Using Social Network-based Trust For Default Reasoning On The Web

Yarden Katz [a,*], Jennifer Golbeck [a]

[a] *8400 Baltimore Ave. Maryland Information and Network Dynamics Lab, University of Maryland, 8400 Baltimore Ave, Suite 200, College Park, Maryland 20740*

Abstract

A drawback of traditional default logic is that there is no general mechanism for preferring one default rule over another. To remedy this problem, numerous default logics augmented with priority relations have been introduced. In this paper, we show how trust values, derived from web-based social networks, can be used to prioritize defaults. We provide a coupling between the method for computing trust values in social networks and the prioritized Reiter defaults of (1), where specificity of terminological concepts is used to prioritize defaults. We compare our approach with specificity-based prioritization, and discuss how the two can be combined. Finally, we show how our approach can be applied to other variants of prioritized default logic.

Key words: Social networks, Nonmonotonic reasoning, Trust-based systems

1 Introduction

We are often given conflicting information from distinct sources, forcing us into a decision about what information to accept. This problem is especially complex on the web, where the information sources are many and varied. Our decision in these cases is sometimes reduced to picking the more highly trusted information source. If we think of the information given by sources as

* Corresponding author.
Email addresses: yarden@umd.edu (Yarden Katz), golbeck@cs.umd.edu (Jennifer Golbeck).
URLs: http://www.mindswap.org/~katz (Yarden Katz), http://www.cs.umd.edu/users/golbeck (Jennifer Golbeck).

a set of default rules, our problem boils down to the following: given defaults from distinct sources which support conflicting conclusions, how should these defaults be prioritized to end up with the most reliable conclusion?

The machinery for expressing priorities between defaults is rich and well-studied, but the question of how these priorities should be generated is frequently left unanswered. Often it is assumed that the preferences are inputted manually by the users of the knowledge base in question. When using trust to prioritize defaults, Web-Based Social Networks (WBSNs) offer an accessible source of trust information. We argue that WBSNs can be used to automatically obtain a set of priorities which reflect the user's levels of trust in the information sources.

Within WBSNs, users often reveal information about their relationships with one another. That includes quantitative values representing how much they trust people they know. Using algorithms presented in this work, trust values can be composed to generate recommendations about how much a user should trust an unknown person in the social network. When default rules are asserted on the web and provenance information is available, these trust values can be used to rate the trustworthiness of the source of each default. That can, in turn, be used as a measure of a default's priority.

In this paper, we show how trust values, derived from web-based social networks, can be used to prioritize defaults. We provide a coupling between the method for computing trust values in social networks given in (2) and the prioritized terminological defaults of (1), where specificity of concepts is used to prioritize defaults. We compare our approach with specificity-based prioritization, and discuss how the two can be combined. We also show that in the resulting coupling, there are several strategies for resolving conflicts that may arise between our own preferences and the preferences of nodes connected to us in the social network. Finally, we discuss how our approach can apply to other methods of prioritizing default logic.

2 Nonmonotonic Reasoning with Default Rules

When we reason, we often use "commonsense" rules that are generally but not universally true. For example, we might infer from (P_1) The flight is scheduled to leave at 11:00 and (P_2) Flights usually leave on time, that we should: (C) Be at the airport in time for an 11:00 flight. While it's certainly not true that *every* flight leaves on time, the premise that this is typically true is what licensed our inference. We can formalize a statement such as (P_2) using *default rules*. Below we briefly describe Reiter defaults and their simple extension to allow priorities. For the sake of simplicity, we will focus on the account of prioritized

defaults given in (1). We later show that our method for combining trust with priorities can be applied to other ways of prioritizing default rules.

2.1 Reiter Defaults

A *Reiter default* (henceforth 'default') is of the form:

$$\frac{\alpha \ : \ \beta}{\gamma}$$

where α, β and γ are formulae of first-order logic. The formula α is the *prerequisite*, β the *justification* and γ the *consequent*. A default rule can be read intuitively as: *if I can prove the prerequisite from what I believe, and the justification is consistent with what I believe, then add the consequent to my set of beliefs.*

Definition 1 (Default Theory) *A default theory T is a pair $\langle \mathcal{W}, \mathcal{D} \rangle$ where $\mathcal{W}$ is a finite set of formulae representing the initial world description (or initial set of beliefs), and $\mathcal{D}$ is a finite set $\{\delta_1, ..., \delta_n\}$ of defaults. T is closed if no free variables appear in either $\mathcal{W}$ or $\mathcal{D}$.*

We will assume for simplicity that free variables in defaults only stand for ground instances. We also, for the sake of exposition, assume that every default has only one justification formula β, though our approach does not rely on this restriction. On these points, we follow (1) where the reader may find a detailed exposition.

The premise (P_2) from our earlier example can be formalized as follows:

$$\delta_f = \frac{Flight(x) \ : \ OnTime(x)}{OnTime(x)}$$

Suppose that $\mathcal{W} = \{Flight(flight714)\}$ and $D = \{\delta_f\}$. Then $\mathcal{W} \vdash Flight(flight714)$, and $\mathcal{W} \cup \{OnTime(flight714)\}$ is consistent, meaning the default δ_f is *active*. Since δ_f is active, we apply it and obtain $\mathcal{W} = \mathcal{W} \ \cup \ \{OnTime(flight714)\}$. The deductive closure of this set is called an *extension*, which we characterize formally below.

