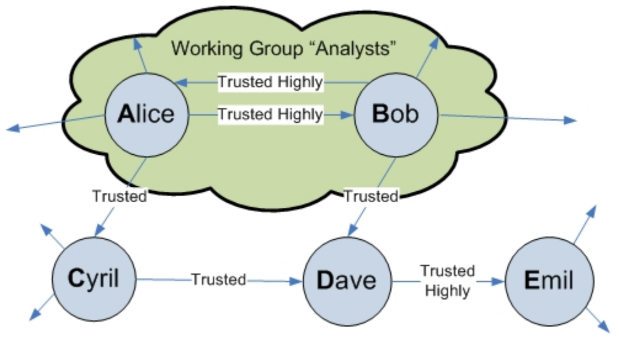

Figure 1: Sample social network Alice (a) is part of.

By convention, letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$ denote the sets of *all* possible individuals, such as the sets of all entities, QA policies, or resources, letters $A, B, C, \ldots$ denote their subsets (if applicable), and small letters $a, b, c, \ldots$ denote the particular individuals.

2. FINANCIAL MOTIVATIONAL SCENARIO

We have chosen the *financial scenario* (FS) to motivate the needs for the WQA model. Alice, a financial analyst and the information consumer, is preparing the financial report focusing on the summarized overview of the economic context of the executives of Fortune500 companies in the first quarter 2010. To achieve her goal, she is using a Linked Data [9] *mashup application* consuming and aggregating information included in stock markets time series, financial reports, government data, demographics, previous analysis, and third-party qualitative and quantitative analysis, physically distributed around the globe in various resources, such as RDF/XML [7] documents, named graphs [14], database tables, and XHTML pages.

Furthermore, we assume that Alice is part of the social network sn depicted in Figure 1. Such a social network consists of trust relations between Alice and (1) her colleague Bob working in the same department, (2) other financial experts, such as Cyril, and (3) her working group Analysts. Alice expresses in all cases that she trusts to some extend in other entities in the social network - users (B, C) or groups (*Analysts*). Furthermore, since she trusts Bob and Cyril, she trusts also David to some extend, because there is a relation between Bob, Cyril and David. All these entities are supposed to have more trust relations not depicted in Figure 1.

The aggregated data Alice is interested in poses many non-trivial questions she has to face, such as:

Q1 Where is the information coming from?

Q2 Is the information coming from trustworthy sources?

Q3 Is the quarterly US GDP projection provided by Bloomberg[1] original or derived from another source? If derived, which source is quoted?

Q4 Is the euro exchange rate up-to-date?

Q5 How many organizations/people support the investment risk analysis score provided for Ireland?

[1]http://www.bloomberg.com/

Q6 Which of several contradicting statements for US unemployment rate should be preferred?

Q7 Does the entity have the rights to reuse the US Government data in a report?

Q8 Are the methods involved in the quantitative analysis of stock prices reliable and compliant to the entity's standards and methodologies?

Q9 How much is the information accurate?

Since provenance or lineage of data (1) provides the necessary contextualization for the information consumer to analyze the quality of the information [41, 20, 26] and (2) allows the transfer of trust from entities behind the resources to the information in the resources [21], the provenance metadata – the data describing the resources at Alice's hands – helps Alice to answer all the questions Q1 – Q9, as long as the provenance metadata is available.[2]

To further clarify the substantial importance of provenance for the information consumer, we point out to an experiment introduced by Pinheiro da Silva et al. [17]. In this experiment, the scientists were trying to identify and explain imperfections of a set of maps. The results show that around 80% of scientists correctly identified the imperfections of the maps when they know provenance of the maps. Without any provenance information at hand, the same scientists were able to identify only around 10% of all map imperfections.

Due to the importance of the data provenance, Alice decides to incorporate Provenance QA Analysis to the QA process. Although the provenance analysis forms the basic analysis Alice is interested in, to rank the resources [1], some of the questions (Q2, Q4, Q6 – Q9) need an extra Alice's consideration or application of further analyses [26, 4, 45, 3], such as the analysis of data timeliness (Q4), reputation of resources, popularity (Q2), or analysis of accuracy (Q9). Nevertheless, in this motivational scenario, we assume that Alice uses only Provenance QA analysis.

Based on the the questions Q1 – Q9 above, Alice constructs her *QA criteria (QAC)* – the criteria which will drive the QA process incorporating the Provenance QA analysis; in Section 4, we will show, how the QA criteria can be straightforwardly converted to the QA policies. The numbers in brackets behind each QAC introduce the questions Q1 – Q9 which motivated the particular QAC:

QAC 1 Raw stock market information published by the official operator NYSE[3] is trustworthy. (Q1, Q2)

QAC 2 Raw stock market information published by the official operators LSE, BOVESPA, and NIKKEI[4] is trustworthy. (Q1, Q2)

QAC 3 Any news excerpts which do not introduce the source they quote are distrusted. (Q3)

QAC 4 Open resources annotated by more DBpedia[5] triples are prioritized. (Q6)

QAC 5 The open resources hosted by GovTrack, Geonames, and US Census Bureau[6] are trustworthy. (Q1, Q2)

QAC 6 In case of resources containing qualitative and quantitative analyses, the hosts must be certified by the Thawte[7] certification authority. (Q2, Q7 – Q9)

QAC 7 Signed resources are preferred when several contradicting statements occur. (Q2, Q6)

QAC 8 Due to the strategic importance of the report, the qualitative analysis generated by junior analysts are considered untrustworthy and are not used in the report. (Q2)

QAC 9 Trends supported by at least three subjects are considered stronger and prioritized. (Q5, Q6)

QAC 10 Resources advocated by more entities are prioritized. (Q5, Q6)

3. TRUST IN WQA MODEL

A *social network* [44, 40] is typically modeled as a directed graph $sn(V, E)$, where the vertices, $V \subseteq \mathcal{V}$, represent entities of the network (users, groups, or organizations) and the edges, $E \subseteq V \times V$, *relations* between them [24, 51]. The social network provides a suitable mechanism for sharing QA policies (grouped into QA profiles) between the entities participating in the social network; for example, an employee in a company may reuse (in his QA process) QA profile of the department he works in or QA profiles of his colleagues. Obviously, the information consumer is more willing to inherit QA profiles of entities he *trusts* more.

3.1 Concept of Trust

Manifestations of trust are easy to recognize because we experience and rely on it every day, but at the same time trust is quite challenging to define because it manifests itself in many different forms and it is used with wide variety of meanings [28, 4, 23, 39]. Definition 3.1 drives the comprehension of the term trust in the further text, because (1) it correlates with the financial scenario, where the information consumer Alice must decide whose QA policies are worth using (will bring good outcome) and (2) it comprehends trust as the subjective opinion of an entity about another entity $v \in \mathcal{V}$, not as the global reputation of the entity v in the social network [51, 31].

Definition 3.1. *(Concept of Trust). Suppose that an entity* ***u*** *encounters a situation where he perceives an ambiguous path (decides whether to use QA policies of an entity* ***v*** *or not). The result of following the path can be good or bad, and the occurrence of the good or bad result is contingent on the action of the entity* ***v****. If the entity* ***u*** *chooses to go down the path (decides to use QA policies of* ***v****), he has made a trusting choice; he* ***trusts*** *that* ***v*** *will take the steps necessary to ensure the good outcome – provide expected QA policies.*[8]

[2]The current activities in the area of data provenance and annotations show that these assumptions are being fulfilled by more and more resources, see [41] and http://twiki.ipaw.info/bin/view/Challenge.
[3]http://www.nyse.com/
[4]http://www.londonstockexchange.com/, http://www.bmfbovespa.com.br/, and http://e.nikkei.com/
[5]http://dbpedia.org/
[6]www.govtrack.us/, www.geonames.org/, www.census.gov/
[7]http://www.thawte.com/
[8]Based on the definition introduced by Deutsch in [19]

The *trusting choice* in Definition 3.1 can be quantified either on a discrete [35, 24] or continuous scale [37, 25, 51, 48, 24]. In general, discrete trust levels are easily seizable by humans when expressing trust between each other, on the other hand, continuous trust values provide more accurate expressions of trust, however, are far below the level of differentiation of the average human.

In [24], Golbeck internally uses (in a social network similar to ours) continuous trust values $tv \in [1, 10]$, however, externally, the information consumer is provided with ten discrete *trust levels* ranging from "absolute trust" ($tv = 10$) to "absolute distrust" ($tv = 1$), with $tv = 5$ expressing the "neutral trust" (neither positive, nor negative) or the absence of trust. We exploited the similar approach.

In our trust model, the trusting choice of the entity u w.r.t. entity v is internally quantified as continuous *trust value* $\tau^t_{u,v} \in [0, 1] \cup \perp$, a frequent representation of trust [48, 25, 51, 4, 23], computed by the *trust function* $\tau : \mathcal{V} \times \mathcal{V} \times \mathbb{N} \rightarrow [0, 1] \cup \perp$; hence $\tau^t_{u,v} = \tau(u, v, t)$. Since trust between entities is evolving in time, t in $\tau^t_{u,v}$ represents the time $t \in \mathbb{N}$ in which the given trust value is valid. We define $\tau_{u,v}$, holding the current trust value, as $\tau_{u,v} = \tau^{t_1}_{u,v}$, so that $\nexists t_2 > t_1$. We discuss the *trust dynamics* in Subsection 3.1.1.

The trust value $\tau^t_{u,v}$ is ranging from "absolute distrust" ($\tau^t_{u,v} = 0$) to "high trust" ($\tau^t_{u,v} \rightarrow 1$), the value $\tau^t_{u,v} > 0.3$ expresses that the entity v is comprehended as trusted (to some extend); $\tau^t_{u,v} = \perp$ if the trust value is unknown. Furthermore, we assume that $\tau^t_{u,u} = 1$, however, similarly as in [37], if $u \neq v$, $\tau^t_{u,v} \neq 1$, because the entity u cannot know with 100% certainty that the entity v will behave like expected. On the other hand, $\tau^t_{u,v} = 0$ can be ascribed through a thoughtful judgement typically based on the bad experience of the entity u with the entity v in the past [37] – the entity u never wants to use any single QA policy of the entity v.

We agree that discrete trust levels are easily seizable by information consumers expressing the trust in other entities in the social network. Regrettably, it is hard to imagine that entities will consistently subjectively map their trust to others on a ten point scale (as proposes Golbeck in [24]) – there is no guarantee for the information consumer that someone "really trusted" is always expressed as 9 or 8. Therefore, we externally use only five trust levels depicted in Table 1 together with the corresponding shortcuts and $\tau^t_{u,v} \in [0, 1]$. The level *TRUSTED_HIGHLY* should be given to close friends of v, family members, or colleagues in a company – the entity u can be almost sure that these entities will make necessary steps to ensure the good outcome according to Definition 3.1. The trust level *TRUSTED* should be assigned to all other entities not classified as TRUSTED_HIGHLY, however, still rather trustworthy – we can expect that they will ensure the good outcome. If the entity u only knows another entity v, however, cannot decide whether v is rather trustworthy or untrustworthy, the trust level *KNOWN* should be used. If the user u wants to explicitly express that the entity v is distrusted, trust level *DISTRUSTED* is used. All other entities are automatically considered as *NOT_KNOWN*, there is no trust relation between entities u and v; these entities are considered more trustworthy than entities explicitly stated as DISTRUSTED.

3.1.1 Trust Dynamics

When the trust value $\tau^t_{u,v}$ is increased by $\beta \in [-1, 1]$ for some $u, v \in \mathcal{V}$, the time counter t is increased[9] and, $\forall x, y \in \mathcal{V}$ for which $\tau^t_{x,y} \neq \perp$, the trust value $\tau^{t+1}_{x,y}$ is set to $\tau^t_{x,y} + \Delta\tau^{t,t+1}_{x,y}$; $\Delta\tau^{t,t+1}_{x,y}$ denotes the amount of change in trust of an entity x in y between the times t and $t+1$ and is set according to Formula 1:

Table 1: Trust Levels

Trust Level Label	Shortcut	$\mathbf{\tau^t_{u,v}}$
DISTRUSTED	D	0.1
NOT_KNOWN	NK	0.2
KNOWN	K	0.3
TRUSTED	T	0.7
TRUSTED_HIGHLY	TH	0.9

$$\Delta\tau^{t,t+1}_{x,y} = \begin{cases} \beta, & \text{if } x = u \wedge y = v \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

3.2 Social Trust Networks

We introduce a concept of *social trust network*[10] – a directed weighted graph $stn(V, E, [\tau]_{stn}) \in \mathcal{STN}$, where $V \subseteq \mathcal{V}$, $E \subseteq \{V \times V\}$, and $[\tau]_{stn} : V \times V \times \mathbb{N} \rightarrow [0, 1] \cup \perp$ represents the *realization* of the trust function in the social trust network *stn*. Each edge – *trust relation* – $(u, v) \in E$ is weighted by the function $[\tau]_{stn}$ determining how much trust an entity u assigns to another entity v in the time t.

