Recommending messages to users in participatory media environments: a Bayesian credibility approach

by

Noel Sardana

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, we address the challenge of information overload in online participatory messaging environments using an artificial intelligence approach drawn from research in multiagent systems trust modeling. In particular, we reason about which messages to show to users based on modeling both credibility and similarity, motivated by a need to discriminate between (false) popular and truly beneficial messages. Our work focuses on environments wherein users' ratings on messages reveal their preferences and where the trustworthiness of those ratings then needs to be modeled, in order to make effective recommendations.

We first present one solution, CredTrust, and demonstrate its efficacy in comparison with LOAR — an established trust-based recommender system applicable to participatory media networks which fails to incorporate the modeling of credibility. Validation for our framework is provided through the simulation of an environment where the ground truth of the benefit of a message to a user is known. We are able to show that our approach performs well in terms of successfully recommending those messages with high predicted benefit and avoiding those messages with low predicted benefit.

We continue by developing a new model for making recommendations that is grounded in Bayesian statistics and uses Partially Observable Markov Decision Processes (POMDPs). This model is an important next step, as both CredTrust and LOAR encode particular functions of user features (viz., similarity and credibility) when making recommendations; our new model, denoted POMDPTrust, learns the appropriate evaluation functions in order to make "correct" belief updates about the usefulness of messages. We validate our new approach in simulation, showing that it outperforms both LOAR and CredTrust in a variety of agent scenarios. Furthermore, we demonstrate how POMDPTrust performs well against real world data sets from Reddit.com and Epinions.com.

In all, we offer a novel trust model which is shown, through simulation and real-world experimentation, to be an effective agent-based solution to the problem of managing the messages posted by users in participatory media networks.

Acknowledgements

First and foremost, I thank my supervisor, Professor Robin Cohen. Robin was an incredible guiding influence throughout the entirety of my graduate studies at the University of Waterloo. She devoted countless hours working with me, providing suggestions for and feedback about my research. I consider myself extremely fortunate to have had the opportunity to work so closely with Robin; she is a stalwart professor, and the motivation she provided me was pivotal to my success in graduate school. I would also like to express my gratitude to Professor Pascal Poupart and Professor Kate Larson for devoting some of their time to serve as readers for this thesis and for their valuable feedback.

Thank you also to Hadi Hosseini for being a great officemate and for always being willing to lend an ear. Likewise, I thank John Doucette, Alan Tsang, and Graham Pinhey for their friendship and moral support. Thanks also to all of the other graduate students (CS Club 7) with whom I shared many laughs; it was truly wonderful to be able to meet and work with such a mix of intelligent, accomplished, and driven people.

My mother, father, and brother (Mary, Pankaj, and Alan) were also important influences and provided unwavering support that has been and continues to be a source of strength and motivation for me. Thank you very much.

Lastly, I am grateful for the financial assistance provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), NSERC's Strategic Networks of Research hSITE project, and the David R. Cheriton Graduate Scholarship.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Today, users are increasingly engaged with online media via the internet. A plethora of available information, ranging from online news networks to various forums to social and participatory media sites, contributes to information overload germane to our information rich society. Moreover, the advent of Massive Open Online Courses (MOOCs), provided by such services as Coursera, and the increasing use of both online retailers (like eBay and Amazon) and information feeds (like Twitter and Facebook) beg the following question: how can we help users sift through excess information to retrieve the most relevant objects of interest?

Much work has been published on the subject of modeling peer trustworthiness, particularly for use in e-marketplace environments (see, for example, [44, 50, 35]). Beyond the e-marketplace domain, the problem of determining the trustworthiness of social network peers can be of particular use when evaluating which messages users might like to see, especially if those peers provide advice and feedback about messages. Peer similarities is at the heart of collaborative filtering techniques used in recommendations; learning which peers are trustworthy is then certainly of value. Accordingly, the trust literature provides a good foundation for developing techniques for message recommendation.

One model in particular, Champaign et al.'s Learning Object Annotation Recommender system (LOAR) [12], drew insipiration from trust models in order to determine which learning object annotations to recommend to students in order to maximize their learning gains when participating in online learning environments. In this domain, LOAR assumes that students can rate annotations positively (1) and negatively (0). Using the ratings left on annotations, LOAR computes similarity scores between students and employs a similarity-weighted combination of annotation ratings to derive the annotation reputation.

Combining this reputation with the annotator reputation, LOAR derives a final predicted benefit of the annotation for a given student.

We demonstrate how LOAR can fall short when making recommendations because it enables false information propagation: if many similar peers provide positive feedback about a given message, it has a higher likelihood of being recommended despite any inaccuracies it may contain. This leads us to develop a model and algorithm called CredTrust, which explores the concept of user credibility and its use in combating the spread of such "folklore". We draw inspiration from Seth et al.'s Bayesian Credibility Model [39] when considering how credibility should be modeled within our framework, though we assume that credibility values can be provided by an oracle (and return to discuss extensions to incorporate deeper reasoning about credibility for future work). In particular, CredTrust combines both similarity and credibility when making recommendations, which helps to avoid recommending false popular messages.

To validate our model, we perform simulations to demonstrate how CredTrust outperforms LOAR under a variety of conditions. These simulations were designed with careful consideration given to the nature and distribution of agents and their rating behaviour, and we incorporated a notion of agent *types* in order to simulate different agent preferences for certain types of messages. The consideration given to simulating a multiagent environment also led us to investigate several properties inherent to LOAR; the insights gleaned during this process leads to our proving two properties about LOAR and developing further simulations that codify precisely the environments in which LOAR has difficulty coping.

While CredTrust is shown to be superior to LOAR in a number of circumstances, we note that both models encode particular linear combinations of user features (specifically, similarity and credibility). As a result, there exist environments that cause both models to underperform. Hence, we continue by developing a new model grounded in Bayesian statistics that draws on concepts from Markov Decision Processes (MDPs) and Partially Observable MDPs (POMDPs) in order to further improve message recommendations. This new model, denoted POMDPTrust, draws upon other Bayesian approaches like BLADE [35], however also explicitly accounts for user utilities when ultimately making message recommendations. In addition to demonstrating its efficacy in simulation, we evaluate the performance of the POMDP model against real world data from Reddit.com and Epinions.com, two websites where users can share and rate messages. These real-world data are meaningful for validation, as they offer a realistic and unique mix of user ratings. We are able to demonstrate that POMDPTrust perfoms well in terms of true/false positives and true/false negatives when set against the ground truth.

In all, this work explores the topic of message recommendations in a social network

of peers wherein users can share and rate messages. To do this, we develop models that draw on ideas from the trust modeling literature. We explore the relationship between message classification/recommender systems and trust modeling, and we provide a concrete evaluation of our models to serve as a backdrop for future work in both trust modeling and recommender system research.

Our POMDP model for message recommendations is interesting from the perspective of trust evaluation, as it uses Bayesian learning to derive an observation function that combines user features in a way that is statistically "correct" given the environment. Moreover, it is novel for the trust community because it incorporates a notion of user utilities in order to determine how to act on the basis of evaluated trust. Ultimately, we believe our research provides a novel perspective on and partial solution to the problem of message recommendation through the application of Bayesian learning using POMDPs and through the exploration of user credibilities in order to help users sift through the raft of information available to them in online messaging environments, and furthermore to ensure that they are exposed to the *right* information.

1.1 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 presents the background concepts that underpin this research, including an in-depth discussion of trust modeling as well as an overview of recommender systems more generally and a discussion of Markov Decision Processes. Chapter 3 introduces the notion of crediblity as a means of combating false information propagation in participatory media networks, inspired by the work of Champaign et al. [12]. Chapter 4 continues by preseting a new model for making recommendations, central to which is the use of a Partially Observable Markov Decision Process. Chapter 5 continues with a detailed discussion of our results. Lastly, Chapter 6 concludes by offering some final points about our work and experiments, as well as a discussion of opportunities for future work.

Chapter 2

Background

In this chapter, we cover some basic terminology and models from related work on trust which form a backdrop to our solutions and to which we return to compare and contrast our work in Chapter 5; we also briefly review Recommender Systems work for which we also offer deeper discussion/contrast (in Chapter 5) after displaying our proposed algorithms in full. We review basic concepts associated with Markov Decision Processes (MDPs) for reasoning under uncertainty, which provide an important background to our POMDP approach to message recommendations developed in Chapter 4. Finally, we discuss Latent Factor Classification Models, to which we compare our POMDP approach as part of our validation in our Chapter 4 simulations. For convenience, we also log definitions for various terms used throughout this thesis in Appendix A.

2.1 Agents and Multiagent Systems

This research is concerned with interactions between users in online messaging environments. Within this setting, *intelligent agents* [48] are entities that represent the users in the system, for example, to act on their behalf, to record their preferences (for types of messages), to aid in decision making, etc. Agents can perceive the environment in which they operate; they respond to stimuli and react on the basis of their perception of their surroundings. They can be fully autonomous, acting on behalf of (but without interacting with) the users they represent, or they can be semi-autonomous, requiring occasional guidance from humans to correct their behaviour or to clarify information they receive from the environment. In our research, our overall goal is to design agents that help users by filtering messages into different classes: those worth seeing, and those not worth seeing.

Since our setting consists of multiple users, and therefore multiple agents that coexist with one another, it is considered to be a *multiagent system* [48]. Multiagent systems can also, depending on the domain, be static with a fixed set of agents, or they can be dynamic, allowing for the agent population to grow and shrink as agents come and go. For our context, the system is dynamic, meaning that agents may come and go in tandum with the coming and going of users in the participatory media network.

2.2 Recommender Systems

Recommender systems aim to recommend new items of use to users by suggesting relevant objects on which to focus one's attention. Such systems can be categorized in one of two ways [23]:

Content-based systems recommend new items to a user based on the characteristics of the items (e.g., the content they contain, when they were published, etc.).

Collaborative-filtering systems recommend items to users based on users' preferences and their similarity to other users' preferences, especially insofar as those preferences are expressed explicitly through ratings on experienced items.

In this work, we focus primarily on collaborative filtering (CF) approaches, which have gained widespread attention in recent years, not the least because of their success and exposure in the recent Netflix competition[1]. However, we believe that an all-encompassing solution could make use of both CF methods as well as content-based methods, especially to cope with cold-start problems and data sparsity. We will revisit this notion in Chapter 5 and in Section 6.2.1.

2.2.1 Trust and Reputation

One flavour of collaborative filtering relies on exploiting a user's *neighbourhood* when making recommendations. The basic idea behind neighbourhood-based approaches is to determine the similarity between users based on past commonly experienced/rated items. For example, in the movie domain, two users A and B may have seen and rated several movies in common. The users' past likes/dislikes serve as evidence as to their similarity with

[1] If interested, see http://www.netflixprize.com/ for details regarding the competition, which was completed in September, 2009.

respect to movie tastes. To the extent that the two users have similar ratings behaviour, new movies can then be recommended to A, for example, by considering other movies that B has seen and rated but that A has not yet seen.

Much research has been conducted to try to determine how to trust other users within a network. The notion of *trust* and the many models that explore its derivation is very much related to the neighbourhood method of collaborative filtering. In this thesis, we then focus on developing a trust-based solution to message recommendation. This section presents several trust models in order to acclimate the reader to some of the concepts and notations used throughout this thesis. In particular, we present in detail Jøsang et al.'s Beta Reputation System (BRS) [28], Teacy et al.'s TRAVOS [44] approach, Zhang et al.'s Personalized Trust Model (PTM), Champaign et al.'s Learning Object Annotation Recommender (LOAR) [12], Seth et al.'s Bayesian Credibility Model [39], and Regan et al.'s BLADE [35]. Two models in particular from this section, LOAR and BLADE, will reappear in later chapters as baseline comparators for our work.

Beta Reputation System (BRS)

The Beta Reputation System, proposed by Jøsang and Ismail in [28] for use in e-marketplace reputation systems, provides a foundation for later trust modelling work carried out by Zhang et al. in [50]. BRS is foundational because it is grounded in probability theory; it uses the Beta probability distribution, which describes probability distributions of binary events (e.g., flipping a coin, or rating a product good versus bad).

Formally, the Beta distribution[2] is a family of distributions and represents the probability $Pr(p\,;\,\alpha,\beta)$, viz., the probability of parameter $p \in [0,1]$ given two hyperparameters, $\alpha, \beta \in (0,\infty)$. In BRS, it is used as a *second-order* probability, or the "probability of a probability", since it is used to describe the prior belief about a probability. The Beta density function is expressed as follows:

$$Pr(p\,;\,\alpha,\beta) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{\mathbf{B}(\alpha,\beta)} \tag{2.1}$$

where $\mathbf{B}(\alpha,\beta)$ is the Euler Beta-function, given by

$$\mathbf{B}(\alpha,\beta) = \int_0^1 x^{(\alpha-1)}(1-x)^{(\beta-1)}dx \tag{2.2}$$

[2] The Beta distribution is the conjugate prior for Bernoulli distributions. For example, it can be used to reason about whether a coin is biased with a 90% chance of turning up heads given that a set of coin tosses $\{x_1,\ldots,x_n\}$ was observed, but given that one has a prior belief about the coin's fairness.

The expected value of a Beta density function is given by the following formula:

$$E[p] = \frac{\alpha}{\alpha + \beta} \tag{2.3}$$

In [28], the authors interpret this expected value as the probability of some positive outcome occuring in the future, where $\alpha = r + 1$ (r being the number of positive outcomes that occurred in the past) and $\beta = s + 1$ (s being the number of negative outcomes that previously occurred). Thus, the expected value of a Beta distribution is a suitable trust metric (i.e., one has high trust in an agent if one expects positive outcomes from that agent in the future). The authors generalize this notion to include real-valued parameters (r, s), which represent the degree of satisfaction/disatisfaction given some interaction between a buyer and a seller in an e-marketplace. Modeling the probability of future behaviour (i.e., the probability of fulfilling future contractual obligations in the e-marketplace setting) as a random variable and thus using Bayesian inference to estimate this probability is more appropriate than using confidence intervals and the frequentist approach in trust models, since there is an implicit assumption that agents' behaviours are dynamic rather than fixed. This view also allows the use of prior knowledge to temper the derived trust values and allows trust beliefs to evolve over time, both of which support the choice to model trust using the Beta distribution.

While the Beta distribution provides the framework for the authors' reputation function[3], BRS is characterized by several further tenets:

1. The notation $\phi_T^X(p)$, $P \sim Beta(r_T^X + 1, s_T^X + 1)$ is used to denote the reputation function for T from the perspective of X.

2. A real-valued reputation is derived from the reputation function by mapping its expected value, $E(\phi_T^X) \in [0, 1]$, onto any desired interval ($[-1, +1]$ in [28]).

3. Feedback from mutiple sources, say $Q = \{(r_T^1, s_T^1), \ldots, (r_T^n, s_T^n)\}$ (where superscripts serve to differentiate feedback sources), can be combined by $(r_T^Q, s_T^Q) = (\sum_{i=1}^n r_T^i, \sum_{i=1}^n s_T^i)$.

4. Feedback/advice from other users, Y, can serve to discount a user X's opinion about a party T, derived from Jøsang's belief model in [27]. In particular, given X's opinion about Y's advice in the form of a 3-tuple (b_Y^X, d_Y^X, u_Y^X) (representing belief, disbelief, and uncertainty), and given Y's opinion about T as (b_T^Y, d_T^Y, u_T^Y), X's opinion of T given Y's advice is $(b_Y^X b_T^Y, d_Y^X d_T^Y, d_Y^X + u_Y^X + b_Y^X u_T^Y)$. The authors interpret this model in the BRS by setting $b = \frac{r}{r+s+2}$, $d = \frac{s}{r+s+2}$, and $u = \frac{2}{r+s+2}$.

[3]We also use Beta distributions as the basis for amalgamating peer advice about messages in our approaches presented in Chapters 3 and 4.

5. Peer advice is weighted according to a "forgetting factor" to lend more weight to recent feedback under the assumption that parties are more likely to behave according to their recent activities. As such, combined feedback can be modified by $(r_T^Q, s_T^Q) = (\sum_{i=1}^{n} r_T^i \lambda^{n-i}, \sum_{i=1}^{n} s_T^i \lambda^{n-i})$ where $0 \leq \lambda \leq 1$, and $\lambda = 1$ corresponds to equally weighting all past advice (never forgetting).

TRAVOS

Teacy et al. developed a system dubbed "TRAVOS" (a trust and reputation model for agent-based virtual organisations) to model trust relationships between agents in virtual organizations [44]. Like BRS, TRAVOS defines a trust metric to be the probability that a trustee will perform on a future obligation and uses a Beta PDF to model relative trust probabilities. Unlike BRS, TRAVOS incorporates the notion of *confidence* in the inferred trust metric. That is, limited past experiences may be inadequate to accurately derive a trust metric, and so γ represents the probability that the true trust metric, τ^*, lies within some margin of error ϵ, i.e., $\tau^* \in [\tau - \epsilon, \tau + \epsilon]$ [44]. τ is again described by the expectation of a Beta distribution given parameters representing the number of positive and negative past outcomes, and γ is given by

$$\gamma = \frac{\int_{\tau-\epsilon}^{\tau+\epsilon} x^{\alpha-1}(1-x)^{\beta-1}dx}{\mathbf{B}(\alpha, \beta)} \tag{2.4}$$

Furthermore, TRAVOS allows the truster to seek third party advice when γ is too small (i.e., below some threshold). Trusters can aggregate reported trust parameters, viz., the number of positive and negative past interactions, from pundits (other agents) to assemble a more confident perspective of a trustee (i.e., one founded on more observations of the trustee's performance). However, a simple aggregation scheme implicitly assumes that pundits report truthfully and that their advice is based on similar rating criteria to the truster. Accordingly, a more robust scheme that adjusts the opinions of a given pundit to account for potential dishonesty works as follows:

1. A trust parameter report is a 2-tuple of positive and negative outcomes between a pundit and another agent. The current (potentially dishonest) report being evaluated is $\widehat{r} = (n_p, n_n)$, with $n_p, n_n \in \mathbb{Z}_{\geq 0}$; it has a Beta distribution with expected value $E_{\widehat{r}}$ that describes the pundit's reported view of the trustee's trustworthiness.

2. Assemble a set of historical reports, $\mathcal{R}$ (note $\widehat{r} \notin \mathcal{R}$). These reports were previously requested of the pundit by the truster for evaluating other agents; accordingly, the

truster has a set of corresponding historical outcomes, $\mathcal{O}$ (one outcome for each report).

3. Each $r \in \mathcal{R}$ forms a Beta distribution B_r that describes the trust metric for that report. Each $o \in \mathcal{O}$ is binary: a satisfactory or unsatisfactory historical interaction.

4. Find a maximal subset of historical reports, $\mathcal{H} \subseteq \mathcal{R}$, that correspond most closely to $\widehat{r}$. Equivalently, $\forall h \in \mathcal{H}$, $E(B_h)$ lies within some margin of error of $E_{\widehat{r}}$.

5. Using the outcomes of the previous interactions corresponding to $\mathcal{H}$, create a new Beta distribution B_o. Compute ρ, the proportion of B_o that lies within the margin of error of $E_{\widehat{r}}$.

6. ρ discounts the pundit's report $\widehat{r}$ by perturbing the expected value and variance of a $Beta(1,1)$, a.k.a. $Uniform(0,1)$, distribution, and then using the perturbed values to derive discounted $\widehat{r}$ trust parameters.

TRAVOS builds on the BRS by including some notion of heterogeneity of agents, viz., by discounting reports from peers who are deemed dishonest or untrustworthy (this is roughly equivalent to discounting advice from dissimilar peers). In gathering advice, TRAVOS relies on the assumption that the truster and pundits have extensive historical dealings with which the truster can evaluate each pundit's expected honesty.

Personalized Trust Model (PTM)

Zhang and Cohen [50] suggest a personalized trust model to determine whom to listen to amongst a network of buyers and sellers in the e-marketplace domain. In particular, they address whether a buyer, b, should purchase a product from a seller, s, based on a combination of *global* advice from other buyers (i.e., advisors, a), and b's own *local* past experiences with s.

The PTM global metric is further broken down to combine *private* and *public* trust estimates of advisors. The intuition is that b may have radically different expectations or preferences regarding s's product than a, and so b should have some notion of how much to trust a. To the extent that b relies on past common experiences to evaluate a's trustworthiness, b uses a *private* trust metric to incorporate a's recommendation. To the extent that b relies on a's similarity to the global rating of various sellers (i.e., how fair are a's ratings), b uses a *public* trust metric to incorporate a's recommendation.

The above overview is made more concrete as follows. In PTM, ratings of a product are binary. The Beta probability distribution is used to estimate the probability that an advisor will provide a fair rating to b. To estimate the private reputation of advisor a, PTM defines:

$$R(a)_{private} = E[Pr_a(\text{fair rating})] = \frac{\alpha}{\alpha + \beta} \tag{2.5}$$

where α and β are parameters to the Beta distribution and can be defined as follows:

$$\alpha = 1 + \sum_{s \in S} \sum_{t} \neg(\vec{r}_{b,s,t} \oplus \vec{r}_{a,s,t}) \cdot \vec{1} \tag{2.6}$$

$$\beta = 1 + \sum_{s \in S} \sum_{t} (\vec{r}_{b,s,t} \oplus \vec{r}_{a,s,t}) \cdot \vec{1} \tag{2.7}$$

The notation in these equations have the following meaning:

- S is the set of sellers rated in common by b and a;
- $\vec{r}_{b,s,t}$ is a ratings vector of b's experiences with seller $s \in S$ in time window t;
- $\vec{r}_{a,s,t}$ is a ratings vector for a such that each rating element $r_i \in \vec{r}_{a,s,t}$ corresponds to the most recent rating prior to $r_i \in \vec{r}_{b,s,t}$;
- $\oplus$ denotes element-wise XOR, and $\neg$ denotes element-wise logical NOT.

Equation 2.6 counts the number of times for which b and a have the same rating of $s \in S$, denoted N_s^a in [50]; Equation 2.7 analogously counts the number of non-similar ratings. (Note that the equations presented here appear different than the algorithmic description in [50], however they are equivalent definitions).

An advisor's public reputation, $R(a)_{public}$, is also calculated using a Beta probability distribution as above, except that α is derived using the number of *fair* ratings, denoted N_f^a, where fair is measured as the degree to which a's ratings vector corresponds to the average ratings vector over all ratings of s.

The overall trust level of advisor a is derived by combining $R(a)_{private}$ and $R(a)_{public}$ according to a weight function, viz., the ratio of the number of common ratings between a and b, N_{all}^a, to the minimum number of common ratings, N_{min}, required to achieve some level of confidence. N_{min} is derived from the Chernoff Lower Bound theorem using a confidence level, $\gamma \in [0, 1]$, and an acceptable error level, ϵ:

$$N_{min} = -\frac{1}{2\epsilon^2} \ln \frac{1 - \gamma}{2} \tag{2.8}$$

A sum over all advisors' experiences with s, weighted according to trust values derived according to the above discussion and according to a "forgetting factor" that weights recent interactions more heavily than distant past interactions, is then computed to determine the global reputation of the seller, s. Lastly, the same Beta distribution derivation is used to arrive at a local reputation of s derived from b's own experiences with s, also weighted by a "forgetting" factor. Analagous to the derivation of global trust values for advisors, the overal trust value for a seller is determined via a weighted combination of global and local reputations, according to the Chernoff Bound. (That is, the more local experience a buyer has, the more he will rely on his own experience).

Learning Object Annotation Recommender (LOAR)

Champaign et al. [12] (see also [11]) develop a model for recommending commentary (annotations) on learning objects (texts or videos) to users in a peer-based, online learning environment, which we term LOAR (Learning Object Annotation Recommendation). The model displays those annotations with the highest predicted learning benefits. This work in turn draws inspiration from Zhang et al.'s PTM trust model [50], which reasons probabilistically about the trustworthiness of sellers in an electronic commerce setting in order to select the one with the highest predicted benefit for a user.

In LOAR, when viewing learning objects, users are allowed to rate the attached annotations as valuable (1) or not (0). The "current" user is presented with peer commentary in a way that is customized according to each annotation's predicted learning benefit for that user. An annotation's predicted benefit is calculated using a combination of the annotator's reputation and explicit ratings the given annotation has recieved. An annotator's reputation is derived as follows:

1. An author q has created a set of annotations $A_q = \{a_1, \ldots, a_n\}$, each of which has an associated set of ratings $R_{a_i} = \{r_1, \ldots, r_{m_i}\}$ left by some number, m_i, of students who have experienced the annotation.

