Novel Directions for Multiagent Trust Modeling in Online Social Networks

by

Alexandre Parmentier

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, 2020

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

This thesis presents two works with the shared goal of improving the capacity of multiagent trust modeling to be applied to social networks. The first demonstrates how analyzing the responses to content on a discussion forum can be used to detect certain types of undesirable behaviour. This technique can be used to extract quantified representations of the impact agents are having on the community, a critical component for trust modeling. The second work expands on the technique of multi-faceted trust modeling, determining whether a clustering step designed to group agents by similarity can improve the performance of trust link predictors. Specifically, we hypothesize that learning a distinct model for each cluster of similar users will result in more personalized, and therefore more accurate, predictions.

Online social networks have exploded in popularity over the course of the last decade, becoming a central source of information and entertainment for millions of users. This radical democratization of the flow of information, while purporting many benefits, also raises a raft of new issues. These networks have proven to be a potent medium for the spread of misinformation and rumors, may contribute to the radicalization of communities, and are vulnerable to deliberate manipulation by bad actors.

In this thesis, our primary aim is to examine content recommendation on social media through the lens of trust modeling. The central supposition along this path is that the behaviors of content creators and the consumers of their content can be fit into the trust modeling framework, supporting recommendations of content from creators who not only are popular, but have the support of trustworthy users and are trustworthy themselves. This research direction shows promise for tackling many of the issues we've mentioned.

Our works show that a machine learning model can predict certain types of anti-social behaviour in a discussion starting comment solely on the basis of analyzing replies to that comment with accuracy in the range of 70% to 80%. Further, we show that a clustering based approach to personalization for multi-faceted trust models can increase accuracy on a down-stream trust aware item recommendation task, evaluated on a large data set of Yelp users.

Acknowledgements

I would like to thank my supervisor, Professor Robin Cohen, for the huge amount of help she has given me in completing my Master's studies and preparing this thesis. Robin has always been available, understanding, and willing to work overtime to help me pursue my goals, and has encouraged and supported my ambitions. I also thank Professor Kate Larson and Professor Peter Van Beek for volunteering their time to read my thesis, and for the insightful comments and questions that have helped to shape this final document.

Thank you to Professor Jie Zhang and Noel Sardana for their contributions to the trust modeling field which have inspired me, and for each taking the time to correspond with me and help me refine my ideas for this work.

Thanks to Wendy Rush and Joe Petrik for their help with the various administrative tasks that have accompanied these works, and for the kindness and empathy with which they have helped me to respond to challenging situations.

I'd like to thank the friends I have made in Waterloo: Ben, Chris, Rishav, and Alister, for their companionship and the support they have given me during my time at Waterloo.

Finally, I thank my partner Despina, whose endless support and patience has made this work possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online sources of information are increasingly relied upon by many. According to yearly studies by the Pew research center, the percentage of American adults using the internet has jumped from 52% in 2000 to 90% in 2019 [11]. In addition to the established institutions that have made the jump from paper and TV to the web, many new blogs, content aggregators, and social networks have become a vital source in the information diet: up to 62% of American adults rely on information shared through social media for their news [58]. A study conducted in the wake of the 2016 American election found that, among American voters, Facebook ranked as the third most relied upon source for news about the election (after Fox News and CNN), far outranking local TV and newspapers and many national stations [29]. It is clear that the power to influence and inform has shifted drastically away from traditional institutions and into the hands of individuals.

While this democratization of information and influence may strike one as appealing, there are reasons to be concerned about this new paradigm. According to Facebook, throughout the 2016 American election thousands of ads designed to incite panic over gun rights and LBGTQ rights were purchased by accounts believed to be funded by the Russian government, some of them specifically targeting voters in swing districts [75]. Also in 2016, a heavily armed man broke into a neighborhood pizza parlor during business hours and fired shots after having become convinced by an online conspiracy popular on Twitter that the basement of the restaurant was used by the Clintons and other Washington elite to murder and rape children [24]. In a 2019 report, the Southern Poverty Law Center has stated that the ability to propagate hateful rumors and rhetoric online is a key factor in the 30% increase in the number of hate crimes reported per year in America since 2014 [74].

As can be seen from the above examples, the radically open information space online allows equal opportunity to foreign government propaganda, paranoid conspiracies, unsubstantiated rumors, and hateful rhetoric[1]. The "top-down" flow of information, where professionals working in reputable institutions provide advice to a few large media companies which then distribute it through well-known channels, has lost much of its relevance in the web era. Now, a "bottom-up" system has a large sway: individuals mixing opinion, fact and style jockey for relevance in a complex network of connections, where the most entertaining, provocative and attractive often win the most attention. Unfortunately, much of this information (especially that related to the examples we gave above) must be considered untrustworthy: that is, information which misleads, incites, and manipulates, which does not represent the truth, and is not helpful to the lives of the people who read it.

1.1 Why Recommendation?

One way to address the existence of untrustworthy information online is to deploy message recommender systems. Rather than showing users a random sampling or chronologically ordered list of the content that has been added to the network since their last visit, artificial intelligence (AI) systems can be designed to reason about which messages should be shown to which users. There are a wealth of possible approaches to this problem. These systems can focus on the content of messages, using black-lists, natural language processing and image recognition to detect unwanted or disturbing content. They can model the reputations of content creators (the authors of messages and tweets), finding patterns and determining which authors have a history of pushing untrustworthy content into the network. Another approach is to focus the view on the network as a whole, noticing messages that are spreading quickly, or information that appears to be being spread in a manner which suggests coordination and subjecting it to further scrutiny. Alternatively, these systems could focus entirely on the preferences of content consumers, recognizing patterns in the reactions to content among the consumers and recommending messages which appear likely to fit into the model of a user's preferences.

