Chapter 3
Models for Trust Inference in Social Networks

Cai-Nicolas Ziegler and Jennifer Golbeck

Abstract Interpersonal trust between any two people in social networks is hard to gauge, and even harder to *infer*, given that these two people are not connected by an immediate social link, such as friendship or acquaintanceship. In order to be able to make accurate inferences for an arbitrary tuple of people in a given social environment, we present an approach, named Appleseed, that is based on mechanics taken from neuropsychology, known as spreading activation models. Compelling in its simplicity, we relate the concept to trust propagation and evaluation in an intuitive fashion. While Appleseed works very well when paths between two arbitrary people in the network can be established, no inference of trust is possible when this is not the case. To this end, we present several algorithms for inferring trust that go beyond network structure and demonstrate their accuracy in real social networks. We also show how these algorithms can be augmented with additional data that may be available in some contexts.

3.1 Introduction

In our world of information overload and global connectivity leveraged through the Web and other media types, social trust [29] between individuals becomes an invaluable and precious good. Hereby, trust exerts an enormous impact on decisions whether to believe or disbelieve information asserted by other peers. Belief should only be accorded to statements from people we deem trustworthy. However, when supposing huge social networks such as the case for social media platforms, trust judgements based on personal experience and acquaintanceship become unfeasible.

The research presented in this chapter has been conducted while being affiliated with Albert-Ludwigs- Universität Freiburg i.Br., Germany.

C.-N. Ziegler (✉)
XING EVENTS GmbH, Sandstraße 33, 80335 Munich, Germany
e-mail: cai-nicolas.ziegler@xing.com

J. Golbeck
University of Maryland, College Park, MD 20742, USA
e-mail: jgolbeck@umd.edu

D. Król et al. (eds.), *Propagation Phenomena in Real World Networks*,
Intelligent Systems Reference Library 85, DOI 10.1007/978-3-319-15916-4_3

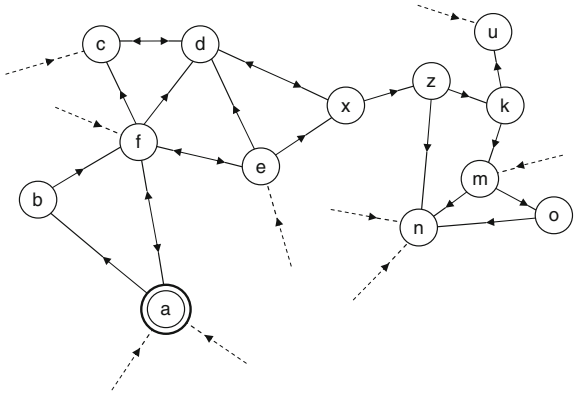

Fig. 3.1 Sample web of trust for agent *a*

In general, we accord trust, which has been defined as the "subjective expectation an agent has about another's future behavior based on the history of their encounters" [30], to only small numbers of people. These people, again, trust another limited set of people, and so forth. The network structure emanating from our person (see Fig. 3.1), composed of trust statements linking individuals, constitutes the basis for trusting people we do not know personally.

We might be tempted to adopt the policy of trusting all those people who are trusted by persons we trust. Trust would thus propagate through the network [21] and become accorded whenever two individuals can reach each other via at least one trust path. However, common sense tells us we should not rely upon this strategy. More complex metrics are needed in order to more sensibly evaluate trust between two persons. Among other features, these trust metrics must take into account social and psychological aspects of trust and suffice criteria of computability and scalability likewise.

When adopting the most basic policy of trust propagation, all those people who are trusted by persons we trust are considered likewise trustworthy. Trust would thus propagate through the network and become accorded whenever two individuals can reach each other via at least one trust path. However, owing to certain implications of interpersonal trust, e.g., attack-resistance, trust decay, etc., more complex metrics are needed to sensibly evaluate social trust. Subtle social and psychological aspects must be taken into account and specific criteria of computability and scalability satisfied.

In this chapter, we aim at designing one such complex trust metric,[1] particularly tailored to social filtering tasks by virtue of its ability to infer continuous trust values through fixpoint iteration, rendering ordered trust-rank lists feasible.

However, one challenge to using network data for trust inference is that when there are few or no paths to a node in the network, the algorithms may not be able

[1] Note that trust concepts commonly adopted for webs of trust, and similar trust network applications, are largely general and do not cover specifics such as "situational trust" [26], as has been pointed out in [13]. For instance, agent a_i may blindly trust a_j with respect to books, but not trust a_j with respect to trusting others, for a_j has been found to accord trust to other people too easily. For our trust propagation scheme at hand, we also suppose this largely uni-dimensional concept of trust.

to infer a value. Using TidalTrust, another network-based trust inference algorithm, and data from FilmTrust, a movie rating website with an underlying trust network, we show that integrating additional, non-network-based information into the trust computation, can improve the number of node pairs for which a trust value can be calculated. Integrating the network and data-based models can also improve the accuracy of these algorithms.

3.1.1 Trust Representation and Model

We assume that all trust information is publicly accessible for any agent in the system, e.g., through machine-readable personal homepages distributed over the network. Agents $a_i \in A = \{a_1, a_2, \ldots, a_n\}$ are associated with a partial trust function $W_i \in T = \{W_1, W_2, \ldots, W_n\}$ each, where $W_i : A \rightarrow [0, 1]^{\perp}$ holds, which corresponds to the set of trust assertions that a_i has stated.

In most cases, functions $W_i(a_j)$ will be very sparse as the number of individuals an agent is able to assign explicit trust ratings for is much smaller than the total number n of agents. Moreover, the higher the value of $W_i(a_j)$, the more trustworthy a_i deems a_j. Conversely, $W_i(a_j) = 0$ means that a_i considers a_j to be *not trustworthy*. The assignment of trust through continuous values between 0 and 1, and their adopted semantics, is in perfect accordance with [26], where possible stratifications of trust values are proposed. Our trust model defines one directed trust graph with nodes being represented by agents $a_i \in A$, and directed edges from nodes a_i to nodes a_j representing trust statements $W_i(a_j)$.

For convenience, we introduce the partial function $W : A \times A \rightarrow [0, 1]^{\perp}$, which we define as the union of all partial functions $W_i \in T$.

3.1.2 Overview of Trust Metrics for Social Networks

Trust and reputation ranking metrics have primarily been used for the Public Key Infrastructure (PKI) [4, 23, 28, 33, 34], rating and reputation systems part of online communities [14, 22, 24], peer-to-peer networks [3, 18–20, 36], and also mobile computing [7]. Each of these scenarios favors different trust metrics. For instance, reputation systems for online communities tend to make use of *centralized trust servers* that compute global trust values for all users on the system [14]. On the other hand, peer-to-peer networks of moderate size rely upon distributed approaches that are in most cases based upon PageRank [18, 36].

Larger social networks, such as the Semantic Web, are made up of millions of nodes, i.e., agents. The fitness of *distributed* approaches to trust metric computation, such as described in [18, 35], hence becomes limited for various reasons:

Trust data storage. Every agent a_i needs to store trust rating information about any other agent a_j on the network. Agent a_i uses this information in order to merge it with own trust beliefs and propagates the synthesized information to his trusted

agents [22]. For a network organized in a decentralized fashion, the number of agents for whom to keep trust information will still exceed the storage capacities of most nodes.

Convergence. The structure of networks not under centralized control is diffuse and commonly not subject to some higher ordering principle or hierarchy. Furthermore, the process of trust propagation is *necessarily asynchronous* as there is no central node of authority. Convergence of trust values might thus take a very long time.

The huge advantage of distributed approaches to trust propagation and computation, on the other hand, is the *immediate availability* of computed trust information about any other agent a_j in the system. Moreover, agents have to disclose their trust assertions only to peers they actually *trust* [35]. For instance, suppose that a_i declares his trust in a_j by $W_i(a_j) = 0.1$, which is very low. Hence, a_i might want a_j not to know about that fact. As distributed metrics only propagate *synthesized* trust values from nodes to successor nodes in the trust graph, a_i would not have to openly disclose his trust statements to a_j.

As it comes to centralized, i.e., *locally computed*, metrics, *full* trust information access is required for agents inferring trust. Hence, online communities based on trust require their users to disclose all trust information to the community server, but not necessarily to other peers [14]. Privacy thus remains preserved. On social networks such as the Semantic Web, however, there is no such central authority that computes trust. *Any* agent might want to do so. Our own trust model, as well as trust models proposed in [1, 7, 13], are hence based upon the assumption of *publicly available trust information*. Though privacy concerns may persist, this assumption is vital, owing to the afore-mentioned deficiencies of distributed computation models. Moreover, centralized *global* metrics, such as depicted in [14, 31], also fail to fit our requirements: because of the huge number of agents issuing trust statements, only dedicated server clusters could be able to manage the whole bulk of trust relationships.

Scalar metrics, e.g., PKI proposals [4, 23, 28, 33, 34] and those metrics described in [13], have poor scalability properties, owing to exponential time complexity [33].

Consequently, we advocate *local group* trust metrics [43] for the Semantic Web and other large-scale decentralized networks. Local group trust metrics do not only compute trust values for a specified pair of agents, $(a_i, a_j) \in V \times V$, but compute trust *ranks* for *sets* of individuals from V. The predicate *local* refers to the metric's network perspective, which is subjective, adopting the position of one of the agents. That is, trust values assigned to $a_j \in V$ are different for two different trust sources a_i.

Local group trust metrics bear several welcome properties with respect to computability and complexity, which may be summarized as follows:

Partial trust graph exploration. Global metrics require a priori full knowledge of the entire trust network. Distributed metrics store trust values for all agents in the system, thus implying massive data storage demands. On the other hand, when computing trusted *neighborhoods*, the trust network only needs to be explored partially: originating from the trust source, one only follows those trust edges that seem promising, i.e., bearing high trust weights, and which are not too far away

from the trust source. Inspection of agent nodes is thus performed in a just-in-time fashion. Hence, prefetching bulk trust information is not required.

Computational scalability. Tightly intertwined with partial trust graph exploration is computational complexity. Local group trust metrics scale well to any social network size, as only tiny subsets of relatively constant size[2] are visited.

3.2 Design of Local Group Trust Metrics

Local group trust metrics, in their function as means to compute trust neighborhoods, have not been subject to mainstream research so far. Significant research has effectively been limited to the work done by Levien [22] who has conceived and deployed the Advogato group trust metric. This section provides an overview of Advogato and introduces our own Appleseed trust metric, eventually comparing both approaches.

3.2.1 Outline of Advogato Maxflow

The Advogato maximum flow trust metric has been proposed by Levien and Aiken [24] in order to discover which users are trusted by members of an online community and which are not. Trust is computed through one centralized community server and considered relative to a seed of users enjoying supreme trust. However, the metric is not only applicable to community servers, but also to *arbitrary* agents which may compute *personalized lists* of trusted peers, not only one single global ranking for the whole community they belong to. In this case, the active agent himself constitutes the singleton trust seed. The following paragraphs briefly introduce Advogato's basic concepts. For more detailed information, refer to [22–24].

3.2.1.1 Trust Computation Steps

Local group trust metrics compute sets of agents trusted by those being part of the trust seed. In case of Advogato, its input is given by an integer number n, which is supposed to be equal to the number of members to trust [24], as well as the trust seed s, which is a subset of the entire set of users A. The output is a *characteristic function* that maps each member to a boolean value indicating his trustworthiness:

$$\text{Trust}_M : 2^A \times \mathbb{N}_0^+ \rightarrow (A \rightarrow \{\text{true}, \text{false}\}) \tag{3.1}$$

The trust model underlying Advogato does *not* provide support for weighted trust relationships in its original version.[3] Hence, trust edges extending from individual x

[2] Supposing identical parameterizations for the metrics in use, as well as similar network structures.