Definition 2 (Reiter Extension) *Given a set of closed formulae $\mathcal{E}$ and a closed default theory $\langle \mathcal{W}, \mathcal{D} \rangle$, let $E_0 = \mathcal{W}$ and $\forall i \geq 0$ define:*

$$E_{i+1} = \{\gamma \mid \frac{\alpha \ : \ \beta}{\gamma} \in D, \alpha \in Th(E_i) \text{ and } \neg\beta \notin \mathcal{E}\}$$

Then $\mathcal{E}$ *is an Reiter extension of* $\langle \mathcal{W}, \mathcal{D} \rangle$ *iff* $\mathcal{E} = \bigcup_{i \geq 0} Th(E_i)$

The above theory has one extension, namely $Th(\mathcal{W} \cup \{Flight(flight714)\})$. Contrast this with the case where $\mathcal{W}$ is:

$$\{Flight(flight714), Delayed(flight714), Delayed(x) \rightarrow \neg OnTime(x)\}$$

In this example, $\mathcal{W} \cup \{OnTime(flight714)\}$ is inconsistent and the inference that $OnTime(flight714)$ is blocked. Thus, this theory has no extension where $OnTime(flight714)$ holds.

2.2 Cases of Conflict

Default rules can conflict. A simple abstract example is when two defaults, δ_1 and δ_2 are applicable (i.e. their justifications are consistent with our knowledge) yet the consequent of δ_1 is inconsistent with the consequent of δ_2. We then typically end up with *two extensions*; one where the consequent of δ_1 holds, and one where the consequent of δ_2 holds. The case of two conflicting defaults is illustrated below, as the "Chomsky Diamond" [1] although it is possible to have arbitrarily many conflicting extensions with a larger set of defaults.

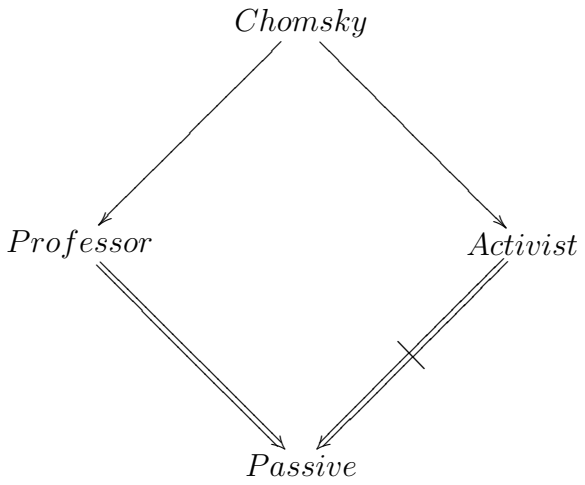

Fig. 1. Chomsky Diamond. Strict "is-a" relations are denoted using '$\rightarrow$', and defeasible (default) implication relations as '$\Rightarrow$'

[1] The diamond is due to its shape when depicted as an inheritance network, illustrated in Figure 1

Example 1 (Chomsky Diamond) *Let $T = \langle \mathcal{W}, \mathcal{D} \rangle$, where:*

$$\mathcal{W} = \{Professor(chomsky), Activist(chomsky)\}$$
$$\mathcal{D} = \{\delta_1, \delta_2\}$$

$$\delta_1 = \frac{Professor(x) \ : \ Passive(x)}{Passive(x)}$$
$$\delta_2 = \frac{Activist(x) \ : \ \neg Passive(x)}{\neg Passive(x)}$$

Note that T has two extensions, $\mathcal{E}_1$ and $\mathcal{E}_2$. In one,

$$Passive(chomsky) \in \mathcal{E}_1$$

while in the other,

$$\neg Passive(chomsky) \in \mathcal{E}_2.$$

It is often desirable to resolve conflicting defaults like δ_1 and δ_2. This can be done by introducing *priorities*, typically expressed as a partial ordering over the set of defaults $\mathcal{D}$. Given a priority relation $<$, we interpret $\delta_1 < \delta_2$ to mean that δ_2 has higher priority than δ_1.

Definition 3 (Prioritized Default Theory) *A prioritized default theory $\mathcal{T}$ is a triple $\langle \mathcal{W}, \mathcal{D}, < \rangle$, where $\mathcal{W}, \mathcal{D}$ are as usual, and $<$ is a partial ordering on $\mathcal{D}$.*

A prioritized version of T would be $\mathcal{T} = \langle \mathcal{W}, \mathcal{D}, < \rangle$. It is easy to see that if $\delta_1 < \delta_2$, then $\mathcal{E}_1$ should not be an extension of $\mathcal{T}$. The reason is that since δ_2 has higher priority, it should be applied first, which in turns blocks the application of δ_1. The definition formalizing this intuition, following (1) again, is given below.