2. Compute a set of average ratings, $V = \{v_{a_1}, \ldots, v_{a_n}\}$, corresponding to each annotation using the associated rating set, i.e., $v_{a_i} = \frac{1}{m_i} \sum r_i$

3. The annotator reputation, T_q, is the mean average rating, i.e., $\frac{1}{n} \sum v_{a_i}$.

Here, parallels between Champaign's annotator model and Zhang's trust model begin to emerge. An annotator in LOAR corresponds to a seller in PTM, and the total annotator

reputation, T_q, is akin to Zhang's global reputation of a seller, $R(S)_{global}$. Moreover, peers in LOAR act as advisors in the system, though the calculation of $R(S)_{global}$ from advisor experiences in PTM requires additional work, involving timewindows. One difference is that LOAR focuses on modeling the trustworthiness of *annotations*, whereas PTM is more focused on what would be annotators. It is important to note that, in LOAR, even if the annotator is highly regarded in a community, the predicted value of a particular annotation will be influenced more heavily by the ratings it receives[4].

In addition, in LOAR what constitutes a "local" annotation reputation depends on the number of votes it receives. In particular, votes for (vF_a) and against (vA_a) an annotation a are weighted according to the similarity between the current user and peer voter, which is calculated according to prior votes the pair have cast in common. The global and local annotation reputations are then combined in one of three ways to derive the predicted benefit for the current user (where those with highest predicted benefit are then shown):

1. Tally: the predicted benefit is entirely determined by the "local" reputation by normalizing $\frac{vF_a - vA_a}{vF_a + vA_a}$.
2. Cauchy: combining local and global reputations using a Cauchy CDF, $\frac{1}{\pi}\arctan\left(\frac{vF_a - vA_a + T_q}{\gamma}\right) + \frac{1}{2}$.
3. Trust: blending T_q with the predicted tally benefit according to a minimum number of ratings N_{min} and the number of ratings on the annotation $|R_{a_i}|$. That is, $\min\left(1, \frac{|R_{a_i}|}{N_{min}}\right) \cdot tally + \max\left(0, 1 - \frac{|R_{a_i}|}{N_{min}}\right) \cdot T_q$.

While LOAR was designed with the goal of operating within the context of an intelligent tutoring system that employs a repository of learning objects (allowing peers to leave commentary on those objects and then reasoning about which commentary to show or to avoid), the system fundamentally determines which messages from peers to present, in order to improve the learning achieved by a user. In this respect, it should be suitable to be applied to the task of recommending messages in online participatory media networks, as well.

Bayesian Credibility Model (BCM)

In this thesis, we advocate the use of a "credibility model" as part of our message recommendation algorithm. One model that provides insights into how to model credibility

[4]The annotator reputation serves only as a proxy for ratings when an annotation has not received votes (e.g., is relatively new).

in social networking environments is that of Seth, Zhang, and Cohen [39], who propose a Bayesian model to derive the crediblity of messages within a social network of peers for the purpose of recommending participatory media content (e.g., blog posts, consumer product reviews, Twitter tweets, etc.) to users. BCM uses the *strength of weak ties* hypothesis from social network theory to categorize clusters of users within a social network, G. The topic-induced subgraph of G, denoted G_t, is a subgraph of users who are interested in some topic, t. Users within G_t can be categorized as belonging to particular clusters, i.e., subgraphs of users that are strongly tied and affect knowledge propagation throughout the cluster in certain ways. Clusters are connected together via weak ties to form the topic-induced subgraph G_t.

For each user $u_i \in G_t$, BCM derives a topic-specific crediblity score for each message m_k, denoted $C_{k,t}$. $C_{k,t}$ depends on Contextual (how easily a message is understood, CN) and Completeness (the depth and breadth of media content, CM) information. Context and Completeness are in turn dependent on four sub-crediblity types (evidence variables):

Cluster credibility (denoted $s_{i,k,t}$) is the credibility the cluster of user u_i (denoted V_{it}) assigns to message m_k authored by some other user, u_j.

Public crediblity (denoted $p_{k,t}$) is the credibility that the entire network of users in G_t assigns to message m_k.

Experienced credibility (denoted $e_{i,k,t}$) is the credibility that u_i assigns to message k based on u_i's past experience with the author of m_k, viz., u_j.

Role-based credibility (denoted $l_{i,k,t}$) is the credibility u_i assigns to m_k given that u_j has some role (and thus has some level of expertise).

More formally, the joint probability distribution for the Bayesian network is as follows (we drop the subscripts t, i, k for brevity):

$$P(C, CN, CM, s, l, e, p) = P(C|CN, CM) \cdot P(CN|s, l, e) \cdot P(CM|p, l, e) \cdot P(s) \cdot P(l) \cdot P(e) \cdot P(p) \tag{2.9}$$

To calculate the evidence variables given a topic t, BCM uses:

- an author matrix $\mathbf{A} \in \{0, 1\}^{k \times n}$ where k is the number of messages and n is the number of users, and a_{ij} indicates m_i was authored by u_j;
- a ratings matrix $\mathbf{R} \in \{0, 1\}^{k \times n}$ where r_{ij} is u_j's rating of m_i; and

- an adjacency matrix $\mathbf{N} \in \{0,1\}^{n \times n}$ where n_{ij} indicates the existence of a link (i.e., relationship) between u_i and u_j.

As an example, the calculation of $\mathbf{P}_t$ (the $k \times 1$ vector of public message credibilities) is derived as follows. Credibilies for the other evidence variables can be derived following a similar process. First, the authors compute credibilities owing to the structure of the topic subgraph as $G = (\beta \cdot \mathbf{N}_r^T + (1-\beta) \cdot \mathbf{Z}_c \cdot \mathbf{1}^T) \cdot G$. In this definition, $\mathbf{Z}_c$ is the column-stochastic form of an $n \times 1$ vector $\mathbf{Z}$ wherein each element z_i is the mean similarity between user u_i and and u_j for all $j \neq i$, given $Sim(u_i, u_j)$ is computed using the Jacquard coefficient. Hence, G can be computed as the dominant eigenvector of $\beta \cdot \mathbf{N}_r^T + (1-\beta) \cdot \mathbf{Z}_c \cdot \mathbf{1}^T$. (Note that β is a blending weight to indicate how much the similarity matrix should be perturb the network matrix).

Next, compute $\mathbf{P'}$, the $n \times 1$ vector of public user credibilities using $\mathbf{P'} = (\alpha \cdot \mathbf{A}_c^T \cdot \mathbf{R}_r + (1-\alpha) \cdot \mathbf{G}_c \cdot \mathbf{1}^T) \cdot \mathbf{P'}$, again by computing the principal eigenvector (e.g., using the power method). Finally, the public message credibilities can be derived from the equation $\mathbf{P} = \mathbf{R}_r \cdot \mathbf{P'}$ ($\mathbf{R}_r$ is the row stochastic form of $\mathbf{R}$ above). Lastly, once the evidence variables are found, the Expectation Maximazation algorithm is used to learn the Bayesian model[5].

BLADE

Our proposed solution in Chapter 4 is a POMDP approach. The following trust model is thus quite relevant: it also tries to learn the evaluation functions of advisors as a stand in for trust modeling and with a Bayesian perspective. As will be revealed in Chapter 5, the primary difference between our approach and BLADE is our consideration of user utilities and the inherent decision-making aspect of our POMDP classification model.

In [35], Regan et al. developed a model called "BLADE" (Bayesian Learning to Adapt to Deception in E-Marketplaces). In BLADE, sellers are evaluated on the basis of several features, say k of them. For example, a particular seller, s, could be evaluated based on product quality, the speed with which purchased products are delivered, the degree to which products correspond to their advertised descriptions, etc. In general, each feature can take on a number of values with some probability (e.g., delivery speed could be "on time", "one day late", "one week late", etc.). Accordingly, each seller feature, denoted F_i^s, can be

[5]It is worth noting here that these fixed point equations were derived in [39] using a variety of heuristics that define credibility recursively (e.g. "a message is credible if credible peers rate it highly" and "a peer is credible if he has many credible messages"). The exact definitions are not reiterated in this paper for brevity, but can be reviewed in [39].

viewed as a random variable that follows a multinomial distribution; each F_i^s is therefore characterized by a corresponding probability vector $\vec{\Theta}_i^s$. The goal of a prospective buyer is to learn each $\vec{\Theta}_i^s$. In particular, $\vec{\Theta}_i^s \sim Dir(\vec{\alpha})$, where the Dirichlet hyperparameters are interpreted as the number of past observations of corresponding events, and buyers perform Bayesian updates on the basis of evidence they observe with each interaction. Then, given beliefs about $\vec{\Theta}^s = \{\vec{\Theta}_i^s\}_{i=1}^k$, buyers can reason about $Pr(\vec{F}^s = \vec{f}|\vec{\Theta}^s)$, and can furthermore make purchasing decisions that depend arbitrarily on utility derived from a given realization of seller features.

Of course, buyers may not always have sufficient past interactions to draw upon when inferring seller feature distributions. In such situations, buyers can elicit feedback from advisors who provide ratings, denoted R_s^a, based on their own past interactions with the seller in focus. The beauty of BLADE lies in its treatment of such advice. In general, advisors report ratings — perhaps stochastically — according to some private reporting function[6]. That is, advisors report ratings that are described by the function $Pr(R_s^a|\vec{f})$. Once again, R_s^a is in general a multinomial random variable, and so ratings can be described as depending on parameters $\vec{\Theta}_s^a \sim Dir(\vec{\alpha})$. The induced Bayesian network can then be queried by marginalizing out unobserved variables to answer questions about, for example, $Pr(\vec{f}^s|\vec{r}_s^a)$, i.e., given the reported ratings, what is the probability of realizing a particular combination of seller features? The true power in this approach is that advisors do not even have to provide ratings *based on the same seller features* as the buyer — so long as advisor criteria are not completely anti-correlated with the seller features upon which the buyer's utility depends, and so long as advisors are consistent (across different sellers) and reasonably deterministic in their ratings, BLADE will correctly learn the parameters that drive the network.

To make these notions more concrete, Figure 2.1 exhibits a plate model of the Bayesian network described above (note that the illustration given here is a more general version of the examples provided in [35]). Ultimately, BLADE can be seen as a generalization of several other trust models. In particular, BRS and TRAVOS can both be described by a particular realization of the BLADE Bayesian network wherein each seller has a single binary feature variable F^s that corresponds to the seller's overall reputation/trustworthiness (i.e., trustworthy or not) and that is fully determined by a single parameter Θ^s. By constrast, BLADE allows buyer utilities to depend arbitrarily on various seller features, and allows advisors to report ratings using whatever (independent) scale they deem most suitable.

[6]In particular, BLADE can handle equally well advisors who report truthfully or deceptively. However, BLADE depends on *consistent* or deterministic ratings; more stochastic rating functions offer less information to be inferred.

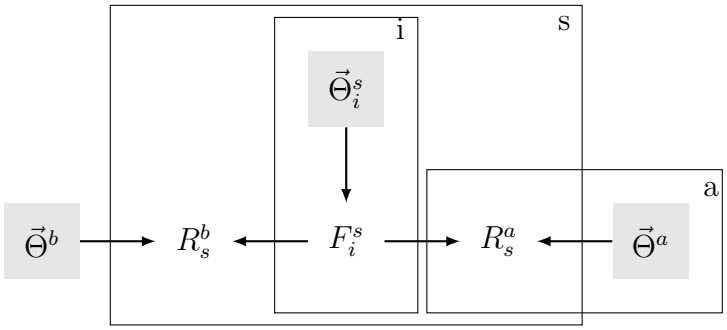


Figure 2.1: The induced BLADE Bayesian network plate model for a single buyer. The highlighted independent parameters are modeled as Dirichlet distributions; a buyer updates his belief about these parameters based on observed evidence about seller features F_i^s and observed advisor ratings R_s^a.

Trust Modeling with Propagation in Social Networks

Hang and Singh [23] and Hang, Zhang, and Singh [24] approach the problem of trust in a social network with a different view. Their work focuses on trust propagation in a network of peers. In [23], the authors create an approach called "LocPat", which recommends trustworthy agents on the basis of a graph similarity. In particular, recommendations are made by suggesting links to other agents (vertices in the graph) based vertex similarity to nodes in a *structure graph*. Structure graphs are essentially heuristics for making recommendations. For example, one such structure graph might encapsulate the notion that a "friend of a friend" is a good candidate for recommendation (in fact, this heuristic is used by Facebook to recommend potential relationships). Thus, agents to which multiple neighbours connect (so-called "friends of friends") with high trust value weights might be good candidates for recommendations.

In [24], the authors develop an approach called "Shin" for propagating trust through unreachable witnesses, viz., those agents in a network graph for which there is no trustworthy connecting path. The main idea is to evaluate the trustworthiness of a witness by evaluating the degree of trust the witness places in some common acquaintance. This trust model describes trust using a Beta distribution and also uses the notion of confidence (similar to TRAVOS) to discount trust values.

This research suggests some future directions for our own work, which we discuss at greater length in Section 6.2.

2.2.2 Latent Factor Models

Latent Factor Models (LFMs) provide an alternate collaborative filtering method to the neighbourhood method for making item recommendations to users. They work by inferring some number of factors, f, from user ratings on items [29]. The meaning and nature of the inferred factors are unknown (hence "latent" factor); they are "uncovered" in the process of analyzing user ratings patterns. Factors essentially characterize the items and users of interest and encode similarities between them.

One particular class of LFMs use matrix factorization. In such models, each item i is mapped to a vector $q_i \in \mathbb{R}^f$ and each user u is mapped to a vector $p_u \in \mathbb{R}^f$ so that the dot product $q_i^T p_u$ is a prediction of u's rating for item i [29].

In order to compute rating predictions under matrix factorization models, it is necessary to derive the vectors q_i and p_u. One way to do this is to perform singular value decomposition after assembling a user-item ratings matrix and imputing missing ratings [37]. However, imputing ratings, which is necessary to ensure the user-item ratings matrix is "dense", can be expensive and can distort the data [29]. Instead, more recent approaches seek to minimize the following objective function given a set of ratings $r_{i,u} \in R$, i.e., ratings by user u on item i:

$$\sum_{r_{i,u} \in R} (r_{i,u} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2.10}$$

Here, $\lambda \geq 0$ is a regularization constant to prevent overfitting. Two typical techniques for minimizing the above quantity include *stochastic gradient descent* (SGD) and *alternating least squares* (ALS) [29]. SGD seeks to minimize the quantity directly by iteratively computing a prediction error and subsequently improving the estimates for q_i and p_u by taking steps in a direction opposite the gradient given the current values for q_i and p_u. ALS, by contrast, alternates between fixing q_i and p_u at each iteration. Doing so transforms Equation 2.10 into a convex function so that the global minimum can be computed exactly [2]. The SGD method is typically faster than ALS (especially when ratings are sparse), easier to implement, and tends to perform well (although it is not guaranteed to converge to a global optimum because the objective function is non-convex). However, ALS can potentially be highly parallelized to great benefit [51], and can be more efficient when considering implicit feedback from users about their preferences, in which case user preference data is not sparse (so that looping through each user/item rating on each iteration, as in SGD, is not practical) [25].

The SGD-based recommendation approach is reintroduced in Section 4.1.4 as a competitor, when we map out a validation of our proposed model.

2.3 Markov Decision Processes (MDPs)

In this section, we briefly introduce Markov Decision Processes and describe one mechanism, Monte Carlo Tree Search, for computing a policy in an online fashion. This background is especially valuable as the backdrop to Chapter 4.

2.3.1 Fully Observable MDPs

A Markov Decision Process (MDP) is a mathematical model for decision making in a sequential, stochastic environment. In an MDP, a decision-making agent navigates through a state space by taking actions in a sequential manner (for example, at discrete time steps). The agent's goal is to maximize the rewards it receives and minimize the costs in incurs as it navigates through the world. More formally, an MDP is characterized by a 6-tuple (S, A, T, R, γ, h), where the components are as follows:

- S is a set of states that encode information about the domain.
- A is the set of actions the agent can take at each time step.
- $T : S \times A \times S \mapsto \mathbb{R}$ is a transition probability kernel that encodes $Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$, the probability with which a state will be reached given the current state and action.
- $R : S \times A \mapsto \mathbb{R}$ is a reward function that dictates the rewards (or equivalently, utility) the agent receives given the current state-action pair.
- $0 \leq \gamma < 1$ is a discount factor that encodes the tradeoff between immediate and future rewards. Setting $\gamma = 1$ has the effect of making future rewards exactly as valuable as present ones (there is no time-weighted discounting). On the other hand, $\gamma = 0$ indicates that the agent derives no utility from future rewards (full discounting).
- h is the horizon for the process, which could be infinite.

In a fully observable MDP, the agent knows the current state and is able to take an action based on this knowledge. For example, an MDP can be used to describe the process by which a robot navigates through a room: the state space corresponds to its current location, it can choose to move forwards or backwards, it receives a negative utility for crashing into a wall, etc.

A policy $\pi : S \mapsto A$ is a function that encodes the action the agent should take given its current state. Since the agent derives utility at various points in time depending on the states it reaches and the actions it takes, we can calculate the maximum expected utility for a given state in the MDP. One technique for accomplishing this, called "value iteration", uses dynamic programming to recursively solving Bellman's equation [3]:

$$\begin{aligned} V^*(S_t = s) &= \max_{a \in A} R(s, a) + \gamma \sum_{s' \in S} Pr(S_{t+1} = s' \mid S_t = s, A_t = a) V(S_{t+1} = s') \\ V^*(S_0 = s) &= \max_{a \in A} R(s, a) \end{aligned} \tag{2.11}$$

This system of equations can be solved by computing the values of states in reverse chronological order. Doing so implies a policy π^*, which yields a maximum expected utility for the agent when followed. Note that the *value function* V, a function of the state, essentially encodes $\max_a Q(s, a)$, the maximum value over all actions that could be taken in a given state. The latter function is known as the Q function.

2.3.2 Partially Observable MDPs (POMDPs)

A Partially Observable MDP (POMDP) is an MDP for which the agent does not know the current state. A POMDP has two additional components to describe the model: O is a set of observations that the agent receives that are in some way correlated with the underlying state, and $\Omega : S \times A \times O \mapsto \mathbb{R}$ is the observation probability kernel that encodes the function $Pr(O_{t+1} = o \mid S_{t+1} = s, A_t = a)$.

The navigation process in a POMDP is slightly different than in an MDP: after each action, the agent makes an observation. This action-observation sequence induces a history $h_t = \langle a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t \rangle$ that describes the nature of the agent's traversal up to time t. Unfortunately, since the states are unobservable, the agent can no longer follow a Markovian policy; rather, policies must dictate actions based on the entire history the agent experiences. That is, a policy for navigating a POMDP is a function $\pi : B_0 \times H_t \to A_t$ from the initial belief space B_0 and the history space H_t to an action. The initial belief space is a probability distribution over the starting state for the process, which encodes the fact that the agent might not know the starting state.

Since the space of histories induces an exponentially large policy tree (in which each root-null path represents a different action/observation sequence) for each initial belief, we need to represent the POMDP differently. POMDPs can be mapped to continuous belief-state MDPs, which allows us to reinforce the Markovian assumption when solving

for an optimal policy. In particular, we designate the agent to maintain a belief about the underlying message state; this belief is updated at every time step in the process, viz., after each observation, according to the following equation[7]:

$$b_{t+1}(s_{t+1}) \propto Pr(o_{t+1} \mid s_{t+1}, a_t) \sum_{s_t} Pr(s_{t+1} \mid s_t, a_t) b_t(s_t) \tag{2.12}$$

By maintaining and updating beliefs, we can devise a policy function $\pi : B \rightarrow A$ from beliefs to actions. Thus, the belief become sufficient statistics that encode a history of actions/observations.

2.3.3 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is an online, stochastic technique for deriving a POMDP policy. The technique constructs a partial search tree by simulating actions over a number of iterations to a certain depth and pruning branches that seem unpromising, viz., those branches that return a minimum mean reward when followed during simulation. When simulating POMDP traversals, the decision-making agent chooses an action that balances the exploration of unknown or less well-searched paths with exploitation of known low-cost, high-reward paths. One elegant mechanism that achieves a good balance relies on the use of *upper confidence bounds* (UCB) [1]:

$$\mu_a \leq \kappa \cdot \bar{\mu}_a + c \cdot \sqrt{\frac{\ln n}{n_a}} \tag{2.13}$$

Here κ is a constant that scales the rewards in case they do not lie in the interval $[0, 1]$ [46], c is a tuning constant, and, in the context of MCTS, n is the total number of simulation runs and n_a is the number of times a particular action was taken (and hence, a particular branch followed)[8]. During simulation, the agent chooses the action a that maximizes the UCB expression.

By balancing exploration and exploitation during simulation and pruning accordingly, MCTS methods construct partial and unbalanced search trees, thus exploring some paths more deeply than others. As the accuracy of Monte-Carlo estimates of tree node values (i.e., estimates of the Q function) improve, the search tree expands in the direction of the

[7] The proportionality hides a normalizing constant that ensures that the beliefs over all states sum to 1, since the beliefs represent probabilities of the underlying state.

[8] The UCB algorithm comes from the multi-armed bandit domain, wherein n_a is the number of times machine a was tested and n is the total number of tests.

most promising nodes [19]. When a simulation reaches a state that is not contained within the simulation search tree, the simulation continues by following a *roll-out policy* [19]. The general process can be outlined as in Algorithm 1 [21]. MCTS resurfaces in Section 4.1.3 as one means for computing a policy in our POMDP classification model.

Algorithm 1: A Generic MCTS Algorithm (details from [21, 41, 47] excluded)

Input: An empty search tree, t, an empty history h, an observation/reward generator $\mathcal{G}$

repeat
  begin Do n simulations starting history h
    Sample state s from current belief state
    if *s has been visited* **then**
      Select action via UCB
      Generate observation/reward from $\mathcal{G}$
      Update search tree nodes
      Recurse from new state s'
    else
      Follow rollout policy
      Update search tree nodes
    end
  end
  Take action $\underset{a}{\text{argmax}}\left(Q(s,h,a)\right)$
  `//` Q `is the Monte-Carlo estimate of the true` Q `function`
  Receive observation and reward from environment
  Update history and belief state
until *process ends (e.g., no more observations)*;

Chapter 3

User Credibility and Folklore

In this chapter, we explore the notion of credibility as it arises to combat the potential proliferation of folklore through a network of agents. We develop a simple, heuristic-based approach to cope with ratings in environments that contain many non-credible users[1]. To begin, we present a motivating example to illustrate where the LOAR model may fall short for the problem of recommending messages to users in participatory media networks. Then, we outline our new CredTrust model, encapsulated in an algorithm and clarified through an example and simulations.

3.1 Motivation: Folklore and Popularity

It would be helpful to refer back to Section 2.2.1 while reading the subsequent text, as we make use of the terminology and notation presented in the original LOAR discussion. To begin, we develop an example that exhibits the "folklore" problem in Champaign's model. This is the scenario where strong similarity among uneducated users is heeded at the expense of valuable opinions of less similar, credible peers. We sketch what LOAR does with this and assume the following:

- A single annotator, a, has created a set of 6 annotations/messages, $M_a = \{m_1, \ldots, m_6\}$. The 6^{th} annotation contains false information (e.g., claims cancer can be cured by magic crystals).

[1] A preliminary version of this work appeared in [36]

Table 3.1: User message ratings

	m_1	m_2	m_3	m_4	m_5	m_6
p_1	0	1	1	1	0	1
p_2	1	1	1	1	0	1
p_3	0	0	0	1	0	0
p_4	0	1	1	0	1	1
s	1	1	1	1	1	?

Table 3.2: Peer similarities to student s

	p_1	p_2	p_3	p_4
s	0.2	0.6	-0.6	0.2

- Four peers, p_1 through p_4, have experienced and rated a's annotations.
- We are trying to determine whether to recommend m_6 to a user s.

Table 3.1 shows the ratings given by each respective participant to each $m_i \in M$. In Table 3.1, a's reputation is simply the mean average rating, T_q, which can be verified[2] to be 0.6583. Next, we calculate the participant pairwise similarity scores per Champaign's model (see Table 3.2). Note that we only show the relevant pairwise similarities (to user s).

To calculate these scores, LOAR finds the number of common ratings that are the same between two agents, denoted vS, and the number of common ratings that are different, denoted vD. The final LOAR similarity metric is given by $\frac{vS - vD}{vS + vD}$.

Next, LOAR uses the similarity scores along with peer scores for the marginal message in order to derive a trust value for that message, for the user. In particular, LOAR records the votes for and against a given message, and combines these. So, for example, p_1's vote for m_6 would increase the "votes for" tally by $1 + 1 * 0.2 = 1.2$. On the other hand, p_2 would increase the "votes for" tally by $1 + 1 * 0.6 = 1.6$, and $p3$ would increase the "votes against" tally by $1 + 1 * (-0.6) = 0.4$. Lastly, the system combines the ratings on m_6 with a's overall reputation (T_q) to derive the predicted benefit, which results in a predicted benefit of 0.86 and 0.76, for Cauchy and Trust, respectively.

[2]Please refer to Appendix B for an deeper clarification of the calculations presented throughout this example.

Clearly then, m_6 has a very high predicted benefit. Accordingly, there is a high likelihood that m_6 will be shown to s (it will likely be over a predetermined threshold of acceptability, and will be among the top annotations). However, the message contains "errors" that could detract from learning, and so should in fact not be shown to s. This example shows how the popular opinion of a message could ultimately enable false information to spread in a network of peers.