[3] Though various levels of peer certification exist, their interpretation does not perfectly align with weighted trust relationships.

to y express *blind*, i.e., *full*, trust of x in y. The metrics for PKI maintenance suppose similar models. Maximum integer network flow computation [8] has been investigated by Reiter and Stubblebine [33, 34] in order to make trust metrics more reliable. Levien adopted and extended this approach for group trust in his Advogato metric.

Capacities $C_A : A \rightarrow \mathbb{N}$ are assigned to every community member $x \in A$ based upon the shortest-path distance from the seed to x. Hereby, the capacity of the seed itself is given by the input parameter n mentioned before, whereas the capacity of each successive distance level is equal to the capacity of the previous level l divided by the average outdegree of trust edges $e \in E$ extending from l. The trust graph we obtain hence contains one single source, which is the set of seed nodes considered as one single "virtual" node, and multiple sinks, i.e., all nodes other than those defining the seed. Capacities $C_A(x)$ constrain nodes. In order to apply Ford-Fulkerson maximum integer network flow [8], the underlying problem has to be formulated as single-source/single-sink, having capacities $C_E : E \rightarrow \mathbb{N}$ constrain *edges* instead of *nodes*. Hence, Algorithm 3.1 is applied to the old directed graph $G = (A, E, C_A)$, resulting in a new graph structure $G' = (A', E', C_{E'})$ (Fig. 3.2).

Figure 3.3 depicts the outcome of converting node-constrained single-source/multiple-sink graphs (see Fig. 3.2) into single-source/single-sink ones with capacities constraining edges.

Conversion is followed by simple integer maximum network flow computation from the trust seed to the super-sink. Eventually, the trusted agents x are exactly those peers for whom there is flow from "negative" nodes x^- to the super-sink. An additional constraint needs to be introduced, requiring flow from x^- to the super-sink whenever there is flow from x^- to x^+. The latter constraint assures that node x does

```
func transform (G = (A, E, C_A)) {
  set E' ← ∅, A' ← ∅;
  for all x ∈ A do
    add node x+ to A';
    add node x− to A';
    if C_A(x) ≥ 1 then
      add edge (x−, x+) to E';
      set C_E'(x−, x+) ← C_A(x) − 1;
      for all (x, y) ∈ E do
        add edge (x+, y−) to E';
        set C_E'(x+, y−) ← ∞;
      end do
      add edge (x−, supersink) to E';
      set C_E'(x−, supersink) ← 1;
    end if
  end do
  return G' = (A', E', C_E');
}
```

Algorithm 3.1. Trust graph conversion

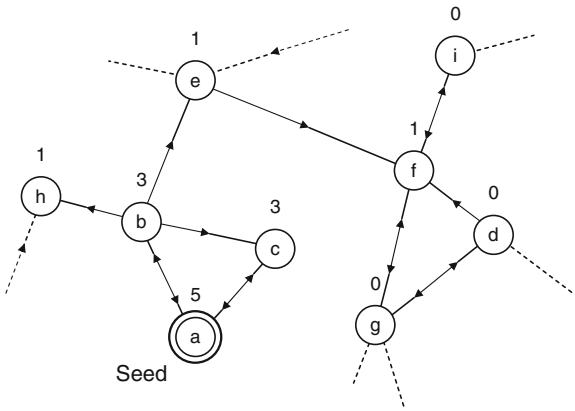

Fig. 3.2 Trust graph *before* conversion for Advogato

not only serve as an intermediate for the flow to pass through, but is *actually added* to the list of trusted agents when reached by network flow. However, the standard implementation of Ford-Fulkerson traces shortest paths to the sink first [8]. The above constraint is thus satisfied implicitly already.

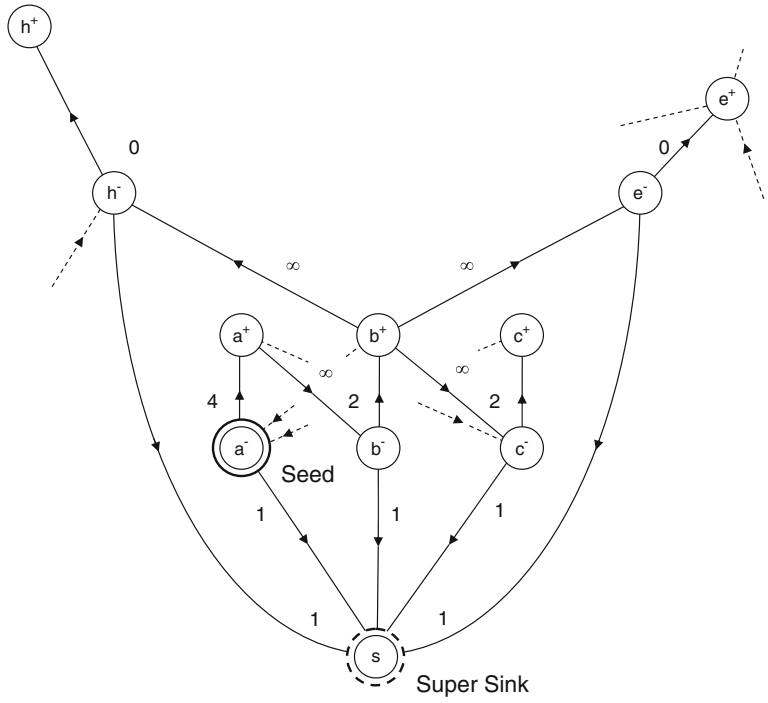

Fig. 3.3 Trust graph *after* conversion for Advogato

Example 3.1 (Advogato trust computation) Suppose the trust graph depicted in Fig. 3.2. The only seed node is a with initial capacity $C_A(a) = 5$. Hence, taking into account the outdegree of a, nodes at unit distance from the seed, i.e., nodes b and c, are assigned capacities $C_A(b) = 3$ and $C_A(c) = 3$, respectively. The average outdegree of both nodes is 2.5 so that second level nodes e and h obtain unit capacity. When computing maximum integer network flow, agent a will accept himself, b, c, e, and h as trustworthy peers.

3.2.1.2 Attack-Resistance Properties

Advogato has been designed with resistance against massive attacks from malicious agents outside of the community in mind. Therefore, an upper bound for the number of "bad" peers chosen by the metric is provided in [24], along with an informal security proof to underpin its fitness. Resistance against malevolent users trying to break into the community can already be observed in the example depicted by Fig. 3.1, supposing node n to be "bad": though agent n is trusted by numerous persons, he is deemed less trustworthy than, for instance, x. While there are fewer agents trusting x, these agents enjoy higher trust reputation[4] than the numerous persons trusting n. Hence, it is not just the *number* of agents trusting an individual i, but also the trust *reputation* of these agents that exerts an impact on the trust assigned to i. PageRank [31] works in a similar fashion and has been claimed to possess properties of attack-resistance similar to those of the Advogato trust metric [22]. In order to make the concept of attack-resistance more tangible, Levien proposes the "bottleneck property" as a common feature of attack-resistant trust metrics. Informally, this property states that the "trust quantity accorded to an edge $s \rightarrow t$ is not significantly affected by changes to the successors of t" [22].

Attack-resistance features of various trust metrics are discussed in detail in [23, 38].

3.2.2 The Appleseed Trust Metric

The Appleseed trust metric constitutes the main contribution of this chapter and is our novel proposal for local group trust metrics. In contrast to Advogato, being inspired by maximum network flow computation, the basic intuition of Appleseed is motivated by *spreading activation models*. Spreading activation models have first been proposed by Quillian [32] in order to simulate human comprehension through semantic memory, and are commonly described as "models of retrieval from long-term memory in which activation subdivides among paths emanating from an activated mental representation" [37]. By the time of this writing, the seminal work of Quillian has been ported to a whole plethora of other disciplines, such as latent semantic indexing

[4] With respect to seed node a.

[5] and text illustration [16]. As an example, we will briefly introduce the spreading activation approach adopted in [5], used for semantic search in contextual network graphs, in order to then relate Appleseed to that work.

3.2.2.1 Searches in Contextual Network Graphs

The graph model underlying contextual network search graphs is almost identical in structure to the one presented in Sect. 3.1.1, i.e., edges $(x, y) \in E \subseteq A \times A$ connecting nodes $x, y \in A$. Edges are assigned continuous weights through $W : E \rightarrow [0, 1]$. Source node s, the node from which we start searching, is activated through an injection of energy e, which is then propagated to other nodes along edges according to some set of simple rules: all energy is *fully divided* among successor nodes with respect to their normalized local edge weight, i.e., the higher the weight of an edge $(x, y) \in E$, the higher the portion of energy that flows along that edge. Furthermore, supposing average outdegrees greater than one, the closer node x to the injection source s, and the more paths lead from s to x, the higher the amount of energy flowing into x. To eliminate endless, marginal and negligible flow, energy streaming into node x must exceed threshold T in order not to run dry. The described approach is captured formally by Algorithm 3.2, which propagates energy recursively.

3.2.2.2 Trust Propagation

Algorithm 3.2 shows the basic intuition behind spreading activation models. In order to tailor these models to trust computation, later to become the Appleseed trust metric, serious adaptations are necessary. For instance, procedure energize(e, s) registers *all* energy e that has passed through node x, stored in energy(x). Hence, energy(x) represents the *relevance rank* of x. Higher values indicate higher node rank. However, at the same time, all energy contributing to the rank of x is passed *without loss* to successor nodes. Interpreting energy ranks as trust ranks thus implies numerous issues of semantic consistency as well as computability. Consider the graph depicted in Fig. 3.4a. Applying spreading activation according to [5], trust ranks of nodes b and d will be identical. However, intuitively, d should be accorded *less* trust than b,

```
procedure energize (e ∈ ℝ₀⁺, s ∈ A) {
  energy(s) ← energy(s) + e;
  e′ ← e / Σ_{(s,n) ∈ E} W(s, n);
  if e > T then
    ∀(s, n) ∈ E : energize (e′ · W(s, n), n);
  end if
}
```

Algorithm 3.2. Recursive energy propagation

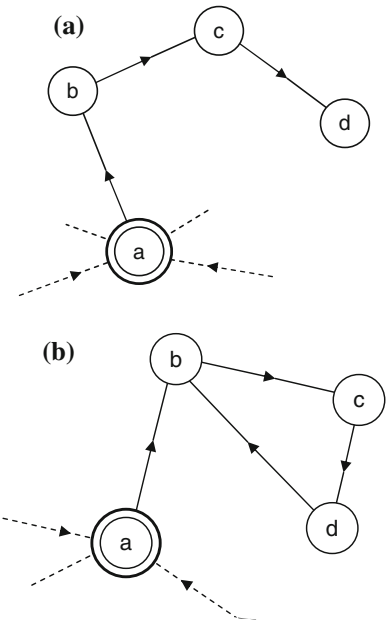

Fig. 3.4 Node chains (**a**) rank sinks (**b**)

since d's shortest-path distance to the trust seed is higher. Trust decay is commonly agreed upon [14, 17], for people tend to trust individuals trusted by immediate friends more than individuals trusted only by friends of friends. Figure 3.4b depicts even more serious issues: all energy, or trust,[5] respectively, distributed along edge (a, b) becomes *trapped in a cycle* and will never be accorded to any other nodes but those being part of that cycle, i.e., b, c, and d. These nodes will eventually acquire infinite trust rank. Obviously, the *bottleneck property* [22] does not hold. Similar issues occur with simplified versions of PageRank [31], where cycles accumulating infinite rank have been dubbed "rank sinks".