Example 3.1. *In the social network visualized in Figure 1, Alice specifies that Bob is TRUSTED_HIGHLY. This trust relation can be written as an edge* $(Alice, Bob) \in E$*, with* $\tau^t_{Alice,Bob} = TH$*. Furthermore, Alice considers Cyril as TRUSTED, i.e.* $\tau^t_{Alice,Cyril} = T$*.*

Since an entity can be involved in more social trust networks (imagine that a user is working during the day and playing cards in the evening with two different groups of people), we define how these networks are merged to simplify the algorithms computing trust between arbitrary two entities. If $stn^1(V_1, E_1, [\tau]_{stn^1}) \in \mathcal{STN}$ and $stn^2(V_2, E_2, [\tau]_{stn^2}) \in \mathcal{STN}$ are arbitrary two social trust networks, so that $V_1 \cap V_2 \neq \emptyset$[11], we merge them to a new social trust network $stn(V_1 \cup V_2, E_1 \cup E_2, [\tau]_{stn})$ and put $\mathcal{STN} = \mathcal{STN} \setminus \{stn^1, stn^2\} \cup \{stn\}$; $[\tau]_{stn}$ is defined according to Formula 2, where $t_1, t_2 \in \mathbb{N}$ are the current times in stn_1, respectively stn_2. This process of merging is iterated as long as no such two social trust networks stn^1 and stn^2 exist; the resulting $\mathcal{STN}$ is denoted as $\mathcal{STN}_{\mathcal{M}}$. As a consequence, every entity $v \in \mathcal{V}$ is involved in at most one social trust network; hence, stn_c – the social trust network $stn \in \mathcal{STN}_{\mathcal{M}}$ the information consumer $c \in \mathcal{V}$ is part of – is determined uniquely.

$$[\tau]_{stn}(u, v, t) = \begin{cases} [\tau]_{stn^1}(u, v, max\{t_1, t_2\}), & \text{if } (u, v) \in E_1 \\ [\tau]_{stn^2}(u, v, max\{t_1, t_2\}), & \text{if } (u, v) \in E_2 \end{cases} \tag{2}$$

[9]We suppose that the time t is counted from 0 and increases when such β occurs.

[10]A concept of social trust network is similar to the web of trust introduced in PGP system

[11]In the test on $V_1 \cap V_2$, two entities with different HTTP URI are considered as equal, if the predicate `owl:sameAs` [18] states so.

Since not all entities in the social trust networks are connected by a trust relation, the information consumer $u \in \mathcal{V}$ can either use QA policies only from trustworthy *neighbors* (entities connected by a trust relation), or we need to define a a function τ_{ta} computing a quantitative estimate – *a derived trust value* – of how much trust an entity u should accord to another entity v. The function $\tau_{ta} : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$ is defined in Formula 3, where $ta.execute(u, v)$ is the result of the particular trust algorithm $ta \in \mathcal{ALG}$ inferring the derived trust value in the social trust network stn_u; the set of trust algorithms, $\mathcal{ALG}$, is particularized in Section 5 and these algorithm are compared between each other in Section 6.

$$\tau_{ta}(u, v) = \begin{cases} \tau_{u,v} & \text{if } \tau_{u,v} \neq \perp \\ ta.execute(u, v) & \text{otherwise} \end{cases} \tag{3}$$

A *trust relation path*, $\pi_{u,v}$, between entities u and v in some $stn(V, E, [\tau]_{stn})$ is a progression of trust relations $(u = v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n = v)$, $v_i \in V$, so that each $e_i = (v_i, v_{i+1}) \in E$ and $\forall e_i, e_j \in \pi_{u,v} : e_i \neq e_j$, where $1 \leqq i \leqq n$; $e \in \pi_{u,v}$ $(x \in \pi_{u,v})$ denotes that $e \in E$ $(x \in V)$ is part of the path $\pi_{u,v}$. The number of trust relations involved in $\pi_{u,v}$ is denoted as $|\pi_{u,v}|$, whereas $\#\pi_{u,v}$ denotes the number of distinct trust relation paths between u and v; two paths $\pi_{u,v}$ and $\pi'_{u,v}$ are distinct if $\exists e \in \pi_{u,v}$ and $\exists e' \in \pi'_{u,v}$, so that $e \neq e'$.

3.2.1 *Persistence of Social Trust Networks*

Since the social trust networks are not valid only during the execution of the information consumer's query, we have to propose a persistence of trust relations between entities. We would like to persist data in the (1) application independent and (2) machine understandable way.

In most cases, social networks coexisting in the Web information space[12] are tightly connected with the applications utilizing them (such as MySpace[13]). Contrary, the Friend of a Friend (FOAF) project[14] is an ontology enabling to describe, using RDF model, (1) entities as instances of RDF class `foaf:Agent` and (2) trust relations with trust level K betweens entities as instances of the predicate `foaf:knows`.

Since there are no other open, application independent, machine readable, and widely used implementations of social networks, we decided to use FOAF ontology as a cornerstone for a representation of entities and trust relations in social trust networks.

The trust level K is directly represented by the `foaf:knows` property in the FOAF ontology, the level *NK* does not need to be explicitly expressed. Unfortunately, FOAF ontology does not define the concept of trust, therefore, we developed an extension of the FOAF ontology, called WQA Trust Module[15], having the capability to express and persist the other trust levels – *TH*, *T*, *D*.

4. WQA MODEL

The Web Quality Assessment Model (WQA) formalizes the QA process. From the information consumer's perspective, the basic workflow of the QA process is depicted in Figure 2.

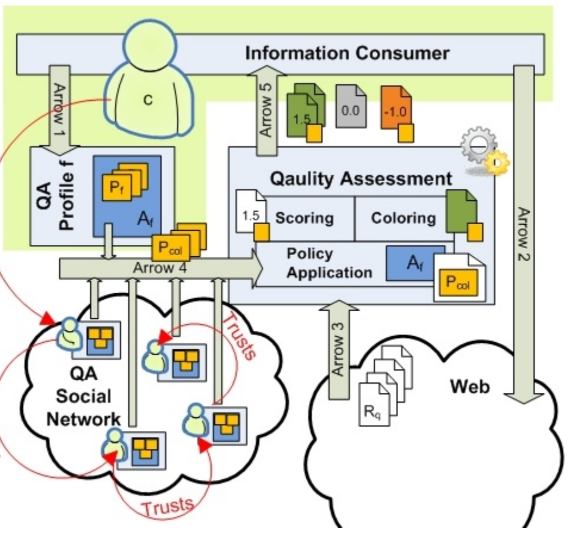

Figure 2: The WQA model workflow

Firstly, the information consumer c, defines his QA profile f involving QA analyses A_f and QA policies P_f (Arrow 1 in Figure 2). Afterwards, the consumer queries the Web (Arrow 2); the set of resources R_q relevant to the query q is returned, based on the result of the conventional search engine, such as Google[16], or semantic search engine, such as Sindice[17] (Arrow 3).

At this moment, the QA process is entered. The set of QA policies P_{col} is collected from the consumer's QA profile and from trustworthy entities (red arrows with "Trusts" labels) forming the QA social network of the information consumer (Arrow 4). After that, the QA analyses A_f are launched, the collected QA policies P_{col} $(P_f \subseteq P_{col})$ belonging to the analyses A_f are tried to be applied to the resources R_q; the successfully applied QA policies are reflected in the QA score and the QA color computed for each processed resource $r \in R_q$ (Blue box "Quality Assessment"). The QA color of the resource serves as a first signal to the information consumer whether the resource r is recommended (green color), recommended with warnings (yellow), unknown, in the sense that no QA policy was successfully applied to this resource (grey), or not recommended at all (red). Afterwards, every resource $r \in R_q$ is ranked according to its QA score and returned to the information consumer together with the information about the set of successfully applied QA policies $r.pol \subseteq P_{col}$ (Arrow 5).

4.1 Resources and Entities

A resource, $r \in \mathcal{R}$, is a basic unit of consumed information to which QA policies can be applied.[18] Every resource is identified by the unique URI in a format which is compatible with Linked Data principles for creating URIs[19] [9]. Every resource r is supplemented with $r.score$ and $r.color$

[12]See [24, p. 10 – 28] for the list of web-based social networks
[13]http://www.myspace.com
[14]http://xmlns.com/foaf/spec/
[15]http://www.ksi.mff.cuni.cz/˜knap/wqa/tm.owl

[16]http://www.google.com
[17]http://sindice.com/
[18]We suppose that all resources are from the financial domain.
[19]http://www.w3.org/DesignIssues/LinkedData.html

Table 2: Quality Requirement Levels

QR level	Sample usage	Financial loss
Crucial (4)	Creating financial report driving the future business	Significant
Important (3)	Completing product dimension in the data warehouse	Small
Normal (2)	Preparing homework for the university course	Insignificant / No
Below Normal (1)	Arguing with a friend about the ice hockey match result	No

for holding the QA score and QA color of the resource.

Let $\mathcal{V}$ denote the set of entities (users, groups of users) on the Web. Every entity $u \in U \subseteq \mathcal{V}$ can be part of several groups $g \in G \subseteq \mathcal{V}$. An user, $c \in U$, who queries the Web and consumes the resources $R_q \subseteq \mathcal{R}$ retrieved from the Web as the answer on the query $q \in \mathcal{Q}$, is called the information consumer.

4.2 QA Profiles

Let $\mathcal{A}$ denote the set of QA analyses and $\mathcal{P}$ the set of QA policies. Every entity $v \in \mathcal{V}$ can specify the QA profile $f \in F_v \subseteq \mathcal{F}$ which drive the QA process (F_v is the set of all QA profiles specified by the entity v). Every QA profile f contains a set of analyses $A_f \subseteq \mathcal{A}$ and a set of QA policies $P_f \subseteq \mathcal{P}$ associated with f.

A QA profile f must be associated with one of the five *quality requirement* (*QR*) *levels* (accessible as $f.qrl$). Various QR levels cover the demand on different requirements on the quality of the resources in different situations. Table 2 lists the admissible QR levels together with their sample usage and the financial loss associated with the chosen level. The financial loss should be the primary factor for determining to which QR level the QA profile belongs.

Every QA analysis is associated with the customizable *weight* – a coefficient determining how much the result of the analysis influences the final QA score (the weight of the analysis is denoted as $w_a \in \mathbb{R}^+$); the bigger number denotes the higher weight.

4.2.1 QA Profiles Persistence

In order to store QA profiles of entities in QA social networks, we developed a FOAF extension module called WQA Profile Module[20]. The main purpose of the WQA Profile Module is the capability to persist the content of QA profiles – QA policies, QA analyses, and QR levels.

Regarding the persistence of the groups, FOAF ontology contains the predicate `foaf:member` to express the membership of the `foaf:Agent` to the given `foaf:Group`. The WQA Profile Module further defines (among others) RDF predicates `foafq:memberOf` to represent the inverse property to the property `foaf:member` (suitable for tracking that a user belongs to a group), and `foafq:hasQAProfile` to associate a QA profile with an instance of the RDF class `foaf:Agent`.

4.3 QA Policies and Terms

We distinguish three types of QA policies: *basic QA policies* $P^B \subseteq \mathcal{P}$, *prioritizing QA policies* $P^P \subseteq \mathcal{P}$ and *counting QA policies* $P^C \subseteq \mathcal{P}$. A QA policy $p \in \mathcal{P}$ always *belongs* to one analysis $a \in \mathcal{A}$ (denoted as $p.belongsTo = a$). Every QA policy $p \in \mathcal{P}$ is associated with its *importance* accessible as $imp_p \in (0, 5)$; the bigger number denotes the higher importance.

A basic QA policy $p \in P^B$ is defined as $p = (type,\ lev,\ \{tle_1, \ldots, tle_i, \ldots, tle_n\})$, where $type = B$ identifies the policy p as the basic QA policy, $tle_i \in \mathcal{Z_T}$, $1 \leqq i \leqq n$, denotes the RDF triple [36], $tle_i = (subj \in \mathcal{Z_S},\ pred \in \mathcal{Z_P},\ obj \in \mathcal{Z_O})$. The admissible values of the sets $\mathcal{Z_S}$, $\mathcal{Z_P}$, $\mathcal{Z_O}$ are determined by the RDF specification [36] (all what is allowed in RDF as subject, predicate, respectively object is allowed here as *subj*, *pred*, respectively *obj*).

We say that the QA policy $p \in P^B$ can be *successfully applied* to the resource $r \in \mathcal{R}$ as part of the QA analysis $a \in \mathcal{A}$, if $p.belongsTo = a$ and the QA policy p is *matching* the resource r according to Definition 4.1. The expression $|r|_p$ denotes the number of successful applications of the QA policy p to resource r (i.e. the distinct number of pairs (t_i, s_j) in Definition 4.1); $|R_q|_p$ denotes the total number of successful applications of the QA policy p to all resources $r \in R_q$.

Definition 4.1. *Let us suppose that a resource $r \in \mathcal{R}$ is associated with a set of RDF triples $\{s_1, \ldots, s_m\}$, $m \in \mathbb{N}$. Further, a QA policy $p \in \mathcal{P}$ defines a set of RDF triples $\{t_1, \ldots, t_n\}$, $1 \leqq i \leqq n$. The QA policy p is matching the resource r as long as there exists a pair (t_i, s_j), so that $t_i.subj = s_j.subj \wedge t_i.pred = s_j.pred \wedge t_i.obj = s_j.obj$, $1 \leqq i \leqq n$, $1 \leqq j \leqq m$, where $t_i(s_j).subj$ is the subject, $t_i(s_j).pred$ is the predicate, and $t_i(s_j).obj$ is the object of the RDF triple according to [36].*

The successful application of the QA policy p to the resource r is represented by a function $a_{pol}(p, r) : \mathcal{P} \times \mathcal{R} \rightarrow \{true, false\}$; $a_{pol}(p, r) = true$ as long as the QA policy p can be successfully applied to the resource r.