Even if untrustworthy content was not an issue, one would still have to contend with the massive glut of information submitted to these networks on a daily basis. There is simply *far* more information being posted on social networks than any individual can hope to keep up with. Therefore, some system for filtering or ranking information is necessary in order

[1] And these are only the types of content which are *allowed* - online stalking, revenge porn, harassment and bullying all exist in a legal gray-area internationally, while other types of blatantly illegal and harmful activities are also facilitated by the internet.

to preserve a modicum of utility of the users of these services. As networks have grown in popularity, many have had to switch away from a chronological ranking of messages (e.g. simply showing information in the order in which it was created), and now deploy complex ranking and user modeling approaches in order to offer a personalized peephole view into the activity of the network[2].

Therefore, either to combat untrustworthy information or to cut through the massive amounts of information, some sort of message recommendation (alternatively but equivalently: filtering, ranking, suggesting) must occur. In this thesis, we take as inspiration the former cause, but both are indeed relevant.

1.2 Why Trust Modeling?

So far we have argued three points: 1) online social networks are a useful and growing tool for accessing and sharing information 2) however, this tool is vulnerable to a host of novel threats, jeopardizing the trustworthiness of information shared though it and 3) message recommendation (i.e. content filtering) is a necessary tool to combat information overload, and a useful one for dealing with untrustworthy information. Finally, we argue that trust modeling of content creators (i.e. message authors) is a useful direction for the enhancement of message recommendation systems.

One reason a trust modeling approach is attractive is to deal with the number of sources of information (individual authors) that are active in social networks. A key difference between the growing online social information sphere and the traditional institutionally driven information sources is the number of voices. Online, there are literally billions of potential "sources" of information, corresponding to all the users of a network, a clear problem when every voice is given roughly equal opportunity to capture attention in a social network. It would be obviously useful for a message recommendation system to have a model of which sources of information are trustworthy or not. A trust modeling approach can provide this model. Under a trust modeling approach, we model attributes of each of these sources, as well as examining their histories of observable behaviour, and make predictions about which sources will be trustworthy in the future.

Another reason is that the trust modeling approach is highly flexible. Modern trust models (which we will introduce in Section 2.2.2) can integrate arbitrarily many different forms of evidence into their modeling, including evidence based on content, interaction

[2]For example, Twitter disabled chronological ordering of tweets in 2016, and only recently re-enabled this feature on an opt-in basis.

behaviour, and network based views. Relevance of evidence can be learned in a data driven way, and personalized for individuals and groups.

Finally, there is already evidence that this approach is effective. Recent work has shown that recommender systems enhanced with the predictions of a trust model can outperform recommender systems that lack this enhancement [68, 22]. This is true in both message and item recommendation tasks.

1.3 Applying Trust Models to Social Networks

There are many challenges that exist in applying trust models to social networks. In this thesis we will highlight two of them, and report on projects designed to alleviate these difficulties.

1. It is difficult to quantify the outcomes of interactions between agents on social networks.

2. Personalized predictions of trust are important yet difficult to achieve.

The first problem is relevant because, as we will show in the next chapter, reasoning about whether an interaction between agents was beneficial to the agent(s) that has accepted risk for the interaction is a critical component of a trust model. This is easier to do in e-marketplaces, where the main interaction of interest is transactions and a large amount of evidence about the quality of the transaction is available (including user reviews and statistics about returns, complaints, charge backs and shipping). On a discussion based social network, we are interested in how an agent's behaviour affects the other agents in the network. For instance, we would like to know whether or not an agent has started a good discussion (e.g. one that illuminates, produces joy, educates) or a bad discussion (e.g. one that devolves into angry arguments, misinforms, or provokes). Quantifying these outcomes, when all the actions and feedback to actions is expressed in natural language, is a difficult task. In Chapter 3, we propose and evaluate an approach for quantifying the outcome of an agent's decision to start a discussion. We do this by predicting whether a discussion-starting comment contains anti-social behaviour by examining the reactions to that comment.

The second problem is relevant because, as we will argue in the next Chapter, the concept of trust must be regarded as fundamentally subjective. The giving of trust from

one agent to another implies that the giver is willing to take on some amount of risk in their interactions with the receiver - this willingness is based on a myriad of personal factors (including the giver's sensitivity to risk). Therefore, when models of trust formulation are designed, it is important to include an aspect of personalization. A "one size fits all" approach to trust prediction, where a single model of trust formulation is applied to all agents in a network, is essentially ignoring the fact that trust is subjective. In Chapter 4, we experiment with incrementally personalizing a multi-faceted trust model. We do this by adding an unsupervised clustering step before trust formulation models are fit, and learning a distinct model for each cluster of users. This approach allows groups of similar users, who potentially express trust in similar ways, to have a model fit for their community, rather than receiving trust predictions that have been smoothed out to apply well to the entire population of the network.

1.4 A Note On Structure

This thesis is organized into four main chapters.

In Chapter 2, we will give a full treatment of the concepts of trust and trust modeling, including a description of a syntax to reference the various important concepts in a multiagent system with respect to trust. We will also survey a number of influential and important trust models, and provide descriptions of the main machine learning techniques used in the following chapters. In Chapters 3 and 4 we will describe the projects we have undertaken with the goal of alleviating the difficulties we have outlined with respect to applying trust models to social networks[3]. These chapters form the main body of the thesis. In Chapter 5, we discuss the value of our particular models. First, we describe one existing solution that inspired us to design our novel directions for applying multiagent trust modeling towards the improvement of social networks. We then compare our work to various approaches of other researchers, and conclude with a philosophical discussion about three outstanding concerns in the trust modeling and message recommendation research areas. Finally, in the Conclusion, we will summarize our works and propose a number of new directions for future work.

This thesis also contains a number of appendices. Appendix A contains extended definitions of multiple "trust-like" concepts that are often confused. Appendix B contains a list of interesting comments from a Reddit data set which our solution indicated were highly likely to contain hateful speech. Appendix C explains our methods for reducing the

[3]Preliminary versions of these works appear in [55, 56].

computational cost of predicting trust in a potentially fully connected network. Appendix D contains a list of symbols.

Chapter 2

Background

This chapter describes background information necessary to understand the works that we present in Chapter 3 and Chapter 4. First, we broadly describe the field of multiagent systems then move into a description of trust modeling. In particular, we describe trust modeling in multiple steps, first in the abstract, then through a summary of previously influential models, and ending with a comparison and contrast with the field of recommender systems. Finally, we describe, in brief, a number of machine learning based concepts used in the works.