3.2.2.3 Spreading Factor

We handle both issues, i.e., trust decay in node chains and elimination of rank sinks, by tailoring the algorithm to rely upon our global *spreading factor* d. Hereby, let $\text{in}(x)$ denote the energy influx into node x. Parameter d then denotes the portion of energy $d \cdot \text{in}(x)$ that node x distributes among successors, while retaining $(1-d) \cdot \text{in}(x)$. For instance, suppose $d = 0.85$ and energy quantity $\text{in}(x) = 5.0$ flowing into node x. Then, the total energy distributed to successor nodes amounts to 4.25, while the energy rank $\text{energy}(x)$ of x increases by 0.75. Special treatment is necessary for

[5] The terms "energy" and "trust" are used interchangeably in this context.

nodes with zero outdegree. For simplicity, we assume all nodes to have an outdegree of at least one, which makes perfect sense, as will be shown later.

The spreading factor concept is very intuitive and, in fact, very close to real models of energy spreading through networks. Observe that the overall amount of energy in the network, after initial activation in^0, does not change over time. More formally, suppose that $\text{energy}(x) = 0$ for all $x \in A$ before injection in^0 into source s. Then the following equation holds in every computation step of our modified spreading algorithm, incorporating the concept of spreading factor d:

$$\sum_{x \in A} \text{energy}(x) = \text{in}^0 \tag{3.2}$$

Spreading factor d may also be seen as the *ratio* between *direct* trust in x and trust in the ability of x to *recommend* others as trustworthy peers. For instance, Beth et al. [4] and Maurer [28] explicitly differentiate between *direct* trust edges and *recommendation* edges.

We commonly assume $d = 0.85$, though other values may also seem reasonable. For instance, having $d \leq 0.5$ allows agents to keep most of the trust they are granted for themselves and only pass small portions of trust to their peers. Observe that low values for d favor trust proximity to the source of trust injection, while high values allow trust to also reach more distant nodes. Furthermore, the introduction of spreading factor d is crucial for making Appleseed retain Levien's bottleneck property, as will be shown in later sections.

3.2.2.4 Rank Normalization

Algorithm 3.2 makes use of edge weight normalization, i.e., the quantity $e_{x \to y}$ of energy distributed along (x, y) from x to successor node y depends on the *relative* weight of $x \to y$, i.e., $W(x, y)$ compared to the sum of weights of all outgoing edges of x:

$$e_{x \to y} = d \cdot \text{in}(x) \cdot \frac{W(x, y)}{\sum_{(x,s) \in E} W(x, s)} \tag{3.3}$$

Normalization is common practice in many trust metrics, among those PageRank [31], EigenTrust [18], and AORank [14]. However, while normalized reputation or trust seem reasonable for models with plain, non-weighted edges, serious interferences occur when edges are *weighted*, as is the case for our trust model adopted in Sect. 3.1.1.

For instance, refer to Fig. 3.5a for unwanted effects: The amounts of energy that node a accords to successors b and d, i.e., $e_{a \to b}$ and $e_{a \to d}$, respectively, are identical in value. Note that b has issued only *one* trust statement $W(b, c) = 0.25$, stating that b's trust in c is rather weak. On the other hand, d assigns *full* trust to individuals e, f, and g. Nevertheless, the overall trust rank for d will be much higher than for any successor of d, for c is accorded $e_{a \to b} \cdot d$, while e, f, and g only obtain $e_{a \to d} \cdot d \cdot 1/3$

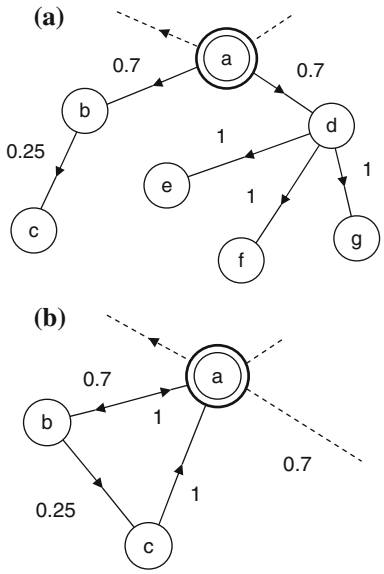

Fig. 3.5 Issues with trust normalization

each. Hence, c will be trusted *three times* as much as e, f, and g, which is not reasonable at all.

3.2.2.5 Backward Trust Propagation

The above issue has already been discussed by Kamvar et al. [18], but no solution has been proposed therein, arguing that "substantially good results" have been achieved despite the drawbacks. We propose to alleviate the problem by making use of *backward propagation* of trust to the source: when metric computation takes place, additional "virtual" edges (x, s) from every node $x \in A \setminus \{s\}$ to the trust source s are created. These edges are assigned full trust $W(x, s) = 1$. Existing backward links (x, s), along with their weights, are "overwritten". Intuitively, every node is supposed to *blindly trust the trust source* s, see Fig. 3.5b. The impacts of adding backward propagation links are threefold:

Mitigating relative trust. Again, we refer to Fig. 3.5a. Trust distribution in the underlying case becomes much fairer through backward propagation links, for c now only obtains $e_{a\to b} \cdot d \cdot (0.25/(1 + 0.25))$ from source s, while e, f, and g are accorded $e_{a\to d} \cdot d \cdot (1/4)$ each. Hence, trust ranks of both e, f, and g amount to 1.25 times the trust assigned to c.

Avoidance of dead ends. Dead ends, i.e., nodes x with zero outdegree, require special treatment in our computation scheme. Two distinct approaches may be adopted. First, the portion of incoming trust $d \cdot \text{in}(x)$ supposed to be passed to successor nodes is completely discarded, which contradicts our intuition of no energy leaving the system. Second, instead of retaining $(1 - d) \cdot \text{in}(x)$ of incoming trust, x keeps

all trust. The latter approach is also not sensible as it encourages users to not issue trust statements for their peers. Luckily, with backward propagation of trust, all nodes are *implicitly linked* to the trust source s, so that there are no more dead ends to consider.

Favoring trust proximity. Backward links to the trust source s are favorable for nodes close to the source, as their eventual trust rank will increase. On the other hand, nodes further away from s are penalized.

3.2.2.6 Nonlinear Trust Normalization

In addition to backward propagation, we propose supplementary measures to decrease the negative impact of trust spreading based on relative weights. Situations where nodes y with poor ratings from x are awarded high overall trust ranks, thanks to the low outdegree of x, have to be avoided. Taking the squares of local trust weights provides an appropriate solution:

$$e_{x \to y} = d \cdot \text{in}(x) \cdot \frac{W(x, y)^2}{\sum_{(x,s) \in E} W(x, s)^2} \tag{3.4}$$

As an example, refer to node b in Fig. 3.5b. With squared normalization, the total amount of energy flowing backward to source a increases, while the amount of energy flowing to the poorly trusted node c decreases significantly. Accorded trust quantities $e_{b \to a}$ and $e_{b \to c}$ amount to $d \cdot \text{in}(b) \cdot (1/1.0625)$ and $d \cdot \text{in}(b) \cdot (0.0625/1.0625)$, respectively. A more severe penalization of poor trust ratings can be achieved by selecting powers above two.

3.2.2.7 Algorithm Outline

Having identified modifications to apply to spreading activation models in order to tailor them for local group trust metrics, we are now able to formulate the core algorithm of Appleseed. Input and output are characterized as follows:

$$\text{Trust}_\alpha : A \times \mathbb{R}_0^+ \times [0, 1] \times \mathbb{R}^+ \to (\text{trust} : A \to \mathbb{R}_0^+) \tag{3.5}$$

The first input parameter specifies trust seed s, the second trust injection e, parameter three identifies spreading factor $d \in [0, 1]$, and the fourth argument binds accuracy threshold T_c, which serves as convergence criterion. Similar to Advogato, the output is an assignment function of trust with domain A. However, Appleseed allows *rankings* of agents with respect to trust accorded. Advogato, on the other hand, only assigns boolean values indicating presence or absence of trust.

Appleseed works with *partial* trust graph information. Nodes are accessed only when needed, i.e., when reached by energy flow. Trust ranks $\text{trust}(x)$, which

correspond to energy(x) in Algorithm 3.2, are initialized to 0. Any unknown node u hence obtains trust$(u) = 0$. Likewise, virtual trust edges for backward propagation from node x to the source are added *at the moment that x is discovered*. In every iteration, for those nodes x reached by flow, the amount of incoming trust is computed as follows:

$$\text{in}(x) = d \cdot \sum_{(p,x) \in E} \left(\text{in}(p) \cdot \frac{W(p,x)}{\sum_{(p,s) \in E} W(p,s)} \right) \tag{3.6}$$

Incoming flow for x is hence determined by all flow that predecessors p distribute along edges (p, x). Note that the above equation makes use of *linear normalization* of relative trust weights. The replacement of linear by nonlinear normalization according to Sect. 3.2.2.6 is straight-forward, though. The trust rank of x is updated as follows:

$$\text{trust}(x) \leftarrow \text{trust}(x) + (1 - d) \cdot \text{in}(x) \tag{3.7}$$

Trust networks generally contain cycles and thus allow no topological sorting of nodes. Hence, the computation of in(x) for reachable $x \in A$ becomes *inherently recursive*. Several iterations for all nodes are required in order to make the computed information converge towards the least fixpoint. The following criterion has to be satisfied for convergence, relying upon accuracy threshold T_c briefly introduced before.

Definition 3.1 (*Termination*) Suppose that $A_i \subseteq A$ represents the set of nodes that were discovered until step i, and $\text{trust}_i(x)$ the current trust ranks for all $x \in A$. Then the algorithm terminates when the following condition is satisfied after step i:

$$\forall x \in A_i : \text{trust}_i(x) - \text{trust}_{i-1}(x) \leq T_c \tag{3.8}$$

Informally, Appleseed terminates when changes of trust ranks with respect to the preceding iteration $i - 1$ are not greater than accuracy threshold T_c.