The $lev \in \mathcal{Z_L}$, $\mathcal{Z_L} = \{$*TRUST(T)*, *DISTRUST(D)*, *PRIORITIZE(P)*, *DEPRIORITIZE(DP)*$\}$, in the definition of the QA policy $p \in P^B$ expresses whether the information consumer wants to trust, distrust, prioritize, or deprioritize the resource to which the policy p can be successfully applied. The elements of the QA policy p are referenced as *p.type*, *p.lev*, and $p.tle_i$, the elements of the $p.tle_i$ are referenced as $tle_i.subj$, $tle_i.pred$, $tle_i.obj$, representing successively the subject, predicate, and object of the RDF triple tle_i, $1 \leqq i \leqq n$.

Example 4.1. *A QA policy $p = (B,\ T,\ \{([r],\ x:publishedBy,\ "http://www.nyse.com/"),\ ("http://www.nyse.com/",\ x:publishes,\ [r])\})$ is generated by QAC 1. The expression $[r]$ in the subject of the first triple and the object of the second triple denotes the resource to which QA policy p is applied; `x:publishedBy` and `x:publishes`, the predicates from the XProv provenance model [20], denote the desired RDF predicates with the namespace prefix x[21]; "http://www.nyse.com/" is the desired object/subject of the predicates. The QA policy p contains two triples, tle_1 and tle_2, because the XProv provenance model enables to express QAC 1 in two different ways.*

Further, let us define a prioritizing QA policy $p \in P^P$, $p = (type,\ lev,\ \{tle_1, \ldots, tle_i, \ldots, tle_n\})$, where $type = P$

[20]The full ontology can be downloaded at http://www.ksi.mff.cuni.cz/~knap/wqam/qapm.owl

[21]$x = "http://prov4j.org/xprov/xdesc\#"$

identifies p as the prioritizing QA policy, $lev \in Z_L^P$, $Z_L^P = \{P, DP\}$, $tle_i \in \mathcal{Z_T}$, $1 \leqq i \leqq n$. The prioritizing QA policy p enables to express that the information consumer prioritizes ($lev = P$) or deprioritizes ($lev = DP$) the resources $r \in R_q$ with the highest $|r|_p$.

Example 4.2. *The prioritizing QA policy* $p = (P, P, \{([r], x{:}supportedBy, x : Agent), (x : Agent, x : supports, [r])\})$ *is generated by QAC 10.*

Finally, let us define a counting QA policy $p \in P^C$, $p = (type, lev, \{tle_1, \dots, tle_i, \dots, tle_n\}, cond, val)$, where $type = C$ identifies p as the counting QA policy, $lev \in \mathcal{Z_L}$, $tle_i \in \mathcal{Z_T}$, $1 \leqq i \leqq n$. The condition $cond \in \mathcal{Z_C}$, $\mathcal{Z_C} = \{Eq, Less, LessEq, More, MoreEq\}$, successively corresponds to the basic arithmetic operators $=, <, \leqq, >, \geqq$. The variable $val \in \mathcal{Z_V}$ can involve special strings "$[N]$", "$[Z]$", "$[R]$", and "$[NOW]$" representing, successively, arbitrary natural number $\mathbb{N}$, whole number $\mathbb{Z}$, real number $\mathbb{R}$, and the current system date. The counting QA policy $p \in P^C$ is matching the resource $r \in \mathcal{R}$ according to Definition 4.1 supplemented with the additional requirement – the condition $p.cond$ must hold for $|r|_p$ as the left operand and $p.val$ as the right operand.

Example 4.3. *The counting QA policy* $p = (C, P, \{([r], x{:}supportedBy, "*"), ("*", x : supports, [r])\}, MoreEq, "3")$ *forms QAC 9 – trends supported by at least three entities (whoever they are) are prioritized.*

All parts of the QA policies support regular expressions according to Java 6 specification[22]. The characters ", [,], \, and metacharacters of the regular expressions are metacharacters in WQA model; these characters can be escaped by the character \.

4.3.1 Restrictions in QA Policies

Every QA policy $p \in \mathcal{P}$ can be supplemented with a set of *restrictions* added as the last element to the n-tuple defining the QA policy p, e.g., if $p \in P^B$, $p = (type, lev, \{tle_1, \dots, tle_i, \dots, tle_n\}, restrictions)$. The *restrictions* in the definition of a QA policy can contain a set of RDF triples with *variables.* The variables are denoted as $[X]$, where X is any capital letter of Greek alphabet; the expressions like $[X|Y|Z]$ are possible with the meaning: either the content of the variable $[X]$, $[Y]$, or $[Z]$ fits there. The variables are always valid only for the given QA policy and can occur in any triple within the QA policy. When the QA policy involves at least one variable (in the form $[X]$) within arbitrary $p.tle_i$, $1 \leqq i \leqq n$, the QA policy must involve the *restrictions* section introducing the same variable(s) as well.

During the application of a QA policy to a resource, the QA process tries to substitute consequently all the variables in the *restrictions* part with the possible values. Suppose for a while that a restriction part of a QA policy involves just one variable $[X]$. Once the possible value X_{val} for the variable $[X]$ is found, it is fixed for the rest of the QA policy. Only after that, the QA process tries to apply the QA policy to the resource; if the QA policy cannot be applied to the resource with the fixed value X_{val}, the substitution is withdrawn and another substitution for the variable $[X]$ is tried. If the QA policy with the fixed value X_{val} can be applied to the resource, no further substitutions are tried and the QA policy is successfully applied to the resource r. Similarly, when more variables are specified within the *restrictions*, all these variables must be fixed before the QA policy is tried to be applied to the resource. The expressions like $[X|Y|Z]$ are possible with the meaning: either the content of the variable X, Y, or Z fits there.

Example 4.4. *The QA policy* $p = (B, D, \{([X], y{:}isJuniorAnalyst, "true")\}, \{([r], x : createdBy, [X])\})$*, generated by QAC 8, involves the variable* $[X]$*, which is substituted by the particular name of the resource's creator. We suppose that there is a predicate* `y:isJuniorAnalyst` *in some ontology.*

Example 4.5. *QA criteria 1 – 10 directly correspond with one or more QA policies; Table 3 lists the QA criteria and all the corresponding QA policies built according to the rules defined. The range of QA policies available is strongly impacted by the used provenance model for tracking provenance on the Web [26, 20]. We suppose that Alice uses X-Prov Provenance Model[24], whenever possible. The* `x:publishedBy` *predicate has an inverse predicate* `x:publishes`*, which is, for simplification, not considered when formulating QA policies.*

4.4 QA Templates

The designer of the analysis $a \in \mathcal{A}$ must define a set of *QA templates*, $T_a \subseteq \mathcal{T}$, in order to prescribe the form in which the QA policies $p \in \mathcal{P}$, $p.belongsTo = a$, $a \in \mathcal{A}$, may be constructed. Every QA policy must be created as an instance of the particular QA template. The QA policy $p^t \in P^B$ denotes the basic QA policy p together with an additional information that the policy was created as an instance of the template $t \in \mathcal{T}$; $t = (B, Z_L^t, \{tle_1^t, \dots, tle_i^t, \dots, tle_n^t\})$, where B denotes the basic QA policy, $p.lev \in Z_L^t \subseteq \mathcal{Z_L}$ and $tle_i^t \in Z_T^t \subseteq \mathcal{Z_T}$, $1 \leqq i \leqq n$, determines the admissible $p.tle_i$. The template triple tle_i^t is of the form $tle_i^t = (Z_{S_i}^t \subseteq \mathcal{Z_S}, Z_{P_i}^t \subseteq \mathcal{Z_P}, Z_{O_i}^t \subseteq \mathcal{Z_O})$ and can be instantiated by $p.tle_i$, if $p.tle_i.subj \in Z_{S_i}^t$, $p.tle_i.pred \in Z_{P_i}^t$, $p.tle_i.obj \in Z_{O_i}^t$, $1 \leqq i \leqq n$. If the special symbol $"*" \in Z_{S_i}^t$, $"*" \in Z_{P_i}^t$, respectively $"*" \in Z_{O_i}^t$, the particular set is equal to $\mathcal{Z_S}$, $\mathcal{Z_P}$, respectively $\mathcal{Z_O}$. The templates for the policies P^P and P^C are defined similarly.

The importance of the QA policy p is prescribed by the QA template t; by default, $imp_{p^t} = 1$.

Example 4.6. *The template t for the QA policy p in Example 4.1 can be* $(B, \{T, D\}, (\{[r]\}, \{x : publishedBy, x : createdBy\}, \{"*"\}))$*. The policy p^t is, therefore, allowed to contain the trust level T or D, the predicate $x : publishedBy$ or $x : createdBy$ and an arbitrary string as the RDF object.*

4.5 QA Social Networks

A QA social network $qn \in \mathcal{QN}$ is a tuple $(\mathcal{STN_M}, \tau, ta)$, written as $qn(\mathcal{STN_M}, \tau, ta)$, where $\mathcal{STN_M}$ is the set of pairwise merged social trust networks defined in Subsection 3.2, τ is the trust function defined in Section 3, and $ta \in \mathcal{ALG}$ is the trust algorithm used to compute τ_{ta}.

4.6 QA Process

Algorithm 1 formally describes the QA process in the WQA model. The input to Algorithm 1 is the consumer $c \in \mathcal{V}$, the consumer's QA profile $cf \in F_c$, the set of resources $R_q \subseteq \mathcal{R}$ returned as a result on the consumer's query

[22]Class `java.util.regex`

[24]http://www.prov4j.org/xprov/

Table 3: QA Policies for the QA Criteria

QAC	QA Policies[23]
1	$(B,\, T,\, \{([r],\, x:publishedBy,\, "http://www.nyse.com/")\})$
2	$(B,\, T,\, \{([r],\, x:publishedBy,\, "http://www.londonstockexchange.com/")\})$, $(B,\, T,\, \{([r],\, x:publishedBy,\, "http://www.bmfbovespa.com.br/")\})$, $(B,\, T,\, \{([r],\, x:publishedBy,\, "http://e.nikkei.com/")\})$
3	$(B,\, T,\, \{([r],\, x:derivedFrom,\, "*")\})$
4	$(P,\, P,\, \{([r],\, dbpedia:*,\, "*"),\, ("*",\, dbpedia:*,\, [r])\})$
5	$(B,\, T,\, \{([r],\, x:hostedBy,\, "*.govtrack.us"),\, ("*.govtrack.us",\, x:hosts,\, [r])\})$, $(B,\, T,\, \{([r],\, x:hostedBy,\, "*.geonames.org"),\, ("*.geonames.org",\, x:hosts,\, [r])\})$, $(B,\, T,\, \{([r],\, x:hostedBy,\, "*.census.gov"),\, ("*.census.gov",\, x:hosts,\, [r])\})$
6	$(B,\, T,\, \{([A\|B],\, x:certifiedBy,\, "http://www.thawte.com/"),$ $("http://www.thawte.com/",\, x:certifies,\, "[A\|B]")\},\, \{([r],\, x:hostedBy,\, [A]),\, ([B],\, x:hosts,\, [r])\})$
7	$(B,\, P,\, \{([r],\, swp:signature,\, "*")\})$
8	$(B,\, D,\, \{([A\|B],\, y:isJuniorAnalyst,\, "true")\},\, \{([r],\, x:createdBy,\, [A]),\, ([B],\, x:creates,\, [r])\})$
9	$(C,\, P,\, \{([r],\, x:supportedBy,\, "*"),\, ("*",\, x:supports,\, [r])\},\, MoreEq,\, "3")$
10	$(P,\, P,\, \{([r],\, x:supportedBy,\, x:Agent),\, (x:Agent,\, x:supports,\, [r])\})$

q, and the QA social network $qn \in \mathcal{QN}$. The output of Algorithm 1 is the modified set R_q, $\widetilde{R_q}$, where each resource $r_i \in \widetilde{R_q}$, $1 \leqq i \leqq |\widetilde{R_q}|$, is supplemented with $r_i.score$ and $r_i.color$

Algorithm 1 QA process

```
Input: $c \in \mathcal{V}$, $cf \in F_c$, $R_q \subseteq \mathcal{R}$, $qn \in QN$, $qn = (\mathcal{STN_M}, \tau, ta)$, $stn_c(V, E, [\tau]_{stn}) \in \mathcal{STN_M}$
Output: $\widetilde{R_q} = qaProcess(c,\, R_q,\, qn)$
 1: $\widetilde{R_q} \leftarrow R_q$
 2: for all entities $v \in V$ do
 3:   if $(\tau_{ta}(c,v) \geqq \kappa_{\tau_{ta}}) \wedge \exists f \in F_v \wedge f.rql \geqq cf.rql$ then
 4:     for all resources $r \in \widetilde{R_q}$ do
 5:       $r_G, r_Y, r_R \leftarrow 0$
 6:       for all policies $p \in P_f \mid p.belongsTo \in A_{cf}$ do
 7:         if $a_{pol}(p,r)$ then
 8:           $r.score \leftarrow r.score + \lambda(v)\tau_{ta}(c,v)s_{pol}(p,r)$
 9:           if $c_{pol}(p,r) = GREEN$ then
10:             $r_G \leftarrow r_G + \tau_{ta}(c,v)$
11:           end if {Similarly for $r_Y$ and $r_R$}
12:           $r.pol \leftarrow r.pol \cup \{p\}$
13:         end if
14:       end for
15:       $r.color \leftarrow c(r_G,\, r_Y,\, r_R)$
16:     end for
17:   end if
18: end for
19: return $\widetilde{R_q}$
```

Before describing Algorithm 1, let us define a scoring function $s_{pol}(p,r) : \mathcal{P} \times \mathcal{R} \rightarrow \mathbb{R}$. If $a_{pol}(p,r) = false$ then $s_{pol}(p,r) = \perp$, i.e. the function s_{pol} is not defined; otherwise, if $a_{pol}(p,r) = true$ then we define the function s_{pol} according to Formula 4 ($p.belongTo = a$, $a \in \mathcal{A}$).

$$s_{pol}(p,r) = imp_p w_a \begin{cases} 1, & \text{if } p \in P^B \cup P^C \wedge p.lev \in \{T, P\} \\ \text{-}1, & \text{if } p \in P^B \cup P^C \wedge p.lev \in \{D, DP\} \\ \frac{|r|_p}{|R_q|_p}, & \text{if } p \in P^P \wedge p.lev \in \{P\} \\ \frac{-|r|_p}{|R_q|_p}, & \text{if } p \in P^P \wedge p.lev \in \{DP\} \end{cases} \tag{4}$$

Furthermore, we define a function $c_{pol} : \mathcal{P} \times \mathcal{R} \rightarrow \{GREY,\, GREEN,\, RED\}$, which qualifies every successful application of the QA policy p to the resource r with the $c_{pol}(p,r)$ assigned according to Formula 5.

$$c_{pol}(p,r) = imp_p w_a \begin{cases} GREY, & \text{if } p.lev \in \{P, DP\} \\ GREEN, & \text{if } p.lev = T \\ RED, & \text{if } p.lev = D \end{cases} \tag{5}$$

In Line 3 of Algorithm 1, $\forall v \in V$, if (1) $\tau_{ta}(c,v)$ is above the given threshold $\kappa_{\tau_{ta}}$ and (2) there exists a QA profile $f \in F_v$ with the QR level at least equal to the QR level of the consumer's QA profile cf, the policies P_f, which belong to the QA analyses A_{cf}, are tried to be applied to the resources $r \in R_q$ in Lines 6 – 14 .