3.2 New Trust Model to Address Folklore

At first glance, it appears as though the folklore problem could be addressed by classifying peers into different roles and weighting peer feedback according to these roles. For example, one could introduce two classes of users, "students" and "professors", and set professors' weights to infinity. This would allow professors to prevent false message propagation; a single bad vote from a professor would outweigh any number of votes from students. But even "professors" can be wrong, so instead of an infinite weight, one could set the weights according to some heuristic that allows for a sufficiently large number of other user roles to outweigh a "super user". Even then, it seems reasonable that the weights of users in any role should be variable, owing to the fact that users can make mistakes and can gain and lose credibility in a community.

Instead, we proceed first by redefining the notion of "trust". In LOAR, a trust metric was annotation-specific, and corresponded to the predicted benefit of a given message, where predicted benefit was a combination of an annotator's mean reputation and the annotation's similarity-weighted rating. Instead of introducing a weight under the same model, we develop a new model drawing on the probability theory used throughout trust literature. We draw inspiration from BRS [28], PTM [50], and TRAVOS [44] in the use of Beta distributions, but continue to model the benefit of annotation objects themselves, drawing inspiration from Seth's Bayesian Credibility Model (BCM) [39] in this regard.

3.2.1 Recasting the trust model

Determining whether an annotation or message will be well-received (i.e., is beneficial) is not deterministic; it can instead be modelled as a Bernoulli process. That is, if M is the event that a message is well-received, then we seek to determine $\psi = Pr(M)$. Moreover, we allow this parameter to itself be represented as a random variable and rely on Bayes' theorem to update prior probability distributions over ψ. In particular, we can use the

Beta distribution[3] to represent the prior $Pr(\psi)$:

$$\psi \sim Beta(\alpha^*, \beta^*) \tag{3.1}$$

However, since we model the trustworthiness of messages (not annotators), the user does not have any prior belief that directly corresponds to the message itself (has yet to experience it, so the only rational belief is to assume that $\alpha^* = \beta^* = 1$, i.e., that ψ is uniformly distributed in the interval $[0, 1]$). Accordingly, we construct a suitable belief by looking to the experiences of peers, as in LOAR[4].

When a user solicits feedback about a message, his peers report binary ratings. Equivalently, peers report parameters α_p and β_p such that $\alpha_p + \beta_p = 1$. In this work, we restrict this report[5] such that $\alpha_p, \beta_p \in \{0, 1\}$. To combine peer reports, we model the similarity between users i and j using Hamming distance. The Hamming distance is a measure of the number of bits by which two binary strings differ, or equivalently, how many changes need to be made to string a to transform it into string b. Here, we can consider the series of common annotation ratings between two users to form "binary rating strings". (Table 3.1 above shows such a set of binary rating strings in the form of a matrix).

We normalize the Hamming distance between i and j to arrive at a similarity metric called the Hamming ratio, denoted h_{ij} (the Hamming distance divided by the length of the binary strings, i.e., the number of common ratings). Since a Hamming distance of 0 means that the two strings are identical, a Hamming ratio of 0 suggests we simply take a peer report as given; in contrast, if the Hamming ratio is 1, we swap the values reported for α_p and β_p. This captures the fact that non-similar peers can still deliver useful information; perfect negative correlations are just as informative as positive ones. We formalize this combination as follows:

$$\alpha^* = 1 + \sum_{p \in P} (1 - h_{sp}) \cdot \alpha_p + h_{sp} \cdot \beta_p \tag{3.2}$$

$$\beta^* = 1 + \sum_{p \in P} (1 - h_{sp}) \cdot \beta_p + h_{sp} \cdot \alpha_p \tag{3.3}$$

[3] In particular, α^* (β^*) represents the strength of a user's belief that a message will be good (bad). Thus, when α^* is high and β^* is low (β^* is high and α^* is low), the user will be very confident that he should see (not see) the message.

[4] One could potentially use a more informed prior, such as the trustworthiness or reputation of the annotator, to inform the distribution. Another use for such an informed prior is presented in Section 3.2.3, and we revisit this notion as future work in Section 6.2.1.

[5] A report of 1 corresponds to the combination $(\alpha_p, \beta_p) = (1, 0)$ whereas a report of 0 corresponds to $(\alpha_p, \beta_p) = (0, 1)$.

Here, P is the set of all peers. This combination capitalizes on the fact that the Beta distribution is well-defined for all real-valued parameters $\alpha, \beta > 0$. Moreover, it allows us to easily extend peer reports to include expectations on message trust values. That is, a report $r \in [0, 1]$ can be translated into parameters $(\alpha, \beta) = (r, 1 - r)$ so that a report of $r = 1$ corresponds to $\alpha = 1, \beta = 0$, a report of $r = 0.5$ corresponds to $\alpha = \beta = 0.5$, and $r = 0$ to $\alpha = 0, \beta = 1$. Thus, a user can solicit feedback from peers about an annotation even if those peers have yet to personally experience the annotation. This is useful if, for example, the current user has no or limited ratings in common with peers who have rated the annotation in focus. (That is, it might be more useful to use a report of expected benefit from a peer who is highly similar to the current user rather than use an explicit report from a peer with whom the user has no history and thus no notion of similarity). This notion, while not explicitly considered in this work, relates to the idea of trust propagation as explored by Hang et al. [24] and trust delegation by Burnett and Oren [10].

3.2.2 Incorporating a measure of credibility

Under the new trust framework described above, we now introduce the notion of credibility. Credibility is a measure of the extent to which users should trust the opinions of peers within the community. Thus, credibility influences the similarity weighted Beta distribution derived above. In particular, we now also seek to determine $\kappa = Pr(C)$, where C is the event that a peer report is credible.

As before, we assume that κ is randomly distributed and can be described by a Beta distribution. Thus, the credibility metric $E[Pr(\kappa)]$ is reported alongside peer ratings of annotations. For now, we assume that this credibility score is made available by an oracle, and we leave a discussion of one possible derivation to Section 6.2.5.

The question that remains is how a user should combine his knowledge of peer credibility and a particular annotation's reputation gleaned through peer reports. When a user is highly similar to the peer from whom he receives a report, this combination is trivial; credibility can directly discount the reported rating. That is, one should listen to the advice of highly credible and similar peers more than the advice of non-credible peers. However, a difficulty arises when a user is dissimilar from a credible peer. In this circumstance, our above model will reverse the opinion of a credible peer. In some instances, this reversal could actually detract from the user's learning.

To make this more explicit, suppose that user i solicits advice from user j about a message m. Suppose further that the Hamming ratio between i and j is 1 (that is, they are completely opposite). Then, if j reports $(\alpha, \beta) = (0, 1)$ (i.e., he thinks the message

Algorithm 2: Deriving a predicted benefit using similarity and credibility (CredTrust)

Input: The current user, u, his set of peers, P, their credibility scores, $c_p \in [0, 1]$, and their corresponding ratings for the annotation in focus, $r_p \in \{0, 1\}$

Output: Parameters α^* and β^* to a Beta distribution describing trust in the current annotation

```
1  α* = β* = 1 // At the start, user has a uniform expectation about
       the message
2  foreach p ∈ P do
3  |   h_up ⟵ computeHammingRatio(u, p)
   |   // Perform a Bayesian update after discounting heuristic
4  |   if r_p == 0 then
   |   |   // Adjust the similarity weight by credibility:
5  |   |   α*+ = h_up(1 − c_p)
6  |   |   β*+ = 1 − h_up · (1 − c_p)
7  |   else
   |   |   // Else simply compute a credibility-dampened trust score
8  |   |   α*+ = c_p · (1 − h_up)
9  |   |   β*+ = c_p · h_up
10 |   end
11 end
```

not useful, or perhaps even incorrect), the similarity weighting scheme described above will reverse this opinion to $(\alpha, \beta) = (1, 0)$ when determining the trust metric from i's perspective. That is, the message, which j thinks should not be shown, will now be more likely to be shown. However, in this case, if j is perfectly credible, his opinion of a message corresponds to a very credible one. Accordingly, his report might be better taken verbatim rather than dampened by the Hamming ratio.

Accordingly, we propose a scheme detailed in Algorithm 2. This algorithm computes a trust metric by discounting peer reports by their community credibility (c_p), except when they report negatively on the given annotation. When this happens, the peer's negative rating is weighted using a combination of similarity and credibility. In particular, the role that similarity plays in blending the reported message rating is linearly reversed as the peer's credibility approaches 1 (i.e., perfect credibility). This credibility weighting scheme helps to address the issue of folklore propagation in an e-learning system. That is, highly

credible peers (like professors and TAs) who report negatively about a given annotation will hold more sway than a number of less credible peers, even if the credible voters usual voting patterns tend to make them dissimilar to the current student.

3.2.3 Incorporating annotator reputation

Lastly, we address the notion of an annotator's reputation. It is useful to model an annotation's reputation using some combination of explicit annotation ratings and the annotator's inherent reputation, especially when the given annotation has little or no explicit ratings. We will assume that an annotator's reputation is the same as his credibility (described and used above). In order to calculate the credibility for user u, we propose the following heuristic, inspired by the recursive derivation of credibility evidence variables in BCM [39]: that a peer is considered credible if credible peers vouch for his annotations. We can derive such a credibility score as follows:

1. User credibility is given by $\kappa_u \sim Beta(\alpha_{c_u}, \beta_{c_u})$. In this paper, we assume that a single, global credibility distribution across all subjects suffices to describe the reputation of an annotator (versus a topic-specific metric as in BCM).

2. When a peer p rates message m_u authored by u, p's report updates α_{c_u} and β_{c_u} as follows: a positive (negative) rating will increment α_{c_u} (β_{c_u}) by 1.

3. A rating by p should also only affect u's credibility to the extent that p is credible. That is, when p rates m_u, the κ_u hyperparameters are incremented as above, except that p's report is discounted by $E(\kappa_p)$. Thus, if p rates m positively and p is perfectly credible, i.e., $E(\kappa_p) = 1$, α_c is incremented by 1. If p is not credible at all, i.e., $E(\kappa_p) = 0$, then α_c is incremented by 0 (i.e., non-credible peers cannot influence u's credibility).

Ultimately, the annotation-specific reputation and annotator credibility can be combined to finalize a trust metric using either of the schemes proposed in LOAR (e.g., Cauchy-based combination), or by using an informed prior[6].

[6]This only works for users who have created annotations and is only useful if those annotations have received ratings.

Table 3.3: Similarities and Hamming Ratios

	p_1	p_2	p_3	p_4
s	0.2	0.6	-0.6	0.2
h	0.4	0.2	0.8	0.4

Table 3.4: α and β reports ($c_i = 1$)

	p_1	p_2	p_3	p_4
h	0.4	0.2	0.8	0.4
(α_p, β_p)	(1,0)	(1,0)	(0,1)	(1,0)
(α'_p, β'_p)	(0.6,0.4)	(0.8,0.2)	(0,1)	(0.6,0.4)

3.2.4 Example Revisited

Returning to the example in Section 3.1, we present the LOAR similarities and Hamming ratios together in Table 3.3. Here we see very clearly the relationship between the similarity metric used by LOAR and the Hamming ratio. In particular, a higher Hamming ratio corresponds to a similarity that is closer to -1. A mapping $f : h \mapsto s$ from row h to row s is defined as $f(h) = 1 - 2 \cdot h$. Hence, these metrics fundamentally measure the same thing and differ only by an affine transformation.

The CredTrust algorithm dampens trust values according to peer credibility. A peer is credible if credible peers rate their annotations highly, or if credible peers rely on their ratings. For example, Table 3.4 shows the initial α_p, β_p reports as well as their credibility-weighted values α'_p, β'_p (as given by Algorithm 2) when peers are all perfectly credible. Using these values, we can see that the reported predicted benefit would be 0.5. Upon reflection, this predicted benefit makes sense. Similarity forms a continuum between "exactly like me" and "exactly opposite me". Peers p_1 and p_4 have Hamming ratios of 0.4, indicating they are centered in that continuum. Accordingly, one cannot gain much insight from their reports. Furthermore, p_2 (p_3) has a Hamming ratio of 0.2 (0.8). On the balance, these two peers' opinions should offset each other. Ultimately, we cannot learn anything about the trustworthiness of the medium in this case, since all peers are equally and perfectly credible.

Table 3.5 illustrates the case where all peers have a credibility score of 0. These combinations result in a trust metric equal to 0.6. In this case, the algorithm completely disregards the votes of peers who liked the given message. However, p_3, who is almost completely opposite to s, disliked the message. Even though p_3 is not credible at all,

Table 3.5: α and β reports ($c_i = 0$)

	p_1	p_2	p_3	p_4
h	0.4	0.2	0.8	0.4
(α_p, β_p)	(1,0)	(1,0)	(0,1)	(1,0)
(α'_p, β'_p)	(0,0)	(0,0)	(0.8,0.2)	(0,0)

or perhaps because he is not credible, the algorithm reverses his opinion, resulting in a predicted benefit of 0.6.

Finally, if $c_1 = c_2 = c_4 = 0$ and $c_3 = 1$, then the message will have a predicted benefit of 0.3; the opinion of the credible p_3 completely outweighs the advice of similar, non-credible peers. This is captured by lines 5–6 in Algorithm 2. This scenario would capture well the case that motivated our exploration of folklore: a dissimilar expert who rates a message as problematic should assist in overriding popularity of messages among similar, less credible peers.

3.3 Agent Simulations

3.3.1 Experimental Design

To evaluate this credibility-based trust model, we conducted simulations and compared the performance of CredTrust verus LOAR. We began by simulating an environment consisting of 20 agents, each of whom create messages and rate messages created by other the agents. Each agent is assigned a credibility score according to one of two schemes depending on the experiment, described below:

Mean Credibility Trials Agents are assigned randomly generated credibility scores at the outset, distributed according to a Binomial distribution. The distribution parameters are varied on each simulation iteration. For example, choosing a mean credibility of 0.5 ensures that approximately half of the agents will have a credibility score greater than 0.5.

Dichotonomous Agent Trials Agents are partitioned into one of two sets: low credibility agents or high credibility agents.

When authoring messages, credibility scores influence the "underlying message credibility" of the messages the agents create. For example, when an agent has a credibility of 0.5,

approximately half of the messages it authors will be simulated to be beneficial and approximately half of the messages will have a "flaw" that detracts from agents' utilities if read[7].

In addition to credibility, agents are randomly assigned a type $\theta_a \in [0, 1]$. The agent's type is a parameter that influences similarity; agents of the same type tend to like the same messages. Moreover, messages have a type $\theta_m \in [0, 1]$ in order to appeal to different agents. In particular, we simulate agents rating messages more highly when those messages correspond to their type. However, agents' evaluation of the credibility of each message is modeled by flipping biased coins with probabilities proportional to their own credibilities; if an agent considers a message to be credible, and that message closely matches the agent's type, it will rate the message highly. The result is that less credible agents tend to rate messages they like highly, irregardless of any misinformation or flaws contained within the message.

Each agent randomly produces between 1 and 10 messages and rates all of the messages produced by other agents. In order to evaluate the quality of the inferred benefits for messages, we randomly partition messages into a training and validation set. The training set is composed of approximately 70% of the messages and is used for the purpose of determining the Hamming distances (for CredTrust) and the similarities and author reputations (for LOAR).

Once all of the algorithm inputs have been computed, the simulation runs each algorithm on the testing set to find the predicted benefits for each message based on the advisory ratings. If the predicted benefit of a message is determined to be high (i.e., greater than 0.5), the message is recommended; otherwise, it is rejected. We compute the number of correctly classified messages (i.e., correctly recommended or correctly rejected) by comparing to the "correct" message classifications (based on the known benefits of each message to each agent) and report the Matthew's Correlation Coefficient (MCC), which relates the true positive, false positive, false negative, and true negative rates[8]. In these simulations, we evaluate four algorithms for predicting benefit, which include a constant prediction scheme (which recommended all messages), a random prediction scheme (which randomly classifies messages), the Tally version of LOAR, and CredTrust.

[7]This treatment of authorship allows for credible (non-credible) agents to potentially create poor (good) messages, as in LOAR.

[8]The MCC metric is in the interval $[-1, 1]$, with higher numbers being better. In particular, 1 implies perfect classification accuracy and 0 means no better than random accuracy.

3.3.2 Results

The simulation results are depicted in Figures 3.1a and 3.1b. Figure 3.1a illustrates the simulation for which user credibilities were randomly generated according to a Binomial distribution. In this scenario, we see that the performance of both CredTrust and LOAR generally improve as mean overall credibility in the system increases. As mean credibility approaches 1, both LOAR and CredTrust converge to approximately the same performance. This result follows by examining the heuristics that both approaches use to ascribe benefits to messages; as credibility tends towards 1, CredTrust increasingly "ignores" credibility information and performs a similar update as LOAR. A widening performance gap is exhibited as mean credibility in the system falls; LOAR outperforms slightly when mean credibility is roughly 0.6, while CredTrust outperforms as mean credibility falls below 0.5. This result makes clear that CredTrust is increasingly able to delineate between useful ratings and stochastic ones, especially in the presence of a large number of non-credible agents. In some sense, the CredTrust heuristic informs which agents are trustworthy and worth listening to in an environment replete with non-credible agents.

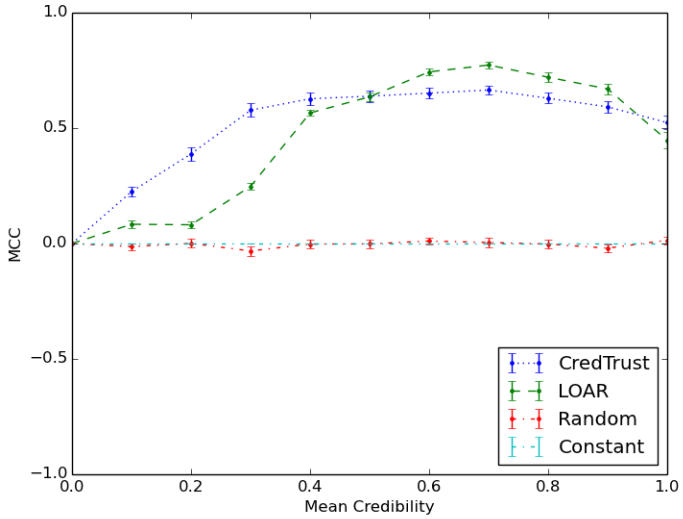
The results in Figure 3.1a suggested a second experiment (shown in Figure 3.1b). In particular, the results of the first experiment make clear that there is some interplay between message credibility and message benefits as the mean credibility in the system evolves, but in particular, LOAR and CredTrust tend to diverge as mean credibility falls. Accordingly, we simulate an environment in which users are partitioned into two general credibility classes: low (corresponding to a credibility of 0.05) and high (corresponding to credibility 0.95). The results of this simulation are make clear that CredTrust is able to correctly account for diverging agent credibilities when ascribing benefits to messages.

3.3.3 Closer look at LOAR

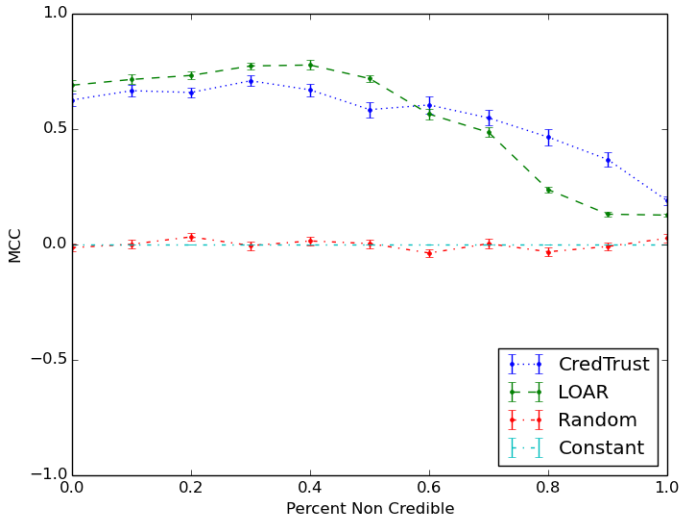
We initially found it surprising that LOAR performed so well in simulation, given that its authors describe the model as heuristic-based and use what appear to be arbitrary (albeit intuitive) functions to combine peer similarity and ratings. We now demonstrate that the "heuristics" used in LOAR in fact have some statistical grounding, which helps to explain why LOAR performs as well as it does.

Lemma 3.1. *The similarity coefficient, S, used in LOAR corresponds to a Maximum Likelihood Estimation (MLE).*

Proof. Let X be the event that two users, A and B, rate a given message the same (note that ratings in LOAR are binary). Then $X \sim Bin(1, \theta)$, where $Pr(X_i = 1) = \theta$ is



(a) MCC versus mean credibility for randomly generated agent credibilities.



(b) MCC versus percentage of non-credible advisors for dichotonomous agent credibilities.

Figure 3.1: Matthew's Correlation Coefficient (MCC) performance for various agent environments (higher MCC is better).

unknown. Let $\{x_1, x_2, \ldots, x_n\}$ be the evidence set of commonly rated messages between A and B (where $x_i = 1$ if and only if A and B rated the i^{th} message the same), and suppose that each instance was drawn independently from X. Then the likelihood function $\mathcal{L}(\theta)$ is as follows:

$$\begin{aligned}
\mathcal{L}(\theta) &= f(x_1, \ldots, x_n; \theta) \\
&= P(X_1 = x_1, \ldots, X_n = x_n; \theta) \\
&= \theta^{x_1}(1-\theta)^{1-x_1} \cdots \theta^{x_n}(1-\theta)^{1-x_n} \\
&= \theta^{\sum_{i=1}^n x_i}(1-\theta)^{n-\sum_{i=1}^n x_i}
\end{aligned}$$

To maximize this expression, we can take the derivative with respect to θ of $\ell(\theta) = \ln(\mathcal{L}(\theta))$, and set the result equal to 0 to solve for the MLE $\hat{\theta}$:

$$\begin{aligned}
\ell(\theta) &= \left(\sum_{i=1}^n x_i\right) \ln(\theta) + \left(n - \sum_{i=1}^n x_i\right) \ln(1-\theta) \\
\Leftrightarrow \frac{d\ell}{d\theta} &= \frac{\sum_{i=1}^n x_i}{\theta} - \frac{n - \sum_{i=1}^n x_i}{1-\theta} \\
\Leftrightarrow \quad \hat{\theta} &= \frac{\sum_{i=1}^n x_i}{n}
\end{aligned}$$

$S_{A,B}$ is simply the affine transormation $2 \cdot \hat{\theta} - 1$ to map the result to the interval $[-1, 1]$:

$$\begin{aligned}
S_{A,B} &= \frac{\sum_{i=1}^n x_i}{n} - \frac{\sum_{i=1}^n (1 - x_i)}{n} \\
&= \frac{\sum_{i=1}^n x_i}{n} - \frac{n - \sum_{i=1}^n x_i}{n} \\
&= \frac{2\sum_{i=1}^n x_i}{n} - \frac{n}{n} \\
&= 2\hat{\theta} - 1
\end{aligned}$$

Thus, $S_{A,B}$ corresponds to an affine transformation of the MLE for θ. □

Lemma 3.2. *The LOAR tally approach for combining peer advice into an estimate of predicted benefit corresponds to performing repeated Bayesian updates to obtain the posterior of a Beta distribution and then summarizing this distribution using its expected value.*

Proof. Let R be the set of ratings on a given annotation, r_i be the binary rating on the annotation from the i^{th} peer, $S_{u,i}$ be the LOAR similarity between the current user

u and the i^{th} peer, and $\hat{\theta}_i$ be the MLE estimator from Lemma 3.1. We first consider the unnormalized LOAR tally benefit computation for a user u for a given message, the numerator of which is as follows:

$$\begin{aligned}
N &= \sum_{i=1}^{|R|} \Big(r_i + r_i S_{u,i} \Big) - \sum_{i=1}^{|R|} \Big((1 - r_i) + (1 - r_i) S_{u,i} \Big) \\
&= \sum_{i=1}^{|R|} \Big(r_i + r_i S_{u,i} - 1 + r_i - S_{u,i} + r_i S_{u,i} \Big) \\
&= \sum_{i=1}^{|R|} \Big(2r_i + 2r_i S_{u,i} - S_{u,i} - 1 \Big) \\
&= \sum_{i=1}^{|R|} \Big(2r_i + 2r_i (2\hat{\theta}_i - 1) - (2\hat{\theta}_i - 1) - 1 \Big) \\
&= \sum_{i=1}^{|R|} \Big(4r_i \hat{\theta}_i - 2\hat{\theta}_i \Big) \\
&= 4 \sum_{i=1}^{|R|} r_i \hat{\theta}_i - 2 \sum_{i=1}^{|R|} \hat{\theta}_i
\end{aligned}$$

In LOAR, this expression is divided by the following quantity:

$$\begin{aligned}
D &= \sum_{i=1}^{|R|} \Big(r_i + r_i S_{u,i} \Big) + \sum_{i=1}^{|R|} \Big((1 - r_i) + (1 - r_i) S_{u,i} \Big) \\
&= \sum_{i=1}^{|R|} \Big(r_i + r_i S_{u,i} + 1 - r_i + S_{u,i} - r_i S_{u,i} \Big) \\
&= 2 \sum_{i=1}^{|R|} \hat{\theta}_i
\end{aligned}$$

The full expression for the LOAR predicted benefit is:

$$\begin{aligned}
\frac{1}{2}\left(\frac{N}{D}+1\right) &= \frac{1}{2D}\Big(N+D\Big) \\
&= \frac{1}{2D}\left(4\sum_{i=1}^{|R|} r_i\hat{\theta}_i - 2\sum_{i=1}^{|R|}\hat{\theta}_i + 2\sum_{i=1}^{|R|}\hat{\theta}_i\right) \\
&= \frac{\sum_{i=1}^{|R|} r_i\hat{\theta}_i}{\sum_{i=1}^{|R|}\hat{\theta}_i} \\
&= \frac{\sum_{j:r_j=1}\hat{\theta}_j}{\sum_{j:r_j=1}\hat{\theta}_j + \sum_{k:r_k=0}\hat{\theta}_k} \\
&= E\Big[Beta\Big(\sum_{j:r_j=1}\hat{\theta}_j, \sum_{j:r_j=0}\hat{\theta}_j\Big)\Big]
\end{aligned}$$

That is, the LOAR predicted benefit is exactly the expectation of a Beta distribution with $\alpha = \sum_{j:r_j=1}\hat{\theta}_j$ and $\beta = \sum_{j:r_j=0}\hat{\theta}_j$, corresponding to the positive and negative peer evidence, respectively. Importantly, each peer opinion has been discounted by the MLE of the probability that the user and the peer vote the same way on messages. Accordingly, the LOAR computation corresponds to summarizing a Beta distribution by its expectation after performing repeated, discounted Bayesian updates[9]. □

These results demonstrate that the methods used by LOAR to predict the benefit of a given message correspond to other well-studied approaches for trust measurement. In fact, this mechanism for deriving trust is exactly the one employed by the Beta Reputation System, with one significant difference: updates are discounted by a maximum likelihood estimator for peer similarity. Unfortunately, this pure discounting is actually not necessarily appropriate in all circumstances. For example, if peer advice about a given message is dominated by non-similar agents, LOAR will be unable to accurately assess the benefit of the message. This fact informs our next experiments.