Moreover, when supposing spreading factor $d > 0$, accuracy threshold $T_c > 0$, and trust source s part of some connected component $G' \subseteq G$ containing at least two nodes, convergence, and thus termination, is guaranteed. The following paragraph gives an informal proof:

Proof (*Convergence of Appleseed*) Assume that f_i denotes step i's quantity of energy flowing through the network, i.e., all the trust that has not been captured by some node x through function $\text{trust}_i(x)$. From Eq. 3.2 follows that in^0 constitutes the *upper boundary* of trust energy floating through the network, and f_i can be computed as follows:

$$f_i = \text{in}^0 - \sum_{x \in A} \text{trust}_i(x) \tag{3.9}$$

Since $d > 0$ and $\exists (s, x) \in E, x \neq s$, the sum of the current trust ranks $\text{trust}_i(x)$ of all $x \in A$ is *strictly increasing* for increasing i. Consequently, $\lim_{i \to \infty} f_i = 0$ holds.

```
func Trust_α (s ∈ A, in^0 ∈ ℝ_0^+, d ∈ [0, 1], T_c ∈ ℝ^+) {
  set in_0(s) ← in^0, trust_0(s) ← 0, i ← 0;
  set A_0 ← {s};
  repeat
    set i ← i + 1;
    set A_i ← A_{i−1};
    ∀x ∈ A_{i−1} : set in_i(x) ← 0;
    for all x ∈ A_{i−1} do
      set trust_i(x) ← trust_{i−1}(x) + (1 − d) · in_{i−1}(x);
      for all (x, u) ∈ E do
        if u ∉ A_i then
          set A_i ← A_i ∪ {u};
          set trust_i(u) ← 0, in_i(u) ← 0;
          add edge (u, s), set W(u, s) ← 1;
        end if
        set w ← W(x, u) / Σ_{(x,u') ∈ E} W(x, u');
        set in_i(u) ← in_i(u) + d · in_{i−1}(x) · w;
      end do
    end do
    set m = max_{y ∈ A_i} {trust_i(y) − trust_{i−1}(y)};
  until (m ≤ T_c)
  return (trust : {(x, trust_i(x)) | x ∈ A_i});
}
```

Algorithm 3.3. Outline of the Appleseed trust metric

Moreover, since termination is defined by some fixed accuracy threshold $T_c > 0$, there exists some step k such that $\lim_{i \to k} f_i \leq T_c$.

3.2.2.8 Parameterization and Experiments

Appleseed allows numerous parameterizations of input variables, some of which are subject to discussion in the section at hand. Moreover, we provide experimental results exposing the observed effects of parameter tuning. Note that all experiments have been conducted on data obtained from "real" social networks: we have written several Web crawling tools to mine the Advogato community Web site and extract trust assertions stated by its more than 8,000 members.[6] Hereafter, we converted all trust data to our trust model proposed in Sect. 3.1.1. The Advogato community server supports four different levels of peer certification, namely OBSERVER, APPRENTICE, JOURNEYER, and MASTER. We mapped these *qualitative* certification levels to quantitative ones, assigning $W(x, y) = 0.25$ for x certifying y

[6] Crawls have been executed in September 2004.

as OBSERVER, $W(x, y) = 0.5$ for an APPRENTICE, and so forth. The Advogato community undergoes rapid growth and our crawler extracted 3,224,101 trust assertions. Preprocessing and data cleansing were thus inevitable, eliminating reflexive trust statements $W(x, x)$ and shrinking trust certificates to reasonable sizes. Note that some eager Advogato members have issued *more than two thousand* trust statements, yielding an overall average outdegree of 397.69 assertions per node.Clearly, this figure is beyond dispute. Hence, applying our set of extraction tools, we tailored the test data obtained from Advogato to our needs and extracted trust networks with specific average outdegrees for the experimental analysis.

Trust Injection

Trust values $\text{trust}(x)$ computed by the Appleseed metric for source s and node x may differ greatly from explicitly assigned trust weights $W(s, x)$. We already mentioned before that computed trust ranks may *not* be interpreted as absolute values, but rather in comparison with ranks assigned to all other peers. In order to make assigned rank values more tangible, though, one might expect that tuning the trust injection in^0 to satisfy the following proposition will align computed ranks and explicit trust statements:

$$\forall (s, x) \in E : \text{trust}(x) \in [W(s, x) - \varepsilon, W(s, x) + \varepsilon] \tag{3.10}$$

However, when assuming reasonably small ε, the approach does not succeed. Recall that *computed* trust values of successor nodes x of s do not only depend on assertions made by s, but also on trust ratings asserted by other peers. Hence, a perfect alignment of explicit trust ratings with computed ones cannot be accomplished. However, we propose a heuristic alignment method, incorporated into Algorithm 3.4, which has proven to work remarkably well in diverse test scenarios. The basic idea is to add another node i and edge (s, i) with $W(s, i) = 1$ to the trust graph $G = (A, E, W)$, treating (s, i) as an indicator to test whether trust injection in^0 is "good" or not. Consequently, parameter in^0 has to be adapted in order to make $\text{trust}(i)$ converge

```
func Trust_heu (s ∈ A, d ∈ [0, 1], T_c ∈ ℝ+) {
  add node i, edge (s, i), set W(s, i) ← 1;
  set in^0 ← 20, ε ← 0.1;
  repeat
    set trust ← Trust_α (s, in^0, d, T_c);
    in^0 ← adapt (W(s, i), trust(i), in^0);
  until trust(i) ∈ [W(s, i) − ε, W(s, i) + ε]
  remove node i, remove edge (s, i);
  return Trust_α (s, in^0, d, T_c);
}
```

Algorithm 3.4. Heuristic weight alignment method

towards $W(s, i)$. The trust metric computation is hence repeated with different values for in^0 until convergence of the explicit and the computed trust value of i is achieved. Eventually, edge (s, i) and node i are removed and the computation is performed one more time. Experiments have shown that our imperfect alignment method yields computed ranks $\text{trust}(x)$ for direct successors x of trust source s which come close to previously specified trust statements $W(s, x)$.

Spreading Factor

Small values for d tend to overly reward nodes close to the trust source and penalize remote ones. Recall that *low d* allows nodes to retain most of the incoming trust quantity for themselves, while *large d* stresses the recommendation of trusted individuals and makes nodes distribute most of the assigned trust to their successor nodes.

Experiment 1 (Spreading factor impact) We compare distributions of computed rank values for three diverse instantiations of d, namely $d_1 = 0.1$, $d_2 = 0.5$, and $d_3 = 0.85$. Our setup is based upon a social network with an average outdegree of 6 trust assignments, and features 384 nodes reached by trust energy spreading from our designated trust source. We furthermore suppose $\text{in}^0 = 200$, $T_c = 0.01$, and *linear* weight normalization. Computed ranks are classified into 11 histogram cells with nonlinear cell width. Obtained output results are displayed in Fig. 3.6. Mind that we have chosen *logarithmic* scales for the vertical axis in order to render the diagram more legible. For d_1, we observe that the largest number of nodes x with ranks $\text{trust}(x) \geq 25$ is generated. On the other hand, virtually no ranks ranging from 0.2 to 1 are assigned, while the number of nodes with ranks smaller than 0.05 is again much higher for d_1 than for both d_2 and d_3. Instantiation $d_3 = 0.85$ exhibits behavior opposed to that of d_1. No ranks with $\text{trust}(x) \geq 25$ are accorded, while interim ranks between 0.1 and 10 are much more likely for d_3 than for both other instantiations of spreading factor d. Consequently, the number of ranks below 0.05 is lowest for d_3.

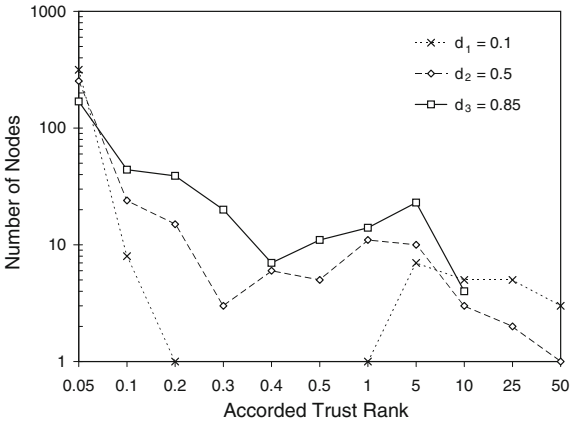

Fig. 3.6 Spreading factor impact

The experiment demonstrates that high values for parameter d tend to distribute trust more evenly, neither overly rewarding nodes close to the source, nor penalizing remote ones too rigidly. On the other hand, low d assigns high trust ranks to very few nodes, namely those which are closest to the source, while the majority of nodes obtains very low trust rank. We propose to set $d = 0.85$ for general use.

Convergence

We already mentioned before that the Appleseed algorithm is *inherently recursive*. Parameter T_c represents the ultimate criterion for termination. We demonstrate through an experiment that convergence is reached very fast, no matter how large the number of nodes trust is flowing through, and no matter how large the initial trust injection.

Experiment 2 (Convergence rate) The trust network we consider has an average outdegree of 5 trust statements per node. The number of nodes for which trust ranks are assigned amounts to 572. We suppose $d = 0.85$, $T_c = 0.01$, and *linear* weight normalization. Two separate runs were computed, one with trust activation $\text{in}_1 = 200$, the other with initial energy $\text{in}_2 = 800$. Figure 3.7 demonstrates the rapid convergence of both runs. Though the trust injection for the second run is 4 times as high as for the first, convergence is reached in only few more iterations: run one takes 38 iterations, run two terminates after 45 steps.

For both runs, we assumed accuracy threshold $T_c = 0.01$, which is extremely small and accurate beyond necessity already. However, experience taught us that convergence takes place rapidly even for very large networks and high amounts of trust injected, so that assuming the latter value for T_c poses no scalability issues. In fact, the amount of nodes taken into account for trust rank assignment in the

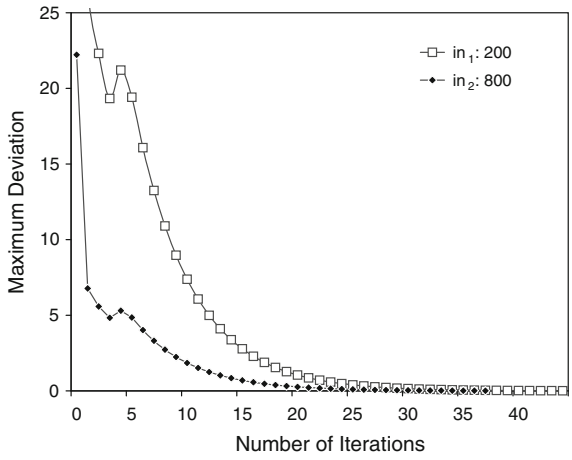

Fig. 3.7 Convergence of Appleseed

above example well exceeds practical usage scenarios: mind that the case at hand demands 572 files to be fetched from the Web, complaisantly supposing that these pages are cached after their first access. Hence, we claim that the actual bottleneck of group trust computation is *not* the Appleseed metric itself, but downloads of trust resources from the network. This bottleneck might also be the reason for selecting thresholds T_c greater than 0.01, in order to make the algorithm terminate after fewer node accesses.

Testbed Design and Experimental Trials

Trust metrics and models for trust propagation have to be *intuitive*, i.e., humans must eventually comprehend *why* agent a_i has been accorded a higher trust rank than a_j and come to similar results when asked for personal judgement. Consequently, we implemented our own testbed, which graphically displays social networks. We made use of the YFILES [39] library to perform complex graph drawing and layouting tasks. The testbed allows for parameterizing Appleseed through dialogs. Detailed output is provided, both graphical and textual. Graphical results comprise the highlighting of nodes with trust ranks above certain thresholds, while textual results return quantitative trust ranks of all accessed nodes, the number of iterations, and so forth. We also implemented the Advogato trust metric and incorporated the latter into our testbed. Hereby, our implementation of Advogato does not require a priori complete trust graph information, but accesses nodes "just in time", similar to Appleseed. All experiments were conducted on top of the testbed application.