Every resource r holds in variables r_G, r_Y, r_R (initiated in Line 5 in Algorithm 1) the number of GREEN, YELLOW, and RED colors as the results of the function c_{pol}. If the policy p can be successfully applied (Line 7), the result of the function s_{pol} mitigated by $\tau_{ta}(c,v)$ is computed and added to $r.score$ (Line 8); $\lambda(v) \in \mathbb{R}^+$ defines a bonus for the consumer's own QA policies, thus, $\lambda(c) > 1$ and $\lambda(x) = 1$, $\forall x \neq c$. Consequently, in Lines 9 – 11, the function c_{pol} is computed and the appropriate color counter (r_G, r_Y, r_R) is increased.

In Line 15, a function $c : \mathcal{R} \rightarrow \{\, GREY,\, GREEN,\, RED,\, YELLOW\}$ computes the QA color for the resource r according to Formula 6 by comparing the variables r_G, r_Y, and r_R.

$$c(r_G, r_Y, r_R) = \begin{cases} \text{GREY}, & \text{if } (r_R = r_Y = r_G = 0) \\ \text{GREEN}, & \text{if } (r_R = r_Y = 0) \wedge (r_G > 0) \\ \text{RED}, & \text{if } (r_R > 0) \wedge (r_R \geqq (r_G + r_Y)) \\ \text{YELLOW}, & \text{otherwise} \end{cases} \tag{6}$$

4.7 Trust Evolution

Since the trust among people in real world evolves during the time, it is natural to implement trust adjustment based on the consumer's feedback. After the QA process returns the resources $\widetilde{R_q}$ to the information consumer c, he can rate how much helpful the resource $r \in \widetilde{R_q}$ is by specifying a *satisfaction level* as the result of the function $sl : \mathcal{V} \times \mathcal{R} \rightarrow \{1,2,3,4\}$: 1 = "Resource r is exactly what I was looking for", 2 = "Resource r is still acceptable as a result", 3 = "Resource r is not what I was looking for", 4 = "Resource r is completely unacceptable". We hypothesize that the precision of the results of the QA process can be increased by adjusting (based on the satisfaction level of the consumer) the trust value of the consumer in his neighbors.

Walter et al. present in [48] a model of a trust-based recommendation system using a social network. After a recommendation from an agent v is deliverer to a consumer c, the trust value changes on all trust relations along the trust relation paths from the consumer c to the recommending agent V. We argue that this approach has security issues – one can change the other's trust values based on his dissatisfaction with the resource – and, more importantly, it is in contrary with Definition 3.1 according to which trust is a personal opinion of every agent.

Therefore, based on the satisfaction $sl(c,r)$, we compute $\Delta\tau_{c,x}^{t,t+1}$, influencing the trust value $\tau^{t+1}(c,x)$ in time $t+1$, according to Formula 7; $\Delta\tau_{c,x}^{t,t+1}$ is computed for the trust relations between the information consumer c and his neighbors (denoted as $n_p(c)$, $x \in n_p(c)$) who are on (some) trust relation path between the consumer c and an entity $v \in \mathcal{V}$, for which $\exists p \in P_v : a(p,r) = 1$.

$$\Delta\tau_{c,x}^{t,t+1} = (\tau_{ta}(x,v))^2(\widehat{\tau^{t+1}(c,x)} - \tau^t(c,x)) \qquad (7)$$

The computation of the trust value $\widehat{\tau^{t+1}(c,x)}$ in Formula 7 is using the auxiliary function $\overline{\tau^{t+1}(c,x)}$ defined in Formula 10[25]; $\overline{\tau^t(c,x)}$ in Formula 10 is obtained from $\tau^t(c,x)$ according to Formula 9. The function *slb* in Formula 10 defines the bonus for the given satisfaction level $sl(c,r)$; the values for the bonuses were set experimentally as: $slb(1) = 1$, $slb(2) = 0.75$, $slb(3) = -0.25)$, $slb(4) = -1$. After the computation of $\overline{\tau^{t+1}(c,x)}$, we apply the backward mapping to $\widehat{\tau^{t+1}(c,x)}$ according to Formula 8, because $\overline{\tau^{t+1}(c,x)} \in [-1,1]$.

$$\widehat{\tau^{t+1}(c,x)} = \frac{1+\overline{\tau^{t+1}(c,x)}}{2} \qquad (8)$$

$$\overline{\tau^t(c,x)} = 2\tau^t(c,x) - 1 \qquad (9)$$

$$\overline{\tau^{t+1}(c,x)} = \begin{cases} \gamma\overline{\tau^t(c,x)} + (1-\gamma)slb(sl(c,r)), & sl(c,r) \leqq 2 \\ (1-\gamma)\overline{\tau^t(c,x)} + \gamma slb(sl(c,r)), & sl(c,r) \geqq 3 \end{cases} \qquad (10)$$

Formula 10 causes, for $0.5 < \gamma < 1.0$, "a slow-positive and a fast-negative effect which usually is a desired property for the dynamics of trust: trust is supposed to build up slowly, but to be torn down quickly" [48]. In our trust model, *gamma* was experimentally set to 0.6.

To further amplify the slow-positive and a fast-negative effect, if $\tau^t(c,x) < K$, we set $\gamma = 0.95$ and lower γ for 0.05 points per the following time unit, finally reaching the original $\gamma = 0.6$. As a result, the revitalization of such entity is even harder, with even bigger penalization for the immediate consequent negative ratings (see Figure 3).

[25]We are inspired by Formula 4 in [48]

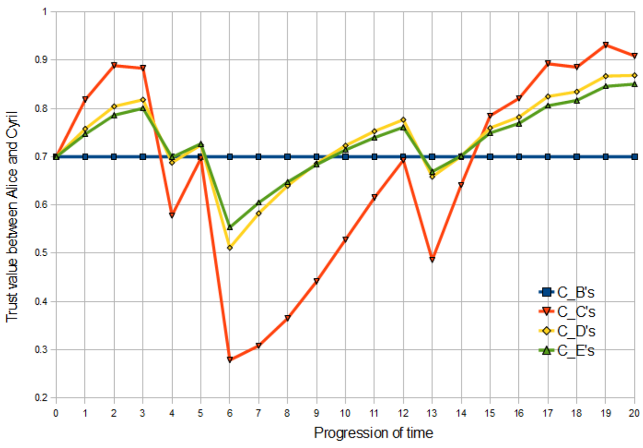

Figure 3: Trust evolution

4.7.1 *Sample Trust Evolution*

Figure 3 illustrates the sample evolution of trust $\tau_{Alice,Cyril}^t$ in time t, $0 \leqq t \leqq 20$, based on the Alice's ratings of the resources r_t to which only QA policies of Bob (in Line $C\_B's$), Cyril's ($C\_C's$), Dave's ($C\_D's$), or Emil's ($C\_E's$) were applied. Let $\vec{sl} = (sl_0, \dots, sl_t, \dots, sl_{19})$, $0 \leqq t \leqq 19$, denotes the vector of satisfaction levels (ratings) Alice assigns to the resources r_t in time t leading to the trust value change in time t+1. We set $\vec{sl} = (1,1,2,3,2,4,2,2,2,2,2,2,3,2,1,2,1,2,1,2)$.

In Figure 3, the x-axis depicts the time units t, the y-axis the trust value $\tau_{Alice,Cyril}^t$ in the given time t. For example, Line $C\_D's$ for $t = 6$ depicts the trust value $\tau_{Alice,Cyril}^6$: $\tau_{Alice,Cyril}^6 < \tau_{Alice,Cyril}^5$, because Alice rated the resource to which the Dave's QA policies were applied with the satisfaction level $\vec{sl}[5] = 4$.

We can see in Figure 3 the slow-positive and fast-negative effect of the trust value change caused by the chosen $\gamma = 0.6$. The consequences of the satisfaction levels are higher (Line $C\_C's$) the closer the entity v whose QA policies were applied to the resource r_t is to *Cyril* (the consequence of $(\tau_{ta}(Cyril,v))^2$ in Formula 7)[26]. In case of Line $C\_B's$, $\tau_{Alice,Cyril}^t$ is not changed for any time t, because Cyril is not on any considered trust relation pats between Alice and Bob. Furthermore, between times $t = 6$ and $t = 12$, the trust value $\tau_{Alice,Cyril}^t$ for Line $C\_C's$ is shaped differently than for Lines $C\_D's$ and $C\_E's$, because the trust value $\tau_{Alice,Cyril}^t$ for Line $C\_C's$ fall below trust level K and, thus, the revitalization is harder.

5. ALGORITHMS FOR DERIVING TRUST

This section overviews the algorithms for computing derived trust value $\tau_{ta}(u,v)$ between arbitrary two entities $u,v \in \mathcal{V}$ in the QA social network $qn(\mathcal{STN_M},\tau,ta)$; the computed $\tau_{ta}(u,v)$ is then used in the QA process to decide, whether the entity u should incorporate the QA policies of the entity v during its QA process described in Algorithm 1. The proposed algorithm ReWA in Subsection5.4 presents one of the contributions of this paper.

The set of chosen algorithms, $\mathcal{ALG}$, is equal to {*SPathMin* (*miSP*), *SPathMulti* (*muSP*), *TidalTrust* (*tt*), *ReWA*

[26]We suppose that the algorithm ta is ReWA (see Section 5)

$(rw)\}$.

Ziegler and Laursen [51] extensively examined the propagation of trust and distrust in social networks and present categorization of trust metrics – they distinguish global and local trust metrics. Local trust metrics comprehend trust as a subjective opinion of the particular entity (information consumer Alice in our scenario). On the other hand, global trust metrics, such as [31, 46] violate our assumption about subjectiveness of the trust (according to Definition 3.1) the information consumer has in the particular entity; global trust metrics compute the *reputation* of the particular entity in the QA social network based on the average of trust estimations the others have in that entity. Therefore, we focus on the local trust metrics (algorithms).

Apart from that, it is questionable if the trust should be transitive, therefore, if the existence of a trust relation between entities A and B and B and C enables to derive a trust value between A and C. As stated in [48] "there have been fierce discussions in the literature whether or not trust is transitive; from the perspective of network security (where transitivity would, for example, imply accepting a key with no further verification based on trust) or formal logics (where transitivity would, for example, imply updating a belief store with incorrect, impossible, or inconsistent statements) it may make sense to assume that trust is not transitive [29, 15, 27]". On the other hand, Golbeck [24] and Guha et al. [25] show experimentally that trust in networks similar to the QA social network is transitive and may propagate along the trust relations.

In contrary to Golbeck's approach, Guha et al. suggest appropriate trust discounting with the increasing lengths of trust relation paths. We agree with this approach, since it realistically models the concept of trust – a user trusts more to his direct friends than to the friends of his friends.

All our considered trust algorithm are local trust metrics satisfying the *personalization* and *asymmetry* properties of trust (already defined in [24]). The trust is considered as transitive based on the discussion above. Properties TP1 and TP2 are discussed for each algorithm separately:

TP1 Transitivity Decay: Is there a trust degradation with the increasing trust relation path lengths?

TP2 Composability: If there exist more trust relation paths between entities A and B, is the derived trust value based on all these paths?