2.1 Multiagent Systems

This thesis primarily concerns trust modeling, which must be considered as a subfield under the field of multiagent systems (MAS). A MAS is a set of intelligent entities (agents) who are connected [82]. This means agents may observe their environment and act rationally in response to their environment in ways which may be observable to the other agents. Agents may be humans, intelligent software, or human-software teams. These agents may be attempting to cooperate, or they may be in competition with each other. Under the former case, problems of coordination and balancing of individual and group goals become paramount. Under the latter case, game theory and modeling of the other agent's goals and behaviour become important.

In this thesis, we are concerned with modeling online social networks as multiagent systems. Social networks may be seen as existing in between the extremes outlined above: human and independent software agents co-mingle, and cooperation and competition exist

in parallel. Trust modeling can be seen as a process that agents perform when tasked with enhancing human decision making in this environment. They do so by processing more data than the human agent can, and making recommendations about which other agents in the network are trustworthy for future interactions.

2.2 Trust Modeling

Here we describe trust modeling at a high level, then provide a rigorous syntax for referring to the different elements of a trust model, then redefine the high level goals in terms of the syntax (which will be used throughout this document).

As we've stated, trust modeling is a subfield in the field of multiagent systems. Broadly, trust models process the behavior of connected agents in an environment with the goal of predicting directed trustworthiness between pairs of agents. Trustworthiness can be based on implicit factors (e.g. the goals and preferences of the two agents appear to be aligned) or explicit ones (e.g. one agent has produced explicit feedback indicating that they trust the other). Typically, trust models can be thought of as collecting data about the explicit behaviors and relationships in a multiagent system in order to identity cases where implicit trustworthiness exists. Phrased differently, the trust model seeks to recommend agents who ought to be trustworthy to a user/agent who is seeking an agent to interact with.

The growth of trust modeling as a research field has largely coincided with the growing popularity of the world wide web [60]. As the number and size of online multiagent environments have grown, so too has the need for methods to predict the trustworthiness of the vast numbers of semi-anonymous agents in these environments. A domain of particular interest, especially for early trust models, was e-commerce platforms, similar in description to eBay and Amazon. The situation where an anxious buyer must choose between a large number of opaque vendors has inspired a number of models [38, 77, 87, 14]. More recent application domains include autonomous vehicle networks [13] and social networks [68, 12].

The high level description above poses a number of questions, chief among them: what exactly do we mean by trustworthiness? While we acknowledge the pioneering work of Castelfranchi and Falcone in framing a definition for multiagent trust [10], we will adopt the cross-domain definition of trust presented in Cho et al. [14].

> Trust is the willingness of the [truster] (evaluator) to take risks based on a subjective belief that a trustee (evaluatee) will exhibit reliable behavior to maximize the [truster's] interest under uncertainty (e.g., ambiguity due to conflicting evidence and/or ignorance caused by complete lack of evidence) of a

given situation based on the cognitive assessment of past experience with the trustee.

Thus, we define "trustworthiness" as the quality of being worthy of trust, as defined above. That is, an entity is trustworthy when a subjective assessment of that entity's qualities and behaviors indicate that the expected benefits of interacting with that entity outweigh the risks of the interaction in question. There is often confusion between the notions of trust and other desirable qualities in an agent. To clarify our position and differentiate these concepts, we have included a brief discussion contrasting the notions of trust, popularity, reputation and credibility in Appendix A.

Here, we introduce a grammar to more accurately describe the situation through the particular lens of multiagent systems, and restate our goal of predicting trustworthiness using this syntax. All variables introduced in this section are also available for reference in the symbol list (Appendix D).

Let A be the set of connected agents in an environment. Agents are assumed to have private goals which may be in conflict. These goals may not be known, but agents have observable behavior and attributes.

The observable attributes of an agent are dependent on the specifics of the environment agents are interacting in. We will define a set of functions for referring to domain specific attributes of agents as these domains are defined. For example, if an agent $a_i \in A$ has 3 friends in a social network, we may write $friendCount(a_i) = 3$.

The semantics of trust relationships among humans implies that trust is asymmetric[1][9]. Thus, it is necessary to unambiguously refer to both ends of the trust relationship. We use the terms "truster"[2] and "trustee" to refer to the giver and receiver of trust respectively. That is, if a_i trusts their auto mechanic a_j, then a_i is a truster and a_j is a trustee. When we wish to emphasize that an agent, a, is acting as a truster, we will use the syntax $\overrightarrow{a}$. Accordingly, we will use the syntax $\overleftarrow{a}$ to emphasize an agent's role as a trustee. The direction of the arrows above the variable combined with left-to-right reading of text is intended to illustrate the direction trust is flowing, either out of the agent ($\overrightarrow{a}$) or in to it ($\overleftarrow{a}$).

The trust model is primarily concerned with interactions between agents. We refer to these interactions as "events". An event can be represented as an ordered tuple that contains one or more agents, a context, and an outcome.

$$e = \langle a_1, a_2, ...a_n, c, o \rangle$$

[1]More will be said about the semantics of trust relationships in Section 2.2.3.

[2]Sometimes spelled "trustor", e.g. in [14] and [10].

An event encapsulates all details of an interaction between agents which are relevant to the trust model. For example, let $e_1 = \langle \overrightarrow{a_j}, \overleftarrow{a_k}, \texttt{PURCHASED:CAMERA}, 2/5 \rangle$. This event may describe the interaction in which $\overrightarrow{a_j}$ purchased a camera from $\overleftarrow{a_k}$, and recorded their satisfaction with the transaction as two out of five possible stars. Alternatively, consider $e_2 = \langle a_j, \texttt{COMMENT}, -3 \rangle$, which may describe the situation where a_j posted a public comment, which received 3 more reactions of disapproval than approval. Specifying only one active agent, a_j, in e_2 is shorthand for an event where all other agents in A are capable of taking part in the event, or where the set of other agents who participated in the event is unknown or unimportant.