3.3.4 Experiments with sparsity

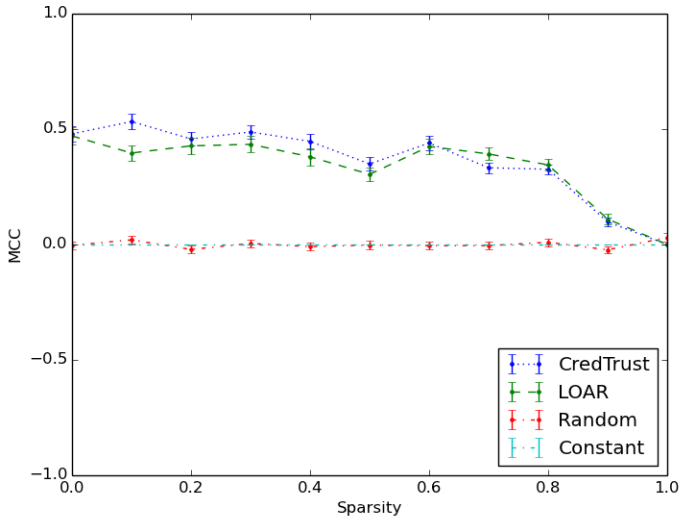
We continue our investigation of LOAR versus CredTrust by considering a new set of experiments where sparsity of ratings plays a role. The agent setup is equivalent to the

[9]Equivalently, this amounts to performing inference in a Bayesian network with a single binomial random variable that has a single Beta random variable parent.

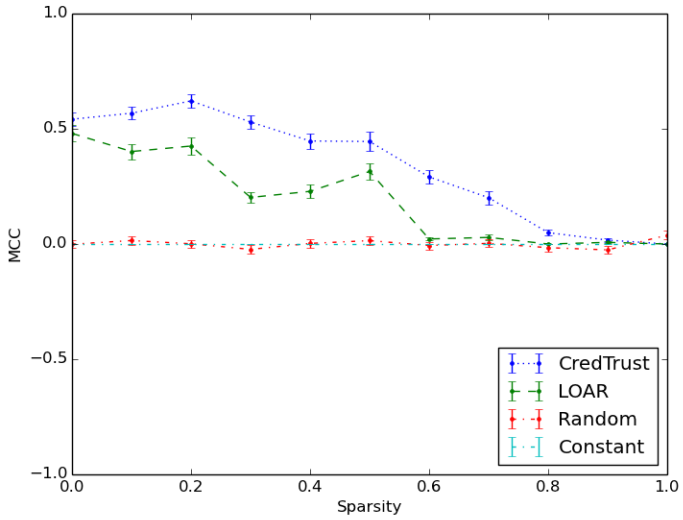
previous experiments, except now we remove the credibility metric from the simulation. In doing so, we assume that all agents have perfect credibility (this assumption has no effect on LOAR). Now, however, agents rate messages sporadically according to a sparsity parameter. When sparsity equals 0, agents rate all messages in the system; when sparsity reaches 1, agents do not rate any messages. Figures 3.2a and 3.2b depict the results of two sparsity experiments.

The first, corresponding to Figure 3.2a, investigates agents who do not discriminate when choosing which messages to rate. That is, when considering whether or not to rate a message, agents simply flip a coin with probability equal to the sparsity metric. Accordingly, agents rate both messages that they like and those that they dislike. The results illustrate that LOAR and CredTrust perform comparably when agents have no bias with respect to ratings selectivity. Both methods deteriorate as the sparsity of ratings increase, to the point where both methods are no better than random when no ratings are present. This of course makes sense, since it corresponds to an environment in which there is no training data that can be used to predict future ratings.

The second experiment, corresponding to Figure 3.2b, investigates another sparsity experiment in which agents are both sporadic and selective in their ratings. In this experiment, agents always report when they like a message, but if they dislike a message, they will flip a coin to determine whether or not to report. When sparsity is 0, agents always report negative opinions; when sparsity is 1, agents never report negative opinions. This captures the logic "if you have nothing nice to say, don't say anything at all". Moreover, it illustrates the relative performance of the algorithms when there is a selection bias in the agents' ratings. These results demonstrate where CredTrust really shines; it allows agents to reinterpret ratings from peers whose opinions are negatively correlated with their own so that useful information is still extracted. On the other hand, LOAR falls short in this environment; rather than reinterpretting ratings from peers whose opinions are negatively with their own, agents simply disregard the potentially useful advice of such peers. Thus, by linearly reversing information received from non-similar peers, CredTrust's treatment of peer advice results in a clearly superior performance compared to how LOAR treats the same advice. Moreover, CredTrust maintains an MCC of approximately 0.5, significantly above the random benchmark, until sparsity in the system reaches 50%, demonstrating robustness in the face of declining participation.



(a) MCC versus sparsity. Agents do not discriminate between rating messages they like versus ones they dislike.



(b) MCC versus sparsity. Agents tend to only rate the messages they like.

Figure 3.2: Matthew's Correlation Coefficient (MCC) performance for sparse ratings environments (higher MCC is better).

Chapter 4

POMDP Classification Model

The results of the simulations presented in Chapter 3 make clear that, while a credibility feature might be useful to consider when evaluating advisor recommendations, the CredTrust scheme does not outperform in all circumstances. Rather, it is finely-tuned for a particular distribution of agent credibilities, which makes sense because it was developed specifically to combat the potential spread of false information in a network wherein messages were falsely promoted by many non-credible peers. Likewise, LOAR encodes a particular function to account for peer similarity when soliciting advice, which works well for a particular mix of agent similarities. However, both approaches might underperform if the mix of agent types and credibilities are inappropriate for the heuristic functions each model encodes. We demonstrated this fact with LOAR in Section 3.3.3, which led us to explore sparsity as part of our simulations. As for CredTrust, while a lot of thought was given to devise a set of formulae, there is no reason to believe that those formulae will always be optimal. Accordingly, we now seek to devise a model that can learn the "correct" function of features by extracting the right amount of information from advisors (i.e., peers who provide ratings) when predicting which messages users might like to see.

We proceed by developing a model for classifying messages that uses a more principled method for considering advisory ratings when making decisions about which messages to recommend to a given user. We demonstrate its efficacy once again in simulation, under the same environment as in Chapter 3. Finally, we conclude this chapter by validating our new model using real world data from Epinions.com and Reddit.com.

4.1 POMDP Classification Model

To begin, we make an observation: all of the trust models and classifiers discussed thus far attempt to classify messages as either beneficial for a user to see or not. They accomplish this by imputing a benefit to each message by modeling the trustworthiness of advisors and their ratings on the message; those messages with the highest imputed benefits are selected to show the user. Interestingly, this entire process can be accomplished by designing a decision-making agent that can decide whether or not to recommend a message to a particular user[1]. That is, the task of classifying messages as useful to show or not can be mapped to a Partially Observable Markov Decision Process (POMDP). Formally, the POMDP is defined as follows:

- $S = \{\text{good}, \text{bad}\}$ :: each message can be either "good" or "bad"
- $A = \{\text{recommend, reject, poll}\}$ :: the agent can choose to recommend a message, to reject it, or to poll advisors
- $O = \{(rating, similarity, credibility)\}$ :: if an agent polls for advice, it receives an observation tuple consisting of the received rating, the advisor similarity (as it appears in LOAR[2]), the advisor credibility, etc.
  - In general, each observation tuple consists of features that are (hopefully strongly) correlated with the underlying message state. The similarity feature, for example, measures the degree to which a user's past ratings agree with a given advisor's past ratings in the hopes that this feature has predictive value for future ratings as well.
  - An assignment of values to an observation tuple corresponds to a row in a conditional probability table (CPT) that reflects the probability of seeing a certain trio of (rating, similarity, credibility) values given the ground truth that a message is good/bad.
- $T : P(s'|s)$ :: the state transition probabilities (an identity function, since the underlying message state does not change)

[1]Note that in this model, we delineate between a "user" in a participatory media network, for whom we are recommending messages, a "simulation agent", which is an entity constructed for the purpose of our simulations, and the "decision-making agent", which corresponds to the entity that traverses this POMDP classifier on behalf of a user. When describing the POMDP model, "agent" refers to the latter.

[2]That is, two users are similar if they rated messages similarly in the past.

- Discount factor $0 \leq \gamma < 1$
- Infinite horizon h
- $\Omega : P(o'|s', a)$ :: the probability of an observation given the state and action
- $R : A \times S \rightarrow \mathbb{R}$ :: a reward function that encodes the desirability of each state for a particular agent

This simple POMDP, which we dub POMDPTrust, describes a process in which a decision making agent participates to determine the desirability of a message. This process can be applied for each message in the network, individually, while still exploiting the overall POMDP structure. When making recommendations for a particular user, a given message can be viewed as having an underlying state that is either "good" or "bad". This state is static, albeit itself unobservable. Said another way, for a given user, a message has only one interpretation as either good or bad, but the decision-making agent cannot directly perceive the correct interpretation. Instead, the agent can choose to perform one of three actions. It can either recommend the message, reject it, or poll an advisor for advice about it. Such advice constitutes the observable states in the POMDP; the states consist of tuples that include the advisor's rating, his similarity, and his credibility. In this setting, the transition function is an identity; a message's underlying suitability is static and does not evolve over time[3].

The benefits of using a POMDP to help classify messages are manifold. First, this model intrinsically encodes the uncertainty in our knowledge of the world and in particular, the desirability of each message for each user. It allows us to express this uncertainty using probability distributions over our beliefs about the state of the the world. Moreover, we can using reinforcement learning techniques to learn the correct distributions over advisory ratings, which allows us to correctly evaluate evidence that we receive from advisors in a principled manner. Furthermore, this formulation of the problem remains tractable, since the space over states, actions, and observations is small and the process is Markovian. Lastly, we can extend this model to specifically account for individual user utilities by modifying the reward function.

As a final consideration, note that the reward function encodes the relative utilities for each action. We assume that the reward for polling should be equivalent in all states, reflecting the fact that the "price" of obtaining more information from advisors is the same regardless of the agent's belief about the underlying message state. We point out

[3]Note, however, that a given message's underlying state can be different for different users, which reflects the fact that different users might derive different utility by experiencing the same message.

that the higher the polling reward, the stronger the belief required in order to make a recommendation/rejection decision (as polling for more advice becomes more attractive in expectation than making a decision). In fact, if the reward for polling is greater than the expected rewards from making a recommendation/rejection decision, then it always becomes more beneficial to poll for advice if possible (i.e., to follow a "greedy" policy). Note as well that we discuss some strategies for learning an appropriate reward function in Section 6.2.2.

4.1.1 POMDP Example

We now illustrate how the POMDP model can be used to classify two messages, m_1 and m_2, for a given user u. For the sake of this example, we make the following assumptions about the POMDP model and about the messages:

- The underlying state for message m_1 is "good", denoted G (m_1 should be recommended).
- The underlying state for message m_2 is "bad", denoted B (m_2 should be rejected).
- The observation function for the poll action is encoded in Table 4.1 (other actions do not elicit any peer advice and so have zero probability).
  - R refers to the rating (either 1 or 0).
  - M refers to similarity (either similar, 1, or not, 0).
  - C refers to credibility (either credible, 1, or not, 0).
  - For example, looking at the first row of Table 4.1, the probability of seeing the trio of (R, M, C) values $(1, 1, 1)$ given that the message is *Good* (*Bad*) is 0.3 (0.025). (Note that the probabilities are entirely fabricated for the sake of this example to demonstrate the belief update process. For a discussion as to how to learn a good observation function, see Section 4.1.2).
- Table 4.2 encodes the user's reward function (table cells are utilities, derived arbitrarily for the purpose of these examples).
- Peer ratings are depicted in Table 4.3
  - Ratings are either positive (1) or negative (0)
  - Peers have a similarity $S \in \{0, 1\}$ to the current user

Table 4.1: Observation function conditional probability table $Pr(O \mid S, A = \text{poll})$ (values are in $[0, 1]$).

Observation			State	
R	M	C	Good	Bad
1	1	1	0.3	0.025
1	1	0	0.15	0.1
1	0	1	0.025	0.3
1	0	0	0.05	0.05
0	1	1	0.025	0.3
0	1	0	0.1	0.15
0	0	1	0.3	0.025
0	0	0	0.05	0.05

Table 4.2: User reward function $R(s, a)$

Action	State	
	Good	Bad
Poll	1.5	1.5
Recommend	1	-1
Reject	-1	1

Table 4.3: Advisor ratings (observations) for messages m_1 and m_2

Observation	Peer				
	p_1	p_2	p_3	p_4	p_5
Rating m_1	1	0	0	1	1
Rating m_2	1	0	1	0	0
Similarity	1	1	0	0	1
Credibility	1	0	1	0	0

– Peers have a credibility $C \in \{0, 1\}$

For both m_1 and m_2, the agent starts with belief $b^{(m_i)}(s_0) = 0.5$, viz., that the message is equally likely to be good or bad. In other words, the agent has no *a priori* knowledge about the message and therefore begins with a uniform belief[4]. From this starting belief state, the agent makes observations about the underlying message by polling for peer advice. In this example, we do not demonstrate how the polling decision might be made; we simply assume that the agent follows a policy to always poll if possible (indeed, this strategy is optimal given that polling offers a higher reward than either recommending or rejecting a message, as can be seen in Table 4.2). The agent can update its belief about the true message state through repeated updates:

$$b_{t+1}(s_{t+1}) \propto Pr(o_{t+1} \mid s_{t+1}, a_t) \sum_{s_t} Pr(s_{t+1} \mid s_t, a_t) b_t(s_t) \tag{4.1}$$

For example, here we demonstrate the belief update for m_1 after polling for advice from p_1 and receiving the observation tuple $\langle R, M, C \rangle = \langle 1, 1, 1 \rangle$ (viz., positive feedback about the message from a peer who is both similar and credible):

$$\begin{aligned} b_1(G) &\propto Pr(\langle 1, 1, 1 \rangle \mid G) \sum_{s_0 \in \{G,B\}} Pr(G \mid s_0, a_0) b_0(s_0) \\ &= 0.3 \cdot (1.0 \cdot 0.5 + 0.0 \cdot 0.5) \\ &= 0.15 \end{aligned}$$

$$\begin{aligned} b_1(B) &\propto Pr(\langle 1, 1, 1 \rangle \mid B) \sum_{s_0 \in \{G,B\}} Pr(B \mid s_0, a_0) b_0(s_0) \\ &= 0.025 \cdot (0.0 \cdot 0.5 + 1.0 \cdot 0.5) \\ &= 0.0125 \end{aligned}$$

After normalizing[5], we arrive at the belief $b_1 = [0.92 \quad 0.08]$, namely that the message is "good" with probability 0.92. Following this same belief update procedure, Figure

[4]In a robust implementation of this model, a domain expert could conceivably augment the starting belief with suitable prior knowledge. For example, perhaps past experience with the author of a given message can inform the starting belief about the suitability of the given message. LOAR does this to some extent by blending together the "global" author reputation with "local" annotation ratings information.

[5]Note that in the observation CPT, the individual columns for states *Good* and *Bad* each sum to 1, but the rows do not sum to 1. Moreover, after applying a belief update as above, the new beliefs do not necessarily sum to 1. Since a belief represents the probability of each respective state, we need to then normalize the values.

4.1 illustrates the agent's belief evolution as it polls for advice from peers $p_1, \ldots, p_5$. In particular, we can see that after polling for advice about each message, the agent's belief about m_1 suggests that the message is "good", while the agent's belief about m_2 suggests that the message is "bad".

Note that after belief updates are complete, decision-making agents have to advise their users. Taking m_1 as an example (Figure 4.1a), the agent would perform the following expected utility calculations:

$$\begin{aligned} EU_{recommend} &= b_5(G) \cdot R_{recommend}(G) + b_5(B) \cdot R_{recommend}(B) \\ &= 0.99 \cdot 1 + 0.01 \cdot (-1) \\ &= 0.98 \end{aligned}$$

$$\begin{aligned} EU_{reject} &= b_5(G) \cdot R_{reject}(G) + b_5(B) \cdot R_{reject}(B) \\ &= 0.99 \cdot (-1) + 0.01 \cdot (1) \\ &= -0.98 \end{aligned}$$

Since the expected utility for recommending the message (0.98) is greater than that for rejecting the message (-0.98), the agent would recommend m_1 to the user.

4.1.2 Learning the Observation Function

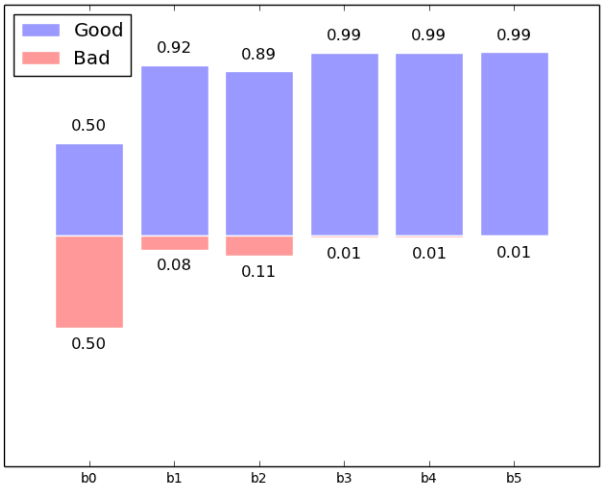
We now examine the treatment of observations under this model. The observation kernel, Ω, encodes the function $Pr(O_{t+1} = o \mid S_{t+1} = s, A_t = a)$. We can denote this function by a matrix of $|A| \times |S|$ multinomial distributions:

$$\vec{\theta} = \begin{bmatrix} \theta_{s_1,a_1} & \cdots & \theta_{s_1,a_{|A|}} \\ \vdots & \ddots & \vdots \\ \theta_{s_{|S|},a_1} & \cdots & \theta_{s_{|S|},a_{|A|}} \end{bmatrix}$$

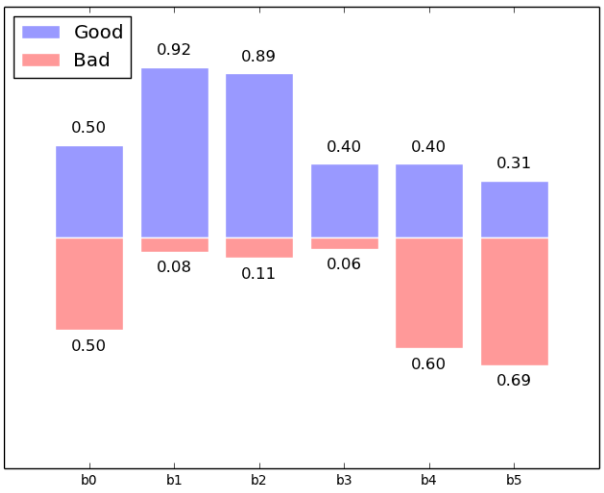
where each distribution contains $|O|$ parameters:

$$Pr(O_{t+1} = o \mid S_{t+1} = s_i, A_t = a_j) \sim \theta_{s_i,a_j} = Mult(m_1, m_2, \ldots, m_{|O|})$$

Since the POMDPTrust model only produces observations when the agent chooses to poll for advice, we only need to encode two multinomial distributions, viz., $\vec{\theta} = [\theta_{\text{good}} \quad \theta_{\text{bad}}]$. In



(a) Belief evolution for m_1: belief converges on S = good so that recommending the message results in a maximum expected utility.



(b) Belief evolution for m_2: belief tends towards S = bad so that rejecting the message results in a maximum expected utility.

Figure 4.1: POMDP model example: belief state tracking.

order to traverse the POMDP, we need to know each of the distributions θ_i. Alternatively, we can allow each of the parameters to be distributed according to a Dirichlet distribution:

$$\theta_i \sim Dir(\alpha_1, \ldots, \alpha_{|O|})$$

Then, we can refine each of these distributions as the agent traverses the POMDP and retrieves evidence from the user. That is, whenever a user provides feedback about a message as to the true state of the message (for example, by rating the message), we can perform a Bayesian update to determine the posterior distribution:

$$Pr(\theta_i \mid d) = \frac{Pr(d \mid \theta_i) Pr(\theta_i)}{Pr(d)}$$

We can perform an update for each peer rating on a message when the current user informs the agent about the true state of the message.

4.1.3 Policy Evaluation

Since the similarity and credibility feature variables, denoted M and C, are continuous in the interval $[0, 1]$, it is non-obvious how such metrics should index the observation function. One technique that can be used to incorporate these metrics is to consider them as parameters to binomial distributions, viz., θ_M and θ_C. Then, whenever the agent solicits advice, it simply samples from the similarity and credibility distributions, respectively, to obtain a discrete, binary observation. For example, if similarity were 0.7 and credibility were 0.9, then this technique would entail observing a similarity of 1 with probability 0.7 and a credibility 1 with probability 0.9. This results in a single, discrete observation. Observe that the mix of sampled discrete observations converge to the parameter from which those observations were sampled in the limit (i.e., as the number of advisors with similarity s and credibility c goes to infinity, a consequence of the central limit theorem), so that this sampling technique performs the right combination of belief updates in the limit.

Since ratings in real world datasets are often sparse (see Section 4.2), another technique

Table 4.4: Separate, discrete belief update results

Observation			State	
R	M	C	$b_{t+1}(good)$	$b_{t+1}(bad)$
1	1	1	0.923	0.077
1	1	0	0.6	0.4
1	0	1	0.077	0.926
1	0	0	0.5	0.5

would be to blend different belief states together by extending Equation 4.1 as follows[6]:

$$b_{t+1}(s_{t+1}) \propto Pr(o_{t+1} \mid s_{t+1}, a_t) \sum_{s_t} Pr(s_{t+1} \mid s_t, a_t) b_t(s_t) \tag{4.2}$$

$$= Pr(o_{t+1} \mid s_{t+1}, a_t) \cdot \kappa \tag{4.3}$$

$$= \sum_{m_{t+1}} \sum_{c_{t+1}} Pr(r_{t+1} \wedge m_{t+1} \wedge c_{t+1} \mid s_{t+1}, a_t) \cdot \kappa \tag{4.4}$$

$$= \sum_{m_{t+1}} \sum_{c_{t+1}} Pr(m_{t+1} \wedge c_{t+1} \mid s_{t+1}, a_t) Pr(r_{t+1} \mid m_{t+1}, c_{t+1}, s_{t+1}, a_t) \cdot \kappa \tag{4.5}$$

$$= \sum_{m_{t+1}} \sum_{c_{t+1}} \theta_{m_{t+1}} \cdot \theta_{c_{t+1}} Pr(r_{t+1} \mid m_{t+1}, c_{t+1}, s_{t+1}, a_t) \cdot \kappa \tag{4.6}$$

Here, 4.3 introduces a constant κ to represent the summation in 4.2 for notational simplicity; 4.4 encodes the fact that the observation function is a joint probability distribution over the elements of the observation tuple, and hence we can marginalize out the unobserved variables M_{t+1} and C_{t+1}; 4.5 follows from the chain rule of probability, and; 4.6 encodes the heuristic in our mechanism, viz., that the similarity and credibility random variables are independent.