5.1 SPathMin Algorithm

The trust algorithm *SPathMin* is represented by a function τ_{miSP}, $miSP \in \mathcal{ALG}$, computing derived trust value as a maximum of the minimums of the weights on all distinct $\pi_{u,v}$, i.e. according to Formula 11.

$$\tau_{miSP}(u,v) = max_{i=1}^{\#\pi_{u,v}}\{min_{j=1}^{|\pi^i_{u,v}|}\{\tau(x_j,x_{j+1}) \mid (x_j,x_{j+1}) \in \pi^i_{u,v}\}\} \tag{11}$$

SPathMin trust algorithm does not satisfy TP1 and TP2; there is no trust discounting with the increasing $|\pi_{u,v}|$ – the shorter paths are not preferred – and only one path is decisive for the resulting τ_{miSP}. The time complexity is $O(|V|log|V| + |E|)$, since this is the time complexity of the Dijkstra algorithm using heap for storing vertices. Since the Dijkstra algorithm is operating with non-negative weights on the edges, during the computation of the algorithm, we try to minimize the resulting $1 - \tau_{miSP}(u,v)$.

5.2 SPathMulti Algorithm

The trust algorithm *SPathMulti* is represented by a function τ_{muSP}, $muSP \in \mathcal{ALG}$, computing derived trust value as a maximum (over all distinct $\pi_{u,v}$) of the multiplications of the weights on the particular path (see Formula 12).

$$\tau_{muSP}(u,v) = max_{i=1}^{\#\pi_{u,v}}\{\prod_{j=1}^{|\pi^i_{u,v}|}\{\tau(x_j,x_{j+1}) \mid (x_j,x_{j+1}) \in \pi^i_{u,v}\}\} \tag{12}$$

SPathMulti trust algorithm does not satisfy TP2 (only one path is decisive for the resulting τ_{miSP}), however, TP1 is satisfied (ensured by multiplication of the trust values along the path). The time complexity is $O(|V|log|V| + |E|)$, for the same reason as in SPathMin.

5.3 TidalTrust Algorithm

The algorithm *TidalTrust* is in detail described in [24], we depict here only the core formula for computing the trust value $\tau_{tt}(u,v)$, $tt \in \mathcal{ALG}$, as a *restricted* weighted average over distinct trust relation paths between u and v (see Formula 13, where $n(u) = \{x \in V : (u,x) \in E\}$). In [24], the entity u is called *source* and v *sink*.

$$\tau_{tt}(u,v) = \begin{cases} \tau(u,v) & (u,v) \in E \\ \frac{\sum_{x \in n(u) \mid \tau_{u,x} \geqq max} \tau_{u,x}\tau_{tt}(x,v)}{\sum_{x \in n(u) \mid \tau_{u,x} \geqq max} \tau_{u,x}} & \text{otherwise} \end{cases} \tag{13}$$

Since entities are more likely to connect with entities they trust highly [24], we restrict the weighted average by defining the threshold $max \in [1,9]$, so that a trust relation path $\pi_{u,v}$, with all trust relation weights higher than max, can still be found.

If there are more trust relation paths from the source u to the sink v, only the shortest paths required to connect u and v are considered in the computation of the restricted weighted average in Formula 13. This approach preserves the benefits of shorter path lengths [24].

The algorithm does not satisfy TP1 (there is no trust decay), however, it satisfies TP2 (composability is ensured by the weighted average).

We implemented Tidal trust algorithm in a more effective way for the purpose of the WQA model. In the original version [24], the time complexity is $O(|V| + |E|)$. Since we need to call the algorithm $|V_{nbors}|$ times, where V_{nbors} is the set of entities having $|\pi(u,x)| < \kappa_{tt}$[27], we can assume $O(|V|(|V|+|E|))$[28]. Therefore, we propose a better approach computing the trust value at once for all entities V_{nbors} having the time complexity $O(|V| + |V||E|) = O(|V||E|)$.

5.4 ReWA Algorithm

The algorithm *ReWA* computes a derived trust value $\tau_{rw}(c,v)$, $rw \in \mathcal{ALG}$, in social trust network $stn_c(V,E,[\tau])$ in three steps:

S1 Obtaining the set $V_{sel} \subseteq V$ of entities by computing $\tau_{muSP}(c,v')$, $v' \in V$, and selecting $\kappa \in \mathbb{N}$ entities with the highest $\tau_{muSP}(c,v')$.

[27] $\kappa_{tt} = 4$ in our implementation of TidalTrust.

[28] Since the QA social network has the properties of small world networks, i.e. the connectance is high

S2 Labeling, $\forall x \in V_{sel}$, all entities $v \in V_{sel}$ in the $stn_c(V_{sel} \cup \{c\}, E_{|V^c_{sel} \times V_{sel}}, [\tau]_{|V^c_{sel} \times V_{sel}})$, $V^c_{sel} = V_{sel} \cup \{c\}$, with a label $l_x \in L$ if $\exists \pi_{c,x}$, so that $v \in \pi_{c,x}$. The result of the algorithm is $\mathcal{L}$, the set of sets of labels $L_v \in \mathcal{L}$, L_v holds all the labels for the entity $v \in V_{sel}$. See Algorithm 2 for details, the algorithm is called as $getLabeledNetwork(c, V_{sel})$.

S3 Processing, for each label $l_x \in L$, all the trust relation path $\pi_{c,x}$, $x \in V_{sel}$, which are using only the trust relations with the label l_x and computing the resulting $\tau_{rw}(c,x)$ as the weighted average. See Algorithm 3 for details, the algorithm is called as $computeAverage(c, l_x)$, $\forall l_x \in L$.

Algorithm 2 S2: Labeling Network

```
Input: $v \in V^c_{sel}$, $V_{sel} \subseteq V$
Output: $\mathcal{L} = getLabeledNetwork(v, V_{sel})$
 1: if $c_v = BLACK$ then
 2:    return $L_v$
 3: else if $c_v = GREY$ then
 4:    return $\emptyset$
 5: else if $c_v = WHITE$ then
 6:    $c_v \leftarrow GREY$
 7:    $L_v \leftarrow \emptyset$
 8:    for all User $n \in neighbors(v)$ do
 9:       $L'_v \leftarrow getLabeledNetwork(n, V_{sel})$
10:       for all Labels $l_x \in L'_v$ do
11:          $neighbors_{l_x}(v) \leftarrow neighbors_{l_x}(v) \cup x$
12:       end for
13:       $L_v \leftarrow L_v \cup L'_v$
14:    end for
15:    $L_v \leftarrow L_v \cup \{l_v\}$
16:    $c_v \leftarrow BLACK$
17:    return $L_v$
18: end if
```

In Algorithm 2, c_v denotes the color of the entity v – at the beginning of the algorithm, all entities $v \in V_{sel}$ are $WHITE$, during the processing of the entity v (Lines 7 – 15), $c_v = GREY$, and after the processing (Line 16), $c_v = BLACK$. In Line 8, $neighbors(v) = \{n \mid (v,x) \in E_{|V^c_{sel} \times V_{sel}}\}$. In Line 11, $neighbors_{l_x}(v) \subseteq neighbors(v)$ contains the set of relations from v to the entities with the label l_x.

Since $\tau_{muSP}(c,v) >= \tau_{rw}(c,v)$, $\forall v \in V_{sel}$, the algorithm SPathMulti presents a good approximation of the ReWA algorithm. According to the experiments, κ determining $|V_{sel}|$ is set to 1000.

The algorithm satisfies TP1 – Line 7 in Algorithm 3 depicts the trust decay $\tau_{v,n}$. Since more trust relation paths $\pi_{c,v}$, $v \in V$, form the resulting $\tau_{rw}(c,v)$ (Line 11 in Algorithm 3), TP2 is satisfied as well.

The time complexity of the individual steps is: S1 = $O(|V| log|V| + |E|)$, S2 = $O(E_{|V^c_{sel} \times V_{sel}})$, and S3 = $O(|V_{sel}| \, |E_{|V^c_{sel} \times V_{sel}}|)$, if implemented reasonably by storing the already computed τ_{rw}. As a result, the time complexity of the algorithm is $O(|V| log|V| + |E|)$ if $|V_{sel}||E_{|V^c_{sel} \times V_{sel}}| < max\{|E|, |V| log|V|\})$, and $O(|V_{sel}||E_{|V^c_{sel} \times V_{sel}}|)$ otherwise.

5.4.1 Correctness of the Algorithm ReWA

To proof the correctness of the algorithm ReWA, we have to show that, in S2, Algorithm 2 (1) finishes after a finite

Algorithm 3 S3: Computing Weighted Average

```
Input: $v \in V^c_{sel}$, $l_x \in L$
Output: $\tau_{rw}(c,v) = computeAverage(v, l_x)$
 1: if $x = l_x$ then
 2:    return 1
 3: else
 4:    $sum_\tau \leftarrow 0$
 5:    $sum_W \leftarrow 0$
 6:    for all User $n \in neighbors_{l_x}(v)$ do
 7:       $sum_\tau \leftarrow sum_\tau + (\tau_{v,n})^2 computeAverage(n, l_x)$
 8:       $sum_W \leftarrow sum_W + \tau_{v,n}$
 9:    end for
10:    if $sum_W > 0$ then
11:       return $\frac{sum_\tau}{sum_W}$
12:    else
13:       return 0
14:    end if
15: end if
```

amount of steps and (2) always returns the correct set of labels $\mathcal{L}$. To show (1), every entity's color is WHITE at the beginning, changed to GREY when the given entity is being processed for the first time and, finally, reaching BLACK when all his neighbors are already BLACK. This coloring ensures that no entity is processed twice.

To proof (2), let us suppose that there is a situation which contravene (2), thus, in the computation of Algorithm 2, L'_x obtains L_y in Line 9, $y \in neighbor(x)$, however, at the end of the computation, there exists a label $l \in L_y$, so that $l \notin L_x$, $x \in V^c_{sel}$, $y \in V_{sel}$. If the entity y was $WHITE$ at the time the entity x obtained L_y, than, the algorithm returned L_y only when y's color changed to $BLACK$, which means that L_y had to be final. If the entity y was $GREY$, we approached the entity y for the second time on some path $\pi_{c,v}$, $v \in V_{sel}$, thus, the cycle was about to occur; as a result, we definitely did not want to propagate L_y back over the entity x and did not consider (x,y). Finally, If the entity y was $BLACK$, then we reached the already fully processed node, L_y had to be final. Therefore, in all cases, there cannot be any $l \in L_y$ so that $l \notin L_x$ in the given situation. Since x and y were chosen arbitrary, the proof is done.

5.5 Summary of the Trust Algorithms

Table 4 summarizes the differences between the trust algorithms $ta \in \mathcal{ALG}$ by showing the satisfaction of TP1 and TP2 properties and the time complexity.

Furthermore, Table 5 illustrates the use of the trust algorithms $ta \in \mathcal{ALG}$ to derive $\tau_{ta}(Alice, Emil)$, where Alice and Emil are two entities in the social trust network depicted in Figure 1. As we can see, the absence of the trust decay (TP1) leads to higher values for $\tau_{ta}(Alice, Emil)$ for the algorithms SPathMin and TidalTrust; $\tau_{rw}(Alice, Emil) < \tau_{muSP}(Alice, Emil)$, because the algorithm SPathMulti calculates only with the best trust relation path $\pi_{Alice,Emil}$, however, the algorithm ReWA incorporates both trust relation paths between Alice and Emil (over Bob and Cyril), which reflects the reality more appropriate.

6. EVALUATION

In our evaluation, a consumer $c \in \mathcal{V}$ constructs a query $q \in \mathcal{Q}$ which yields to the ordered set of resources $R_q \subseteq$

Table 4: Trust Properties and Time Complexities of the Trust Algorithms $ta \in \mathcal{ALG}$

Trust algorithm	TP1	TP2	O(n)
$SPathMin$	NO	NO	$O(\|V\|log\|V\| + \|E\|)$
$SPathMulti$	YES	NO	$O(\|V\|log\|V\| + \|E\|)$
$TidalTrust$	NO	YES	$O(\|V\|\|E\|)$
$ReWA$	YES	YES	See Subsection 5.4

Table 5: Derived Trust Value $\tau_{ta}(Alice, Emil)$

Trust algorithm	$\tau_{\mathbf{ta}}(\mathbf{Alice}, \mathbf{Emil})$
$SPathMin$	0.7
$SPathMulti$	0.57
$TidalTrust$	0.9
$ReWA$	0.51

$\mathcal{R}$. Consequently, R_q is an input to the QA process (see Algorithm 1) resulting in the ordered set $\widetilde{R_q}$.

As part of the QA process, we use the trust algorithms $ta \in \mathcal{ALG}$ to select the set of QA policies tried to be applied to the resources R_q. In order to compare the suitability of these trust algorithms, we define a set of properties P1 – P4 observed for each algorithm:

P 1 Accuracy: The algorithm should be as much accurate as possible – the results of the computed ranking must be as close as possible to the results of the ideal ranking (see Subsection 6.2).

P 2 Robustness: Since anybody can join the social trust network, the trust algorithm must be resistant against malicious entities and entities, who simply provide misleading information. We call both these entities as *bad entities*.

P 3 Complexity: The algorithm must not be too complex, the information consumer needs to get the response (the ordered set of resources $\widetilde{R_q}$) in the usual response time on the Web.

P 4 Scalability: The algorithm should scale well with the increasing number of entities in the social trust network.

It is obvious that it is hard to fulfill all properties P1 – P4, for example, the more accurate the results are, the more complex the algorithm tends to be.