Definition 4 (Prioritized Extension) *Let $\mathcal{T} = \langle \mathcal{W}, \mathcal{D}, < \rangle$ be a prioritized default theory, and $\mathcal{E}$ a set of formulae. Let $\mathcal{E}_0 = \mathcal{W}$, and $\forall i \geq 0$ define:*

$$E_{i+1} = E_i \ \cup \ \{\gamma \mid d = \frac{\alpha \ : \ \beta}{\gamma} \in \mathcal{D}, \alpha \in Th(E_i), \neg\beta \notin \mathcal{E}, \\ \textit{and every } d' > d \textit{ is not active in } E_i\}$$

Then $\mathcal{E}$ is a prioritized extension of $\langle \mathcal{W}, \mathcal{D}, < \rangle$ iff $\mathcal{E} = \bigcup_{i \geq 0} Th(E_i)$

It is clear now that in the above example, if $\delta_1 < \delta_2$, then $\mathcal{E}_1$ is not an extension. Similarly, if $\delta_2 < \delta_1$ were true, then $\mathcal{E}_2$ would not be an extension.

There have been many other approaches to prioritized default logic, where a priority relation is introduced in either the object or the meta language. We refer the reader to (3) for an extensive survey.

Regardless of the specifics of a given approach, some kinds of priority relations are undesirable. In particular, it is unrealistic to require the priority relation to be a total ordering over the defaults, especially if we are dealing with a large and changing collection of defaults. We follow the more common and flexible approach which only requires the priority relation to be a partial ordering.

In previous approaches, the priority relation was usually taken as a given, and sometimes compiled into the object language and reasoned over. In contrast, our priorities are based on the trust rating of the sources of the defaults—i.e. their creators—in a web-based social network. The next section introduces the concept of trust in web-based social networks, and a corresponding algorithm for computing trust ratings. In section 4 we apply this work to the case of prioritizing defaults.

3 Trust in Web-based Social Networks

Web-based social networks (WBSNs) are online communities where users maintain lists of people they know. Other users can browse those connections, and access contact and profile information about people in the network. The popularity of WBSNs has grown dramatically over the last few years, with hundreds of networks that have hundreds of millions of members. Within WBSNs, a variety of features are available to allow users to annotate their relationship; trust is one of these.

When trust is assigned on a quantitative scale, we can make computations with trust values in the network. If we choose a specific user and look at all of the trust ratings assigned to that person, we can see the average opinion about the person's trustworthiness. Trust, however, is a subjective concept where averages are often unhelpful. Consider the simple example of asking weather the President is trustworthy. Some people believe very strongly that he is, and others believe very strongly that he is not. In this case, the average trust rating is not useful to either group. However, given provenance information about the trust annotations, we can significantly improve on the average case. If someone (the *source*) wants to know how much to trust another person (the *sink*), we can look at the who trusts the sink, see how much the source trusts the intermediate people, and produce a result that weights ratings from trusted people more highly than those from untrusted people.

In this section, we present a description of and algorithm for inferring trust

values, and show how the results can be applied.

3.1 Background and Related Work

We present an algorithm for inferring trust relationships in social networks, but this problem has been approached in several ways before. Here, we highlight some of the major contributions from the literature and compare and contrast them with our approach.

Trust has been studied extensively in peer-to-peer systems including (4), (5), (6). There are basic differences in the meaning of trust in P2P networks and social networks that makes these algorithms inappropriate for social use. In P2P systems, trust is a measure of performance, and one would not expect the performance of $peer_a$ to be very different when it is interacting with $peer_b$ vs. $peer_c$. Thus, one global recommendation about the trustworthiness of $peer_a$ will usually be sufficient. Socially, though, two individuals can have dramatically different opinions about the trustworthiness of the same person. Our algorithms intentionally avoid using a global trust value for each individual to preserve the personal aspects that are foundations of social trust.

There are several algorithms for computing trust in social networks specifically. A thorough treatment can be found in (2). Our algorithm differs from most existing algorithms in one of three major ways: we output recommendations in the same scale that users assign trust (vs. eigenvector based approaches like (7)), our computations are about people (vs. trust in statements as in (8)), and we create personalized recommendations (vs. global ratings as are used in P2P systems and (9)).

3.2 Issues for Inferring Trust

When two individuals are directly connected in the network, they can have trust ratings for one another. Two people who are not directly connected to not have that trust information available by default. However, the paths connecting them in the network contain information that can be used to infer how much they may trust one another.

For example, consider that Alice trusts Bob, and Bob trust Charlie. Although Alice does not know Charlie, she knows and trusts Bob who, in turn, has information about how trustworthy he believes Charlie is. Alice can use information from Bob and her own knowledge about Bob's trustworthiness to infer how much she may trust Charlie. This is illustrated in Figure 2.

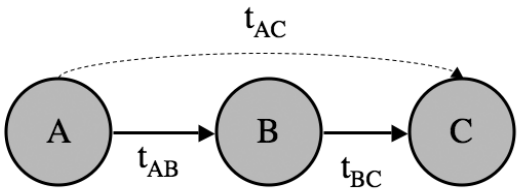

Fig. 2. An illustration of direct trust values between nodes A and B (t_{AB}), and between nodes B and C (t_{BC}). Using a trust inference algorithm, it is possible to compute a value to recommend how much A may trust C (t_{AC}).

To accurately infer trust relationships within a social network, it is important to understand the properties of trust networks. Certainly, trust inferences will not be as accurate as a direct rating. There are two questions that arise which will help refine the algorithm for inferring trust: how will the trust values for intermeidate people affect the accuracy of the inferred value, and how will the length of the path affect it.