To make this more concrete, consider the following small example. Suppose that the observation function is once again given by Table 4.1 and that we receive a positive rating about a message ($r_{t+1} = 1$) from a peer for whom we've calculated a similarity score of 0.7 and a credibility score of 0.9. The belief updates for the different discrete assignments to observation variables are exhibited in Table 4.4. To arrive at a final belief, we combine the

[6]Note that by using this blending technique, Monte-Carlo Tree Search to derive a policy becomes more difficult, since the observation corresponds to a real number. One way to get around this would be to discretize the observation interval.

beliefs according to the weights given by the similarity and credibility parameters (shown in parentheses)[7]:

$$\begin{aligned} b_{t+1}(G) &= (0.7 \cdot 0.9) \cdot 0.923 + (0.7 \cdot 0.1) \cdot 0.6 + (0.3 \cdot 0.9) \cdot 0.077 + (0.3 \cdot 0.1) \cdot 0.5 \\ &= 0.664 \end{aligned}$$

$$\begin{aligned} b_{t+1}(B) &= (0.7 \cdot 0.9) \cdot 0.077 + (0.7 \cdot 0.1) \cdot 0.4 + (0.3 \cdot 0.9) \cdot 0.923 + (0.3 \cdot 0.1) \cdot 0.5 \\ &= 0.336 \end{aligned}$$

4.1.4 Simulation Results

We perform the same simulations[8] as in Chapter 3, this time observing the performance of the POMDP model in an environment with 20 users. We assume that credibility values are provided by an oracle, similarities are computed on the basis of common ratings, and ratings are simulated by virtue of agent types. We run experiments wherein we vary the user credibilities or the sparsity of ratings.

For our experiments, we instantiate POMPDTrust with a single, global reward function for all agents. This reward function matches the one from Table 4.2, i.e., one where recommending good messages yields a reward of 1, while recommending a bad message yields a reward of -1.

Beyond this, we use a simple greedy policy that polls for as much advice as possible[9], and then takes the action that maximizes the expected utility for agents based on the POMDP belief state. This is roughly the same policy that is found via Monte Carlo Tree Search (MCTS), depending on the relative utilities between polling and recommending/rejecting messages, except it is also significantly faster in simulation due to the small ratings set per message (namely, at most 20, since there are 20 agents in our simulations, as is explained in Section 3.3.1 and reviewed in this section). We postulate that MCTS would be beneficial when the number of ratings on a given message is very large (in which case the MCTS might discover the better course of action to make an early decision rather than considering all ratings attached to the message, and thus could prove faster). While explicit ratings tend to be rather sparse (so that dealing with a large rating set is typically not an issue),

[7] These weights/parameters were chosen arbitrarily for the sake of this example.

[8] Please refer to Section 3.3.1 for a more detailed discussion of the simulation setup.

[9] This is equivalent to setting the reward for polling to be higher than the reward for recommending/rejecting a message.

considering implicit ratings makes the data much more dense, and so MCTS could prove useful under such conditions. (See Section 6.2.1 for a discussion of implicit ratings).

The results of our simulations are depicted in Figures 4.2a, 4.2b, 4.3a, and 4.3b. Once again, we split the simulation into a training phase and a testing phase. During the training phase, we use the training messages as evidence to learn the POMDP observation function by performing repeated Bayesian updates (illustrated in more detail in Appendix D). We also use the training data to compute a similarity metric (for all three models) and assume once again that the underlying credibilities of the agents are known.

Overall, the new POMDP model tends to outperform in all of the experiments, which makes sense given that it uses Bayesian updates to extract exactly the right amount of information from a given set of peer advice. The POMDP model only underperforms CredTrust in one instance: the lower spectrum in Figure 4.2a.

We also compare the performance of POMDPTrust versus a latent factor model using stochastic gradient descent (see Section 2.2.2), which we dub the "SGD" model[10], by repeating the same four simulations. In order to perform a fair comparison, we first tuned the SGD model by running a separate set of simulations to determine which combination of SGD parameters yielded the best results. We varied the number of latent features from 1 to 20 and experimented with regularization parameters from the set $\{0.1, 0.3, 1, 3, 10\}$. Our experiments revealed that a regularization parameter $\lambda = 3$ yielded the best results by sufficiently minimizing residuals while avoiding overfitting the training data. Furthermore, the number of features beyond $f = 7$ had little impact on the overall performance of the SGD model. Accordingly, we chose parameters $f = 10$ and $\lambda = 3$ as the parameters that yielded the best results for the SGD model.

For SGD, the training data is used to optimize the feature vectors for both items and users. In particular, the training data is used to compute the cost function described in Equation 2.10 and gradient; these are used to solve an optimization problem that finds cost-minimizing values for the feature vectors.

The results of these runs are exhibited in Figures 4.4a, 4.4b, 4.5a, and 4.5b, which depict only the SGD curves and POMDPTrust curves. These results are slightly more mixed. In Figure 4.4b, the two algorithms trace a similar performance curve until credibility in the system reaches 0.5, after which the SGD algorithm significantly outperforms the POMDP model. Figure 4.4b tells a slightly different story; in the dichotomous case, SGD

[10]Strictly speaking, our SGD model implementation does not perform *stochastic* gradient descent, as the cost function is evaluated in "batch" fashion (wherein the cost contribution of all the training examples is calculated before modifying the factor parameters on each iteration of gradient descent); however, this distinction is pedantic and we hereafter refer to the latent factor model as "SGD".

outperforms until the percentage of non-credible peers reaches 50%, after which point the POMDP model outperforms. The sparsity simulations depict a smaller difference in performance between the two approaches, however the story remains similar in that SGD outperforms in lower sparsity settings while the POMDPTrust model modestly outperforms in the higher sparsity settings.

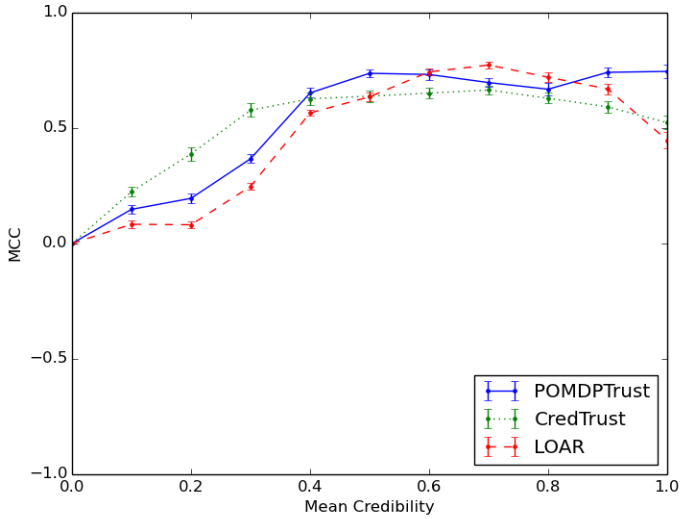
We save a more detailed discussion of these and the previous results for Chapter 5, after exploring performance on two real world datasets in the next section.

4.2 Real World Experiments

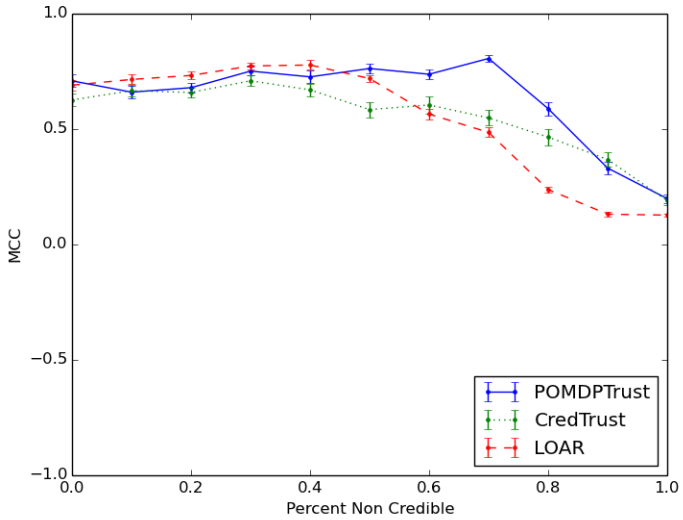
Having demonstrated the POMDP model's effectiveness in simulation against LOAR, CredTrust, and to a lesser extent, the SGD model, we now turn our attention to validation using real-world data available from Reddit.com and Epinions.com. While we believe the simulations presented in Section 4.1.4 were an important first step in examining our POMDP model, it is also important to consider models against real-world data, which have their own unique characteristics (for example, with respect to the distribution of ratings, discussed below). Reddit.com is a website that allows users to create and post messages for other users to see; other users in the Reddit community can then choose to "upvote" the message (i.e., rate it positively) or "downvote" the message (rate it negatively). Messages are then more or less likely to be seen by other users based on the mix of positive and negative feedback the messages have received. Epinions.com is a website that allows users to submit reviews about real-world products they have used. Other users can then read and rate the reviews (not the products themselves) on a 5-point scale (1–5 stars), as well as provide concrete, written feedback. Data from these websites has been made available for research purposes, and given the nature of the services they provide, the data seems appropriate for the problem our research seeks to address (more discussion as to the suitability of these data for our experiments is provided in Appendix C). Accordingly, through these real-world experiments, we hope to demonstrate that the POMDP model is an effective model for use in classifying messages as either beneficial or not for users given a realistic distribution of users and their ratings.

The Reddit.com data contains a total of approximately 7.4 million user ratings by approximately 31 thousand different users on approximately 2.0 million messages[11]. Of these ratings, approximately 76% were positive. Table 4.5 provides a more detailed summary

[11]The raw, anonymized Reddit data can be retrieved from http://www.reddit.com/r/redditdev/comments/dtg4j/want_to_help_reddit_build_a_recommender_a_public/

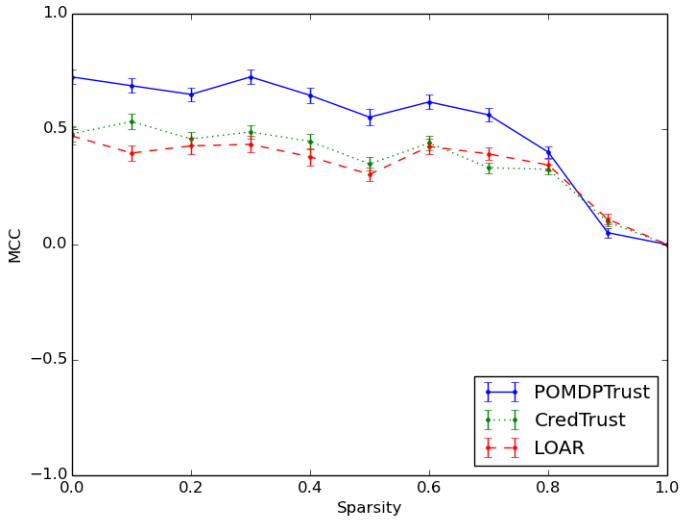


(a) MCC versus mean credibility for randomly generated agent credibilities.

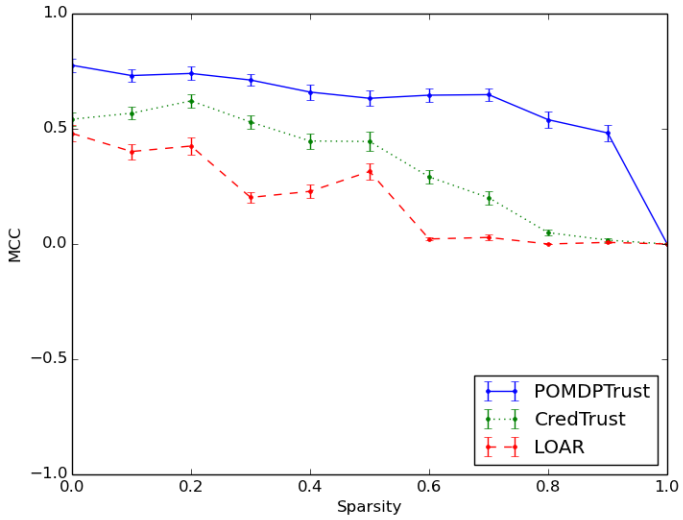


(b) MCC versus percentage of non-credible advisors for dichotonomous agent credibilities.

Figure 4.2: Matthew's Correlation Coefficient (MCC) performance for various agent environments (higher MCC is better).

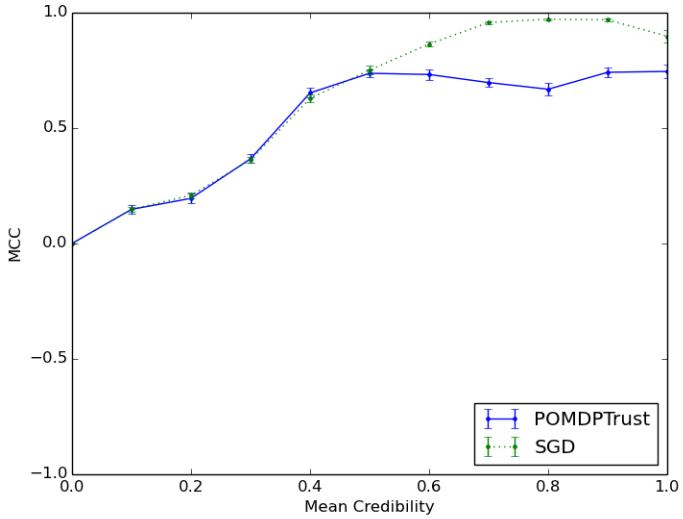


(a) MCC versus sparsity. Agents do not discriminate between rating messages they like versus ones they dislike.

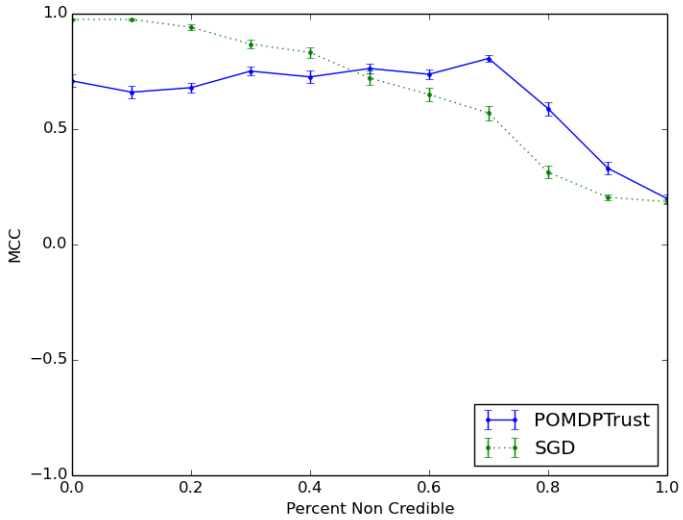


(b) MCC versus sparsity. Agents tend to only rate the messages they like.

Figure 4.3: Matthew's Correlation Coefficient (MCC) performance for sparse ratings environments (higher MCC is better).

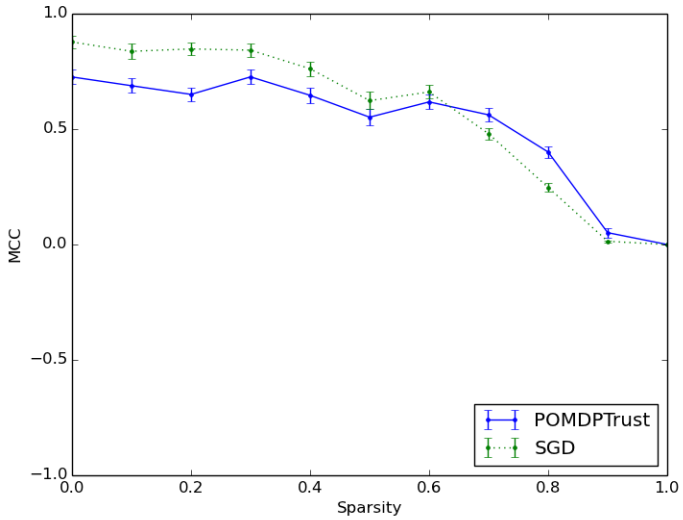


(a) MCC versus mean credibility for randomly generated agent credibilities.

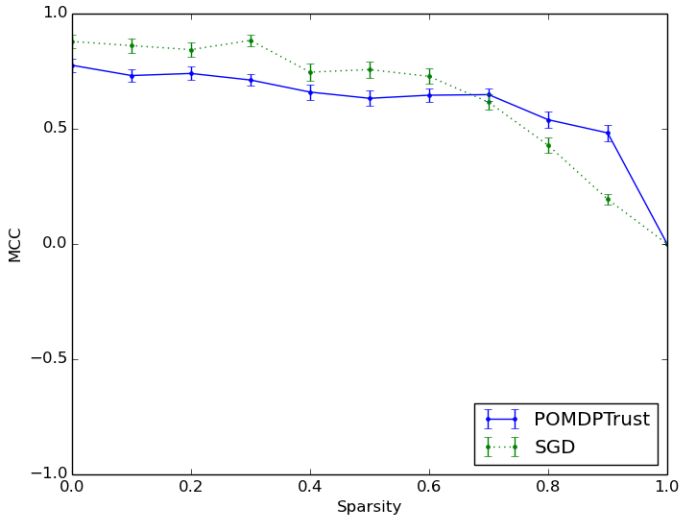


(b) MCC versus percentage of non-credible advisors for dichotonomous agent credibilities.

Figure 4.4: Matthew's Correlation Coefficient (MCC) performance for various agent environments (higher MCC is better).



(a) MCC versus sparsity. Agents do not discriminate between rating messages they like versus ones they dislike.



(b) MCC versus sparsity. Agents tend to only rate the messages they like.

Figure 4.5: Matthew's Correlation Coefficient (MCC) performance for sparse ratings environments (higher MCC is better).

Table 4.5: Reddit.com dataset descriptive statistics

	User Ratings	Common Ratings	Advisors
count	31553	31553	31553
mean	234.66	2.3269	2315.99
std	446.64	4.1893	3285.83
min	1	0.0000	0
Q_1	3	0.0000	0
Q_2	20	1.1176	320
Q_3	194	2.5579	4058
max	2000	188.00	13322

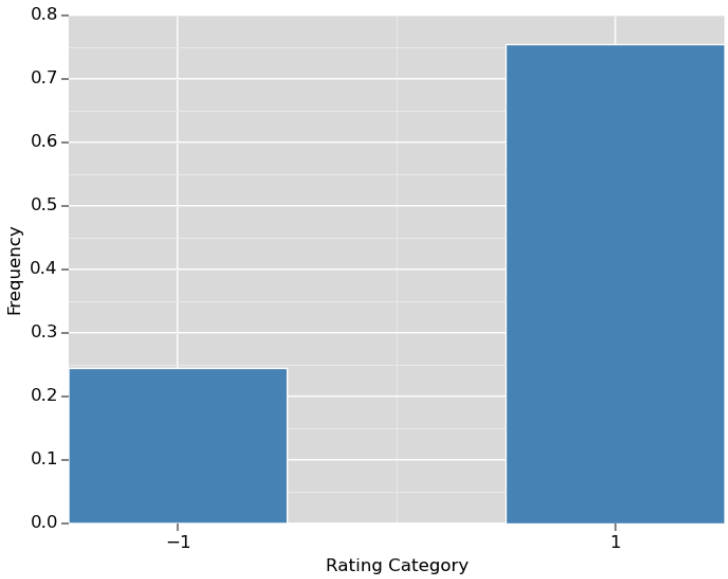
of the dataset by reporting the means, standard deviations, and quartiles for various features. There were 31,553 users connected with this dataset; the first row, "count", shows the number of users over which the remaining statistics were calculated. "User Ratings" in this table summarizes the number of ratings left by each user in the dataset. "Common Ratings" summarizes the average number of ratings each user had in common with each other user. "Advisors" summarizes the average number of ratings that were left on messages rated by a given user[12]. Q_1, Q_2, Q_3 means that $25\%, 50\%, 75\%$ of the users had the given number or lower.

Figure 4.6a shows the distribution of user ratings. Figure 4.7a shows the distribution of the number of ratings per user; this graph makes clear that most users rate a small number of messages. Figure 4.8a shows the distribution of average commonly rated messages between users, and again, users tend to rate a small number common ratings (perhaps because users tend to rate a small number of messages at all). Lastly, Figure 4.9a depicts the average number of advisors per user for each message. A sample of messages from this data set can be viewed in Appendix C.

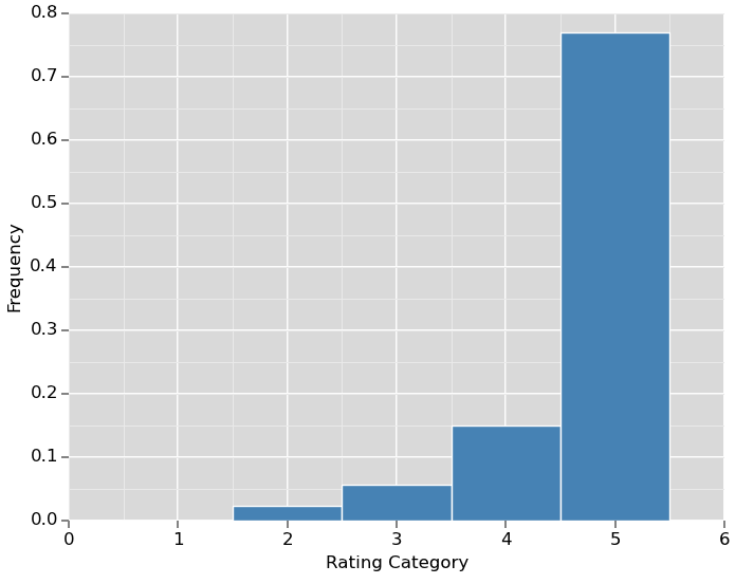
The Epinions.com data contains a total of approximately 13.7 million user ratings by 132 thousand users on approximately 1.56 million different messages[13]. As before, Table 4.6 provides a more detailed summary of the dataset, and Figure 4.6b shows the distribution

[12]To draw a distinction between "Common Ratings" and "Advisors", consider the case where a user A has rated only 1 message. It may be the case that all other users also rated that message, in which case the user has a large number of advisors (for that message). However, since A only rated 1 message, it can only have on average 1 common rating with each other user.

[13]The raw, anonymized Epinions data can be retrieved from http://www.trustlet.org/wiki/Extended_Epinions_dataset, and was generously donated to the scientific community for research purposes as a result of the work of Paolo Massa [32].

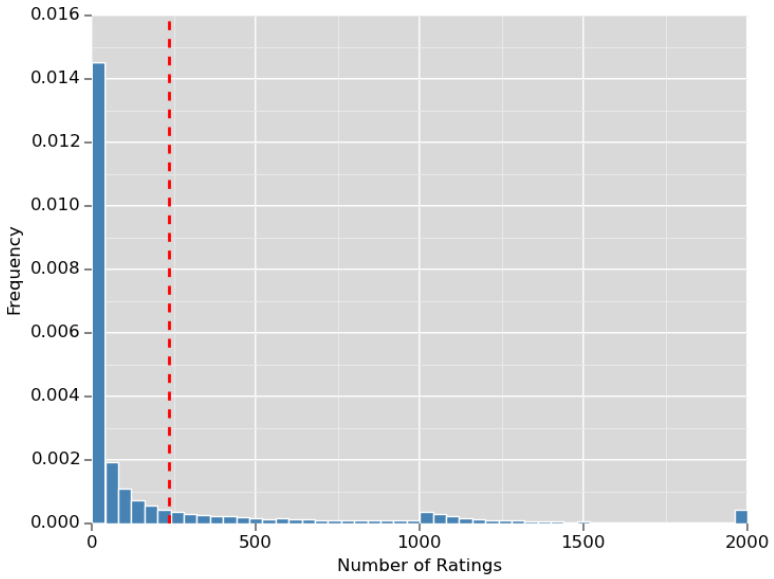


(a) Reddit.com dataset ratings distribution.