3.2.3 Comparison of Advogato and Appleseed

Advogato and Appleseed are both implementations of local group trust metrics. Advogato has already been successfully deployed into the Advogato online community, though quantitative evaluation results have not been provided yet. In order to evaluate the fitness of Appleseed as an appropriate means for group trust computation, we relate our approach to Advogato for qualitative comparison:

(F.1) *Attack-resistance.* This property defines the behavior of trust metrics in case of malicious nodes trying to invade into the system. For evaluation of attack-resistance capabilities, we have briefly introduced the "bottleneck property" in Sect. 3.2.1.2, which holds for Advogato. In order to recapitulate, suppose that s and t are nodes and connected through trust edge (s, t). Node s is assumed good, while t is an attacking agent trying to make good nodes trust malevolent ones. In case the bottleneck property holds, manipulation "on the part of bad nodes does not affect the trust value" [22]. Clearly, Appleseed satisfies the bottleneck property, for nodes cannot raise their impact by modifying the structure of trust statements they issue. Bear in mind that the amount of trust accorded to agent t *only* depends on his predecessors and does not increase when t adds more nodes. Both,

spreading factor d and normalization of trust statements, ensure that Appleseed maintains attack-resistance properties according to Levien's definition.

(F.2) *Eager truster penalization.* We have indicated before that issuing multiple trust statements dilutes trust accorded to successors. According to Guha [14], this does not comply with real world observations, where statements of trust "do not decrease in value when the user trusts one more person [...]". The malady that Appleseed suffers from is common to many trust metrics, most notably those based upon finding principal eigenvectors [18, 31, 35]. On the other hand, the approach pursued by Advogato does *not* penalize trust relationships asserted by eager trust dispensers, for node capacities do not depend on *local* information. Remember that capacities of nodes pertaining to level l are assigned based on the capacity of level $l-1$, as well as the *overall* outdegree of nodes part of that level. Hence, Advogato *encourages* agents issuing numerous trust statements, while Appleseed *penalizes* overly abundant trust certificates.

(F.3) *Deterministic trust computation.* Appleseed is deterministic with respect to the assignment of trust rank to agents. Hence, for any arbitrary trust graph $G = (A, E, W)$ and for every node $x \in A$, linear equations allow for characterizing the amount of trust assigned to x, as well as the quantity that x accords to successor nodes. Advogato, however, is *non-deterministic*. Though the *number* of trusted agents, and therefore the computed maximum flow size, is determined for given input parameters, the set of agents is not. Changing the order in which trust assertions are issued may yield different results. For example, suppose $C_A(s) = 1$ holds for trust seed s. Furthermore, assume s has issued trust certificates for two agents, b and c. The actual choice between b or c as trustworthy peer with maximum flow *only depends on the order* in which nodes are accessed.

(F.4) *Model and output type.* Basically, Advogato supports non-weighted trust statements only. Appleseed is more versatile by virtue of its trust model based on *weighted* trust certificates. In addition, Advogato returns one set of trusted peers, whereas Appleseed assigns *ranks* to agents. These ranks allow to select most trustworthy agents first and relate them to each other with respect to their accorded rank. Hereby, the definition of thresholds for trustworthiness is left to the user who can thus tailor relevant parameters to fit different application scenarios. For instance, raising the application-dependent threshold for the selection of trustworthy peers, which may be either an absolute or a relative value, allows for enlarging the neighborhood of trusted peers. Appleseed is hence more adaptive and flexible than Advogato.

The afore-mentioned characteristics of Advogato and Appleseed are briefly summarized in Table 3.1.

Table 3.1 Characteristics of Advogato and Appleseed

	Feature **F.1**	Feature **F.2**	Feature **F.3**	Feature **F.4**
Advogato	Yes	No	No	Boolean
Appleseed	Yes	Yes	Yes	Ranking

3.3 Distrust

The notion of distrust is one of the most controversial topics when defining trust metrics and trust propagation. Most approaches completely *ignore* distrust and only consider *full* trust or *degrees of trust* [4, 23, 28, 30, 33, 35]. Others, among those [1, 3, 6, 13], allow for distrust ratings, though, but do not consider the subtle semantic differences that exist between those two notions, i.e., trust and distrust. Consequently, according to [9], "distrust is regarded as just the other side of the coin, that is, there is generally a symmetric scale with complete trust on one end and absolute distrust on the other". Furthermore, some researchers equate the notion of distrust with *lack of trust information*. However, in his seminal work on the essence of trust, Marsh [26] has already pointed out that those two concepts, i.e., lack of trust and distrust, may *not* be intermingled. For instance, in absence of trustworthy agents, one might be more prone to accept recommendations from non-trusted persons, being non-trusted probably because of lack of prior experiences [27], than from persons we explicitly *distrust*, the distrust resulting from bad past experiences or deceit. However, even Marsh pays little attention to the specifics of distrust.

Gans et al. [9] were among the first to recognize the importance of distrust, stressing the fact that "distrust is an irreducible phenomenon that cannot be offset against any other social mechanisms", including trust. In their work, an explicit distinction between confidence, trust, and distrust is made. Moreover, the authors indicate that distrust might be highly relevant to social networks. Its impact is not inherently negative, but may also influence the network in an extremely positive fashion. However, the primary focus of this work is on methodology issues and planning, not considering trust assertion evaluations and propagation through appropriate metrics.

Guha et al. [15] acknowledge the immense role of distrust with respect to trust propagation applications, arguing that "distrust statements are very useful for users to debug their web of trust" [14]. For example, suppose that agent a_i blindly trusts a_j, which again blindly trusts a_k. However, a_i completely distrusts a_k. The distrust statement hence ensures that a_i will *not* accept beliefs and ratings from a_k, irrespective of him trusting a_j trusting a_k.

3.3.1 Semantics of Distrust

The non-symmetrical nature of distrust and trust, being two dichotomies, has already been recognized by recent sociological research [25]. In this section, we investigate the differences between distrust and trust with respect to inference opportunities and the propagation of beliefs.

3.3.1.1 Distrust as Negated Trust

Interpreting distrust as the negation of trust has been adopted by many trust metrics, among those trust metrics proposed by Abdul-Rahman and Hailes [1, 2], Jøsang

et al. [17], and Chen and Yeager [6]. Basically, these metrics compute trust values by analyzing *chains* of trust statements from source s to target t, eventually merging them to obtain an aggregate value. Each chain hereby becomes synthesized into one single number through *weighted multiplication* of trust values along trust paths. Serious implications resulting from the assumption that trust concatenation relates to multiplication [35], and distrust to negated trust, arise when agent a_i distrusts a_j, who distrusts a_k[7]:

$$\neg\,\mathrm{trust}(a_i, a_j) \wedge \neg\,\mathrm{trust}(a_j, a_k) \models \mathrm{trust}(a_i, a_k) \tag{3.11}$$

Jøsang et al. [17] are aware of this rather unwanted effect, but do not question its correctness, arguing that "the enemy of your enemy could well be your friend". Guha [14], on the other hand, indicates that two distrust statements canceling out each other commonly does *not* reflect desired behavior.

3.3.1.2 Propagation of Distrust

The *conditional transitivity* of trust [1] is commonly agreed upon and represents the foundation and principal premiss that trust metrics rely upon. However, no consensus in literature has been achieved with respect to the *degree* of transitivity and the decay rate of trust. Many approaches therefore explicitly distinguish between *recommendation* trust and *direct* trust [1, 4, 6, 17, 28] in order to keep apart the transitive fraction of trust from the non-transitive. Hence, in these works, only the *ultimate* edge within the trust chain, i.e., the one linking to the trust target, needs to be direct, while all others are supposed to be recommendations. For the Appleseed trust metric, this distinction is made through the introduction of spreading factor d. However, the conditional transitivity property of trust does not equally extend to distrust. The case of double negation through distrust propagation has already been considered. Now suppose, for instance, that a_i distrusts a_j, who trusts a_k. Supposing distrust to propagate through the network, we come to make the following inference:

$$\mathrm{distrust}(a_i, a_j) \wedge \mathrm{trust}(a_j, a_k) \models \mathrm{distrust}(a_i, a_k) \tag{3.12}$$

The above inference is more than questionable, for a_i penalizes a_k simply for being trusted by an agent a_j that a_i distrusts. Obviously, this assumption is not sound and does not reflect expected real-world behavior. We assume that distrust does not allow for making direct inferences *of any kind*. This conservative assumption well complies with [14].

[7] We oversimplify by using predicate calculus expressions, supposing that trust, and hence distrust, is fully transitive.

3.3.2 Incorporating Distrust into Appleseed

We compare our distrust model with Guha's approach, making similar assumptions. Guha computes trust by means of *one global* group trust metric, similar to PageRank [31]. For distrust, he proposes two candidate approaches. The first one directly integrates distrust into the iterative eigenvector computation and comes up with one single measure combining both trust and distrust. However, in networks dominated by distrust, the iteration might not converge [14]. The second proposal first computes trust ranks by trying to find the dominant eigenvector, and then computes separate distrust ranks in one single step, based upon the iterative computation of trust ranks. Suppose that D_{a_i} is the set of agents who distrust a_i:

$$\text{DistrustRank}(a_i) = \frac{\sum_{a_j \in D_{a_i}} \text{TrustRank}(a_j)}{|D_{a_i}|} \tag{3.13}$$

The problem we perceive with this approach refers to *superimposing* the computation of distrust ranks *after* trust rank computation, which may yield some strange behavior: suppose an agent a_i who is highly controversial by engendering ambiguous sentiments, i.e., on the one hand, there are numerous agents that *trust* a_i, while on the other hand, there are numerous agents who *distrust* a_i. With the approach proposed by Guha, a_i's impact for distrusting other agents is huge, resulting from his immense positive trust rank. However, this should clearly not be the case, for a_i is subject to tremendous distrust himself, thus leveling out his high trust rank.

Hence, for our own approach, we intend to *directly* incorporate distrust into the iterative process of the Appleseed trust metric computation, and not superimpose distrust afterwards. Several pitfalls have to be avoided, such as the risk of non-convergence in case of networks dominated by distrust [14]. Furthermore, in absence of distrust statements, we want the distrust-enhanced Appleseed algorithm, which we denote by Trust_{α^-}, to yield results identical to those engendered by the original version Trust_{α}.

3.3.2.1 Normalization and Distrust

First, the trust normalization procedure has to be adapted. We suppose normalization of weights to the power of q, as has been discussed in Sect. 3.2.2.6. Let $\text{in}(x)$, the trust influx for agent x, be *positive*. As usual, we denote the global spreading factor by d, and quantified trust statements from x to y by $W(x, y)$. Function $\text{sign}(x)$ returns the sign of value x. Note that from now on, we assume $W : E \rightarrow [-1, +1]$, for degrees of *distrust* need to be expressible. Then the trust quantity $e_{x \rightarrow y}$ passed from x to successor y is computed as follows:

$$e_{x \rightarrow y} = d \cdot \text{in}(x) \cdot \text{sign}(W(x, y)) \cdot w, \tag{3.14}$$

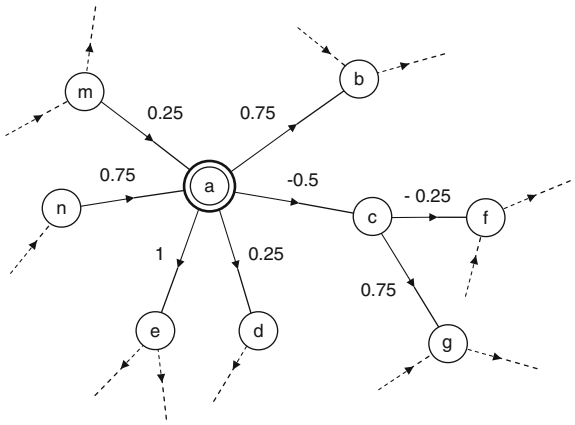

Fig. 3.8 Network augmented by distrust

where

$$w = \frac{|W(x, y)|^q}{\sum_{(x,s) \in E} |W(x, s)|^q}$$

The accorded quantity $e_{x \to y}$ becomes *negative* if $W(x, y)$ is negative, i.e., if x distrusts y. For the relative weighting, the *absolute* values $|W(x, s)|$ of all weights are considered. Otherwise, the denominator could become negative, or positive trust statements could become boosted unduly. The latter would be the case if the sum of positive trust ratings *only slightly* outweighed the sum of negative ones, making the denominator converge towards zero. An example demonstrates the computation process:

Example 3.2 (Distribution of Trust and Distrust) We assume the trust network as depicted in Fig. 3.8. Let the trust energy influx into node a be $\text{in}(a) = 2$, and global spreading factor $d = 0.85$. For simplicity reasons, backward propagation of trust to the source is *not* considered. Moreover, we suppose *linear* weight normalization, thus $q = 1$. Consequently, the denominator of the normalization equation is $|0.75| + |-0.5| + |0.25| + |1| = 2.5$. The trust energy that a distributes to b hence amounts to $e_{a \to b} = 0.51$, whereas the energy accorded to the distrusted node c is $e_{a \to c} = -0.34$. Furthermore, we have $e_{a \to d} = 0.17$ and $e_{a \to e} = 0.68$.