We expect that people who the user trusts highly will tend to agree with the user more about the trustworthiness of others than people who are less trusted. To make this comparison, we can select triangles in the network. Given nodes n_i, n_j, and n_k, where there is a triangle such that we have trust values t_{ij}, t_{ik}, and t_{kj}, we can get a measure of how trust of an intermediate person can affect accuracy. Call Δ the difference between the known trust value from n_i to n_k (t_{ik}) and the value from n_j to n_k (t_{ik}). Grouping the Δ values by the trust value for the intermediate node (t_{ij}) indicates on average how trust for the intermediate node affects the accuracy of the recommended value. Several studies (10),(2) have shown a strong correlation between trust and user similarity in several real-world networks.

It is also necessary to understand how the paths that connect the two individuals in the network affect the potential for accurately inferring trust relationships. The length of a path is determined by the number of edges the source must traverse before reaching the sink. Does the length of a path affect the agreement between individuals? Specifically, should the source expect that neighbors who are connected more closely will give more accurate information than people who are further away in the network?

In previous work (2),(11) this question has been addresses using several real networks. The first network is part of the Trust Project, a Semantic Web-based network with trust values and approximately 2,000 users. The FilmTrust network [2], see Figure 3, is a network of approximately 700 users oriented around a movie rating and review website. We will use FilmTrust for several examples in this paper. Details of the analysis can be found in the referenced work, but we present an overview of the analysis here.

[2] Available at http://trust.mindswap.org/FilmTrust

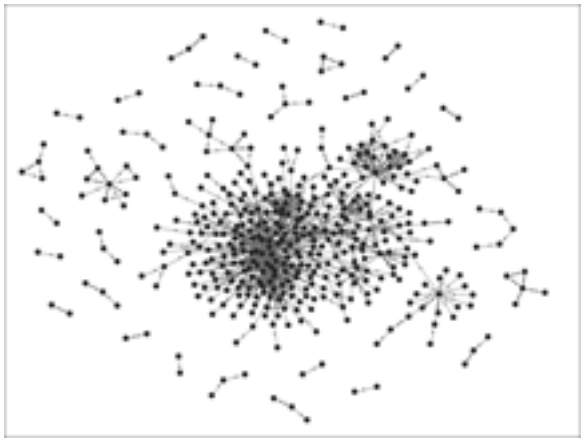

Fig. 3. This figure illustrates the social network in the FilmTrust website. There is a large central cluster of about 450 connected users, with small, independent groups of users scattered around the edges.).

Table 1
Minimum $\overline{\Delta}$ for paths of various lengths containing the specified trust rating.

Trust Value	Path Length			
	2	3	4	5
10	0.953	1.52	1.92	2.44
9	1.054	1.588	1.969	2.51
8	1.251	1.698	2.048	2.52
7	1.5	1.958	2.287	2.79
6	1.702	2.076	2.369	2.92

To see the relationship between path length and trust, we performed an experiment. We selected a node, n_i, and then selected an adjacent node, n_j. This gave us a known trust value t_{ij}. We then ignored the edge from n_i to n_j and looked for paths of varying lengths through the network that connected the two nodes. Using the trust values along the path, and the expected error for those trust values, as determined by the analysis of the correlation of trust and similarity determined in (2). Call this measure of error Δ. This comparison is repeated for all neighbors of n_i, and for all n_i in the network.

For each path length, Table 1 shows the minimum averageΔ ($\overline{\Delta}$). These are grouped according to the minimum trust value along that path.

In Figure 4, the effect of path length can be compared to the effects of trust ratings. For example, consider the $\overline{\Delta}$ for trust values of 7 on paths of length 2. This is approximately the same as the $\overline{\Delta}$ for trust values of 10 on paths of length 3 (both are close to 1.5). The $\overline{\Delta}$ for trust values of 7 on paths of length 3 is about the same as the $\overline{\Delta}$ for trust values of 9 on paths of length

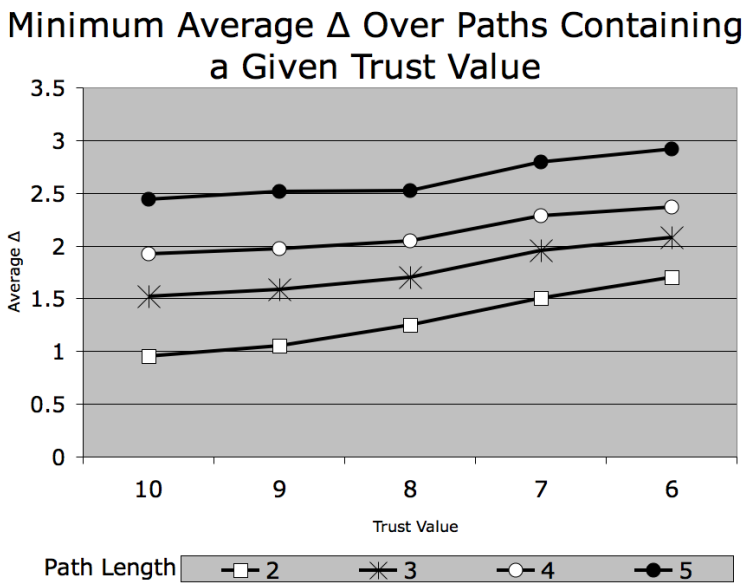

Fig. 4. Minimum $\overline{\Delta}$ from all paths of a fixed length containing a given trust value. This relationship will be integrated into the algorithms for inferring trust presented in the next section.