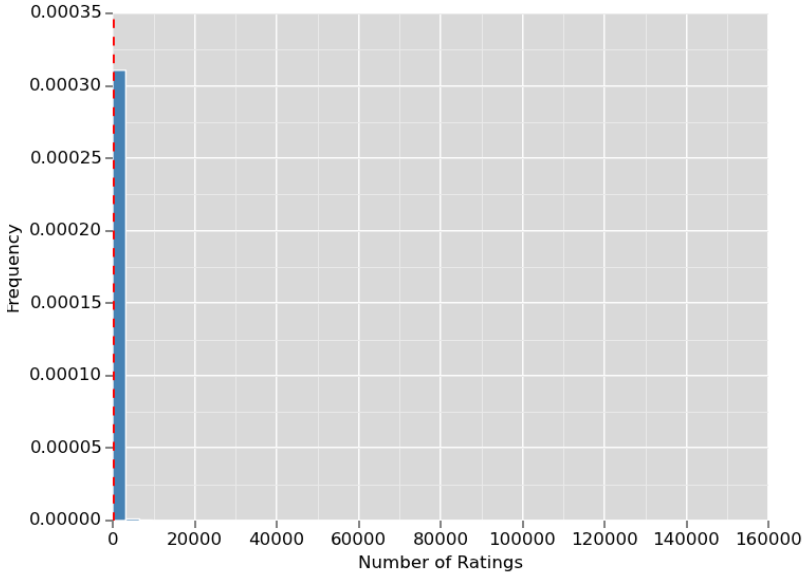


(b) Epinions.com dataset ratings distribution.

Figure 4.6: Ratings distributions for real-world datasets

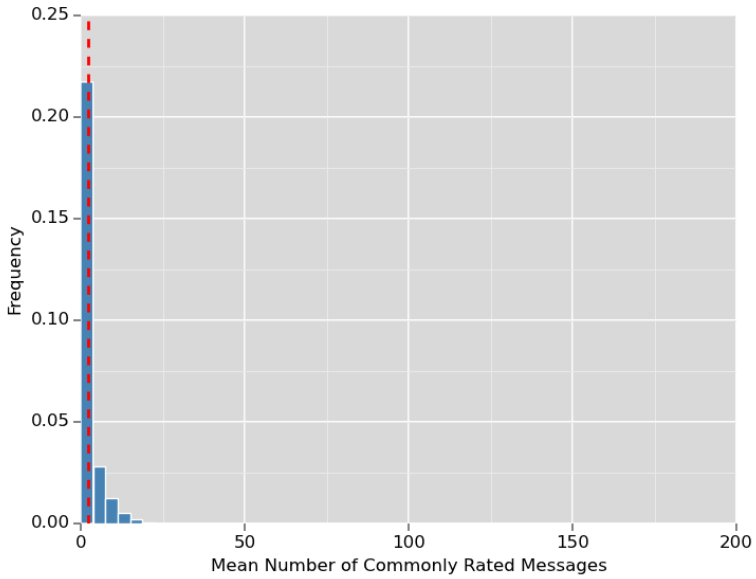


(a) Reddit.com dataset: number of ratings per user.

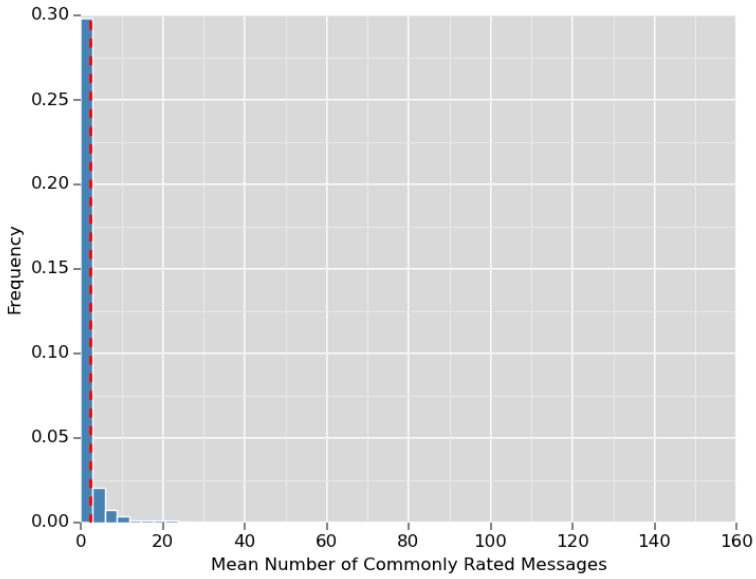


(b) Epinions.com dataset: number of ratings per user.

Figure 4.7: Number of ratings per user for real-world datasets

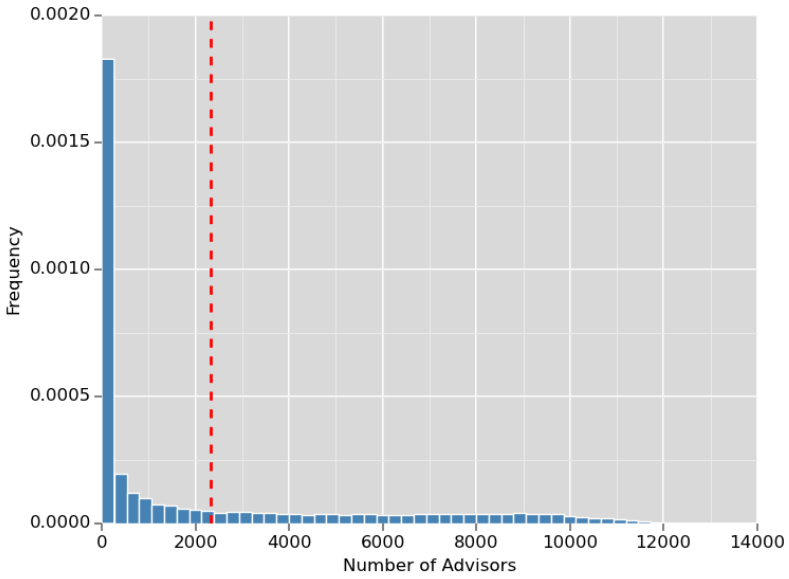


(a) Reddit.com dataset: average number of commonly rated messages per user.

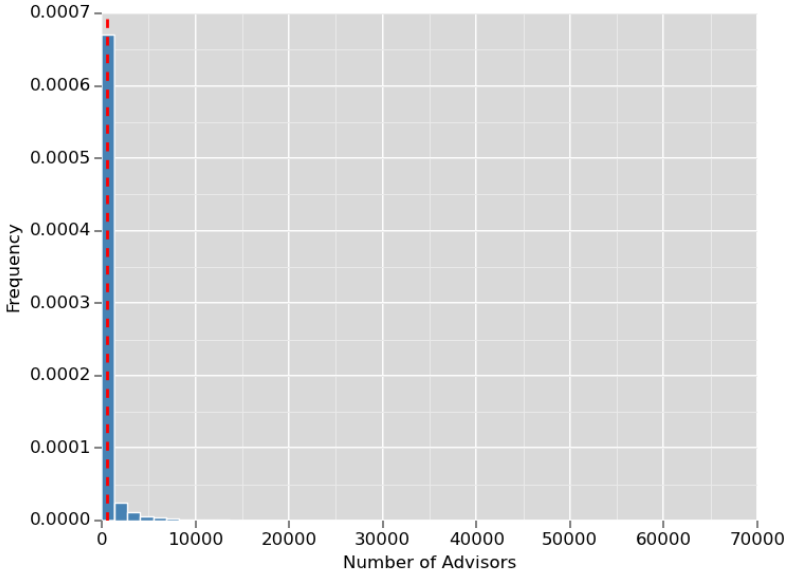


(b) Epinions.com dataset: average number of commonly rated messages per user.

Figure 4.8: Average number of commonly rated messages per user for real-world datasets



(a) Reddit.com dataset: number of advisors per user.



(b) Epinions.com dataset: number of advisors per user.

Figure 4.9: Number of advisors per user for real-world datasets

Table 4.6: Epinions.com dataset descriptive statistics

	User Ratings	Common Ratings	Advisors
count	119,901	119,901	119,901
mean	112.571	2.27976	538.931
std	1052.70	5.92368	1794.91
min	1	0.0000	0
Q_1	1	1.0000	19
Q_2	3	1.0471	67
Q_3	12	1.5192	258
max	159,607	148.14	68,563

of user ratings. Figure 4.7b shows the distribution of the number of ratings per user; this graph makes clear that most users rate a small number of messages. Figure 4.8b shows the distribution of average commonly rated messages between users, and again, users tend to rate a small number common ratings (perhaps because users tend to rate a small number of messages at all). Lastly, Figure 4.9b depicts the average number of advisors per user for each message. Note that these reports do not include the users who left no ratings (so that the minimum number of ratings is 1 as opposed to 0). A sample of messages from this data set can be viewed in Appendix C.

4.2.1 Experimental Design

In these experiments, we compare three approaches for classifying messages: LOAR, BLADE, and our POMDP model. We show the LOAR results as a baseline estimate of classification accuracy, as LOAR has been discussed throughout this thesis and exhibited in our simulations as well. We chose BLADE as another comparator because it uses Bayesian techniques to evaluate peer trustworthiness (see Section 2.2.1). Accordingly, it has several similarities to our own POMDP approach (discussed further in Chapter 5). We do not include CredTrust in these experiments because we view our POMDP approach as an improved version of CredTrust (and indeed, have demonstrated in Section 4.1.4 that the POMDP approach outperforms CredTrust in all of our simulations). Furthermore, since the real-world data does not contain explicit credibility metrics (which we assumed were made available through an oracle in simulation), CredTrust cannot be validated against these data.

In order to evaluate the effectiveness of these models against the Reddit.com and Epin-

ions.com datasets, we use repeated random sub-sampling validation. In particular, we run 10 trials for each algorithm, and on each trial, we randomly partition the data into a training and testing set consisting of 70% and 30% of the ratings, respectively. The training data is used for the purpose of determining the similarities between peers (for both LOAR and POMPTrust) and updating the BayesNet Dirichlet priors in BLADE. The testing set is withheld from the training phase and then used to evaluate the efficacy of each model. In particular, after training the models, each algorithm (i.e., LOAR, BLADE, and POMDP-Trust) develops predictions for the ratings that users give to messages in the testing set. The actual ratings that users give to messages in the testing set is known and therefore considered to be the ground truth message benefit.

We run the LOAR tally algorithm against both datasets. However, LOAR was only developed to work with binary data. Accordingly, for the Epinions dataset, we substitute the LOAR similarity metric with a standard cosine similarity metric and accumulate evidence in five separate buckets (representing one through five star ratings). Thus, whereas we demonstrated that the standard LOAR tally approach essentially uses a Beta distribution to amalgamate peer advice, we extend LOAR to use a Dirichlet distribution for the Epinions data.

We also run the BLADE model against both datasets. Since information about message authors is unavailable for the Reddit dataset, we assume that each message is submitted by an anonymous author. The result is that users are unable to develop strong posterior beliefs about the message authors, since at most each user interacts only once with a given anonymous author (viz., either the user rates a message in the training set or not). In the original BLADE model, it is important for buyers to develop strong posterior beliefs about sellers so that the buyers can more accurately learn advisor evaluation functions. Projected into our context, this requires us to model beliefs about authors. However, in this setting, since there is at most a single data point with which to arrive at a posterior belief about a given anonymous author, the hope is that there is sufficient additional training data (i.e., additional common ratings) to improve estimates of advisor evaluation functions.

For the Epinions dataset, we hybridized the BLADE model. If the message author was known, then BLADE was set up to learn the author's features; if the message author was unknown, BLADE proceeded as in the Reddit case and assumed that the message was authored anonymously. In this case, BLADE also made use of posterior belief about authors when reasoning about the probability of author features for messages in the testing set. (That is, if a message in the testing set was known to have been authored by a particular user, then that user's posterior feature distribution was used as a prior in addition to the explicit advisor ratings on the message for the purpose of classifying the message).

4.2.2 Results

Tables 4.7 and 4.8 depict the confusion matrices[14] for each algorithm for the Reddit and Epinions experiments, respectively. The rows of each table show what each algorithm predicted for messages in the testing set, while the columns show the ground truth message class (with $1,679,101$ ($542,177$) messages for which the ground truth was positive (negative)). For example, in Table 4.7, the first row under the LOAR heading reports the number of messages that LOAR classified as positive; the first column depicts the number of messages that were actually positively rated, while the second column reports the number of messages that were actually negatively rated. Using LOAR as an example, there were $1,599,762$ true positives, $414,036$ false positives, $79,339$ false negatives, and $128,141$ true negatives. The reported numbers are averaged over all trials, with standard deviations reported in parentheses. For Reddit, we also compute the Matthew's Correlation Coefficient (MCC) and exhibit the results in Figure 4.10a. For Epinions, we compute the Mean Absolute Error (MAE); these results are depicted in Figure 4.10b.

For the Reddit experiment, we report the results of two separate POMDPTrust instantiations, corresponding to two separate global reward functions. The first ("POMDP") rewards true positives slightly more than true negatives to reflect the positive skew of the data set. The second ("POMDP2") rewards true positives and true negatives equally. We begin by reporting the positive/negative hit rates:

- LOAR achieves accuracies of 95% and 24%, respectively
- BLADE achieves 88% and 46%
- POMDPTrust achieves 81% and 56% for the first instantiation, versus 63% and 79% for the second instantiation.

In both instantiations, we see that POMDPTrust achieves a significantly higher true negative rate at the expense of a lower true positive rate versus both LOAR and BLADE. Accordingly, based on hit rates alone, it is difficult to distinguish which algorithms perform better[15]. As a result, and also due to the skewed nature of the data, the Matthew's Correlation Coefficient is another important metric to report since it balances the true

[14] A confusion matrix, or classification table, depicts the performance of prediction algorithms in terms of true/false positives and true/false negatives (for the binary case). In the general case, it depicts the hit rates for each given prediction class [26] (see Section 14.5).

[15] In many scenarios, correctly rejecting bad messages (i.e., a higher true negative rate) is a more important consideration than ensuring that all good messages are shown (i.e., a higher true positive rate).

positive and true negative rates to arrive at an overall accuracy that corrects in some sense for class skewness. For this data, the MCC is important to examine because a scheme that simply recommends all messages will achieve true positive rate of 100% and an overall accuracy of 76% (since 76% of the messages were rated positively in the original dataset). Figure 4.10a illustrates how the first POMDP instantiation has a 23% higher MCC versus LOAR and a 6% lower MCC versus BLADE. On the other hand, the second instantiation outperforms LOAR by 31%, and achieves a not significantly different MCC versus BLADE. Overall, POMDPTrust performs very well against the Reddit data, despite using a single, global reward function (as opposed to tailoring a reward function to each individual user).

For the Epinions experiment, we extend the two-state POMDPTrust to be able to classify messages on a scale of 1 through 5 stars. (This is done very simply by extending the number of states to 5 and increasing the transition and observation functions accordingly). We only run a single instantiation of POMDPTrust with a specific global reward function for all agents that accounts for the skewness of the Epinions data (e.g., by emphasizing a higher reward for classifying messages as "5 star"). As a result, we observe that POMDPTrust achieves higher individual hit rates and is able to correctly classify messages with an overall accuracy of 82% versus LOAR (80%) and BLADE (78%)[16]. Additionally, Figure 4.10b depicts the mean absolute error (MAE) achieved by each algorithm. The MAE is a measure of the average distance each classification is from the true message classification. Note that a low MAE is desirable. From this figure, we can see that POMDPTrust outperforms BLADE by approximately 20%, while it slightly underperforms LOAR by 7%. Again, POMDPTrust performs very well against the Epinions data despite using a single, global reward function for all advisors (which presumably hurts performance, since not all advisors have the same utility functions). We offer some discussion as to how the results can be improved further by considering individual reward functions in Section 6.2.2.

In all, the results of these experiments demonstrate that POMDPTrust performs very well in real-world scenarios, outperforming LOAR in the Reddit experiment and BLADE in the Epinions one. A deeper discussion of POMDPTrust as it compares to both LOAR and BLADE follows in Chapter 5.

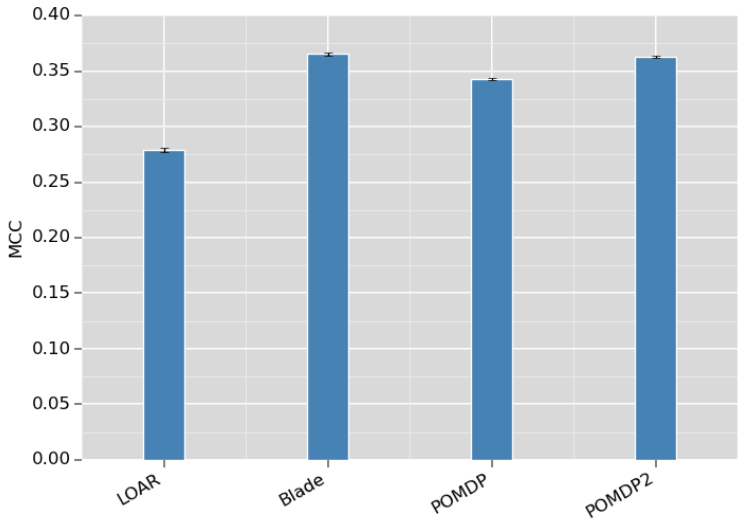
[16]The overall accuracy rates can be computed by dividing the sum of the diagonal entries in the respective classification matrices by the total number of messages in the testing set (which has been pre-computed for the convenience of the reader and is located in the bottom right corner of the respective classification matrices).

Table 4.7: Reddit.com dataset results classification matrices.

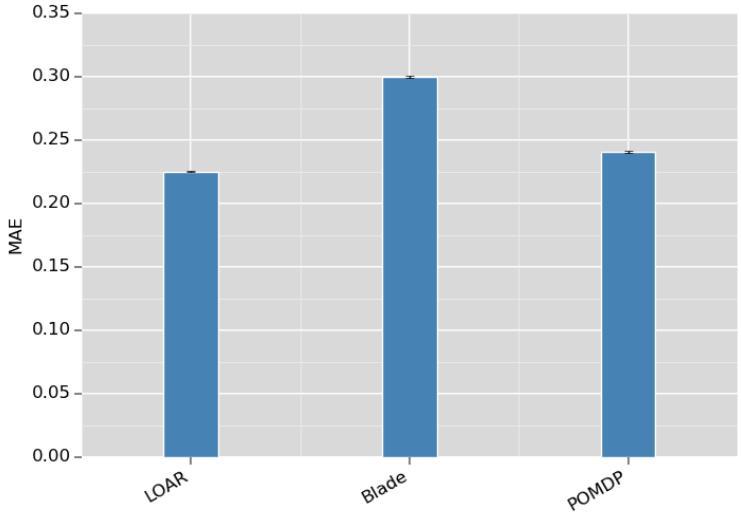
	Observed				
Predicted	Positive Ratings (1)		Negative Ratings (-1)		*Total*
LOAR					
Positive Ratings (1)	1,599,762	($\sigma = 6,120$)	414,036	(1,075)	*2,013,798*
Negative Ratings (-1)	79,339	(342.1)	128,141	(829.3)	*207,480*
Total	*1,679,101*		*542,177*		*2,221,278*
BLADE					
Positive Ratings (1)	1,476,253	(5,930)	290,701	(795.9)	*1,766,954*
Negative Ratings (-1)	202,847	(1,010)	251,476	(1,108)	*454,323*
Total	*1,679,100*		*542,177*		*2,221,278*
POMDP					
Positive Ratings (1)	1,347,475	(5,518)	239,535	(520)	*1,587,010*
Negative Ratings (-1)	331,626	(1,832)	302,641	(1,160)	*634,263*
Total	*1,679,101*		*542,179*		*2,221,280*
POMDP2					
Positive Ratings (1)	1,056,839	(5,172)	111,392	(407)	*1,168,230*
Negative Ratings (-1)	622,262	(2,015)	430,786	(1,240)	*1,053,047*
Total	*1,679,101*		*542,176*		*2,221,277*

Table 4.8: Epinions.com dataset results classification matrices.

	Observed										
Predicted	(1)		(2)		(3)		(4)		(5)		*Total*
LOAR											
(1)	198	($\sigma = 33$)	25	(3)	10	(2)	6	(2)	5	(0)	*243*
(2)	199	(32)	27,413	(553)	5,316	(50)	1,553	(29)	2,106	(44)	*36,587*
(3)	112	(18)	28,131	(269)	92,190	(670)	34,824	(118)	19,047	(226)	*174,305*
(4)	99	(13)	21,743	(285)	98,605	(755)	321,695	(2,107)	290,820	(2,359)	*732,961*
(5)	138	(18)	16,783	(204)	30,640	(192)	250,196	(1,345)	2,803,527	(6,086)	*3,101,282*
Total	*746*		*94,094*		*226,760*		*608,274*		*3,115,505*		*4,045,379*
BLADE											
(1)	20	(8)	14	(5)	25	(5)	22	(6)	20	(5)	*101*
(2)	44	(8)	20,338	(240)	3,923	(74)	1,989	(48)	2,132	(82)	*28,426*
(3)	25	(25)	4,671	(4,671)	21,662	(21,662)	14,857	(14,857)	10,084	(10,084)	*51,299*
(4)	39	(12)	3,081	(74)	23,708	(237)	96,649	(602)	79,495	(641)	*202,972*
(5)	618	(58)	65,990	(429)	177,442	(1,071)	494,757	(2,495)	3,023,774	(7,402)	
Total	*746*		*94,094*		*226,760*		*608,274*		*3,115,505*		*4,045,379*
POMDP											
(1)	395	(64)	1,024	(53)	531	(17)	618	(41)	1,548	(102)	*4,116*
(2)	72	(9)	45,447	(500)	14,567	(189)	10,173	(167)	26,263	(331)	*96,522*
(3)	38	(8)	13,268	(207)	100,620	(835)	52,135	(438)	40,864	(695)	*206,925*
(4)	39	(8)	8,298	(80)	45,933	(421)	241,784	(1,633)	127,770	(698)	*423,824*
(5)	201	(21)	26,057	(185)	65,109	(524)	303,565	(1,709)	2,919,059	(7,015)	*3,313,991*
Total	*745*		*94,094*		*226,760*		*608,275*		*3,115,504*		*4,045,378*



(a) Reddit.com results: Matthew's Correlation Coefficient (MCC), by algorithm (higher is better)



(b) Epinions.com results: Mean Absolute Error (MAE), by algorithm (lower is better)

Figure 4.10: Real-world experiment results.

Chapter 5

Discussion of Results

We will now discuss in greater detail the results of our experiments and will explore how our model relates to other trust research. As a reminder, in Chapter 4, we introduced a POMDP model called POMDPTrust to reason about which messages should be recommended to a user. We ran several simulations and carried out experiments using real-world data from Reddit.com and Epinions.com, and we compared the accuracy (using the Matthew's Correlation Coefficient and the Mean Absolute Error) of the recommendations using the POMDP approach versus the accuracy of several other approaches, including CredTrust, LOAR, SGD, and BLADE.

POMDPTrust vs. CredTrust

In our view, the POMDP model is an improved version of the CredTrust model for a variety of reasons. First, the POMDP approach introduces a mathematical framework for reasoning about which messages to show to users by drawing on well-established methods from the (PO)MDP literature for reasoning and making decisions under uncertainty. Unlike in CredTrust, the update rules used in POMDPTrust are principled in that they go beyond more ad-hoc heuristics. Instead, POMDPTrust learns an appropriate observation function in order to amalgamate advice from peers in the "right" way given the environment and distribution of users (viz., the mix of user credibilities and preferences). As a result, we see that POMDPTrust outperforms CredTrust in nearly all of the simulations.

However, POMDPTrust does slightly underperform CredTrust in the simulation depicted in Figure 4.2a when the mean credibility of peers in the simulation falls below 0.4. We believe this to be the case because CredTrust discounts, and hence filters out,

the advice of less credible peers whose ratings tend to be more stochastic. In particular, in such cases, POMDP sometimes learns an incorrect observation function because the ratings of peers are less deterministic and hence less indicative of the true probabilities. Perhaps seeding the POMDP model with an appropriate prior for the observation functions could help to boost POMDPTrust's performance in this particular simulation (and in such circumstances). The development of an appropriate prior is typically considered a domain-specific challenge; this is another area that could be considered as future work (see Section 6.2.1).

POMDPTrust vs. LOAR

It is clear that our POMDP approach outperforms LOAR in all of the simulations presented in Chapter 4 as well as in the real-world experiments we performed (see Figures 4.2, 4.3, 4.10a, and 4.10b). In some sense, POMDPTrust can thus be viewed as an evolution of LOAR; it corrects for the biases that LOAR introduces in the form of heuristic updates (see, for example, Section 3.3.3). Furthermore, unlike LOAR, the POMDPTrust model can be easily extended to include additional user features by simply extending the observation function to include the features. We demonstrated this in simulation where POMDPTrust was able to make use of both similarity and credibility information, whereas LOAR only makes use of user similarities. But POMDPTrust can be easily made to handle any additional information (e.g., message-specific features like message length or topic). As long as there is sufficient data from which to learn the observation function, POMDPTrust will always be able to make use of any additional information. In order to boost the performance of POMDPTrust, it is therefore important to identify features that are strongly correlated with the underlying message states. Further discussion of this is included in Section 6.2.1.