Observe that trust energy becomes *lost* during distribution, for the sum of energy accorded along outgoing edges of a amounts to 1.02, while 1.7 was provided for distribution. The effect results from the negative trust weight $W(a, c) = -0.5$.

3.3.2.2 Distrust Allocation and Propagation

We now analyze the case where the influx $\text{in}(x)$ for agent x is *negative*. In this case, the trust allocated for x will also be negative, i.e., $\text{in}(x) \cdot (1 - d) < 0$. Moreover, the

energy $\text{in}(x) \cdot d$ that x may distribute among successor nodes will be negative as well. The implications are those which have been mentioned in Sect. 3.3.1, i.e., distrust as negation of trust and propagation of distrust. For the first case, refer to node f in Fig. 3.8 and assume $\text{in}(c) = -0.34$, which is derived from Example 3.2. The trusted agent a distrusts c who distrusts f. Eventually, f would be accorded $d \cdot (-0.34) \cdot (-0.25)$, which is *positive*. For the second case, node g would be assigned the *negative* trust quantity $d \cdot (-0.34) \cdot (0.75)$, simply for being trusted by f, who is distrusted. Both unwanted effects can be avoided by not allowing distrusted nodes to distribute *any energy at all*. Hence, more formally, we introduce a novel function $\text{out}(x)$:

$$\text{out}(x) = \begin{cases} d \cdot \text{in}(x), & \text{if } \text{in}(x) \geq 0 \\ 0, & \text{else} \end{cases} \tag{3.15}$$

This function then has to replace $d \cdot \text{in}(x)$ when computing the energy distributed along edges from x to successor nodes y:

$$e_{x \to y} = \text{out}(x) \cdot \text{sign}(W(x, y)) \cdot w, \tag{3.16}$$

where

$$w = \frac{|W(x, y)|^q}{\sum_{(x,s) \in E} |W(x, s)|^q}$$

This design decision perfectly aligns with assumptions made in Sect. 3.3.1 and prevents the inference of unwanted side-effects mentioned before. Furthermore, one can see easily that the modifications introduced *do not affect* the behavior of Algorithm 3.3 when not considering relationships of distrust.

3.3.2.3 Convergence

In networks largely or entirely dominated by distrust, the extended version of Appleseed is still guaranteed to converge. We therefore briefly outline an informal proof, based on Proof 3.2.2.7:

Proof (*Convergence in presence of distrust*) Recall that only *positive* trust influx $\text{in}(x)$ becomes propagated, which has been indicated in Section 3.3.2.2. Hence, all we need to show is that the overall quantity of *positive* trust distributed in computation step i cannot be augmented through the presence of distrust statements. In other words, suppose that $G = (A, E, W)$ defines an arbitrary trust graph, containing quantified trust statements, but *no distrust*, i.e., $W : E \to [0, 1]$. Now consider another trust graph $G' = (A, E \cup D, W')$, which contains additional edges D, and weight function $W' = W \cup (D \to [-1, 0[)$. Hence, G' augments G by additional distrust edges between nodes taken from A. We now perform two parallel computations with the extended version of Appleseed, one operating on G and the other on G'. In every step, and for every trust edge $(x, y) \in E$ for G, the distributed energy $e_{x \to y}$ is greater or equal to the respective counterpart on G', because the denominator

of the fraction given in Eq. 3.16 can only become *greater* through additional distrust outedges. Second, for the computation performed on G', negative energy distributed along edge (x, y) can only *reduce* the trust influx for y and may hence even accelerate convergence.

However, as can be observed from the proof, there exists one serious implication arising from having distrust statements in the network: the overall accorded trust quantity does *not* equal the initially injected energy anymore. Moreover, in networks dominated by distrust, the overall trust energy sum may even be *negative*.

Experiment 3 (Network impact of distrust) We observe the number of iterations until convergence is reached, and the overall accorded trust rank of 5 networks. The structures of all these graphs are identical, being composed of 623 nodes with an average indegree and outdegree of 9. The only difference applies to the assigned weights, where the first graph contains no distrust statements at all, while 25 % of all weights are negative for the second, 50 % for the third, and 75 % for the fourth. The fifth graph contains nothing but distrust statements. The Appleseed parameters are identical for all 5 runs, having backward propagation enabled, an initial trust injection $in^0 = 200$, spreading factor $d = 0.85$, convergence threshold $T_c = 0.01$, *linear* weight normalization, and no upper bound on the number of nodes to unfold. Figure 3.9a clearly demonstrates that the number of iterations until convergence, given on the vertical axis, *decreases* with the proportion of distrust increasing, observable along the horizontal axis. Likewise, the overall accorded trust rank, indicated on the vertical axis of Fig. 3.9b, decreases rapidly with increasing distrust, eventually dropping below zero. The same experiment was repeated for another network with 329 nodes, an average indegree and outdegree of 6, yielding similar results.

The effects observable in Experiment 3 only marginally affect the ranking itself, for trust ranks are interpreted *relative* to each other. Moreover, compensation for lost trust energy may be achieved by boosting the initial trust injection in^0.

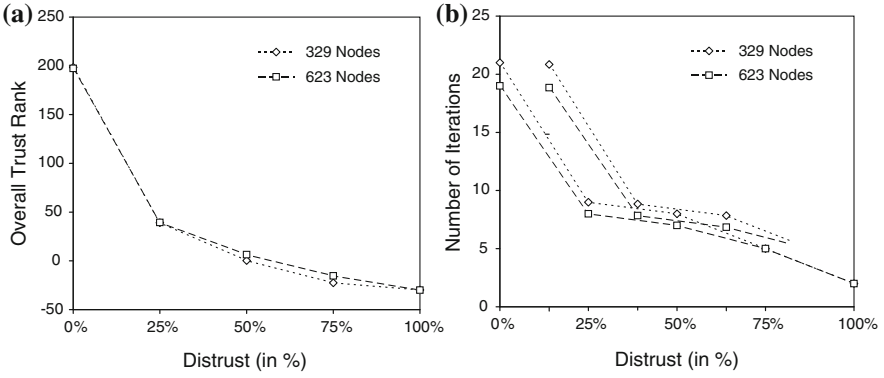

Fig. 3.9 Network impact of distrust

3.4 Expanding Network Coverage

Among the network-based algorithms for computing trust, there is one common problem: coverage. In social networks there are often many users who are disconnected from the main cluster or who are connected in a way that computing an accurate trust value would be difficult. This naturally leads to the question of how we can improve network coverage, and potentially accuracy, in trust computation.

We propose one solution to this which incorporates similarity measures grounded in our sociological understanding of trust. Sociological definitions of trust have two components: a belief and a commitment. For example, in a context, if Alice trusts Bob, it implies that Alice believes that Bob will provide useful information *and* that she is willing to take action based on that information [10]. If we consider this in the context of information on the Web, trust in a person means that the user is willing to take actions, like buying a product, based on others' reviews. This, of course, gets to the core of why we want to compute trust in the first place; if we know how much the user trusts the author of some online content, we can use that to help optimally sort, filter, and aggregate the information.

Network flow-based algorithms, like Appleseed and Advogato presented above work very well on connected graph components, but they cannot infer trust between people who are not connected by paths in the social network. In those situations, trust can be inferred from other sources of information. In our previous work, we identified a series of similarity measures drawn from underlying data that can estimate trust effectively. We will discuss this work further in Sect. 3.4.1. This method can infer trust between any two users as long as they have rated a common set of items upon which to compute similarity. However, computing trust based only on nuanced similarity measures loses some of the insights that come from the network.

In some cases, we will be able to compute trust values with both algorithms for a pair of users. In other cases, we may be able to compute only one (or perhaps neither). A combination of the methods will obviously achieve better coverage, but how to effectively use both values when available is an open question. In this section, we present a model that integrates trust computed from social networks and trust inferred from data similarity. We show results on how to optimally use both models and demonstrate their accuracy on a real-world dataset.

3.4.1 Revisiting Trust Inference Algorithms

Above, we discussed two major trust inference algorithms: Advogato and Appleseed. For the experiments here, we used the TidalTrust [12] algorithm, a simple trust inference algorithm that gives a good indication of how many pairs of users a network-based algorithm can reach. Readers will note similarities between TidalTrust and the algorithms presented above.

3.4.1.1 TidalTrust: An Algorithm for Inferring Trust

TidalTrust is a modified breadth-first search-base algorithm. The source's inferred trust rating for the sink, denoted $t_{s,p}$, is a weighted average of source s's neighbors' ratings of sink p. The source node begins a search for the sink. It will poll each of its neighbors to obtain their rating of the sink. If the neighbor has a direct rating of the sink, that value is returned. If the neighbor does not have a direct rating for the sink, it queries all of its neighbors for their ratings, computes the weighted average, and returns the result. Each neighbor repeats this process. Essentially, the nodes perform a breadth-first search from the source to the sink, and then inferred values are passed back to the source. The basic process of values for the sink flowing back to the source are shown in Fig. 3.10.

Network-based inference algorithms rely on the social network. This provides a benefit because recommendations can be made for users who have rated no items because trust is inferred from the social connections. However, it has a corresponding drawback that trust can only be computed when users are connected in that network.

TidalTrust incorporates two factors to limit the size of the search and improve accuracy. Previous research has shown the following [11]:

- Shorter paths have a lower error.
- Using nodes with higher trust ratings leads to lower error.