4. A precise rule cannot be derived from these values because there is not a perfect linear relationship, and also because the points in Figure 4 are only the minimum $\overline{\Delta}$ among paths with the given trust rating.

3.3 TidalTrust: An Algorithm for Inferring Trust

The effects of trust ratings and path length described in the previous section guided the development of TidalTrust, an algorithm for inferring trust in networks with continuous rating systems. The following guidelines can be extracted from the analysis of the previous sections: 1. For a fixed trust rating, shorter paths have a lower $\overline{\Delta}$. 2. For a fixed path length, higher trust ratings have a lower $\overline{\Delta}$. This section describes how these features are used in the TidalTrust algorithm.

3.3.1 Incorporating Path Length

The analysis in the previous section indicates that a limit on the depth of the search should lead to more accurate results, since the $\overline{\Delta}$ 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 con-

nect 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.

3.3.2 Incorporating Trust Values

The previous results also indicate that the most accurate information will come from the highest trusted neighbors. As such, we may want the algorithm to limit the information it receives so that it comes from only the most trusted neighbors, essentially giving no weight to the information from neighbors with low trust. If the algorithm were to take information only from neighbors with the highest trusted neighbor, each node would look at its neighbors, select those with the highest trust rating, and average their results. However, since different nodes will have different maximum values, some may restrict themselves to returning information only from neighbors rated 10, while others may have a maximum assigned value of 6 and be returning information from neighbors with that lower rating. Since this mixes in various levels of trust, it is not an ideal approach. On the other end of possibilities, the source may find the maximum value it has assigned, and limit every node to returning information only from nodes with that rating or higher. However, if the source has assigned a high maximum rating, it is often the case that there is no path with that high rating to the sink. The inferences that are made may be quite accurate, but the number of cases where no inference is made will increase. To address this problem, 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.

3.3.3 Full Algorithm for Inferring Trust

Incorporating the elements presented in the previous sections, the final TidalTrust algorithm can be assembled. The name was chosen because calculations sweep forward from source to sink in the network, and then pull back from the sink to return the final value to the source.

$$t_{is} = \frac{\sum_{j \in adj(j) \mid t_{ij} \geq max} t_{ij} t_{js}}{\sum_{j \in adj(j) \mid t_{ij} \geq max} t_{ij}} \tag{1}$$

TidalTrust is a modified breadth-first search. The source's inferred trust rating for the sink ($t_{source,sink}$) is a weighted average if the source's neighbors' ratings of the sink (see Forumula 1). 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

Table 2
$\overline{\Delta}$ for TidalTrust and Simple Average recommendations in both the Trust Project and FilmTrust networks. Numbers are absolute error on a 1-10 scale.

Algorithm		
Network	TidalTrust	Simple Average
Trust Project	1.09	1.43
FilmTrust	1.35	1.93

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 in Formula 1, and returns the result. Each neighbor repeats 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, computed as 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 completes the calculations of a weighted average by taking information from nodes that they have rated at or above the max threshold. Those values are passed back to the neighbors who queried for them, until the final result is computed at the source.

3.4 Accuracy of TidalTrust

As presented above, TidalTrust strictly adheres to the observed characteristics of trust: shorter paths and higher trust values lead to better accuracy. However, there are some things that should be kept in mind. The most important is that networks are different. Depending on the subject (or lack thereof) about which trust is being expressed, the user community, and the design of the network, the effect of these properties of trust can vary. While we should still expect the general principles to be the same−shorter paths will be better than longer ones, and higher trusted people will agree with us more than less trusted people−the proportions of those relationships may differ from what was observed in the sample networks used in this research.

There are several algorithms that output trust inferences, but none of them

produce values within the same scale that users assign ratings. Some trust algorithms form the Public Key Infrastructure (PKI), such as Beth-Borcherding-Klein (12), are more appropriate for comparison. Due to space limitations that comparison is not included here, but can be found in (2). One direct comparison to make is to compare the $\overline{\Delta}$ from TidalTrust to the $\overline{\Delta}$ from taking the simple average of all ratings assigned to the sink as the recommendation. We made this comparison using two real world networks. As shown in table2, the TidalTrust recommendations outperform the simple average in both networks, and these results are statistically significant with $p < 0.01$.

4 Basing Priority on Trust Values

Given a social network, an ordinary default theory T, and a source node Src in the network, we can now now prioritize the defaults according to trust values.

procedure $TrustPrioritize(W, D, Src, Prov)$:

Input:

(1) A set of initial formulae W

(2) A source node Src

(3) A set $D = \{\delta_1, ..., \delta_n\}$ of defaults,

(4) A function $Prov \ : \ D \rightarrow Nodes$

Output:

A set of extensions

$P := \emptyset$

for every $d, d' \in D$:

if $TidalTrust(Src, Prov(d)) < TidalTrust(Src, Prov(d'))$:

$P = P \cup \{d < d'\}$

if $Prov(d) = Src$ **and** $Prov(d') \neq Src$:

$P = P \cup \{d' < d\}$

return $ComputeExtensions_{\mathcal{PL}}(W, D, P)$

Fig. 5. Generating Priorities from Trust Values

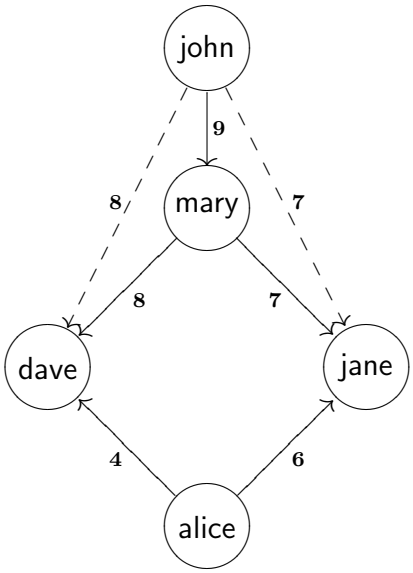

Fig. 6. The social network between John, Mary, Dave, Jane and Alice

4.1 Algorithm

The simple algorithm for generating extensions based on trust values is given in Figure 4. Note that our method does not make any assumptions about the specifics of the base default logic language $\mathcal{PL}$. We do, however, assume the following are available:

(1) A function $ComputeExtensions_{\mathcal{PL}}$ for computing the extensions of $\mathcal{PL}$, which takes a prioritized default theory as input.
(2) A *source node*, which in our case is the node according to which priorities will be generated. Intuitively, this can be thought of as our 'viewpoint' in the social network−we reason from the perspective of the source node.

If restricted to normal form, any prioritized default theory of (1) is always guaranteed to have an extension. In addition, every prioritized normal default extension is also a Reiter extension. Since we have not in any way changed the semantics of the prioritized defaults, it is obvious that the same desirable properties hold true for our approach. For this reason, we restrict ourselves to normal defaults for the remainder of the paper.

4.2 Example: Using Trust for Choosing a Film

Suppose that we are dealing with a film knowledge base. A group of friends−John, Mary, Dave, Jane and Alice−each input their film preferences, such as preferred genre or directors/actors, in the form of default rules. Their preferences are as follows:

$$\mathcal{W} = \{IndieFilm(hce), SpanishFilm(hce), DirectedBy(hce, Almodovar)\}$$

$$\mathcal{D} = \{\delta_{john}, \delta_{dave}, \delta_{jane}\}$$

$$\delta_{john} = \frac{Comedy(x)}{\neg Watch(x)}$$

$$\delta_{jane} = \frac{IndieFilm(x) \wedge SpanishFilm(x)}{\neg Watch(x)}$$

$$\delta_{dave} = \frac{IndieFilm(x) \wedge DirectedBy(x, Almodovar)}{Watch(x)}$$

We assume that every Spanish film is a film, and similarly that every film directed by anyone (in our case, Almodovar) is also a film.

In our scenario, John, Mary, Dave and Alice are part of a social network, shown in Figure 6. The direct trust values between two nodes in the network are given in bold, while inferred trust values are italicized and are shown as a dotted edge.

Suppose that John is trying to decide whether or not he should watch the film *hce*, the only film currently in our knowledge base. John's only preference is not to watch comedies, which does not apply to *hce*. Simply looking at the defaults in $\mathcal{D}$, a conflict arises. According to δ_{jane}, John should not watch the movie since it is a Spanish film. On the other hand, according to δ_{dave}, John should watch the film since it is directed by Almodovar.

Note that John did not directly rate Dave and Jane. John's only connection to the two is via Mary, who he highly trusts. Mary does not have any film preferences, and so we cannot use her to resolve the conflict. According to TidalTrust, John's inferred trust values for Dave and Jane are 8 and 7, respectively. Thus, the relevant priority yielded in this case is $\delta_{jane} < \delta_{dave}$, which allows John to conclude that he should watch *hce*.

Consider the same scenario, except this time with Alice as the source node. Unlike John, Alice has direct trust ratings for Dave and Jane, and unlike Mary, Alice has stated that Jane is more trusted than Dave Therefore, there will be an extension where Alice's conclusion, based on the generated priorities $\delta_{dave} < \delta_{jane}$, is *not* to watch *hce*. Clearly, this extension is not possible if we pick John as the source node, differentiating between the two nodes' relations to the rest of the social network.

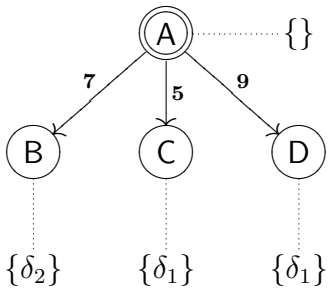

Fig. 7. A social network and default configuration that violates the partial order. A is the source node in the example. The dotted lines indicate the set of defaults asserted by a node.

5 Resolving Conflicting Priorities

In the algorithm as given (see Figure 4), we can easily have situations where due to conflicting priorities, the generated prioritized default theory has no extensions. This can be illustrated by an abstract example. Consider the social network and default configuration in Figure 5, where A is assumed to be the source node. As indicated, A has no preferences, so the set of its asserted defaults (i.e. defaults that A endorses) is empty. Node B asserted the default δ_2, and both C and D assert the default δ_1. The combined algorithm will then generate, if A is assumed as the source node, the following priorities:

(1) $\delta_1 < \delta_2$ since A trusts B more than C, and,
(2) $\delta_2 < \delta_1$ since A trusts D more than C

Since (1) and (2) are clearly inconsistent, $<$ is no longer a partial order, and the resulting default theory will have no extensions. This can be avoided by picking, for every default $\delta \in \mathcal{D}$, the most highly trusted node that asserted it, and using this value in the computation of the priorities for a given source node. In the current case, the most highly trusted node that asserted δ_1 is D. Thus, when computing the priorities for A, it is D's trust value that is considered, giving us the single correct priority $\delta_2 < \delta_1$.