POMDPTrust vs. SGD

In Chapter 4 we also compared the POMDPTrust model to a Latent Factor Model that we abbreviated as "SGD". The results of these comparisons depicted SGD as outperforming POMDPTrust in a variety of scenarios (see Figures 4.4 and 4.5). Recall from Section 2.2.2 that the SGD model learns a number of hidden features of both messages and users. Essentially, the SGD model solves a nonlinear optimization problem to find a least-squares solution that situates each user and each message in a vector space for users and messages, respectively. Accordingly, we believe that part of the reason why the SGD model outperforms is because it performs a number of computations that the POMDPTrust model does not. For example, POMDPTrust does not consider any message-specific features.

Furthermore, in the simulations, POMDPTrust only considered two user-specific features: user similarity and credibility (whereas SGD learned 10 latent features). We postulate that POMDPTrust could be improved by considering message specific features in addition to user features, and by searching for additional, informative, and correlated metrics. Moreover, our implementation of POMDPTrust used a single, global reward function for all users. We postulate that using a user-specific reward function that caters to each user's individual preferences could help boost the model's performance. Additional discussion of this extension is offered in Section 6.2.2.

One method that could be considered for learning user reward functions draws on research concerned with inverse reinforcement learning [14, 33] (viz., the problem of learning the reward function that an expert is maximizing given the expert's policy). In particular, one could perform something akin to logistic regression: for a given user, we can use past messages that the user has rated as training data. Moreover, we can compute the POMDP beliefs for each training example. Taken together, we should be able to learn the rewards that cause the POMDP to classify messages based on the computed beliefs in such a way that a logistic cost function is minimized. We believe that such a technique could be used to derive user-specific reward functions that improves the overall performance of POMDPTrust.

As a final point, consider that SGD requires finding the least-squares minimizer solution given an $M \times N$ ratings matrix (M being the number of messages and N being the number of users). Whenever a new message is created or a new user enters the system, their associated feature variables are unknown. Accordingly, a new solution needs to be computed every time the system changes (additional messages are created or new users enter); given that real-world participatory media networks like Reddit can potentially have many thousands of messages created and millions of new votes cast per day, recomputing a least-squares solution may be prohibitive, and thus the SGD approach may not be the most desirable solution for such dynamic contexts. In contrast, the POMDPTrust approach can be done entirely online (and thus is perhaps more appropriate for online message recommendations). It would be interesting to perform a head-to-head experiment in the future to empirically validate this hypothesis and quantify the relative benefits of these two methods for different environments.

POMDPTrust vs. BLADE

The POMDPTrust model has many parallels to the Bayesian approach for inferring trust in e-marketplaces espoused by Reagan et al. in their BLADE model [35]. The most striking similarity is in the use of probability distributions to encode uncertainty about the

underlying parameters that drive inference about trust in the network. Both approaches essentially learn the "right" way to treat peer advice. That is, POMDPTrust performs Bayesian updates to learn the correct observation function for a given user. Likewise, BLADE performs Bayesian updates to learn the evaluation functions of peers in the system.

However, there are some differences, too. One difference is that the POMDP model incorporates some notion of user utilities[1] for making decisions about messages by using a reward function. In particular, POMDPTrust computes the expected utility for recommending a message versus rejecting it, and chooses the action that maximizes a user's overall expected utility. On the other hand, BLADE does not incorporate any model for reasoning about decisions or actions that agents should take[2].

Accordingly, in order to make recommendations using the BLADE model (for our experiments), we used a maximum likelihood estimate based on BLADE's prediction of author features in order to classify messages. This strategy worked reasonably well in the case of Reddit.com (where BLADE matched the MCC performance of POMDPTrust). However, BLADE underperformed in the Epinions experiment, where ratings sparsity is higher as a result of a larger number of classification classes (i.e., 1 through 5 stars); this hampered BLADE's ability to learn a good evaluation function, which resulted in lower overall performance in terms of MAE and individual hit rates. In contrast, the POMDP model's use of utilities gives rise to a natural means for making recommendation decisions, and was shown to perform very well in both simulation and against real-world data. Moreover, the model could potentially be extended to explicitly account for individual user preferences; individualized reward functions could serve to boost POMDPTrust's performance. (One potential avenue that could be explored for developing appropriate user reward functions was suggested in the SGD discussion above; another approach could be to specifically ask users about their preferences and encode there responses using suitable reward functions).

A second difference is that POMDPTrust is able to make use of a number of different features that could be correlated to the underlying message state. In particular, POMDPTrust can be extended to make use of message-specific features, social network information, etc., simply by extending the observation function. In fact, it is quite possible that POMDPTrust could operate in environments where there are no ratings provided for messages (we discuss these ideas further in Section 6.2.1). BLADE on the other hand is

[1]Note that we use utilities interchangeably with the rewards from the reward function specified by POMDPTrust.

[2] We consider [35] to be the standard reference to BLADE. We note, however, that Regan's thesis [34] includes a future work section that expands on possible decision making solutions. Interestingly, these suggest the use of POMDPs. We developed our POMDPTrust model independently (and later discovered the overlap with [34]).

tuned to make use of the evaluation functions of advisors to learn seller features, which are represented by Dirichlet priors; it offers no way to take advantage of other features.

Finally, BLADE offers little insight into how one might prune the advisor space; if all peers in a network provide feedback about a given message (or if implicit feedback is used to impute ratings), BLADE offers no insight as to how to choose which ones are most important to listen to. POMDPTrust, on the other hand, can follow a policy that "short circuits" once it develops a strong enough belief about a message state, thereby potentially avoiding a large amount of unnecessary computation. For example, consider a case where there exist hundreds of thousands of imputed ratings for a given message. In such a scenario, BLADE will consider every single rating before coming to a decision. POMDPTrust, on the other hand, might only need to view a small percentage of the ratings set, at which point it might obtain a belief about the message that is strong enough to make the expected utility for recommending/rejecting the message high enough that further polling becomes unnecessary[3]. Of course, this approach does not go as far as the work of Gorner et al. [20], whose work explicitly considers a careful selection of advisors to boost the overall accuracy of trust imputation; we discuss this in detail in Section 6.2.5.

Relationship with Trust Models

Since we approached the problem of recommending/classifying messages from the perspective of imputing trust to users in participatory media networks, it comes as no surprise that our work is closely related to the work of other trust researchers. One might reasonably draw a comparison between POMDPTrust and models like BRS, PTM, and TRAVOS, given that all of these models make use of Beta distributions (or the more general Dirichlet distribution) in one form or another during trust evaluation. It is important, however, to draw a distinction between the mechanisms used by POMDPTrust to amalgamate peer reports and the methods employed by the latter models. In particular, POMDPTrust uses Bayesian updates in order to learn an observation function, which is in turn used to extract the "correct" amount of information from subsequent advisor reports. In other words, POMDPTrust chooses the weights for its belief updates entirely on the basis of the data. On the other hand, BRS, PTM, and TRAVOS weight belief updates according to particular formulas which are shown to perform well in simulations (much like LOAR weighted its own belief updates; see Section 3.3.3). In would be interesting to further compare and contrast these different approaches and to concretely evaluate whether POMDPTrust outperforms these models in simulation. Morever, the notion of "uncertainty" used by BRS might be

[3]This is of course less of a concern when considering explicit ratings alone, given the sparse nature of real-world ratings on data.

a useful future addition as part of the POMDPTrust observation function to codify the degree to which advice from a particular peer is stochastic. In this way, POMDPTrust could be extended so that it learns to avoid stochastic peers' opinions[4].

Relationship to Recommender Systems

As a final consideration, we draw attention back to recommender systems more generally, which the reader will recall were introduced in Section 2.2. In that discussion, we briefly touched on the distinction between content-based and collaborative filtering methods for recommendations. We are now in a position to more fully explore the relationship between such recommender systems and the trust models with which our research is concerned.

The problem of recommending messages (or more generally, items) and the problem of evaluating peer trustworthiness are inexorably linked: both problems are defined by their goal of helping users access the right information or make the right choices. Recommender systems are tasked with finding for users the items that will provide them with the highest utilities; trust evaluation would be useful to perform so that users know which peers to interact with. Collaborative filtering recommender systems in particular try to exploit user similarities in order to recommender new messages of interest. In this context, the notion of *user similarity* is closely related to the notion of *user trustworthiness*; however, we believe that trustworthiness goes further, since it must incorporate notions like user expertise (i.e., credibility) and reputation (e.g., to avoid issues like folklore).

Clearly, there are many parallels that can be drawn between the two topics of research. Given this, our POMDPTrust model straddles the two, both providing recommendations to users as to which messages are likely to be the most desirable to see, while also incorporating notions of trustworthiness from other models, most notably in its Bayesian belief update model. POMDPTrust is therefore an interesting model from the perspective of trust researchers, as it begins to bridge the gap between the evaluation of trustworthiness and the ability to act on the basis of derived trust. It is also an interesting model from the perspective of recommender systems, because it provides some insight into exactly what types of messages users should be concerned with seeing; making decisions solely on the basis of similar-peer advice may not be the most appropriate in all domains. Moreover, the POMDP reward function is a convenient mechanism for incorporating user preferences.

[4] Note that this is a pitfall in both BLADE and POMDPTrust, which cannot differentiate well between those advisors whose evaluation functions are deterministic versus those whose ratings are provided randomly.

Contribution to a Comprehensive Trust Module

In [38], Sen discusses how trust research has focused primarily on the *a posteriori* evaluation of trust and how research has for the large part ignored issues regarding establishing, engaging, and using trusted relationships. He envisions a holistic multiagent systems architecture that incorporates a trust module as a core component to facilitate and shape interactions with other agents in the environment. With this goal in mind, Sen proposes a comprehensive trust management scheme (CMTS) that involves four major components to address trust issues and manage trusted relationships:

1. The ability to evaluate other agents' trustworthiness.
2. The ability for an agent to establish trustworthiness by taking certain actions.
3. The ability to strategically engage with agents on the basis of evaluated trustworthiness and for the purpose of further evaluating trust.
4. The ability to use trust relationships to select future actions.

To date, trust literature has largely focused on the evaluation of agent trustworthiness and has placed little emphasis on the other facets of a comprehensive trust module as proposed by Sen. Our POMDPTrust model is therefore novel with respect to the CTMS, as it explores the usage of trust information to select future actions by incorporating notions of user utilities that are associated with different outcomes (e.g., recommending a "good" message or rejecting a "bad" message). POMDPTrust also provides a novel approach to the evaluation of agent trustworthiness through the belief update mechanism inherent to POMDPs. Taken together, our work offers insight into various aspects of Sen's comprehensive trust module [38] and is therefore, we feel, of particular interest and value to trust researchers.

Chapter 6

Conclusions and Future Work

6.1 Summary

In this thesis, we explored the problem of recommending messages to users in participatory media networks. We approached the problem from the perspective of modeling the trustworthiness of peers in the network, which we argued is related to the neighbourhood collaborative filtering approach to recommender systems. This led to our developing CredTrust, an algorithm that incorporates peer similarity and credibility in order to recommend messages and stop the spread of false information. We demonstrated that CredTrust outperformed LOAR [12], another recommender system developed specifically to recommend annotations on learning objects in online learning environments so as to maximize the learning gains of students in the system. We also demonstrated that the LOAR model corresponded closely to other well-established models in trust literature (specifically, the Beta Reputation System [28]), and thus can be classified as a simple collaborative filtering algorithm for message recommendation.

From CredTrust, we moved on to develop POMDPTrust, a model grounded in Bayesian probability that uses a POMDP to account for peer ratings on messages. We pursued this avenue after observing that the linear functions proposed by CredTrust, LOAR, and several other trust models amount to heuristics that necessarily underperform under specific circumstances. POMDPTrust, by contrast, learns the statistically "correct" evaluation function for belief updates. We demonstrated the merits of this approach in simulation versus LOAR and CredTrust (wherein the POMDP model outperformed), and also considered its performance against a class of Latent Factor Models (wherein the POMDPTrust

stood its ground). Apart from simulation, we also validated our approach against real world data. Taken together, our efforts make clear the value of the POMDPTrust model.

In all, we have developed a new approach for making recommendations to users in participatory media networks regarding messages of interest with applications to both trust modeling and message recommendation. Specific contributions which have been produced in this thesis include the following:

- An exploration of *credibility* as it applies to trust evaluation in order to combat false information propagation. In particular, considering a user's credibility or expertise is important when trying to recommend the *right* messages, not necessarily just those (popular) messages that a user wants to see.
- A method for capturing the value of a trust modeling framework through simulation that varies agents' preferences by using *types*; this allows us to explore environments with different mixes of agents and their preferences.
- Proofs of two properties about LOAR that tease out some subtleties in the model, and as a result of these proofs, a demonstration of exactly where the model struggles.
- The insight that modeling rating sparsity is valuable to validate the performance of trust-based message recommendation algorithms (used as part of the validation of our approach)
- A demonstration of the application of POMDPs to recommender systems and trust modeling domains, as well as a concrete model that was shown to perform well in a variety of simulated and real-world environments.
- An investigation into two particular real-world data sets for use in validating message recommending algorithms, and a discussion of the properties required to carry out validation on other real-world data sets.
- An implementation of competing models so that they could become competitors as part of our evaluation of POMDPTrust.

6.2 Future Work

We conclude this thesis with a discussion of several next steps that could build on the work we have presented.

6.2.1 Extending POMDPTrust to Use Additional Features

In our simulations and real-world experiments, we calibrated POMDPTrust by using similarities and credibility[1]. It would be interesting to explore the use of additional features in order to further improve the recommendations made by POMDPTrust.

For example, recall that POMDPTrust emerged as a result of our exploration of CredTrust, but that model was focused primarily on advisor similarity and credibility. Investigating author similarity and credibility would also be valuable for future work. Inspired by observations such as these, different features that might be interesting to explore include the following:

- The distance a user is from the message author and peer raters in an associated social network. The idea behind considering social network properties stems from work like Seth et al. [39], in which the authors contend that strong and weak links between peers in a network contribute to the credibility associated with feedback about a particular message.
- Similarity between users and raters on the basis of implicit preferences/cues (as opposed to ratings). For example, in the online learning scenario, it might be interesting to look at the similarity between users on the basis of test scores, or on the basis of the length of time spent reading material versus watching learning videos, etc. Deriving user preferences on the basis of their actions outside of leaving ratings naturally brings in an element of user modeling that can serve to augment POMDPTrust (by more richly specifying similarities between users).
- The topic of the message, or number of different "entities" the message contains. Certain users might be more interested in different topics than others, or in messages that cover a number of different subjects.

In addition to considering additional features, it might also be useful to explore the use of non-uniform priors. In particular, for a given message, POMDPTrust starts with a uniform belief over the message state. However, given the often skewed nature of the data in real-world environments, it might be beneficial to start with a non-uniform prior that reflects this skewness. Moreover, if information regarding message authors is available, it might be useful to start with a prior that reflects a given user's past experience with the author (i.e., does a user typically find a given author's message to be useful or not). Using

[1]Credibility was only used in simulation, since we assumed the credibility information was available through use of an oracle.

a non-uniform prior can also be helpful in cold-start scenarios (when there is not a lot of evidence about a particular message). This would then afford modeling the author a greater role in message recommendation; the focus to date has been on modeling the raters (advisors) instead.

A related extension would be to consider implicit feedback by users to improve classification accuracy, as in the work of Hu et al. [25]. Implicit data can potentially be used to make rating sets more "dense", and might be helpful in systems where explicit feedback simply is not available. For example, on the social media site Twitter, users cannot explicitly rate *tweets* (the name given to messages). However, users can "retweet" messages, can choose to "favourite" a tweet, and can create new tweets that highlight subjects by using "hashtags". Perhaps such actions can be indicative of positive or negative feedback about a given message, and thus could be used in order to provide message recommendations for Twitter messages. Likewise, other implicit factors like the time a user spends reading certain threads of messages could be indicative of positive/negative feedback about the messages or the entities they contain. These implicit factors could then be used to extend the features included in the POMDP observation function.

Ultimately, considering additional features could be useful to improve recommendation accuracy, especially given the sparse ratings nature of real-world environments. In particular, such features might be able to help boost performance in cold-start scenarios, viz., when users have little common interactions with advisors (and thus have little evidence on which to judge and learn advisor evaluation functions). In addressing such problems, we might also benefit by drawing upon the work of Guo et al. [22], who discuss the use of social trust to improve collaborative filtering performance. In particular, their "Merge" method boosts performance in such cold-start scenarios by merging the ratings of trusted neighbours in order to improve similarity calculations between two particular users (who otherwise might have a very common small ratings set on the basis of which their similarity can be measured). Incorporating a similar notion to the *web of trust* as used by Guo et al. could find application in our own work, in addition to the ideas suggested above.

6.2.2 Learning a Good Reward Function

It is clear that the reward function has a very important role to play in POMDPTrust. In the Reddit.com experiment, a minor change in the relative true positive versus true negative rewards caused a 5% improvement in MCC. Accordingly, there is some evidence that the careful selection of rewards can improve POMDPTrust's performance immensely. We believe that future work could explore different ways to derive a good reward function in order to further boost POMDPTrust's classification accuracy:

- One idea would be to use standard regression techniques in order to find the rewards that minimize a cost function that quantifies the classification error made by POMDPTrust on the training data (see, for example, Chapter 4 of [4]). Such an approach could also integrate inverse reinforcement learning techniques, as explored by Ng and Russel in [33] or by Choi and Kim in [14].
- Another idea is to explore the use preference elicitation techniques in order to determine a good utility function for a given user, as is done by, for example, Craig Boutilier in [5].

Overall, we believe that this would be a promising step to consider in future work, as it seems reasonable that using user-specific reward functions should serve to improve the classification accuracy achieved by POMDPTrust, which in this thesis only uses a single, global reward function, as a first step.

6.2.3 Trust-Related Extensions

Applying POMDPTrust to Trust-Specific Problems

Our POMDPTrust model is related to trust research more generally. In particular, the POMDP agent's belief about message states is akin to a trust evaluation. One interesting application to explore would be e-marketplaces, especially given the intrinsic consideration of user utilities in POMDPs. In an e-marketplace domain, user utilities have a natural interpretation with respect to an agent's profit (from a fulfilled contract/interaction) and loss (from a failed transaction). We postulate that this explicit integration of user utilities would be useful in trust scenarios where evaluating the trustworthiness of sellers/advisors and deciding how to act on the basis of estimated trustworthiness (e.g., to make a purchase) is important and has economic consequences. In fact, this is similar to what was proposed by Shani, Heckerman, and Brafman [40], who present an MDPs to make recommendations in such e-commerce environments. Their work is thus related to our own POMDPTrust, however it is separate, as it focuses on recommending particular lists (sequences) of items that users are likely to purchase in a given order, and considers a state space of ordered purchase sequences. However, it nonetheless provides some interesting insights that would be useful in applying our own techniques to such a domain, and indeed also for simply considering extensions to our own model (e.g., with respect to new features).

Aside from extending POMDPTrust in the context of e-marketplaces, it would also be interesting to incorporate some notions of uncertainty as espoused by Jøsang and Ismail in

their Beta Reputation System [28]. Perhaps their notion of uncertainty could be used as an additional feature to capture the stochasticity of certain raters' evaluation functions, and hence allow POMDPTrust to learn to avoid stochastic advisors in favour of deterministic ones. Similarly, Şensoy et al. [16] use of a subset of Description Logics combined with Dempster-Shafer theory to reason about uncertain information and make decisions about deleting and/or discounting peer advice. This line of work could also prove informative when reasoning about the stochasiticity of peer feedback, as the authors offer specific methods for reasoning about how to discount feedback based on evidence.

Drawing on Latent Factor Models to Improve Trust Evaluation

Given the relationship between trust modeling and recommender systems as discussed in Chapter 5, we postulate that there could be some interesting crossover between the two topics of research. In particular, we have shown how notions of trust can influence recommendations in our POMDPTrust model; however, we also believe that recommender systems could provide some interesting insights into trust derivation. In particular, in Section 4.1.4, we demonstrated how SGD models can achieve great accuracy by combining a modeling of both messages and users; it would be very interesting to see if some of those same concepts can be ported for use in evaluating peer trustworthiness[2]. One challenge that might arise when exploring this relationship would be in finding an appropriate ground truth to use for in the least squares optimization required by, for example, gradient descent methods. Moreover, Latent Factor Models typically consider both users *and items*; trust models tend to focus more on users alone, though researchers such as Tran [45] discuss the potential value of item-specific trust. Accordingly, more consideration may need to be given to the items of exchange in trust modeling scenarios (e.g., products being sold in an e-marketplace) in order to apply LFM concepts, but if applicable, LFM concepts might be able to improve trust evaluation performance.

At the very least, it is clear that the problems of recommendation and trust evaluation are related, and this relationship could provide inspiration for future work in both domains.

6.2.4 Additional Experiments

As a final consideration, we briefly outline some ideas on different experiments that we think would be interesting to perform to further test and refine our models. One would be

[2]Indeed, we have already suggested in Section 6.2.1 that a hybrid content-based/collaborative approach could be useful for improving the performance of POMDPTrust by considering message-specific features.

to integrate social network features (like explicit friendship links between users) into a simulation environment in order to test how social links can affect message recommendations, and how POMDPTrust can be adapted to make use of social network information. Another simulation that would be interesting to run would be one that varies rating stochasticity, to see how various models cope with stochastic advisors.

Aside from new simulations, we also believe that it would be valuable to consider additional real-world messaging environments for further validation. In particular, it would be very interesting to experiment with environments like Twitter, Facebook, or YouTube. Unfortunately, these websites do not have readily available datasets for research purposes, so obtaining the appropriate data could be a challenge. Moreover, the websites just mentioned do not all make explicit positive/negative ratings information available (if they even record such information at all), so such experimenting with data from such sites would require developing and using implicit feedback (as described previously). Moreover, evaluating such data would also be challenging, given that implicit feedback might not contains a ground truth against which accuracy can be measured.

Lastly, it would also be useful to integrate several additional models into our experiments, including BRS [28], TRAVOS [44], PTM [50], BCM [39], and latent factor models [29], to provide additional comparitors. Adjusting some existing parameter values in our current experiments could be integrated as well. Another future comparator will be HABIT [43], a the state-of-the-art model that adopts a BLADE-like Bayesian model but with statistical inference leveraged by third party relationships. HABIT integrates user behaviour modeling, considering user utilities when making decisions on the basis of evaluated trustworthiness. Our model remains separate given its explicit consideration of user utilities in the decision making framework that we provide. That said, a more in-depth evaluation would be interesting to conduct.

6.2.5 Long-Term Considerations

The original vision for this thesis was to not only complete a trust-inspired model for message recommendation that was sensitive to similarity and credibility but also to accomplish two related aims. The first was to revisit the notion of credibility in order to move beyond the placeholder for it in Chapter 3: a value that would be provided by an oracle. The second was to resolve not only which messages are best shown to users but also to weigh in on what the ideal set of peers within a social network would be; this would then provide another avenue for reducing the flood of messages, considering those provided only by a subset of valued peers.

We have therefore delegated a bit more space to these two items of future work, within this chapter. Our current ideas for these subtopics form a useful detailed starting point for future research that could ultimately form the basis of an expanded proposal for message recommendation in social networks.

Further Exploration and Derivation of Credibility

Our exploration of user credibility relates to stereotypical trust and role-based trust derivation. In these contexts, a user's trustworthiness can be inferred on the basis of their stereotypical class. For example, in [7, 9], Burnett et al. propose the use of stereotypes to improve trust evaluations when there is little evidence about a given trustee (i.e., a limited number of past interactions between a given truster and trustee). They use decision trees to encode stereotyping functions that provide *a priori* trust evaluations and propose deriving stereotype features by observing the relationships between agents [8]. Their work on stereotypes is related to our notion of credibility in that stereotypes are features that are correlated with agent biases that provide insight into how agents act. Likewise, Liu et al. [31, 30] present StereoTrust, which also explores the use of stereotypes for improving trust evaluations. Essentially, stereotypes learn about different classes of users by capturing semantic information about them (for example, by considering information contained in their profiles).

Fang et al. [17] generalize these ideas by constructing a semantic ontology of seller attributes in e-marketplaces. Stereotypes can then be learned (using a "fuzzy" semantic decision tree) on the basis of identified seller attributes, resulting in more accurate trust evaluations when limited data is available (to address cold-start issues). While our notion of credibility was not developed with the cold-start issue in mind, it relates to the use of stereotypes in that we use user credibilities to improve message recommendations and to prevent false information propagation. It is important to note, however, that our notion of credibility goes beyond that of stereotypes, as a user can be atypical with respect to their stereotype. Our approaches allow for learning a user-specific metric, and thus to evaluate each user individually. Future work might explore how to integrate existing stereotype classifications of users into reasoning about credibility or to suggest our methods as a way of fine-tuning existing stereotypical trust proposals.