6.1 Initialization of the Model

For the evaluation of the QA process utilizing the particular trust algorithm $ta \in \mathcal{ALG}$, we examined 1000 queries $q \in Q_I \subseteq \mathcal{Q}$. The resources R_q, $|R_q| = 10$, in the answer for each $q \in Q_I$ are chosen randomly from a pool of 1000 resources $R_I \subseteq \mathcal{R}$[29]; we suppose that all resources are from the financial domain. The information consumer $c \in V_I$ is chosen randomly from a pool of $|V_I|$ users[30], $V_I \subseteq U$, and the rest of the users V_I form the social trust network $stn(V, E, [\tau]_{stn}) \in STN$, $V \subseteq V_I$, $E \subseteq \{V_I \times V_I\}$. All the assignments above are performed using random selection, with replacement (i.e. the covariance between two consumers and two sets of resources selected for two different queries is zero), from uniform distributions over the sets R_I and V_I.

The QA process incorporates one QA analysis $a \in A$ – the analysis of data provenance – which defines a single template $t \in T_a \subseteq \mathcal{T}$, $t = (B, \{TRUST, DISTRUST\}, \{([r], \{Prop_i\}$[31], $\{Val_j\}$[32]$)\})$, $1 \leqq i \leqq |Z^t_{P_1}|$, $1 \leqq j \leqq |Z^t_{O_1}|$. We set $|Z^t_{P_1}| = 10$ and $|Z^t_{O_1}| = 20$. Every resource $r \in R_q$ is associated with RDF triples $r.triple_x \in \mathcal{Z}_{\mathcal{T}}$, $1 \leqq x \leqq |Z^t_{P_1}|$, having each $w \in Z^{t_1}_P$ as the RDF property and the chosen $v \in Z^t_{O_1}$ as the property value; v is chosen for each triple at random with replacement from uniform distribution over $Z^t_{O_1}$.

We obtained two types of social networks $sn_a(V_I, E_a)$ and $sn_b(V_I, E_b)$, $E_a, E_b \subseteq \{V_I \times V_I\}$, which form the basis for our social trust networks used in the evaluation.

The social network sn_a is a social network expressed using FOAF ontology and downloaded from the Web; we extracted instances of the class `foaf:Person` and instances of the predicate `foaf:knows` between them to obtain the users and the relations for our social network sn_a. To achieve this, we implemented a simple crawler, starting from the FOAF profile of Tim Berners-Lee[33] and extracting $|V_I| = 2887$ users with the average number of 8.47 relations between them from the social network behind the Advogato web site[34].

The social network sn_b is generated according to the approach introduced in [5], which realistically approximates real world social networks. Starting from an initial seed of randomly interconnected users (in our case 50 users), the rest of the users are progressively connected with the users in the initial seed of users and already processed users; the probability that a user will be connected with a particular user u depends on the number of relations the user u already has (the more relations, the higher probability). Furthermore, there is a 20% chance that the user u creates a backward trust relation to the newly added user. The total number of users in sn_b is $|V_I| = 10000$. The number of acquaintances each user in the social network sn_b has is a random number chosen from a normal distribution with mean $\mu_A = 8$ and standard deviation $\sigma_A = 1$, the values for μ_A and σ_A are inspired by the properties of the downloaded social network sn_a.

Based on the obtained social networks, we define social trust networks $stn_a(V_I, E_a, [\tau]_{stn_a})$ and $stn_b(V_I, E_b, [\tau]_{stn_b})$. In both social trust networks, the trust relations are weighted using trust levels TRUSTED_HIGHLY, TRUSTED, KNOWN, and DISTRUSTED according to randomly chosen values of the function τ.

The number of QA policies a user $u \in V_I$ in a QA social network defines is a random number chosen from a normal distribution with the chosen mean $\mu_P = 7$ and standard deviation $\sigma_P = 1$. All the generated policies are policies $p^t \in \mathcal{P}$, $p^t.belongsTo = a$, $imp_{p^t} = 1$, $t \in T_a$. For each such generated policy p^t, $p^t.tle_1.pred$, respectively $p^t.tle_1.obj$, are selected at random with replacement from uniform distributions over $Z^t_{P_1}$, respectively $Z^t_{O_1}$. To deduce $p^t.lev$, we

[29]We have tried various $|R_q|$ with similar resulting graphs.

[30]For simplicity, we do not consider groups or organizations.

[31]The particular names of admissible RDF properties are not important, we denote them as $Prop_1 \ldots Prop_n$, $n = |Z^t_{P_1}|$.

[32]The particular names of the objects in RDF triples are not important, we denote them as $Val_1 \ldots Val_m$, $m = |Z^t_{O_1}|$.

[33]http://www.advogato.org/person/timbl/foaf.rdf

[34]Advogato.com, a free software developer's advocate and a research testbed for Advogato trust algorithm [35]

Table 6: Probabilities for Generated Trust Levels

$\mathbf{v \in Z_O^{t_1}}$	**Trust prob**	**Distrust prob**
$Val_1 - Val_5$	$60 * wav_u^\tau$	$40 * (1 - wav_u^\tau)$
$Val_6 - Val_{10}$	$58 * wav_u^\tau$	$42 * (1 - wav_u^\tau)$
$Val_{11} - Val_{15}$	$56 * wav_u^\tau$	$44 * (1 - wav_u^\tau)$
$Val_{16} - Val_{20}$	$54 * wav_u^\tau$	$46 * (1 - wav_u^\tau)$
$Val_{21} - Val_{25}$	$52 * wav_u^\tau$	$48 * (1 - wav_u^\tau)$
$Val_{26} - Val_{30}$	$48 * (1 - wav_u^\tau)$	$52 * wav_u^\tau$
$Val_{31} - Val_{35}$	$46 * (1 - wav_u^\tau)$	$54 * wav_u^\tau$
$Val_{36} - Val_{40}$	$44 * (1 - wav_u^\tau)$	$56 * wav_u^\tau$
$Val_{41} - Val_{45}$	$42 * (1 - wav_u^\tau)$	$58 * wav_u^\tau$
$Val_{46} - Val_{50}$	$40 * (1 - wav_u^\tau)$	$60 * wav_u^\tau$

Table 7: Determination of the Values for the Number of Policies (#Pol), Number of Neighbors (#Nbor), and the trust value τ

Range	#Pol	#Nbor	τ
$\lvert gv\rvert < 0.5\sigma$	7	8	T
$(-1)\sigma < gv \leqq (-0.5)\sigma$	6	7	K
$0.5\sigma \leqq gv < \sigma$	8	9	TH
$(-2)\sigma < gv \leqq (-1)\sigma$	5	6	D
$\sigma \leqq gv < 2\sigma$	9	10	D
$(-3)\sigma < gv \leqq (-2)\sigma$	3	4	K
$2\sigma \leqq gv < 3\sigma$	11	12	TH
$gv \leqq (-3)\sigma$	1	2	K
$gv \geqq 3\sigma$	13	14	TH

introduce an ideal ordering of $Z_{O_1}^t$ values according to a magical oracular that knows general and objective trustworthiness of the values $v \in Z_{O_1}^t$. The ideal ordering is governed by $\gamma : Z_{O_1}^t \to 1, 2, \dots, |Z_{O_1}^t|$, ranking the $Z_{O_1}^t$ values from the most trusted to the most distrusted. For simplicity, we choose that $\gamma(Val_i) = i$, $1 \leqq i \leqq |Z_{O_1}^t|$, i.e. Val_i is ranked to the i-th position in the ideal ordering. We suppose that Val_i values for $1 \leqq i \leqq (|Val_i|/2)$ are considered trustworthy (to some extent) and Val_i for $(|Val_i|/2) < i \leqq |Val_i|$ are distrusted.

Table 6 describes how probable is that a user u will choose the right $p^t.lev$ for the particular $v \in Z_{O_1}^t$, i.e. $TRUST$ for the first half of the values and $DISTRUST$ for the rest of the values $Z_{O_1}^t$ according to γ. The trust and distrust probabilities in Table 6 are based on the ideal ordering (the better the rank according to γ, the bigger the probability that the user chooses the right $p^t.lev$) and meliorated by the weighted average wav_u^τ of the trust values $\tau_{x,u}$, $(x, u) \in E$.

Example 6.1. *Suppose that $wav_u^\tau = 6$, further, the user u defines the policy p^t, where $p^t.tle_1.obj = Val_{32}$. According to Table 6, the probability of choosing the trust level $TRUST$, $prob_t$, is equal to $t/t + d$, where $t = 46 * (1 - 0.7)$ and $d = 54 * 0.7$ (Line 7 in Table 6). Therefore $prob_t \approx 0.27$ and $prob_d \approx 0.73$, where $prob_d$ is the probability of choosing trust level $DISTRUST$. The resulting probabilities correspond with the fact that $wav_u^\tau > 5$ – the user u is more trustworthy then untrustworthy (as seen by his neighbors), and $\gamma(Val_{32}) = 32$, thus, the selected $p^t.tle_1.obj$ is in the second half according to the function γ, where the distrusted values occur.*

Table 7 depicts how the values for the number of policies, number of neighbors (only in stn_b), and trust values are chosen according to the randomly generated value gv of the normal distribution and it's distance from the mean measured in the multiples of the standard deviation σ.

6.2 Metrics

Let us define a permutation $\phi : \widetilde{R_q} \to \widehat{R_q}$, mapping the ordered set of resources $\widetilde{R_q}$ to $\widehat{R_q}$; ϕ is applied to rank the set $\widetilde{R_q}$ according to the decreasing QA scores $r_i.score$ of the resources $r_i \in \widetilde{R_q}$, $1 \leqq i \leqq |\widehat{R_q}|$. In order to compare the accuracy of trust algorithms, we compare for every query $q \in Q_I$ the ordered set $\widehat{R_q}$ with the ordered set $\widehat{IR_q}$. In $\widehat{IR_q}$, the resources $r_i \in R_q$, $1 \leqq i \leqq |R_q|$, are ordered according to the decreasing values of the function $iScore : \mathcal{R} \to \mathbb{Z}$ computed according to Formula 14, where $\kappa : Z_{O_1}^t \to \{-1, 1\}$ is defined in Formula 15. Further, we define a permutation $\chi : \widetilde{R_q} \to \widehat{IR_q}$ mapping the ordered set of resources $\widetilde{R_q}$ to $\widehat{IR_q}$. Obviously, $\widehat{IR_q}$ differs for different $q \in Q_I$, therefore, we call the ranking according to $\widehat{IR_q}$ as an *ideal ranking* and the ranking according to $\widehat{R_q}$ as *computed ranking* for some $q \in Q_I$.

$$iScore(r) = \sum_{j=1}^{|Z_{P_1}^t|} \kappa(r.triple_j.obj) \tag{14}$$

$$\kappa(x) = \begin{cases} 1, & \text{if } \gamma(x) \leqq (|Z_{O_1}^t|/2) \\ -1, & \text{otherwise} \end{cases} \tag{15}$$

Suppose that for the particular $q \in Q_I$, $\chi(r) = a_q$ and $\phi(r) = b_q$ for the particular $r \in R_q$. Then, to measure the accuracy of the trust algorithm in QA process, we define the set of *average distance metrics* δ_x, $1 \leqq x \leqq |R_q|$, according to Formula 16. The expression a_q^x in Formula 16 denotes the rank x of the resource $a_q \in \widehat{IR_q}$ according to the decreasing results of the function *iScore*; similarly, b_q^y denotes the rank y of the resource $b_q \in \widehat{R_q}$ according to the decreasing $b_q.score$ of the resources $r \in R_q$. The metrics δ_x express the average relative distance of $\phi(r)^x$ and $\chi(r)^y$ measured $\forall q \in Q_I$ for resources $r \in R_q$.

$$\delta_x = (\sum_{q=1}^{|Q_I|} a_q^x - b_q^y)/|Q_I| \tag{16}$$

The set of metrics δ_x, $1 \leqq x \leqq |R_q|$, is further used to compute the *average global distance metric* Δ according to Formula 17. The absolute value in Formula 17 occurs because we do not want to distinguish positive and negative δ_x values – they would falsify the metric Δ.

$$\Delta = \sum_{x=1}^{|R_q|} |\delta_x| \tag{17}$$

Furthermore, we define the *applied policies metric* ($\#Pol$) for computing the average number of successfully applied QA policies per resource, *users reached metric* ($\#Users$) for computing the average number of users visited during the execution of the particular trust algorithm, and *time metric* ($Time$), with results in seconds, computing the average time necessary to perform one query $q \in Q_I$.

6.3 Simulation & Results

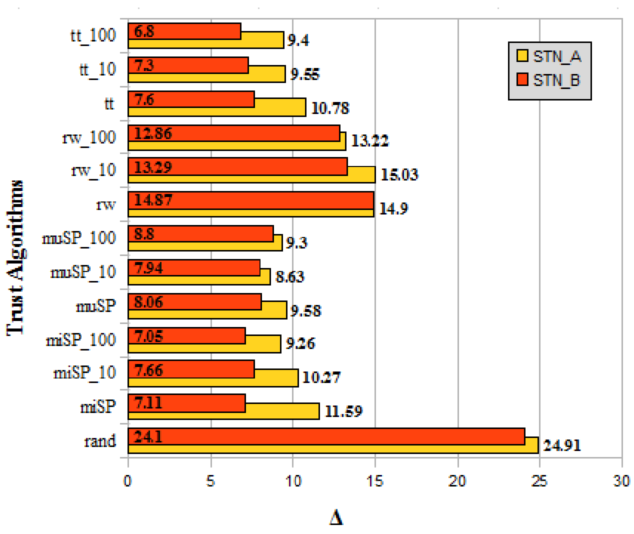

Figure 4: Accuracy – the metric Δ

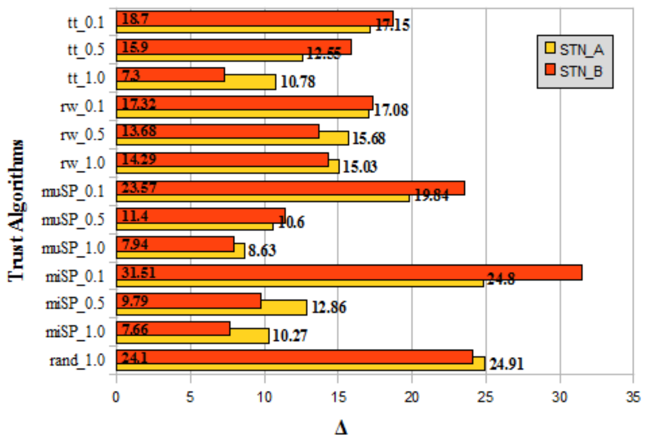

Figure 5: Robustness – the metric Δ

In Subsections 6.3.1 – 6.3.4, we compare the accuracy, robustness, complexity, and scalability of the trust algorithms $ta \in \mathcal{ALG}$. Subsection 6.3.5 summarizes the results.