6 Discussion and Conclusions

6.1 Priority of the Source Node

Cases can arise where the source node has a default that conflicts with another node's default. In our approach, we chose to prioritize the defaults of the source higher than the defaults of other nodes in the social network. This is reflected in the algorithm, where we explictly add to the default theory that the defaults

associated with the source have higher priority than all others. We believe this is the most appropriate choice for the case when dealing with social networks.

If the choice to explicitly prefer the source's defaults is not made, then new cases of conflict can arise. Consider the following abstract example. Suppose we have a root node A with an edge AB. Assume that A has one default whose consequent is $\varphi(x)$, i.e. $\delta_A = \frac{\top}{\varphi(x)}$, and that B has one default $\delta_B = \frac{\top}{\neg\varphi(x)}$. Regardless of the value t_{AB} (or the value of any other edges A might have) we are guaranteed to have an extension where $\varphi(x)$ holds. The reason is that A does not necessarily have an explicit trust rating for itself, i.e. there is no t_{AA} value. Note that this is very different from the usual reason for why δ_A and δ_B would generate two extensions in ordinary default logic. Therefore, in systems where this value is not present or assumed, it seems there is no way to determine the priority of δ_A compared with other defaults in the system. This issue will arise whenever the source node has an applicable default whose consequent might conflict with defaults of other nodes in the system.

In such cases, at least two simple resolutions are possible:

(1) Make the assumption that the source node has "infinite" credibility−i.e. one always trusts oneself over all others, or alternatively,
(2) Make the assumption that when getting a recommendation from other nodes, one should ignore one's own preferences.

In our approach, the first choice was made. We contrast this with the case where specificity is used as a measure of priority.

6.2 Priority and Specificity

In (1), priorities between defaults are induced by the specificity of their justifications. While this approach is useful, it cannot resolve every case. In our first example where John is the source node, a specificity-based approach will not decide between Dave's default rule and Alice's. In this case, our approach can be used to *supplement* the priorities generated by specificity-based approach.

Going back to the issue raised by the preferences of the source in the film example, we see that specificity might be altogether inappropriate. For example, suppose that John is the source node and we know that in general his preference not to watch any film that is a comedy. Let's assume that we have one given film, c, and that $RomanticComedy(c)$ and $RomanticComedy(x) \rightarrow Comedy(x)$. In this case, it does not make sense for John's choice to not watch c, based on his preference, to be defeated by another node X, where

$\delta_X = \dfrac{RomanticComedy(x)}{Watch(x)}$ simply because δ_X is more specific. John's preference, while defined more generally than that of node X, should still apply.

In the Tweety triangle, specificity clearly leads to the desirable extension. In fact, whenever dealing with a set of defaults that are meant to *classify* objects and their properties most accurately, the specificity-based approach is generally more appropriate. However, as we have shown, such an approach may fail if we use a set of defaults to express user preference.

6.3 Other approaches to priorities

We have focused on the method of prioritization of defaults in (1). Our motivation for focusing on the system of (1) has been two-fold: (i) the system has straight-forward and simple semantics, which was clearer for the sake of exposing our method, and (ii) it is focused on description logics, which are naturally relevant to Semantic Web applications due to their correspondence to subsets of OWL.

Following the classification of approaches to priorities offered in (3), we see that priorities in this system have the following properties:

(1) *Range over rules* – the priority relation $<$ is an irreflexive partial ordering over the set $\mathcal{D}$ of defaults.
(2) *Static* – the priority relation $<$ for a given default theory is fixed and may not change in the course of reasoning.
(3) *Meta-level* – the priority relation $<$ is given "outside" of the language of default logic, as a separate component of the default theory.
(4) *Prescriptive* – the priority relation $<$ specifies that defaults with higher priority should be *applied* first.

Some have argued against the prescriptive priorities (see (13)), showing examples where they lead to counter-intuitive results. In these examples, defaults that are ranked lower than others are sometimes applied first, because the more highly prioritized defaults are not–in that stage of reasoning–applicable. The *descriptive* approach to preferences aim to resolve such problems, by interpreting the set of priorities as representing "a ranking on desired outcomes: the desirable (or: preferred) situation is one where the most preferred defaults are applied." (3). One could easily apply our approach to these logics where the priority relation is interpreted descriptively.

Similarly, many systems with different properties than the above (putting aside descriptive v. prescriptive distinctions) have been investigated in the literature. For example, systems with dynamic priorities, and priorities in the

object-language, are naturally appealing for their increased expressivity. In such systems, applying our trust-induced priorities is less obvious. The reason is that in these systems, the priority relations among defaults can be inferred, and more importantly, might change in the course of reasoning (i.e. the priorities are, like defaults themselves, defeasible.) While such systems can use our trust-based method for generating the priorities to start with, it is unclear what the interaction between the inferred priority and trusted one will be. In other words, since one will be able to talk about priorities in the object-language, conflicts between priorities recommended by the given knowledge base (*inferred* priorities) and priorities induced by the trust values from the social networks can arise.