We use the functions $\text{out}(e)$ and $\text{con}(e)$ to refer to the context and outcome of events. In the common case where an event has clearly defined truster(s) and trustee(s), we will use the functions $\overrightarrow{\text{tr}}(e)$ and $\overleftarrow{\text{te}}(e)$ to refer to them, respectively. When a context or an outcome is a nested structure with multiple members, we will use the "dot" notation common in programming languages to navigate this structure. For example, if an event context describing a transaction consists of a time, a product type and a price, we will use, for example, $\text{con}(e).\textit{price}$ to refer to the price.

It is clear that the concrete definitions of the space of contexts and outcomes are once again dependent on the domain in which the model is intended to be deployed, and the examples given above are merely illustrative. However, the event structure outlines a number of important properties of trust models. For one, it emphasizes that trust models are employed in examining previous behavior of agents in order to predict likely future behavior, that is, there is an assumption that agents are acting rationally with respect to some goal and that useful patterns can be extracted from their behavior. Second, it underlines the fact that the outcome of interactions is a critical element for trust models (emphasized importantly in [71]). It is necessary to know whether a truster was helped or harmed by their interaction with a trustee in order to judge whether the trustee ought to be considered worthy of future trust. Finally, the event structure makes clear the context-bound nature of trust. It is incorrect to assert that because an agent is a trusted mechanic, then they must also be a trusted babysitter. Clearly defining and considering the context(s) of trust that are relevant in an application domain is an important aspect of defining a trust model[3].

As most trust models are typically computed at a fixed point in time, the total set of known events up to the point the model is computed can be referred to as E.

Above, we stated that a trust model "seeks to recommend agents who ought to be

[3]While this contextual nature of trust has been asserted previously [37], practical systems for dealing with multiple contexts are an active research area [83, 1].

trustworthy to a user/agent who is seeking an agent to interact with", and admitted that this was vague. We can now be more specific. A trust model models a distribution over agents, contexts and outcomes, $P_{A,E}$, which is derived from the set of known agents (including their attributes) and previous events. When a truster, $\overrightarrow{a_i}$ wishes to engage in some kind of future interaction with context c that requires the participation of another agent, $\overleftarrow{a_j}$, (e.g. making a purchase, accepting a news article recommendation), the trust model recommends an agent with a high probability that the ensuing event will have a "good" outcome. That is the trust model estimates the probability:

$$P_{A,E}(o = good|\overrightarrow{a_i}, c, \overleftarrow{a_j}) \tag{2.1}$$

for all relevant trustees, and recommends the agent or agents who are most likely to produce a positive outcome for the truster. For a trust model, an agent $\overleftarrow{a_i}$ is *trustworthy* for $\overrightarrow{a_j}$ under context c if Equation 2.1 is high. If we imagine that trustworthiness is measured on a closed interval from $[0, 1]$ where 1 is "very trustworthy" and 0 is "very untrustworthy" then:

$$\mathrm{T}(\overrightarrow{a_i}, \overleftarrow{a_j}, c) = P_{A,E}(o = good|\overrightarrow{a_i}, c, \overleftarrow{a_j}) \tag{2.2}$$

where the T function stands for "trustworthiness". A trust model is working well when the predictions of the distribution it derives, $P_{A,E}$, are often correct. That is, when good outcomes are predicted for future events, those events tend to have good outcomes when/if they actually occur.

Here again, we have left the exact specifications of the outcome and context elements of an event vague. We have merely insinuated that some outcomes can be regarded as more desirable than others (i.e. "good" outcomes). We propose that this assumption holds in situations where trust modeling is useful (otherwise, if all outcomes were equally desirable, why bother recommending any particular agent over another?). Equation 2.1 is based upon a binary event outcome. Of course, as we've argued that the definition of event outcomes will vary based on implementation domain, it is worth considering how the equation must change if outcomes are non-binary (e.g. a 5-star scale) or even continuous. Here we can simply point out that, so long as outcomes are ordered with respect to desirability, it suffices to instead maximize the expected outcome of the above equation. Thus, a more general (but not much more illuminating) version of Equation 2.2 might be:

$$\mathrm{T}(\overrightarrow{a_i}, \overleftarrow{a_j}, c) = \mathbb{E}_{P_{A,E}}[O|\overrightarrow{a_i}, c, \overleftarrow{a_j}] \tag{2.3}$$

where O is the random variable corresponding to event outcome and $\mathbb{E}_{P_{A,E}}$ is the expectation with respect to $P_{A,E}$. Once again, the implication is that the most trustworthy trustee is the agent that maximizes the T function. Note, Equation 2.3 bears a good deal

of similarity to the solution developed in the Beta Reputation System [38], which we will discuss below.

While not all trust models motivate their methodology in terms of the probabilistic approach implied by here, we propose that this framework is largely accurate for describing the high level goals of the trust modeling field. In the next subsection, we will provide an overview of some influential trust models and state their goals in terms of this framework.

It must be noted that an important quality of a trust model is that it produces novel recommendations. Specifically, it does not suffice to simply recommend that agents interact with only the other agents for which they have already explicitly stated their trust. A trust model can be compared, at a high level, to a recommender system (e.g. like the systems Amazon uses to recommend new items to purchase) that recommends agents rather than content[4]. Specifically, it is important that a trust model can help trustees to discover new agents to interact with profitably and steer them away from interaction partners who might harm them.

As prediction of directed trust among members of an online community is closely related to the field of social network link prediction [27], we will use the term "trust link" to describe the case when a relationship of trust is either (explicitly) stated or (implicitly) predicted between agents. In certain domains, the asymmetry of trust relationships is ignored, and all valid trust relationships are considered to be mutual (e.g. when the mutual friend relationship on a social network is taken to constitute a trust relationship). In these cases, the term trust link is quite natural.

2.2.1 Some influential trust models

In this section, we provide an overview of three influential trust models, describing their basic functioning, and showing how their operations fit into the syntax we have described above.