There is also merit in conducting a deep and detailed study of how credibility should be modeled. We note that credibility should not simply be equated with trustworthiness. Some trust models, e.g., [50], make an effort to combine both private and public reputation when calculating trustworthiness, but then the calculation will include a user-specific component. We would like to instead respect some of the tenets proposed in [39], requiring,

for example, the influence of *credible people* on determining who is credible. If we were to incorporate restrictions like this, it would help considerably in combating the "tyranny of the majority" (which already motivated our inclusion of credibility in CredTrust). We note that attempting a detailed model of credibility will be useful both for the modeling of authors and the modeling of raters, towards message recommendation. In particular, if authors are determined to be credible, then one might expect that most of their messages will be credible. However, even credible people may occasionally leave a bad message; this would be one challenge to address in our future research. An additional challenge that we can imagine is attempting to model credibility in environments where users are very homogeneous. However, we may be able to consider additional characteristics: if a user exhibits stochastic behaviour, for instance, they would be less reliable and therefore one would be less inclined to consider them as credible.

There are also some parallels between our notion of credibility and the notion of learning influence in social networks, where more credible peers may correspond to more influencial ones. Thus, work by, for example, Franks et al. [18] and Karthik et al. [42] could serve as a backdrop for a future investigation of identifying credible peers. In particular, Franks et al. [18] provide a general methodology for learning the network value of a node in terms of influence, and Karthik et al. [42] consider measuring the social capital of peers in the network. Both of these approaches could be useful in deriving the credibility of peers in our contexts.

Choosing Good Advisors

Research considering the issue of advisory network composition and size also provides interesting avenues for future work. In [20], Gorner et al. demonstrate through simulation that choosing a subset of trusted advisors can actually improve performance when evaluating agent trustworthiness especially in contexts such as e-marketplaces. In particular, they outline an approach to empirically determine the "optimal" network size. In their work, the authors vary the size of a buyers advisor network according to two separate criteria: a trustworthiness threshold and a maximum number of neighbours. Gorner et al. evaluate their approach to optimizing network size by considering the performance gains realized in two separate trust models: TRAVOS [44] and Zhang et al.s personalized trust model (PTM) [50]. In particular, for each model, the authors simulate an agent network consisting of a single buyer, 80 advisors, and 100 sellers. Sellers are simulated as having an inherent trustworthiness probability $T \in [0.1, 1]$, and advisors report ratings of sellers, lying with some probability (in particular, the authors examine cases wherein advisors lie 30% and 60% of the time). The authors also examine the impact that advisor referrals

have in calculating trust metrics, drawing inspiration from the work of Yu and Singh [49] (i.e., if a trusted advisor does not have direct experience with a given seller, the buyer can transitively elicit feedback from the advisors advisor network). They are able to demonstrate how the size of advisor networks can be set to minimize the usage of untrustworthy advisor reports.

Although such techniques have not been tested in sparse ratings environments, it seems reasonable that carefully selecting advisors on behalf of a user could be beneficial for improving classification accuracy. Moreover, relying on a small set of advisors for advice regarding all messages opens opportunities to integrate research that examines trust propagation as it is examined by Hang et al. [24] (since it is unreasonable to expect advisors to have first-hand experience with all messages).

One framework that might prove useful for reasoning about the optimal number of advisors in a user's advisory network and for expanding on the findings presented by Gorner et al. [20] that we have explored builds on certain game-theoretic concepts. Cheap-talk is a model for communication between an informed sender and an uninformed receiver: the idea is that the receiver makes some decision (for example, reading a message, or buying a product from a seller) after receiving messages from senders (i.e., polling for advice from advisors), and the decision results in a payoff for both the sender(s) and the receiver. Crawford and Sobel [15] and Chen et al. [13] formalize a model for cheap-talk communication and characterize various solution concepts for the game. Drawing on such work, Buntain et al. [6] describe a game theoretic framework that can be applied to study and address certain vulnerabilities in trust systems. The authors present a model under which agents can be one of three types ("honest", "fraudulent", and "saboteurs"); this model provides an interesting foundation upon which the process of trust evaluation and establishment can be studied. With these concepts in mind, we propose the following repeated Bayesian game:

- N is the set of players, which includes the current agent a and its peers $p \in P$
- As in POMDPTrust, the message has an intrinsic underlying state $s^* \in [0, 1]$, considered here to be deterimined by Nature, drawn from some probability distribution S
- Advisors partially observe this state and then send messages $m \in M = [0, 1]$ to a (such reporting might be stochastic); the agent a chooses to set some threshold R that determines which reports should be heeded

- Advisors have hidden types θ_p (known to themselves but not to a) that dictates their reporting behaviour and determines whether or not their advice should be heeded; advisors report with expected error $\epsilon = |s^* - E[m_p]|$

- Each $p \in P$ has a utility function $u_p : S \times M \times R \times \Theta \mapsto \mathbb{R}$ that depends on the error in its report $\epsilon : S \times M \mapsto \mathbb{R}$, a's threshold setting (i.e., whether or not a listens to p), and on p's own type We propose this utility function is decreasing in R (i.e., $\frac{\partial}{\partial_R} u_p(\cdot) < 0$), and convex in both θ and ϵ (i.e., $\frac{\partial^2}{\partial^2_\Theta} u_p(\cdot) > 0$ and $\frac{\partial^2}{\partial \epsilon^2} u_p(\cdot) > 0$)

- Similarly, a has a utility function $u_a : S \times \vec{M} \times R \times \vec{\Theta} \mapsto \mathbb{R}$ that depends on the error in its inference of the state of the world, the advisor threshold it sets, and the types of the solicited advisors. One potential utility function might take the form $u_a(s^*, \vec{m}, r, \vec{theta}) = \sum_{p \in P} H(\theta_p - r) f(|s^* - m_p|)$ where the unit step function $H(\cdot)$ serves to discriminate those reports from agents who do not meet the threshold r and $\frac{d}{d\epsilon}(\cdot) < 0$.

We present this preliminary and very general framework in order to motivate future work on choosing the right advisors to listen to. In particular, the problem of choosing advisors is akin to setting the threshold R in this game; reasoning about how to set this threshold could be done by finding a strategy for u that maximizes u's expected utility. It is our hope that these ideas can provide a starting point for further investigation into choosing a good (optimal) set of advisors. Perhaps developing this or a related model further might lead to a specific strategy that agents can follow for choosing their advisors, without the need to simulate an agent environment and determine empirically an appropriate set of advisors.

Trust delegation chains offer another interesting avenue for future work regarding eliciting peer feedback, as in the work of Burnett and Oren [10]. In particular, delegation chains allow advisors to recursively delegate a given task (in our case, the task of opining about a message) to other users in a network. The notion of trust delegation is therefore also useful when considering the size and composition of one's advisory network: we postulate that maintaining a small set of advisors for whom one can accurately estimate an evaluation function (on the basis of a lot of past evidence/interactions), and allowing those advisors to recursively delegate feedback to other similarly maintained advisor sets, could improve ratings even further (especially given the sparse nature of real-world data).

6.3 Closing Remarks

In closing, we believe we have provided a number of novel contributions in this thesis by examining the problem of recommending messages to users in online, participatory media environments. Our decision to integrate both credibility and similarity into our approach is motivated in part by the desire to prevent popular messages from similar but perhaps less credible peers to be a major influence on what users will see. Handling this problem is especially valuable in contexts such as peer-based education or, even more dramatically, self-help online health networks. The model that we have presented in this thesis works towards finding applications in recommender systems as well as in trust evaluation and management. We are excited by the prospects of further developing this model and have proposed several concrete directions for doing so. Taken together, we believe that we have provided research that begins to address the "use" and "evaluation" components of Sen's proposed comprehensive trust module [38], and as such offer an attractive starting point for future trust modeling research. In addition, we believe we have provided new insights as well for researchers who are focused on POMDPs: we have shed some light on how this paradigm can be used for trust modeling, providing both a deep and detailed model and a validation that clarifies how to measure the intrinsic value of this approach.

APPENDICES

Appendix A

Glossary

Agent An (semi-)autonomous entity that acts on behalf of a human user.

Bayesian Network A directed graph that encodes probabilistic dependencies (edges) between random variables (nodes).

Collaborative Filtering An overarching paradigm that considers the user similarities (e.g., ratings similarities) in order to make recommendations.

Credibility A global measure of the extent to which a user's messages are believable.

Latent Factor Model A class of collaborative filtering models that learns hidden features for users and items in order to make recommendations.

Markov Decision Process A mathematical model for decision making that incorporates uncertainty about the environment in the form of probabilistic state transitions.

Monte-Carlo Simulation A stochastic technique that is used to approximate some value (e.g., the expectation of a random variable) by averaging sample values obtained through repeated trials in a simulation.

Monte-Carlo Tree Search A method for approximating the optimal policy for a POMDP that constructs a biased search tree by using *Monte-Carlo Simulation* to evaluate the desirability of different actions.

Partially Observable MDP A mathematical model for decision making wherein the agent cannot directly perceive the state of the environment, but rather can only make correlated observations. See also *Markov Decision Process*.

Similarity A measure of the degree to which two users have the same preferences (e.g., for messages).

Stochastic Gradient Descent A technique for optimizing an objective function. Also, the name given to our implementation of a particular *latent factor model* for recommending messages.

Trust A measure that can be interpreted as the probability with which the trust*ee* will fulfill some future obligation to the trust*er*, e.g., by performing adequately on a contract in an e-commerce setting.

Appendix B

Calculations for LOAR Folklore Example

Here we illustrate various calculations for the example presented in Section 3.1. To begin, we demonstrate how to calculate the LOAR similarity metric between s and p_1. Since both users voted the same for messages $\{m_2, m_3, m_4\}$, $vS = 3$, and since both users voted differently for messages $\{m_1, m_5\}$, $vD = 2$. Thus, the final similarity metric is:

$$S = \frac{vS - vD}{vS + vD} = \frac{3-2}{3+2} = \frac{1}{5} = 0.2$$

The remaining similarity metrics displayed in Table 3.2 can be calculated in the same manner.

Next we demonstrate how to calculate the overall annotator reputation, T_q. Recall that T_q is calculated by taking the mean annotation reputation over all annotations created by author q. This requires first calculating each annotation's general reputation, v_a, which is simply the average annotation reputation (i.e., the number of positive ratings divided by the number of ratings left on the annotation). For m_1, this equates to the following:

$$v_{a_1} = \frac{0+1+0+0+1}{5} = 0.4$$

The remaining ratings for $(v_{m_2}, v_{m_3}, v_{m_4}, v_{m_5}, v_{m_6})$ can be calculated in a similar fashion, yielding $(0.8, 0.8, 0.8, 0.4, 0.75)$, respectively. Then, T_q is simply the mean over all the v_{m_i}'s:

$$T_q = \frac{0.4+0.8+0.8+0.8+0.4+0.75}{6} = 0.6583$$

Finally, we demonstrate how to arrive at the predicted benefit for m_6 under the Tally, Trust, and Cauchy approaches. Each approach requires calculating the "votes for" m_6, denoted vF, and the "votes against", denoted vA. Each positive rating on m_6 increments vF by 1 plus the similarity between s and the peer from whom the rating came. So,

$$vF = (1 + 1 * (0.2)) + (1 + 1 * (0.6)) + (1 + 1 * (0.2)) = 4$$

$$vA = (1 + 1 * (-0.6)) = 0.4$$

Given these counts, the Tally predicted benefit is as follows:

$$\begin{aligned} Tally &= \frac{1}{2}\left(\frac{vF - vA}{vF + vA} + 1\right) \\ &= \frac{1}{2}\left(\frac{4 - 0.4}{4 + 0.4} + 1\right) \\ &= 0.9091 \end{aligned}$$

The Cauchy predicted benefit is as follows (arbitrarily setting $\gamma = 2$):

$$\begin{aligned} Cauchy &= \frac{1}{\pi} \cdot \arctan\left(\frac{vF - vA + T_q}{\gamma}\right) + 0.5 \\ &= \frac{1}{\pi} \cdot \arctan\left(\frac{4 - 0.4 + 0.6583}{0.2}\right) + 0.5 \\ &= 0.8602 \end{aligned}$$

And lastly, the Trust predicted benefit is computed as follows (setting N_min to 10 as in [11], and letting nV represent "number of votes"):

$$\begin{aligned} Trust &= \frac{\min(N_{min}, nV)}{N_{min}} \cdot Tally + \left(1 - \frac{\min(N_{min}, nV)}{N_{min}}\right) \cdot T_q \\ &= 0.4 \cdot 0.9091 + 0.6 \cdot 0.6583 \\ &= 0.7586 \end{aligned}$$

Appendix C

Real World Datasets

C.1 Suitability for Experiments

The data from Reddit.com was appropriate for the purpose of validating our POMDP approach because it contains associations between users, messages, and ratings. In particular, from the data, we can determine which user rated which message. Moreover, the ratings themselves are binary and contain both positive and negative ratings. Likewise, the data from Epinions was appropriate because the same user-message-rating associations are present. The Epinions data also contains authorship associations between users and messages, so it is possible to determine author reputations (for deriving message priors in BLADE and for determining the overall author reputations in LOAR). That said, the authorship information was only used in the BLADE model (we did not experiment with either of the LOAR Cauchy or Trust approaches).

In order to validate our approach, a dataset must contain at minimum an association between users, messages, and their ratings (so that the observation functions and similarities/peer evaluation functions can be learned and exploited). Furthermore, there must be a way of interpreting ratings as belonging to two or more classes (e.g., binary, as in the case of Reddit, or multinomial, as in the case of Epinions). This precludes datasets from services in which users can only express positive feedback about messages (e.g., by "liking" the message), unless the absence of positive feedback can be reasonably interpreted as negative feedback. This point is important, as several services (e.g., Facebook, Digg, Twitter) do no allow users to provide negative feedback. Accordingly, a mechanism for imputing multiclass feedback would be required in order to carry out experiments against such data.

C.2 Data Samples

This section exhibits a short selection of data from each of the Reddit and Epinions data sets. Note that the data in each case has been anonymized by the original data providers.

Table C.1: Reddit article ratings information

Username	Link ID	Vote
00ash00	t3_as2t0	1
00ash00	t3_ascto	-1
00ash00	t3_asll7	1
00ash00	t3_atawm	1
00ash00	t3_avosd	1
0-0	t3_6vlrz	1
0-0	t3_6vmwa	1
0-0	t3_6wdiv	1
0-0	t3_6wegp	1
0-0	t3_6wegz	1

Table C.2: Epinions article author information

Article ID	Author ID	Subject ID
1,445,594	718,357	149,002,425,217
1,445,595	220,568	149,003,604,865
1,445,596	717,325	5,303,145,344
1,445,597	360,156	192,620,893,057
1,445,598	718,857	149,002,163,073
1,445,600	513,114	34,252
1,445,601	718,997	5,576,168,320
1,445,602	719,278	34,252
1,445,603	372,997	44,674
1,445,604	298,574	70,012

Table C.3: Epinions article ratings information

Article ID	Member ID	Rating
139,431,556	591,156	5
139,431,556	1,312,460,676	5
139,431,556	204,358	5
139,431,556	368,725	5
139,431,556	277,629	5
139,431,556	246,386	5
139,431,556	293,732	5
139,431,556	525,858	5
139,431,556	237,120	5
139,431,556	971,935,620	5

Appendix D

Learning the Observation Function

In this appendix, we clarify the Bayesian update procedure for learning the observation function in POMDPTrust.

The goal is to refine our belief about each θ_s so that the observation function converges on one that represents a good belief about the probability of obtaining a particular observation tuple given the state s of the message. Repeatedly, for each message in the training set, we elicit feedback from each advisor a, which results in obtaining an observation $o_s^a = (r_s^a, m^a, c^a)$, where m^a and c^a are the similarity and credibility metrics, respectively, for the advisor. (We assume that these metrics can be interpreted as the parameters θ_m^a and θ_c^a to binomial distributions that dictate whether or not agents are similar/credible, as in Section 4.1.3, and that these values are independent of the message state). Since we know the ground truth benefit of each training message, we then can perform a Bayesian update to learn the posterior probability as follows:

$$Pr(\theta_s \mid o_s^a) \propto Pr(o_s^a \mid \theta_s) \cdot Pr(\theta_s) \tag{D.1}$$

$$= \sum_m \sum_c Pr(r_s^a \wedge m_s^a \wedge c_s^a \mid \theta_s) \cdot Pr(\theta_s) \tag{D.2}$$

$$= \sum_m \sum_c Pr(m^a \wedge c^a) Pr(r_s^a \mid m^a, c^a, \theta_s) \cdot Pr(\theta_s) \tag{D.3}$$

$$= \sum_m \sum_c (\theta_m \cdot \theta_c) \cdot \theta_{s,o_s^a} \cdot Pr(\theta_s) \tag{D.4}$$

$$= \sum_m \sum_c \kappa_{m,c} \cdot D_{s,o_s^a}(\theta_s) \tag{D.5}$$

Here, D.3 follows from the chain rule of probability and D.4 follows because we assume that similarity and credibility are independent parameters. Moreover, since we model $Pr(\theta_s)$ as a Dirichlet distribution $D_s(\theta_s)$, D.5 follows because $\theta_{s,o_s^a} Pr(\theta_s) = D_{s,o_s^a}(\theta_s)$, i.e., is the same Dirichlet distribution except that the hyperparameter that corresponds to the observation o_s^a has been incremented by 1. Accordingly, the updated observation function is a mixture of Dirichlets, where each Dirichlet is weighted by $\kappa_{m,c}$. As in [35], we can preserve the expectations over this Dirichlet mixture by instead storing a single Dirichlet distribution whose hyperparameters are a weighted mixture of each individual Dirichlet hyperparameters.

References

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, pages 235–256, 2002.

[2] Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *International Conference on Data Mining*, pages 43–52, 2007.

[3] Richard Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 6:679–684, 1957.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Craig Boutilier. A pomdp formulation of preference elicitation problems. In *AAAI/IAAI*, pages 239–246, 2002.

[6] Cody Buntain, Jennifer Golbeck, Dana Nau, and Sarit Kraus. Advice and trust in games of choice. In *Tenth Annual International Conference on Privacy, Security, and Trust*, pages 157–158, 2012.

[7] Chris Burnett, Timothy J. Norman, and Katia Sycara. Bootstrapping trust evaluations through stereotypes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 241–248, 2010.

[8] Chris Burnett, Timothy J. Norman, and Katia Sycara. Sources of stereotypical trust in multi-agent systems. In *Proceedings of the AAMAS 2011 international workshop on Trust in Agent Societies (TRUST11)*, 2011.

[9] Chris Burnett, Timothy J. Norman, and Katia Sycara. Stereotypical trust and bias in dynamic multi-agent systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(2):26, 2013.

[10] Chris Burnett and Nir Oren. Position-based trust update in delegation chains. In *Proceedings of the AAMAS 2013 international workshop on Trust in Agent Societies (TRUST13)*, pages 51–62, 2013.

[11] John Champaign. *Peer-Based Intelligent Tutoring Systems: A Corpus-Oriented Approach*. PhD thesis, University of Waterloo, 2012.

[12] John Champaign, Jie Zhang, and Robin Cohen. Coping with poor advice from peers in peer-based intelligent tutoring: The case of avoiding bad annotations of learning objects. In *Proceedings of User Modeling, Adaptation and Personalization (UMAP)*, pages 38–49, 2011.

[13] Ying Chen, Navin Kartik, and Joel Sobel. Selecting cheap-talk equilibria. *Econometrica*, 76(1):117–136, 2008.

[14] Jaedeug Choi and Kee-Eung Kim. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, pages 691–730, 2011.

[15] Vincent P. Crawford and Joel Sobel. Strategic information transmission. *Econometrica*, 50(5):1431–1451, 1982.

[16] Murat Şensoy, Achille Fokoue, Jeff Z. Pan, Timothy J. Norman, Yuqing Tang, Nir Oren, and Katia Sycara. Reasoning about uncertain information and conflict resolution through trust revision. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 837–844, 2013.

[17] Hui Fang, Jie Zhang, Murat Sensoy, and Nadia Magnenat Thalmann. A generalized stereotypical trust model. In *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom)*, 2012.

[18] Henry Franks, Nathan Griffiths, and Sarabjot S. Anand. Learning influence in complex social networks. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 447–454, 2013.

[19] Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michèle Sebag, David Silver, Csaba Szepesvári, and Olivier Teytaud. The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113, 2012.

[20] Joshua Gorner, Jie Zhang, and Robin Cohen. Improving trust modelling through the limit of advisor network size and use of referrals. *Electronic Commerce Research and Applications (ECRA)*, page accepted., 2012.

[21] Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. *Advances in Neural Information Processing Systems (NIPS)*, 2:1025–1033, 2012.

[22] Guibing Guo, Jie Zhang, and Daniel Thalmann. A simple but effective method to incorporate trusted neighbors in recommender systems. In *Proceedings of the 20th international conference on User Modeling, Adaptation, and Personalization*, pages 114–125, 2012.

[23] Chung-Wei Hang and Munindar P. Singh. Generalized framework for personalized recommendations in agent networks. *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1–27, 2011.

[24] Chung-Wei Hang, Zhe Zhang, and Munindar P. Singh. Generalized trust propagation with limited evidence. *IEEE Computer*, pages 1–8, 2012.

[25] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining*, pages 263–272, 2008.

[26] Carl J. Huberty and Stephen Olejnik. *Applied MANOVA and Discriminant Analysis*. Wiley, 2006.

[27] Audun Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, 2001.

[28] Audun Jøsang and Roslan Ismail. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*, pages 324–337, 2002.

[29] Yehuda Koren, Robert M. Bell, and Chris Volinksy. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, 2009.

[30] Xin Liu, Anwitaman Datta, and Krzysztof Rzadca. Trust beyond reputation: A computational trust model based on stereotypes. *Electronic Commerce Research and Applications*, 12(1):24–39, 2013.

[31] Xin Liu, Anwitaman Datta, Krzysztof Rzadca, and Ee-Peng Lim. Stereotrust: a group based personalized trust model. In *18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 7–16, 2009.

[32] Paolo Massa and Paolo Avesani. Trust aware recommender systems. In *ACM conference on Recommender Systems*, pages 17–24, 2007.

[33] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

[34] Kevin Regan. A social reputation model for electronic marketplaces sensitive to subjectivity, deception and change. Master's thesis, University of Waterloo, 2006.

[35] Kevin Regan, Pascal Poupart, and Robin Cohen. Bayesian reputation modeling in e-marketplaces sensitive to subjectivity, deception and change. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 1206–1212, 2006.

[36] Noel Sardana, Robin Cohen, Jie Zhang, and John Champaign. Credibility-based trust in social networks. In *Proceedings of the AAMAS 2013 international workshop on Trust in Agent Societies (TRUST13)*, pages 12–23, 2013.

[37] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system — a case study. In *ACM WEBKDD Workshop*, 2000.

[38] Sandip Sen. A comprehensive approach to trust management. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 797–800, 2013.

[39] A. Seth, J. Zhang, and R. Cohen. Bayesian credibility modeling for personalized recommendation in participatory media. In *Proceedings of the International Conference on User Modeling, Adaptation and Personalization (UMAP)*, pages 279–290, 2010.

[40] Guy Shani, David Heckerman, and Ronen I. Brafman. An mdp-based recommender system. In *Journal of Machine Learning Research*, pages 1265–1295, 2005.

[41] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2164–2172, 2010.

[42] Karthik Subbian, Dhruv Sharma, Zhen Wen, and Jaideep Srivastava. Social capital: The power of influencers in networks. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 1243–1244, 2013.

[43] W. T. Luke Teacy, Michael Luck, Alex Rogers, and Nicholas R. Jennings. An efficient and versatile approach to trust and reputation using hierarchical bayesian modelling. *Artificial Intelligence*, 2012.

[44] W. T. Luke Teacy, Jigar Patel, Nicholas R. Jennings, and Michael Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 12(2):183–198, 2006.

[45] Thomas Thanh Tran. *Reputation-Oriented Reinforcement Learning Strategies for Economically-Motivated Agents in Electronic Market Environments*. PhD thesis, University of Waterloo, 2004.

[46] Joel Veness, Kee Siong Ng, Marcus Hutter, and David Silver. A monte carlo aixi approximation. *J. Artif. Intell. Res*, pages 95–142, 2010.

[47] Ngo Anh Vien, Wolfgang Ertel, Viet-Hung Dang, and TaeChoong Chun. Monte-carlo tree search for bayesian reinforcement learning. *Applied Intelligence*, 39:345–353, 2013.

[48] Gerhard Weiss. *Multiagent Systems*. MIT Press, 2013.

[49] Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *Proceedings of the Fourth International Workshop on Cooperative Information Agents*, pages 154–165, 2000.

[50] Jie Zhang and Robin Cohen. Evaluating the trustworthiness of advice about seller agents in e-marketplaces: A personalized approach. *Electronic Commerce Research and Applications*, pages 330–340, 2008.

[51] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *International Conference on Algorithmic Aspects in Information and Management*, pages 337–348, 2008.