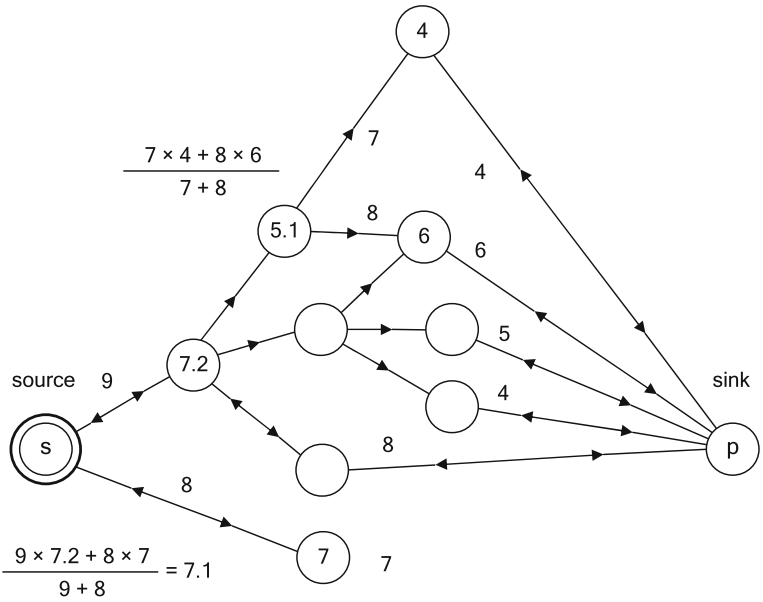

Fig. 3.10 An illustration of direct trust values between nodes a and b, $t_{a,b}$, and between nodes b and c, $t_{b,c}$. Using a trust inference algorithm, it is possible to compute a value to recommend how much a may trust c, $t_{a,c}$

Limiting the depth of TidalTrust's search should lead to more accurate results, since the error often increases as depth increases. If accuracy decreases as path length increases, as the earlier analysis suggests, then shorter paths are more desirable. However, the tradeoff is that fewer nodes will be reachable if a limit is imposed on the path depth. To balance these factors, the path length can vary from one computation to another. Instead of a fixed depth, the shortest path length required to connect the source to the sink becomes the depth. This preserves the benefits of a shorter path length without limiting the number of inferences that can be made.

The previous results also indicate that the most accurate information will come from the highest trusted neighbors. To incorporate this into the algorithm, we establish a minimum trust threshold, and only consider connections in the network with trust ratings at or above the threshold. This value cannot be fixed before the search because we cannot predict what the highest trust value will be along the possible paths. If the value is set too high, some nodes may not have assigned values and no path will be found. If the threshold is too low, then paths with lower trust may be considered when it is not necessary. We define a variable, max, that represents the largest trust value that can be used as a minimum threshold such that a path can be found from source to sink. Our max is computed while searching for paths to the sink by tracking trust values that have been seen.

TidalTrust is a modified breadth-first search. The inferred trust rating of source s for sink p, denoted $t_{s,p}$, is a weighted average of the source's neighbors' ratings of the sink. This is succinctly represented as follows:

$$t_{s,p} = \frac{\sum_{j \in \text{adj}(s) \wedge t_{s,j} \geq \max} t_{s,j} \times t_{j,p}}{\sum_{j \in \text{adj}(s) \wedge t_{s,j} \geq \max} t_{s,j}} \tag{3.17}$$

The source node begins a search for the sink. It will poll each of its neighbors to obtain their rating of the sink. If the neighbor has a direct rating of the sink, that value is returned. If the neighbor does not have a direct rating for the sink, it queries all of its neighbors for their ratings, computes the weighted average as shown above, and returns the result .

To improve the accuracy of the algorithm, path length and path strength considerations are included. Each node that is reached performs this process, keeping track of the current depth from the source. Each node will also keep track of the strength of the path to it. Nodes adjacent to the source will record the source's rating assigned to them. Each of those nodes will poll their neighbors. The strength of the path to each neighbor is the minimum of the source's rating of the node and the node's rating of its neighbor. The neighbor records the maximum strength path leading to it. Once a path is found from the source to the sink, the depth is set at the maximum depth allowable. Since the search is proceeding in a breadth-first search fashion, the first path found will be at the minimum depth. The search will continue to find any other paths at the minimum depth. Once this search is complete, the trust threshold (max) is established by taking the maximum of the trust paths leading to the sink. With the max

value established, each node can complete the calculations of a weighted average by taking information from nodes that they have rated at or above the max threshold.

The accuracy of this algorithm is addressed in depth in [10, 12]. While the error will very from network to network, our experiments in two real world social networks show the results to be accurate to within about 10 %.

3.4.1.2 Similarity-Based Trust Inference

It has been long known in the sociological literature and more recently shown in the computer science literature that trust correlates strongly with similarity between people [42, 43]. In our previous work [40] we showed that in addition to overall similarity, there is also a correlation between trust and several nuanced similarity measures. In a context where people rate items, those ratings can be used to compute values that go beyond simple similarity. Specifically, trust between people is tied to the largest single difference over items they have both rated, and to the agreement on movies where one user has given extreme ratings. We also showed that some people tend to be more trusting than others, and thus inferred trust values can be adjusted up or down to account for this. We used these nuanced similarity measures in this research.

3.4.1.3 Experimental Network

When working with trust, data is usually one of the greatest challenges. Trust information is private, and for that reason there are no publicly available datasets with this information. In 2004, we developed and launched FilmTrust,[8] a Web-based social network centered around movies. Users create profiles and link to friends and rate how much they trust each friend's opinion about movies. Users can also rate and review movies. The network has been live on the web and growing on its own since 2004. Thus, it serves as a useful real-world dataset upon which we can run experiments.

The FilmTrust network has 1,610 total members. Many do not have any friends in the social network; 712 people have at least one friend and there are 1,465 edges in the network. The average trust rating is 6.83. The network has a central giant component, and many small subnetworks.

Most users have rated movies in the network; 1,250 people have rated at least one movie. These movie ratings are used in the similarity-based trust inference techniques. In total, we have either trust ratings or movie ratings from 1,339 of the 1,610 users.

3.4.2 Experimental Analysis

Our experimental analysis executes the network-based and similarity-based algorithms over the FilmTrust network, and then follows that with the integrated trust

[8] See http://trust.mindswap.org/FilmTrust.

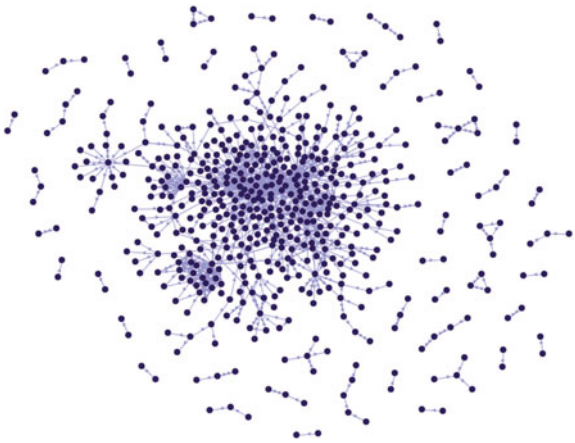

Fig. 3.11 A visualization of the FilmTrust network

inference algorithm. We compare these methods for accuracy and coverage of the network. We begin this section by describing the setup for each algorithm and then present the results of our experiments. We show that an integrated model does produce significantly more accurate results and better coverage than either method alone (Fig. 3.11).

3.4.2.1 Trust Inference Setup

The *network*-based trust inference algorithm, FilmTrust, was able to run directly on the FilmTrust network, so no special setup was required. For the *similarity*-based algorithm, we needed to develop a method for integrating our earlier insights on similarity measures that relate to trust into an algorithm. In that mentioned earlier work we identified four measures made over users' item ratings that relate to trust: overall similarity, similarity on extremes (items that received very high or very low ratings from a user), the single largest difference between users on a given item, and the source's propensity to trust. We computed similarity measures in two ways: as mean average error (MAE) and using the Pearson correlation. Thus, we had six total measures: the average difference (AD), overall correlation (COR), average difference on extremes (XD), correlation on extremes (XCOR), the single largest difference (MD), and the source's propensity to trust (PT). A linear combination of these values can predict trust and is given in Eq. 3.18, where ω indicates the weight given to each measure.

$$\begin{aligned} t_{s,p} = {} & \omega_{\text{AD}} \times \text{AD} + \omega_{\text{COR}} \times \text{COR} + \omega_{\text{XD}} \times \text{XD} \\ & + \omega_{\text{XCOR}} \times \text{XCOR} + \omega_{\text{MD}} \times \text{MD} + \omega_{\text{PT}} \times \text{PT} \end{aligned} \tag{3.18}$$

It is worth noting that for some pairs of people, some of these values may be unavailable. For example, it is common for users to have no movies in common

Table 3.2 Weights for similarity measures determined in a multivariate linear regression analysis

Weight	Extreme values available	No extreme values
ω_{AD}	−1.8084	−0.5951
ω_{COR}	1.0589	0.8269
ω_{XD}	0.1751	
ω_{XCOR}	0.0655	
ω_{MD}	0.2489	0.1145
ω_{PT}	1.0568	0.9946

where the source has assigned an extreme rating. Thus, the weights will be different depending on whether or not the two measures on extreme-rated items are available. The weights (ω values) will vary between networks based on the behavior of the users and context of the data. To determine the optimal weights for the FilmTrust dataset, we ran a multivariate linear regression analysis. To achieve the most meaningful results from the regression, we selected a subset of node pairs who had at least 10 items in common. To compute values for XD and XCOR, we required at least 3 items in common with extreme ratings from the source. The results of this regression analysis are shown in Table 3.2.

3.4.2.2 Integrated Trust Model

Combining the network-based and similarity-based trust inference algorithms into an integrated algorithm has two major benefits. First, it provides a more thorough coverage so trust can be inferred for a greater number of individuals. If someone is not in the social network, the ratings similarity method can be used. If they have not rated enough items but have friends, the social network method can be used. The second benefit is the potential for improved accuracy when trust can be inferred using both the network-based trust inference algorithm and the similarity-based inference algorithm.

Our approach was to use a linear combination of the trust values produced from each base algorithm. To combine these values, we ran a multivariate linear regression analysis using known trust values as a ground truth. We found $\omega_{sim} = 0.869$ and $\omega_{net} = 0.195$. Thus, the integrated trust value was computed as follows:

$$T_{int} = \omega_{sim} * T_{sim} + \omega_{net} \times T_{net} \tag{3.19}$$

3.4.3 Results

The next sections present results on coverage and accuracy of the presented approaches. These experiments were executed on real-world social networks on the Web.

3.4.3.1 Coverage

The FilmTrust network we used for our experiments has 1,610 nodes, but only 1,339 have input any data to the system. Thus, we have $1{,}339 \times 1{,}338 = 1{,}791{,}582$ total pairs for which trust can be inferred. The network has a somewhat high rate of members who have no friends in the social network. Of the 1,339 participating members, only 712 (53.17 %) have any social connections in the network. Recall also that the edges are directed in the network. Nodes must have outgoing edges in order to infer trust to any other nodes. Only 480 nodes have outgoing edges. Thus, we would expect to be able to infer trust values for no more than $480 \times 712 = 341{,}280$ pairs of users if we are using a network-based trust inference algorithm. Note that *any* algorithm that infers trust by searching paths in the network will have this limit. TidalTrust, which infers trust for any sink reachable from the source, was able to compute values for 69,016 pairs. While this is less than 4 % of the total number of pairs, it is just over 20 % of the nodes who have some social network data. The other 80 % of pairs for which a value represent nodes that have no indirect connections in the network (e.g. pairs of nodes not connected to the giant component).