In all Figures, yellow (brighter) color represents the results in the social trust network stn_a (real social trust network with $|V_I| = 2887$), whereas orange (darker) color represents the results in stn_b (artificial social network with $|V_I| = 10000$). Apart from the algorithms $ta \in \mathcal{ALG}$, we introduce an algorithm *rand*, a dummy algorithm randomly ranking the resources at its hand, which serves as a reference algorithm.

6.3.1 Accuracy

When measuring *accuracy*, we assume that all users act fairly; thus, they decide whether to trust or distrust the particular $v \in Z^t_{O_1}$ according to their best conscience. Accuracy is based on the results of the metric Δ.

Figure 4 depicts the metric Δ (x-axis, the lower, the better). The y-axis represents the trust algorithms $ta \in \mathcal{ALG}$ probed; if the trust algorithm ta's name is in the form *ta_t*, the trust evolution occurs between times 0 and $t - 1$, and, the depicted metric Δ is computed for τ_{ta} in time t.

As we can see, the algorithms operating in the social trust network stn_b are generally more accurate, which is caused by more users in the network, thus, more QA policies applied (see Figure 7B). Since all users act fairly, there is a direct proportion between the metrics $\#Pol$ and Δ. We can observe that in most cases, for the algorithms rw and tt, the longer the trust evolution, the more accurate the algorithms are. On the other hand, especially in case of the algorithm $muSP$, the trust evolution has rather negative effect, because it lowers the trust value of the information consumer in its neighbors, thus, causing that the algorithm applies less QA policies, which leads to worse accuracy in case of fair users.

6.3.2 Robustness

To measure the *robustness* of the trust algorithms, we again compare the results of the metric Δ. Nevertheless, every user $u \in V_I$, when created or downloaded from the FOAF social network, has the probability $isGood \in [0, 1]$ to act fairly during the creation of his QA policies. If he does not act fairly, we call the user as a *bad user*, who intensionally or unintentionally behaves in an opposite way, i.e. $\forall p^t \in P_u$, $p^t.lev = T$, if $p^t.lev$ was originally D and conversely.

Figure 5 represents the results of the metric Δ (x-axis, the lower, the better) in both social trust network, stn_a and stn_b, with various portions $isGood$ of fair users. As in the previous case, the trust evolution implicitly occurs between times 0 and $t - 1$, and, the depicted metric Δ is computed for τ_{ta} in time t. The y-axis holds the list of trust algorithms probed; the trust algorithm ta's name is in the form *ta_isGood*, where the variable $isGood \in [0, 1]$ represents the portion of fair users.

As we can observe, for $isGood = 0.1$ – 90% of bad users, which is a realistic assumption on the Web – the trust algorithm rw outperforms all the other trust algorithms *ta*_0.1, $ta \in \{miSP, muSP, tt\}$. The average global distance metric for the algorithm $miSP$ with $isGood = 0.1$ is even worse than the result of the algorithm *rand*, because, as depicted in Figure 7B, the amount of $\#Pol$ is the highest for this algorithm leading to the biggest amount of misleading QA policies applied. For the same reason, we can observe that, if $isGood < 0.5$, the accuracy is higher (for the same algorithms) in case of the smaller social trust network stn_a, because of less misleading QA policies collected and applied. For the majority of trust algorithms, the accuracy is increasing with the increasing portion of $isGood$ users; the increase is faster for trust algorithms $miSP$ and $muSP$, because they generally apply more QA policies. Furthermore, with the decreasing portion of $isGood$ users, the effect of trust evolution on the accuracy is increasing.

6.3.3 Complexity

To measure the *complexity* of the trust algorithms, we compare the results of the metric $Time$. Figure 6 represents the results of the metric $Time$ (y-axis, the lower time, the better) in both social trust networks and for all $ta \in \mathcal{ALG}$ (x-axis); we suppose that before measuring the metric $Time$, the trust evolves between times 0 and $t - 1$ and, as in Figure 4, all users behave fairly ($isGood = 1.0$).

As we can see, all the trust algorithms, except of the algo-

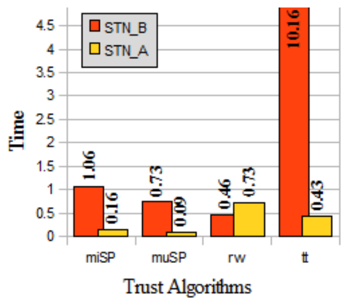

Figure 6: Complexity – the metric *Time*

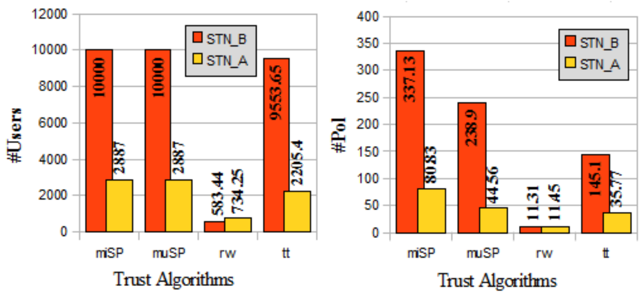

Figure 7: (A) Scalability - the metric *#Users* (B) Scalability - the metric *#Pol*

rithm tt in the social trust network stn_b, behave reasonably with a response time below or around 1 second. Much more higher complexity of the trust algorithm tt in stn_b, when compared with the algorithm rw having the similar complexity of computation, is caused by more $\#Users$ visited and more $\#Pol$ applied (see Figure 7). The results of the metric $Time$ are generally similar for various portions of $isGood$ nodes.

6.3.4 Scalability

According to the semantic web index Sindice[35] (indexing 161.01 millions of semantic web documents to the date 1.2.2011), the property `foaf:knows` is used by 5.21 millions of the documents and we can assume even more users in the future. Therefore, the trust algorithm's scalability is an important factor.

To measure the *scalability* of the trust algorithms, we compare the metrics $\#Users$ and $\#Pol$. Figures 7A and 7B depict the results of the metrics $\#Users$, respectively $\#Pol$ (y-axis) in both social trust networks and for all $ta \in \mathcal{ALG}$ (x-axis); we suppose that before measuring the metric $Time$, the trust evolves between times 0 and $t-1$ and, as in Figure 4, all users behave fairly ($isGood = 1.0$). From the scalability point of view, the lower $\#Users$ and $\#Pol$, the better.

Results of the metric $\#Users$ show that the algorithm rw utilizes a similar number of users in both networks (bounded by a threshold κ introduced in Subsection 5.4), which is a reason for similar results of the metric Δ for both social trust networks in Figures 4 and 5. The algorithms $miSP$, $muSP$, and tt utilize for its computation (almost) all users in the given social trust network. Obviously, it is possible to limit the number of users visited by these algorithms in a similar way as in the algorithm rw, however, this has a negative impact on the reached accuracy – the accuracy is then lower than the accuracy of the algorithm rw.

In Figure 7B, the results of the metric $\#Pol$ follows the results of the metric $\#Users$, the more users visited, the more QA applied. Nevertheless, we can observe that the algorithm $muSP$, although reaching the same amount of users, applies less policies. This is caused by the trust decay with the increasing trust relation paths in the computation of the algorithm $muSP$ leading to more QA policies, which could have been successfully applied to the resources, however, were not applied, because of low trust value of the consumer in the creator of these QA policies.

From the scalability point of view, the trust algorithm rw performs the best, reaching less than κ users in arbitrarily large networks. Obviously, it is not so easy, because the algorithm rw uses, for its selection of suitable users, the algorithm $muSP$ possibly reaching much more nodes than allowed by κ. That is the reason, why the results of the metric $Time$ of the algorithms rw, $miSP$, and $muSP$ are quite similar. In bigger social networks, the amount of users reached by the algorithm $muSP$ selecting the suitable users for the algorithm rw should be bounded by a multiplication of κ.

6.3.5 Summary

To summarize the results, in terms of accuracy and robustness, in the network of relatively fair users ($isGood \geqq 0.5$), the algorithms $miSP$, $muSP$, and tt have better accuracy than rw. On the other hand, in case of networks with the portion of fair users $isGood < 5$, which is supposed to be the case of the networks on the Web, the accuracy is higher for the algorithm rw. In terms of complexity, except of the algorithm tt, which behaves unacceptably, all the algorithm show up to be reasonably complex (the information consumer receives response in less than 1 second).

Nevertheless, from the scalability point of view, which is a very important property in the environment of the Web, the algorithm rw wins and the other algorithms (especially the algorithms $miSP$ and tt not utilizing the trust dacay when computing trust between users) should be bounded in the number of users visited/QA policies applied to be usable in the Web environment. Unfortunately, after introduction of such κ to the algorithms $miSP$, $muSP$, and tt, they are beaten by the algorithm rw in terms of the reached accuracy.

The trust evolution influences the accuracy positively, in case of networks with $isGood < 0.5$, the accuracy is influenced significantly.

7. RELATED WORK

This chapter compares the particular concepts of the WQA model (QA policies and social networks in Subsection 7.1, trust in Subsection 7.2) and the WQA as a whole (Subsections 7.3 and 7.4) with related work.

7.1 QA Policies and Social Networks

Researchers have developed and investigated various policy languages to describe trust and security requirements on the Web [10, 30, 47, 8]; a variety of access control mechanisms generally based on policies and rules have been developed [34, 42, 13]. Although these approaches support customization of the policies, to the best of our knowledge,

[35]http://www.sindice.com

they do not utilize the power of the social network to share policies.

Paper [12] coins the term Data Quality Social Network. Nevertheless, their social network is not intended to be used for sharing policies – it is used for adjusting weights of the quality dimensions participating in the QA process. Moreover, each Data Quality Social Network is restricted to entities collaborating on the similar tasks.

7.2 Trust in Social Networks

Kuter and Golbeck propose in [33] an algorithm SUNNY for trust inference in social networks; the computed trust value is supported by a measure of confidence in the computed value; the confidence measure is derived probabilistically based on similarities of entities' ratings. They evaluated the algorithm on the FilmTrust network [24], where the entities rated various films, and compared the algorithm with TidalTrust algorithm – SUNNY's average error was 6.5% lower, performing much more better for $p < 0.05$ in the standard two-tailed t-test. Nevertheless, the computation of the confidence values seems to be not trivial (unfortunately, no big O notation is given) and the resulting improvement is rather minor.

Ziegler and Laursen [51] proposed Appleseed trust metric (as an improved version of the Advogato[36] trust metric [35]) calculating trust for a collection of entities at once by energizing the selected entity (an information consumer) and spreading the energy to other entities connected by trust relations. The problem of this algorithm is the normalization of the trust values - the more trust relations the entity defines, the less energy (trust) each target entity of these trust relations receives. Trust algorithms in Section 5 do not have this problem.

Guha et al. [25] developed a framework of trust propagation schemes. They introduced several ways of propagating trust in a social network. Except of *direct propagation* (use of trust transitivity), they propose other atomic propagations – *co-citation* (if $\tau_{i_1,j_1} \geqq K$, $\tau_{i_1,j_2} \geqq K$, and $\tau_{i_2,j_2} \geqq K$, we can infer $\tau_{i_2,j_1} \geqq K$), *transpose trust* (if $\tau_{u,v} \geqq K$, then $\tau_{v,u} \geqq K$), and *trust coupling* (if $\tau_{u_1,v} \geqq K$, $\tau_{u_2,v} \geqq K$, then $\tau_{u,u_1} \geqq K$ implies $\tau_{u,u_2} \geqq K$). In the trust algorithms in Section 5, we use the transitivity of trust and, in Section 6, the transpose trust propagation with a probability 20%. The co-citation and trust coupling atomic propagations are not used, because they are vulnerable to attacks – a malicious entity can easily simulate the prerequisites of co-citation or trust coupling propagations and obtain an extra trust.

Furthermore, Guha et al. [25] and Ziegler and Laursen [51] dealt with the propagation of distrust and observed several problems in approaches dealing with distrust in the same way as with trust. Namely, the semantics of transitivity is not clear (Should I trust to a distrusted friend x of the distrusted friend of mine or should I distrust x even more?). Moreover, if $\tau_{u,v} < K$ and $\tau v, w \geqq K$, the trust value $\tau_{u,w} < K$, although we obtained the trust recommendation in the entity w from the distrusted entity v. The latter case can be meliorated by computing global reputations of all entities in the QA social network, however, this stands against our comprehension of trust as a subjective opinion of an entity in another.