The Beta Reputation System (BRS)

The BRS, developed by Jøsang and Ismail [38], is an early and influential trust model targeted at e-marketplaces. BRS uses the beta probability distribution (Equation 2.4) to

[4]We have more to say about the similarities and differences between trust models and recommender systems in Section 2.2.3.

model expected outcome of future events based on historical outcomes. The probability density function of a beta distribution, $B(\alpha, \beta)$, is defined as:

$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{2.4}$$

where Γ is the gamma function:

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx \tag{2.5}$$

This function is used to represent the "reputation" (i.e. trustworthiness) of an agent. Specifically, let r_i and s_i be the respective sums of reports of positive and negative experiences with agent a_i from other agents in A. Then, under (simplified) BRS, a_i's reputation (trustworthiness) is:

$$\phi(r_i, s_i) = \mathbb{E}[B(r_i + 1, s_i + 1)] \tag{2.6}$$

Note that the expectation of a beta distribution can be easily calculated, therefore:

$$\phi(r_i, s_i) = \mathbb{E}[B(r_i + 1, s_i + 1)] = \frac{r_i + 1}{r_i + s_i + 2} \tag{2.7}$$

The value of $\phi(r_i, s_i)$ is interpreted as the probability[5] of a satisfactory future interaction with a_i.

Extensions to Equation 2.6 are proposed for combining feedback from multiple parties, discounting feedback (weighing reports from a peer based on the reputation of that peer) and forgetting feedback (decreasing the weight of old reports).

Seen through the framework defined above, BRS defines the space of events to be generic transactions between e-marketplace participants, where context is kept static and outcomes correspond to the reviews left by buyers. An agent's trustworthiness is simply the expectation of the Beta distribution parameterized by the sum of positive and negative feedback that agent has received. Thus, when translated into the event model described above:

$$T_c(\overrightarrow{a_i}, \overleftarrow{a_j}) = \phi(r_j, s_j)$$

[5] As α corresponds to reports of positive experiences, and β corresponds to reports of negative experiences, it follows that the expectation of this Beta distribution will improve (i.e. the reputation of the user will improve) when α is high and β is low.

Where:

$$r_j = \sum_{e_i \in E} \mathbb{1}(\overleftarrow{\text{te}}(e_i) = a_j) \cdot \text{out}(e_j).pos$$

$$s_j = \sum_{e_i \in E} \mathbb{1}(\overleftarrow{\text{te}}(e_i) = a_j) \cdot \text{out}(e_j).neg$$

Where $\text{out}(e_j).pos$ and $\text{out}(e_j).neg$ are real number submitted by the truster of event e_j (i.e. the buyer) which describe the mix of positivity and negativity they have expressed with respect to the event (i.e. the transaction).

Personalized Trust Model (PTM)

The PTM developed by Zhang and Cohen [87] can be thought of as an extension of the BRS that combines evidence from personal experience with reports received from peers (public information). This system assumes that agents exist in a system where they are frequently advising each other about whether other agents are trustworthy or not. The problem then becomes to determine how to combine advice from multiple advisers, given that these advisers themselves may be untrustworthy.

Under this model, when agent $\overrightarrow{a}$ is considering advice from adviser $\overleftarrow{b}$, a considers both a private notion of reputation and a public one when deciding whether or not to trust b's advice. In particular, b's private reputation according to a, $R_{pri}(a, b)$, is modeled by the expectation of a beta distribution (see Equation 2.7 above) where α is the number of times a and b have agreed in the past about the reputation of other agents, and β corresponds to how many times they have disagreed. The public reputation of b, $R_{pub}(b)$, is again modeled by the expectation of a beta distribution, where α corresponds to the number of times b's advice has agreed with majority opinion, and β the number of times it has not.

The final reputation of b for a is then a linear combination of the private and public reputation of b, weighted by how much comparable experience a has had with b (i.e. the number of agents commonly rated agents).

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

$$w = \begin{cases} \frac{N^b_{all}}{N_{min}} & \text{if } N_{a,b} < N_{min} \\ 1 & \text{otherwise} \end{cases} \tag{2.9}$$

$$T(a, b) = wR_{pri}(a, b) + (1 - w)R_{pub}(b) \tag{2.10}$$

Where ϵ is an error bound, γ is a confidence measure and $N_{a,b}$ is the number of times a and b have advised on the same targets. As can be seen, this equation allows a trustee to incorporate more personal experience with an adviser into their reasoning gradually and according to their own preferences for error bounds, eventually allowing them to form a completely private opinion of the adviser. The truster can then combine the advice from multiple advisers (again, using a beta distribution), weighing the relevance of their advice based on their reputation, in order to form a final belief about another agent (e.g. evaluating a seller based on reports from buyers).

For PTM, the context of trust could potentially be considered fluid (e.g. asking for different advice under different circumstances). The events of interest are instances of advice being given. For the private reputation function, a three agent event is considered, where a and b have both given advice on c. The outcome of this event is positive if a and b agreed on their assessment of c, and negative otherwise. The public reputation function considers the double-agent event where b gives advice on agent c. The outcome is positive if b's advice is in line with consensus in the community (e.g. the mode of advice from all other agents) and negative otherwise.

Bayesian Learning to Adapt to Deception in E-Marketplaces (BLADE)

The BLADE system, developed by Regan et al. [59], is another model aimed at evaluating the advice of other agents in an e-marketplace situation. Under this model, each agent acting as a seller of goods is assumed to have a set of features that they exhibit in their transactions $F^s = \{F_1^s, ..., F_k^s\}$. Each of these features can take on a finite set of discrete values: for example, F_1 might correspond to the item being shipped on time, and can take on values *late* or $onTime$. Since these features differ from transaction to transaction, a multinomial distribution parameterized by θ_i^s is associated with each feature of each seller. As a buyer's satisfaction is expected to be a function of the features of a transaction, the buyer's goal is then to learn the probabilities of each feature of each relevant seller and choose a seller who maximizes the expected outcome of their own utility function.