Using the similarity-based method, trust can be inferred for any pair of users who have data in common. In the FilmTrust network, 503,912 (28 %) pairs of users have at least one item in common. However, one single item is a very weak basis for computing trust, and is insufficient for computing the correlation measures we need for our method. We set a lower threshold of 3 items in common for computing a similarity-based trust value. With this restriction, 302,336 pairs had an inferred trust value, which is 16.88 % of the total number of pairs

Not surprisingly, when used together, these methods give better coverage than either achieves on its own. We could infer trust for 342,504 pairs, just shy of 20 % of the network. This is less than the sum of the coverage of the two methods, since some pairs have inferred trust from both algorithms. Of the 342,504 pairs for which trust could be inferred, values from *both* methods were available for 28,878 pairs (8.43 % of the covered pairs, and 1.6 % of all pairs).

It is important to note that these coverage rates are unique to the network we are examining here. Other networks may have vastly different coverage rates based on the behavior of the users. Previous work has shown dramatically different rates of participation in the social networking component of websites. However, the improved coverage using two types of algorithms is not surprising and should expected in other datasets (Table 3.3).

Table 3.3 Pairs of nodes for which trust can be inferred using different methods

Method	Coverage	(in % of all pairs)
Similarity-based	302,366	(16.88 %)
Network-based	69,016	(3.85 %)
Integrated	342,504	(19.12 %)

3.4.3.2 Accuracy

Improved coverage is useful, but can results from two algorithms also improve the accuracy of trust inferences? We investigated this by using the 1,465 pairs of nodes with a known trust value, i.e. where one user had assigned a trust rating to another in the social network. These were our ground truth values against which the inferred values were compared.

With the TidalTrust algorithm, we tested accuracy by selecting a pair of nodes with a known trust rating, ignoring the edge between them in the network, and then running the algorithm to infer a trust value. This would allow us to see what the algorithm would infer if the relationship did not exist. The similarity-based algorithm was run for any pair of nodes with 3 or more rated items in common.

Of the 1,465 pairs, TidalTrust inferred values for 881 pairs. The similarity-based approach found values for 763 pairs. The intersection of these sets where both methods inferred results comprises 490 pairs. We used those 490 pairs for our analysis so we were comparing the accuracy of all three methods over the same users.

We compared the accuracy of the inferred trust value computed with the integrated trust equation to the accuracy of the trust value inferred using each algorithm individually. Accuracy was measured as both mean absolute error (MAE) and root mean square error (RMSE). For each accuracy measure, an ANOVA indicated statistically significant differences among the three methods. For both accuracy measures, a two-tailed Student's t-test showed that the integrated trust method was significantly more accurate than both the network-based and similarity-based trust estimates alone.

This indicates that using both trust inference techniques, the results not only provide inferred trust values for more pairs of users than either method could do alone, but they also can be combined to provide more accurate trust inferences for pairs where both methods generate results.

3.5 Discussion and Outlook

In this chapter, we advocated the need for local group trust metrics, presenting our metric Appleseed. Appleseed's nature largely resembles Advogato, bearing similar complexity and attack-resistance properties, but offers one particular feature that makes Appleseed much more suitable for certain applications than Advogato: the ability to compute *rankings* of peers according to their trustworthiness rather than *binary* classifications into trusted and untrusted agents (Table 3.4).

Table 3.4 Accuracy of trust inference methods

Method	Accuracy (MAE)	Accuracy (RMSE)
Similarity-based	1.36	1.78
Network-based	1.88	2.47
Integrated	**1.27**	**1.67**

Originally designed as an approach to social filtering within our decentralized recommender framework [41], Appleseed suits other application scenarios as well, such as group trust computation in online communities, open rating systems, ad-hoc and peer-to-peer networks.

For instance, Appleseed could support peer-to-peer-based file-sharing systems in reducing the spread of self-replicating inauthentic files by virtue of trust propagation [18]. In that case, explicit trust statements, resulting from direct interaction, would reflect belief in someone's endeavor to provide authentic files.

We also showed that augmenting group trust metrics with additional information that can indicate trust, such as nuanced similarity over rated items, can improve the number of user pairs for whom trust can be inferred.

References

1. Abdul-Rahman, A., Hailes, S.: A distributed trust model. In: New Security Paradigms Workshop, pp. 48–60. Cumbria, UK (1997)
2. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: Proceedings of the 33rd Hawaii International Conference on System Sciences. Maui, HI, USA (2000)
3. Aberer, K., Despotovic, Z.: Managing trust in a peer-2-peer information system. In: Paques, H., Liu, L., Grossman, D. (eds.) Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 310–317. ACM Press, Atlanta, GA, USA (2001)
4. Beth, T., Borcherding, M., Klein, B.: Valuation of trust in open networks. In: Proceedings of the 1994 European Symposium on Research in Computer Security, pp. 3–18. Brighton, UK (1994)
5. Ceglowski, M., Coburn, A., Cuadrado, J.: Semantic search of unstructured data using contextual network graphs (2003)
6. Chen, R., Yeager, W.: Poblano: a distributed trust model for peer-to-peer networks. Technical report, Sun Microsystems, Santa Clara, CA, USA (2003)
7. Eschenauer, L., Gligor, V., Baras, J.: On trust establishment in mobile ad-hoc networks. Technical report MS 2002–10, Institute for Systems Research, University of Maryland, MD, USA (2002)
8. Ford, L., Fulkerson, R.: Flows in Networks. Princeton University Press, Princeton (1962)
9. Gans, G., Jarke, M., Kethers, S., Lakemeyer, G.: Modeling the impact of trust and distrust in agent networks. In: Proceedings of the Third International Bi-Conference Workshop on Agent-Oriented Information Systems. Montreal, Canada (2001)
10. Golbeck, J.: Computing and applying trust in Web-based social networks. Ph.D. thesis, University of Maryland, College Park, MD, USA (2005)
11. Golbeck, J.: Combining provenance with trust in social networks for semantic web content filtering. In: Proceedings of the International Provenance and Annotation Workshop, LNCS, vol. 4145, pp. 101–108. Springer, Chicago, IL, USA (2006)
12. Golbeck, J.: Generating predictive movie recommendations from trust in social networks. In: Proceedings of the Fourth International Conference on Trust Management, LNCS, vol. 3986, pp. 93–104. Springer, Pisa, Italy (2006)
13. Golbeck, J., Parsia, B., Hendler, J.: Trust networks on the Semantic Web. In: Proceedings of Cooperative Intelligent Agents. Helsinki, Finland (2003)
14. Guha, R.: Open rating systems. Technical report, Stanford Knowledge Systems Laboratory, Stanford, CA, USA (2003)
15. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: Proceedings of the Thirteenth International World Wide Web Conference. ACM Press, New York, NY, USA (2004)

16. Hartmann, K., Strothotte, T.: A spreading activation approach to text illustration. In: Proceedings of the 2nd International Symposium on Smart Graphics, pp. 39–46. ACM Press, Hawthorne, NY, USA (2002)
17. Jøsang, A., Gray, E., Kinateder, M.: Analysing topologies of transitive trust. In: Proceedings of the Workshop of Formal Aspects of Security and Trust. Pisa, Italy (2003)
18. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in P2P networks. In: Proceedings of the Twelfth International World Wide Web Conference. Budapest, Hungary (2003)
19. Kinateder, M., Pearson, S.: A privacy-enhanced peer-to-peer reputation system. In: Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies, LNCS, vol. 2378. Springer, Prague, Czech Republic (2003)
20. Kinateder, M., Rothermel, K.: Architecture and algorithms for a distributed reputation system. In: Nixon, P., Terzis, S. (eds.) Proceedings of the First International Conference on Trust Management, LNCS, vol. 2692, pp. 1–16. Springer, Crete, Greece (2003)
21. Król, D.: Propagation phenomenon in complex networks: theory and practice. New Gener. Comput. **32**(3–4), 187–192 (2014)
22. Levien, R.: Attack-resistant trust metrics. Ph.D. thesis, University of California at Berkeley, Berkeley, CA, USA (2004)
23. Levien, R., Aiken, A.: Attack-resistant trust metrics for public key certification. In: Proceedings of the 7th USENIX Security Symposium. San Antonio, TX, USA (1998)
24. Levien, R., Aiken, A.: An attack-resistant, scalable name service. Draft Submission to the Fourth International Conference on Financial Cryptography (2000)
25. Lewicki, R., McAllister, D., Bies, R.: Trust and distrust: new relationships and realities. Acad. Manag. Rev. **23**(12), 438–458 (1998)
26. Marsh, S.: Formalising trust as a computational concept. Ph.D. thesis, Department of Mathematics and Computer Science, University of Stirling, Stirling, UK (1994)
27. Marsh, S.: Optimism and pessimism in trust. In: Ramirez, J. (ed.) Proceedings of the Ibero-American Conference on Artificial Intelligence. McGraw-Hill, Caracas, Venezuela (1994)
28. Maurer, U.: Modelling a public key infrastructure. In: Bertino, E. (ed.) Proceedings of the 1996 European Symposium on Research in Computer Security, LNCS, vol. 1146, pp. 325–350. Springer, Rome, Italy (1996)
29. McKnight, H., Chervany, N.: The meaning of trust. Technical report MISRC 96–04, Management Information Systems Research Center, University of Minnesota, MN, USA (1996)
30. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation. In: Proceedings of the 35th Hawaii International Conference on System Sciences, pp. 188–196. Big Island, HI, USA (2002)
31. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the Web. Technical report, Stanford Digital Library Technologies Project (1998)
32. Quillian, R.: Semantic memory. In: Minsky, M. (ed.) Semantic Information Processing, pp. 227–270. MIT Press, Boston (1968)
33. Reiter, M., Stubblebine, S.: Path independence for authentication in large-scale systems. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 57–66. ACM Press, Zürich, Switzerland (1997)
34. Reiter, M., Stubblebine, S.: Toward acceptable metrics of authentication. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 10–20. IEEE Computer Society Press, Oakland, CA, USA (1997)
35. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the Semantic Web. In: Proceedings of the Second International Semantic Web Conference. Sanibel Island, FL, USA (2003)
36. Sankaralingam, K., Sethumadhavan, S., Browne, J.: Distributed PageRank for P2P systems. In: Proceedings of the Twelfth International Symposium on High Performance Distributed Computing. Seattle, WA, USA (2003)
37. Smith, E., Nolen-Hoeksema, S., Fredrickson, B., Loftus, G.: Atkinson and Hilgards's Introduction to Psychology. Thomson Learning, Boston (2003)

38. Twigg, A., Dimmock, N.: Attack-resistance of computational trust models. In: Proceedings of the Twelfth IEEE International Workshop on Enabling Technologies, pp. 275–280. Linz, Austria (2003)
39. Wiese, R., Eiglsperger, M., Kaufmann, M.: yFiles: Visualization and automatic layout of graphs. In: Proceedings of the 9th International Symposium on Graph Drawing, LNCS, vol. 2265, pp. 453–454. Springer, Heidelberg, Germany (2001)
40. Ziegler, C.N., Golbeck, J.: Investigating interactions of trust and interest similarity. Decis. Support Syst. **43**(2), 460–475 (2007)
41. Ziegler, C.N., Lausen, G.: Paradigms for decentralized social filtering exploiting trust network structure. In: Meersman, R., Tari, Z. (eds.) Proceedings of the DOA/CoopIS/ODBASE Confederated International Conferences (2), LNCS, vol. 3291, pp. 840–858. Springer, Larnaca, Cyprus (2004)
42. Ziegler, C.N., Lausen, G.: Spreading activation models for trust propagation. In: Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service. IEEE Computer Society Press, Taipei, Taiwan (2004)
43. Ziegler, C.N., Lausen, G.: Propagation models for trust and distrust in social networks. Inf. Syst. Front. **7**(4–5), 337–358 (2005)