6.4 Conclusions

In summary, we have presented a coupling between traditional default logic with priorities and a method for inferring trust in web-based social networks. We argue that the latter provides a good way to generate priorities for default rules. This approach makes it possible to make use of the many large and readily available existing web-based social networks, thus grounding the priorities in real web data. Such an approach differs from the more traditional approaches to priority, where the priorites are taken as specifically tailored to the set of defaults at hand.

While the more traditional approach is appropriate for closed knowledge representation systems (where priorities are assumed as given or inputted manually) our approach reuses existing web data, which makes the introduction of prioritized defaults into established web systems less demanding. Furthermore, we emphasize that in a system where default rules use a different mechanism for priorities (such as specificity), user preferences, encoded as a web-based social network, can be used as an alternative. That is, when the first mechanism of priority might be incomplete, the priorities generated from the social network can be used to possibly fill in the gap. In addition, we have also shown a case where a specificity-based approach is likely to be inappropriate, and where a trust value based approach shows more promise.

7 Future Work

On the theoretical front, at least two issues are left open. The first, discussed in an earlier section, is the application of our method to default logic systems where the priorities are both dynamic and expressible in the object-language. Such systems are more complex than the prioritized default logic explored

here, and strategies for resolving conflicts between priorities as described the object-language and priorities determined by trust will be needed.

A second theoretical issue arises from a limitation of our approach. Currently, we assume that the priorities for the source node are determined solely by the trust relations of that node in the social network. However, it is natural to allow the source nodes in some domains to prioritize *its own* set of asserted defaults. If this is allowed, another source of conflict is possible, between the priorities of the source for its own set of defaults, and the priorities generated by its trust relations to other nodes who are committed to rules of the same set. For example, suppose that a source node Src is committed to both δ_1 and δ_2, and specifies that $\delta_2 < \delta_1$. Now assume that node $t_{Src,A} = 5$, and $t_{Src,B} = 9$. If B is committed to δ_2 and A to δ_1, then applying our algorithm as given will generate the conflicting preference $\delta_1 < \delta_2$, leading to a default theory with no extension. It would be interesting for this case to apply work on revising priorities and preference relations in default logics (such as (14)), to allow the source node to intelligently pick between its own priorities and that of its neighbors in the network.

On the practical front, the quality of the results obtained by prioritizing with trust can be determined empirically when they are applied within applications. One of the main networks we have used for testing is part of the FilmTrust system. FilmTrust currently uses inferred trust values to compute predictive movie ratings customized to each user based on who they trust. However, the current system does not allow for users to specify any default rules about their preferences. Such a default rule system fits well in the context of films.

As part of our future work, we will be deploying a rule system in the FilmTrust system [3], a social network about movies. These defaults will be used in two ways. First, they can help tailor recommendations for the user who asserted rules. They can also be used to filter recommendations for others who trust the user who asserted the rules. In this application, it will be common for defaults to conflict. In such cases, trust is an obvious option for determining which rules to apply.

This will allow us to quantitatively and qualitatively measure the performance of using trust for prioritizing defaults. Showing that the trust-prioritized defaults improve performance will validate how our approach can be used to develop intelligent applications.

[3] http://trust.mindswap.org/FilmTrust/

References

[1] F. Baader, B. Hollunder, Priorities on Defaults with Prerequisites, and Their Application in Treating Specificity in Terminological Default Logic., J. Autom. Reasoning 15 (1) (1995) 41–68.

[2] J. Golbeck, Computing and Applying Trust in Web-based Social Networks, Ph.D. Dissertation, University of Maryland, College Park, 2005.

[3] J. P. Delgrande, T. Schaub, Expressing preferences in default logic., Artif. Intell. 123 (1-2) (2000) 41–87.

[4] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The eigentrust algorithm for reputation management in p2p networks, Proceedings of the 12th International World Wide Web Conference.

[5] K. Aberer, Z. Despotovic, Managing trust in a peer2peer information system., Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM01) (2001) 301–317.

[6] S. Lee, R. Sherwood, B. Bhattacharjee, Cooperative peer groups in nice., IEEE Infocom.

[7] C. Ziegler, G. Lausen, Spreading activation models for trust propagation., Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service.

[8] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, Proceedings of the Second International Semantic Web Conference.

[9] R. Levien, A. Aiken, Attack resistant trust metrics for public key certification, 7th USENIX Security Symposium.

[10] C.-N. Ziegler, J. Golbeck, Investigating Correlations of Trust and Interest Similarity., Decision Support Services.

[11] J. Golbeck, Generating Predictive Movie Recommendations from Trust in Social Networks., Proceedings of The Fourth International Conference on Trust Management.

[12] T. Beth, M. Borcherding, B. Klein, Valuation of trust in open networks, Proceedings of ESORICS 94.

[13] G. Brewka, T. Eiter, Prioritizing default logic., in: S. Hölldobler (Ed.), Intellectics and Computational Logic, Vol. 19 of Applied Logic Series, Kluwer, 2000, pp. 27–45.

[14] M. Freund, On the revision of preferences and rational inference processes., Artif. Intell. 152 (1) (2004) 105–137.