If we are not working with the social network of the financial experts, however, with a generic social network, we propose in [32] an extension of the trust model, enabling to express that an entity trusts another entity in the particular area of the human expertise, such as ”Finances” or ”Computers”. Unfortunately, designing an appropriate generally acknowledged topic ontology, which would be used by the majority of entities when specifying a trust in another entity w.r.t the particular topic, is problematic. Therefore, better approach seems to be the utilization of the trust model not including the concept of topics (as proposed in this paper) together with a social network algorithm for finding experts in particular areas of human expertise [11].

7.3 QA Frameworks

Bizer [8] propose the Web Information Quality Assessment Framework (WIQA), a framework tailored for a web-based scenario supporting local storage of information from web pages visited by an information consumer. Users can specify policies in the form of RDF graph patterns using the WIQA-PL policy language, they can filter the information in their local storage according to the selected policy, and get justifications for ”why” a given information satisfies a set of policies. When comparing WIQA with the reference implementation of the WQA model, (1) WIQA does not support the concept of QA social networks, a crucial concept in the WQA model and in the QA process on the Web and (2) WIQA-PL is inspired by SPARQL, thus, its syntax may be confusing for the majority of information customers.

The Inference Web[37] (IW) is a Semantic Web based knowledge provenance infrastructure providing interoperable explanations of sources using Proof Markup Language (PML) and, therefore, supporting entity's trust decisions [38, 16]. Furthermore, IW translates PML machine explanations to natural language and displays them to human entities using the application Probe-It![38]. IWTrust [50], the trust component of IW, is based on Web of Trust principle for expressing entity's trust in other entities and data sources. Comparing with our approach, IW does not support customizable trust (QA) policies, having an important usability issue. Furthermore, Inference Web can be viewed as a community of actively reasoning agents cooperatively deriving answers based on the shared knowledge. On the other hand, the WQA model is tailor for rather passive average users of the Web.

7.4 Recommender Systems

Recommender systems suggest which resources should be used by the information consumer. In [22, 48], two types of recommender systems are highlighted – those using (1) content-based methods, which suggest resources based on their similarity with the resource the information consumer already expressed interest in, and (2) collaborative filtering methods, which calculate the similarities between entities in the system and recommend resources that are liked by similar entities.

Since the particular WQA model implementation is de facto a recommender system, we observed how the content-based and collaborative filtering methods can be accommodated in the WQA model. The content-based methods can be exploited by collecting metadata associated with the processed resources with high satisfaction levels and, consequently, if a resource containing a subset of this metadata is

[36]http://www.advogato.org/

[37]http://inference-web.org/

[38]http://trust.utep.edu/probeit/

consumed by the same information consumer in the future, it can be scored higher. The collaborative filtering methods can be applied by comparing the similarities of the sets of derived trust values of two trustworthy entities and, consequently, allowing these (trustworthy!) entities to share their derived trust values.

In [48], Walter et al. propose a model of trust based recommender system involving the set of agents (entities) connected with each other by trust relations; the agents have ratings of the subset of objects (resources) belonging to the particular categories of objects and the agents can be experts in one or more categories. When an agent (the information consumer) seeks for the best object (resource) in the given category, he asks his neighbors in the social network. If the neighbor (1) is an expert on the given category or (2) has an experience with objects from this category, he returns to the information consumer a recommendation of the chosen object; if neither (1) nor (2) holds, the request for recommendation is submitted farther along the relations in the social network. Based on the set of recommendations returned, the information consumer chooses one of them with a probability corresponding to the amount of trust between him and the recommending agent.

When comparing with WQA model, instead of applying policies collected from the neighbors, they ask the neighbors for their opinion. Nevertheless, their approach assumes that all agents behave according to their best conscience. As a result, the algorithms and methods proposed in [48] can show up unsuitable in the real world deployment. In WQA model, we evaluated the robustness of our approach on the artificial and real world network and, based on that, decided for the algorithm ReWA, which shows up to be the most robust one. On the other hand, Walter et al. use the algorithm SPathMulti which behaves much worse.

Moreover, the experience the agent gained from having an item (resource) recommended leads to an increase in trust along the path from the information consumer to the recommending agent; therefore, trust evolves to a metric expressing the similarity between two entities. As a result, they conflate trust with similarity, which is wrong from the conceptual point of view – trust is a personal feeling, something which underpins every interaction between agents, an agent can trust another agent because of their similarity or despite of their dissimilarity. Furthermore, changing the trust along the whole path from the information consumer to the recommending agent has security issues – one can change the other's trust values based on his dissatisfaction with a recommendation. In WQA model, based on the satisfaction of an entity with a resource, the trust value is changed, however, only for the trust relations between the entity and the direct neighbors on trust relation paths from the entity to the entities whose policies were applied to the rated resource; this approach is more secure and is not in contrary with Definition 3.1.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a generic Web Quality Assessment model for assessing the quality of information on the Web. Since the unprecedented volume of information is being published as open data in the Web information space and millions of information consumers are assessing the quality of information, the QA process in the WQA model is driven by a set of customizable QA policies grouped to QA profiles suitable for different tasks at the consumers' hands. Since we argue that provenance of information presents the cornerstone aspect in assessing the quality of information, the proposed WQA model was demonstrated on the financial motivational scenario incorporating the Provenance QA analysis.

As part of the WQA model, we proposed and detailed a crucial concept of QA social networks enabling the sharing of QA policies between trustworthy entities in social trust networks. To enable this, we (1) defined a trust model where an entity can express trust in another entity, i.e. create a trust relation to another entity, and (2) proposed a trust algorithm ReWA capable of deriving trust between two entities not connected with a trust relation. To experimentally evaluate the accuracy, robustness, complexity, and scalability of the algorithm ReWA, we developed the WQA model's reference implementation, Grian, and compared the algorithm ReWA with other trust algorithms. The evaluation confirmed that ReWA is a suitable trust algorithm to the environments with thousands of entities and high portion of intentionally or unintentionally malicious entities. Furthermore, the evaluation revealed that the trust evolution proposed in Subsection 4.7 is a promising approach.

Future work involves further experiments with huge real world data obtained from FOAF social networks. We will consider incorporation of algorithms for finding experts and content-based and collaborative filtering methods to further improve the QA process. In parallel, we are working on user interfaces for easy management of QA profiles and QA social networks. We are persuaded that the ability to assess the quality of information on the exponentially growing Web will play a fundamental role in the continued adoption of Linked Data principles.

9. ACKNOWLEDGMENTS

The work presented in this article has been funded in part by the Czech Science Foundation (GACR, grant number 201/09/H057) and GAUK 3110.

10. REFERENCES

[1] B. Aleman-Meza, C. Halaschek-Wiener, I. B. Arpinar, C. Ramakrishnan, and A. P. Sheth. Ranking Complex Relationships on the Semantic Web. *IEEE Internet Computing*, 9:37–44, 2005.

[2] J. E. Alexander and M. A. Tate. *How to Evaluate and Create Information Quality on the Web*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1999.

[3] S. ann Knight and J. Burn. Developing a Framework for Assessing Information Quality on the World Wide Web. *Informing Science Journal*, 8:159–172, 2005.

[4] D. Artz and Y. Gil. A Survey of Trust in Computer Science and the Semantic Web. *Web Semant.*, 5(2):58–71, 2007.

[5] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286(5439):509–512, October 1999.

[6] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. Methodologies for Data Quality Assessment and Improvement. *ACM Comput. Surv.*, 41(3):1–52, 2009.

[7] D. Beckett. RDF/XML Syntax Specification (Revised). W3C Recommendation, 10 February 2004.

http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[8] C. Bizer and R. Cyganiak. Quality-driven Information Filtering Using the WIQA Policy Framework. *Web Semant.*, 7(1):1–10, 2009.

[9] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[10] P. Bonatti and D. Olmedilla. Driving and Monitoring Provisional Trust Negotiation with Metapolicies. In *POLICY '05: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 14–23, Washington, DC, USA, 2005. IEEE Computer Society.

[11] J. G. Breslin, U. Bojars, B. Aleman-meza, H. Boley, L. J. Nixon, A. Polleres, and A. V. Zhdanova. Finding Experts using Internet-based Discussions in Online Communities and Associated Social Networks. In *First International ExpertFinder Workshop*, 2007.

[12] I. Caballero, E. Verbo, M. A. Serrano, C. Calero, and M. Piattini. Tailoring Data Quality Models Using Social Network Preferences. In *DASFAA Workshops*, pages 152–166, 2009.

[13] S. Cantor, J. Kemp, R. Philpott, and E. Maler. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report, 2005.

[14] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, Provenance and Trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM.

[15] B. Christianson and W. S. Harbison. Why Isn't Trust Transitive? In *Proceedings of the International Workshop on Security Protocols*, pages 171–176, London, UK, 1997. Springer-Verlag.

[16] P. P. da Silva, D. L. Mcguinness, and R. Fikes. A proof markup language for semantic web services. *Inf. Syst.*, 31(4):381–395, June 2006.

[17] P. P. da Silva, D. L. McGuinness, N. D. Rio, and L. Ding. Inference Web in Action: Lightweight Use of the Proof Markup Language. In *International Semantic Web Conference*, pages 847–860, 2008.

[18] M. Dean and G. Schreiber. OWL Web Ontology Language Reference. W3C recommendation, W3C, February 2004.

[19] M. Deutsch. Cooperation and Trust: Some Theoretical Notes. 1962.

[20] A. Freitas, S. O'Riain, E. Curry, and T. Knap. W3P: Building an OPM based provenance model for the Web. *Future Generation Computer Systems*, In Press, Accepted Manuscript:Online on http://www.sciencedirect.com/, 2010.

[21] Y. Gil and D. Artz. Towards Content Trust of Web Resources. *Web Semant.*, 5(4):227–239, 2007.

[22] J. Golbeck. Generating Predictive Movie Recommendations from Trust in Social Networks. In *Trust Management*, pages 93–104. 2006.

[23] J. Golbeck. Trust on the world wide web: a survey. *Found. Trends Web Sci.*, 1(2):131–197, 2006.

[24] J. A. Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, College Park, MD, USA, 2005.

[25] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of Trust and Distrust. In *International World Wide Web Conference*, 2004.

[26] O. Hartig. Provenance Information in the Web of Data. April 2009. LDOW2009, 2009, Madrid, Spain.

[27] J. Huang and M. S. Fox. An ontology of trust: formal semantics and transitivity. In *Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet*, ICEC '06, pages 259–270, New York, NY, USA, 2006. ACM.

[28] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, March 2007.

[29] A. Jøsang and S. Pope. Semantic constraints for trust transitivity. In *Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43*, APCCM '05, pages 59–68, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[30] L. Kagal, T. Finin, and A. Joshi. A Policy Based Approach to Security for the Semantic Web. In *2nd International Semantic Web Conference (ISWC2003)*, September 2003.

[31] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA, 2003. ACM.

[32] T. Knap and I. Mlýnková. Towards Topic-Based Trust in Social Networks. 7th International Conference on Ubiquitous Intelligence and Computing, 2010. (http://www.ksi.mff.cuni.cz/˜knap/wqam/tt10.pdf).

[33] U. Kuter and J. Golbeck. SUNNY: a new algorithm for trust inference in social networks using probabilistic confidence models. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1377–1382. AAAI Press, 2007.

[34] K. Lawrence and C. Kaler. WS-Trust Specification. Technical report, March 2007.

[35] R. Levien. Attack-Resistant Trust Metrics. pages 121–132. 2009.

[36] F. Manola and E. Miller. RDF Primer, 2004.

[37] S. Marsh. Formalising Trust as a Computational Concept, 1994.

[38] D. Mcguinness and P. Silva. Infrastructure for Web Explanations. pages 113–129. 2003.

[39] D. H. Mcknight and N. L. Chervany. The Meanings of Trust. Technical report, University of Minnesota, Carlson School of Management, 1996.

[40] S. Milgram. The Small World Problem. *Psychology Today*, 1:61, 1967.

[41] L. Moreau. The Foundations for Provenance on the Web. *Foundations and Trends in Web Science*, November 2009.

[42] T. Moses. Extensible Access Control Markup Language Version 2.0. Technical report, 2005.

[43] F. Naumann and C. Rolker. Assessment Methods for Information Quality Criteria. In *Proceedings of the*

International Conference on Information Quality, pages 148–162, 2000.

[44] M. E. J. Newman. Models of the Small World. *J. Stat. Phys*, pages 819–841, 2000.

[45] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[46] M. Richardson, R. Agrawal, and P. Domingos. Trust Management for the Semantic Web. In *In Proceedings of the 2nd International Semantic Web Conference*, pages 351–368, 2003.

[47] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. KAoS Policy and Domain Services: Toward a Description-logic Approach to Policy Representation, Deconfliction, and Enforcement. In *Proceedings of Policy 2003*, pages 93–96, 2003.

[48] F. Walter, S. Battiston, and F. Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, February 2008.

[49] R. Y. Wang and D. M. Strong. Beyond Accuracy: What Data Quality Means to Data Consumers. *J. Manage. Inf. Syst.*, 12(4):5–33, 1996.

[50] I. Zaihrayeu, P. P. da Silva, and D. L. McGuinness. IWTrust: Improving User Trust in Answers from the Web. In P. Herrmann, V. Issarny, and S. Shiu, editors, *Trust Management*, volume 3477 of *Lecture Notes in Computer Science*, chapter 27, pages 384–392. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2005.

[51] C.-N. Ziegler and G. Lausen. Propagation Models for Trust and Distrust in Social Networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.