When a buyer does not have enough prior history with a seller to accurately estimate these probabilities, BLADE describes a system for combining advice from other agents. BLADE assumes that each adviser has a private reporting function, that is, their choice to recommend or not recommend a seller is based on their own personal function of the features of the transaction they experienced with that seller. This private function is also modeled as a multinomial distribution that is conditioned on the set of features actually observed in a transaction. A Bayesian network for estimating the features of sellers and reporting functions of advisers in parallel is proposed. This learning of adviser reporting

function is conjectured to handle deception and mismatched priorities in a principled way, and should indeed be effective so long as adviser reporting behaviour is consistent.

The real strength of the BLADE approach is that an agent, a, can make use of reports from another agent, b, even if there are important subjective differences in how they judge other agents. Learning the evaluation functions of advisers as the estimates of the qualities of sellers are refined is what makes this possible.

As this model is concerned with e-marketplaces, like the previous models, the events of interest are transactions and buyer reports of satisfaction. Each event consists of a buyer, a seller, and a report of either satisfaction of dissatisfaction from the buyer.

2.2.2 Multi-faceted trust modeling

Multi-faceted trust modeling (MFTM) is a flexible and data driven approach to trust modeling. Inspired by work in the social sciences which have outlined the numerous variables which influence the formation of trust relationships [51], MFTM incorporates arbitrarily many indicators of trustworthiness into a single (optionally context-dependent) trustworthiness score. Operationalizing this core idea for trust and social tie prediction has been proposed by multiple researchers (e.g. [45, 27, 36, 22, 50]). As is evident in these works, there is little agreement over whether this technique should be called multi-dimensional, multi-faceted or composite trust modeling, and this confusion has likely led to some difficulty in coordinating efforts in this research direction. We use the term "multi-faceted", in keeping with the most recent works.

The defining feature of an MFTM is a customizable vector of trust indicators, where each indicator is a real number based on two agents:

$$\boldsymbol{\Psi}(a_1, a_2) = \langle \psi_1(a_1, a_2), \psi_2(a_1, a_2), ..., \psi_n(a_1, a_2) \rangle \tag{2.11}$$

A "trust indicator" can be thought of as a piece of evidence for or against trusting an agent under a particular context. For example, $\psi_1(a_1, a_2) = friendCount(a_2)$ may be relevant to assessing the reputation of a_2 in a domain where only popular and reputable agents can accrue large numbers of friends. The indicators ψ_i must be computable given A and E (the set of agents and their attributes and the history of events). One important feature of MFTM is its flexibility to tune its parameters to different domains. The customizability of MFTM is highly attractive for application to social networks, as it is rare to find explicit statements of trust encoded into the feature set of online environments[6]. Instead, an arbi-

[6]For example, social network designers could elicit explicit statements of context-bound trust from users, but such a feature is not currently popular online.

trary number of "imperfect" indicators of trustworthiness, such as popularity, friendship, reputation, interaction history, preference similarity and institutional credibility can be considered as each contributing to a final tally of trustworthiness. Clearly, the underlying assumption of this model is that the existence of trustworthiness between two agents can be predicted based on a comparison of the attributes and behaviors of those agents.

As we've mentioned, the consideration of multiple indicators of trust is in emulation of the way in which humans consider multiple sources of evidence when deciding to trust or not [51]. For example, consider the problem of choosing an auto mechanic shortly after having moved to a new town. In this case, one has no interaction history with any nearby mechanics and must weigh available evidence in order to choose which mechanic to trust. In a simple case, one might only consider two pieces of evidence towards or against a mechanic: has any colleague recommended them (ψ_1), and have their prices been posted clearly online (ψ_2). In this case both indicators are binary, and it seems likely that the mechanic a_j who satisfies both indicators, $\Psi(\overrightarrow{a_i}, \overleftarrow{a_j}) = \langle 1, 1 \rangle$, will be a good candidate to trust.

In order to predict trustworthiness, the relevance of each indicator can be learned using an off-the-shelf machine learning technique given A and E to train with. To do this, a trust link is chosen as a target of prediction : y (e.g. explicit statements of trust/friendship, high degrees of preference alignment). Then, given the set of existing implicit/explicit trust links, a machine learning model fits a classifier $\hat{f}$ to the function that determines how trust indicators are related to trust links $f : \Psi(a_1, a_2) \rightarrow y$. For example, in the case where logistic regression (see Section 2.3.1) is used, y will be binary and we have:

$$T_c(\overrightarrow{a_1}, \overleftarrow{a_2}) = P_{A,E}(\overrightarrow{a_1}, \overleftarrow{a_2}, c) = \frac{1}{1 + \exp^{-(\theta \cdot \Psi(a,b))}} \tag{2.12}$$

where θ is the vector of weights learned through the logistic regression process and T_c is trustworthiness under context c. We wish to emphasize that while logistic regression is an elegant and natural choice with some popularity in the literature (e.g. [22]), it is by no means the only choice.

The ability to define custom indicators appropriate to whichever application domain one is pursuing offers a tremendous amount of flexibility. As we will show in Chapter 4, both highly generic as well as application-specific trust indicators can be defined.

Finally, we wish to emphasize how MFTM can be seen as a generalization of a number of existing trust modeling techniques. Primarily this is because many trust modeling techniques do in fact consider multiple sources of evidence, but they weigh or combine this evidence in a non-data-driven manner. For example, the beta reputation system can be

configured so that old advice is considered less important than new advice. However, a method for specifying *how much more* important newer advice should be treated compared to older advice is not specified. A similar situation occurs in the Personalized Trust Model, where private and public reputation are weighed against each other. The weighting function chosen has a good statistical justification[7], but ultimately does not specify how error bounds should be chosen, and thus how exactly to weigh personal and private reputation. MFTM can consider arbitrarily many sources of information, and learns the weights for them directly from data. For example, PTM could be roughly replicated by treating private and public reputation as trust indicators, and learning an appropriate function for combining them.

Another example of how MFTM is data driven is that it does not specify which distributions should be used to model beliefs. For example, both the BRS and PTM rely heavily on the beta distribution. While this choice is statistically justified (assuming the behaviour of agents is governed by a random process[8]), it also constitutes a form of bias, and is vulnerable to abuse. For example, if one knew that the BRS was being used to model seller reputation on an e-marketplace, they could increase profits and maintain a high reputation simply by only scamming every 10th customer, or by acting honestly for every small purchase and scamming the less frequent buyers of expensive items [40]. By allowing arbitrary machine learning methods to combine many forms of trust evidence into prediction, MFTM loses Bayesian rigor, but gains a large degree of flexibility and generalizability.

2.2.3 Trust modeling and recommender systems

There is sometimes confusion regarding the differences between trust modeling and recommender systems. Indeed, there are some similarities; however, there are also fundamental differences. As our works consider both trust models and recommender systems, we will briefly outline some of these points. We first note that recommender systems largely consider similarity between the preferences of users who are not in any kind of significant competition (e.g. users on a music or video streaming platform), while trust models are usually targeted at environments where agents are somewhat adversarial and have the potential to harm each other (e.g. an e-marketplace).

We argue that trust models and recommender systems are similar because:

[7] Based on the Chernoff bound theorem.

[8] Generally it is not.

- Ultimately, both systems consider a pool of possible entities for recommendation to a human agent. For trust models, these entities are other agents for future interactions, while for recommender systems they are items to consume.

- Collaborative filtering (techniques for aggregating the feedback or preferences of many agents in order to advise one agent) is a dominant approach in both areas. To see this, consider that polling peers for advice and discounting their advice by your trust in the peer (as the BRS and PTM do) is analogous to polling your peers for movie ratings and discounting their ratings by that peer's similarity (as many recommender systems do). However, trust models usually also consider an aspect of personal experience, distinct from mere similarity [53].

- Both approaches are often applied to similar data sets, and trust models can be used in tandem with recommender systems [81, 22].

We argue that trust models and recommender systems are different because:

- Trust, as a concept gleaned from human interactions, has a number of important semantics, including asymmetry and (limited) transitivity [9, 14]. These semantics should be considered when designing a trust model. Distrust, as distinct from a lack of trust, also has unique semantics which bear consideration [10].

- Trust varies under context [14]. Recommender systems typically ignore many aspects of context, while doing so for a trust model would be inappropriate.

- Trust models present significant opportunities for abuse, which must be considered. Sophisticated cheating patterns can be employed to trick trust models [40]. Meanwhile recommender systems chiefly consider the similarity between users expressed preferences (which presents less opportunity for abuse.)

- Trust is dynamic [10]. A single abuse of trust can quickly sour trust with an agent [78]. This is very different from preferences for movies or music, which are relatively slow to change.

- Trust modeling generally places the well being of the agents it advises as a core goal. Recommender systems are often optimized for the goals of the administrators of the system they are employed on (e.g. user engagement and profits).

2.3 Machine Learning Techniques

Here we briefly overview the machine learning models we use in the projects we present.

2.3.1 Logistic regression

Logistic regression is a straightforward probabilistic binary classification machine learning algorithm. Given each point has a vector of features $x_i \in R^n$ and a binary class label $y_i \in \{0, 1\}$, logistic regression models the probability that a point x_i is in class 1 as follows:

$$Pr(y_i = 1|x_i; \theta) = h_\theta(x) = \frac{1}{1 + \exp(-\theta x_i)} \tag{2.13}$$

where θ is a vector of weights to be learned to fit a data set. This weight vector can be learned by maximizing the likelihood function:

$$L(\theta|X) = \prod_i h_\theta(x_i)^{y_i}(1 - h_\theta(x_i))^{(1-y_i)} \tag{2.14}$$

This function does not admit an analytical solution, and needs to be optimized via gradient descent or some other iterative optimization method. In practice, this optimization is often very fast, as the number of parameters to optimize is low compared to other machine learning methods.

Logistic regression is valuable as it admits a simple probabilistic interpretation, is quick to optimize, frequently very effective, and the weight vector learned, θ, is highly interpretable. However, this method is fundamentally limited by the assumption of linearly separable data and inability to compute interesting feature combinations (data must be preprocessed if this is desired).

2.3.2 k-means clustering

k-means clustering is a simple and popular method for partitioning a set of vectors into k cohesive groups. Given a set of vectors $X = \{x_1, ..., x_n\}$ and a $k \leq n$, partition the vectors into k sets $S = \{s_1, ..., s_k\}$ such that the within cluster sum of squares is minimized. Formally:

$$\arg\min_S \sum_{i=1}^{k} \sum_{x \in s_i} ||x - \mu_i||^2 \tag{2.15}$$

where μ_i is the mean of the vectors in s_i. Restated, the goal is to find the k groups of elements from X such that the sum of the squared distances between each point in a group and the mean of that group is minimized. Finding the exact solution to Equation 2.15 is NP-hard, but many effective heuristic algorithms are available.

In Chapter 4 we will use algorithms inspired by k-means, although modified to be applicable to data that is not represented in a vector format. Note that this is possible so long as a distance function between a point and a group of points is defined to stand in for the term $||x - u_i||^2$ in Equation 2.15.

2.3.3 Feed forward neural networks

In Chapter 3 we make use of a small feed forward neural network for classification. Here we will very briefly summarize the basics of these models.

Feed forward neural networks are a simple type of neural network that are amenable to classification of fixed length input data. The building blocks of neural networks are artificial neurons: entities which receive multiple inputs, compute a linear weighted sum of those inputs, then output a function of this weighted sum[9]. When multiple layers of neurons are organized in a pipeline, where the outputs of the neurons in layer i are the inputs to the neurons in layer $i+1$, the resulting structure is a feed forward neural network. The number of neurons in a layer is referred to as the width of the layer, and the number of layers is referred to as the depth of the network. The width of the first layer must be equal to the length of input vectors, while layers in between the input and output layer ("hidden" layers) can be of arbitrary width. The final output layer should be of a width that is appropriate for the function to be approximated (e.g. one neuron for each class).

Under a supervised learning task, the weights the individual neurons apply to their inputs can be optimized via the back-propogation algorithm [62] in order to learn a cohesive network that approximates the function. Under basic assumptions, a feed forward network of arbitrary finite size can approximate the continuous relationships between independent and dependent real variables to an arbitrary degree of precision [19].

As the input sizes of these networks must be fixed, in Chapter 3 we propose a set of feature extraction functions and procedures for aggregating these features that we apply to arbitrary length data (representing online discussions) in order to produce fixed length vectors of real numbers representing the discussion. We also make use of a number of common tools in the neural network tool belt in order to construct our network:

[9]This function is called the "activation" and is usually non-linear, for purposes of optimization.

- Dropout: The process of randomly excluding outputs from a layer of neurons. 50% dropout means the results of half of the neurons on a layer are randomly excluded in each training pass. This makes the training process more noisy, preventing overfitting to the training set.
- ReLu: Rectified linear units, a simple non-linear activation function that has useful properties for optimization via backpropogation.
- Softmax: A function that normalizes a vector, allowing it to be interpreted as a probability distribution. Commonly used as the final layer of a neural network for classification problems.

2.3.4 Random forest classifiers

A random forest classifier is an ensemble learning algorithm for classification. Random forests are based on an ensemble technique called "bagging": rather than training a single, complex model on all available data, a large set of relatively weak classifiers (e.g. decision trees with very low max height, often 1-3) are each trained on a random subset of the data. After training the weak classifiers, predictions can be made by simply averaging out the predictions across the set of all classifiers. If M weak classifiers are used, then:

$$\hat{f}(x) = \frac{1}{M} \sum_{m=1}^{M} \hat{f}_m(x) \tag{2.16}$$

For a random forest, low-height decision trees are always used as the weak classifiers, and, in addition to bagging, each decision point of each decision tree is only exposed to a random subset of features (this can be called "feature bagging"). This forces some of the weak classifiers to train without the advantage of the most predictive variables, allowing weaker correlations in the data to be modeled (this fixes the problem where a decision stump exposed to all features would always split on the most predictive feature).

It is possible to interpret feature importance in a random forest. For example, one can calculate the the Gini impurity of a node in a decision tree as follows:

$$G = \sum_{i=1}^{C} f_c(1 - f_c) \tag{2.17}$$

Where C is the set of classes at a node and f_c is the frequency of that class at that node. A feature's importance can be taken as the average decrease in impurity from nodes that test

on that feature and the children of those nodes, weighted by the probability of reaching the nodes. That is, the importance of node i, given that it's children are nodes j and k is then:

$$imp_i = w_i G_i - w_j G_j - w_k G_k \tag{2.18}$$

where w_i is the weighted sum of samples which can reach node i and G_i is the Gini impurity (Equation 2.17) of node i.

2.3.5 Natural language processing

Natural language processing (NLP) is an extremely large and diverse field which we do not endeavor to fully summarize here. However, having used some techniques from this field in Chapter 3, we will briefly introduce it.

The central supposition underlying the statistical approach to NLP is that there are significant statistical relationships between word usage and ordering in phrases and the meaning of those phrases [49]. These relationships are expected to be strong enough that useful language models can be constructed without formally specifying any language rules (such as grammar or syntax). This approach, once highly controversial, has been significantly buoyed by the advances in deep learning over the past ten years.

In Chapter 3, we use some very basic statistical NLP techniques to extract features from comments. For instance, we tokenize (split into distinct words) comment text and count the number of pronouns of each degree, as well as profane and hateful words, punctuation marks, etc. We also make use of pre-trained models for detecting hateful speech, profanity and sentiment. The dominant approach in the models we've used is to represent a text as a bag-of-words vector: a vector of fixed length where each index corresponds to a word, and the value at a given index is the number of times that word appears in the text. These bag-of-word vectors are then used to train linear models (such as SVMs) on hand labeled data sets, producing a classifier.

2.3.6 Latent factor models

Latent factor models for recommendation are a popular approach to collaborative filtering based recommendation derived from matrix factorization technique called Singular Value Decomposition (SVD) [69]. Specifically, by applying an SVD technique, a $m \times n$ matrix R of rank ℓ can be decomposed into three matrices of rank $k \leq \ell$:

$$R_k = Q \cdot S \cdot V$$

Where Q is $m \times k$, S is $k \times k$ and V is $k \times n$. While S has a number of interesting mathematical properties, in recommender system literature it is is frequently ignored by substituting $U = Q \cdot S$:

$$R_k = U \cdot V \tag{2.19}$$

This decomposition is guaranteed to exist and provide the best rank-k approximation of the matrix R with respect to the Frobenius norm [69]. That is, $R_k = U \cdot V$ is the matrix that minimizes $||R - R_k||_F$ where $||A||_F = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2$. Phrased otherwise, this decomposition procedure can be used to "compress" a large matrix by splitting it into the two rank-k matrices which, when multiplied together, best reproduce the original matrix.

SVD can be applied to recommender systems when R is the user-item matrix of review scores such that r_{ij} is the rating user i gave to item j (e.g. on a binary "recommend or not" scale, or on an ordinal 1-5 rating system). Naturally, this matrix is sparse - in practice, the vast majority of the entries in R are unknown, as most users have only given feedback on a small number of items. While SVD cannot be applied directly to a sparse matrix like R, we can imagine that the defined entries in R comprise a subset of the entries in the (unknown) dense matrix R' where every user has expressed an opinion on every item. By SVD, R', is guaranteed to have a minimal rank-k decomposition. This line of reasoning serves as inspiration for the following loss function [43]:

$$\min_{u_*, v_*} \sum_{(i,j) \in \kappa} (r_{ij} - u_i^T v_j)^2 + \lambda(||u_i||^2 + ||v_j||^2) \tag{2.20}$$

where u_i is a length k vector corresponding to user i and v_j is a length k vector corresponding to item j, and κ is the set of indices (i, j) such that r_{ij} is defined in R. λ simply controls the strength of the regularization penalty. By optimizing Equation 2.20, one constructs matrices $\hat{U}$ and $\hat{V}$, where the i'th row of $\hat{U}$ is u_i^T and the j'th column of $\hat{V}$ is v_j. Then, $\hat{R} = \hat{U} \cdot \hat{V}$ is a matrix where the distance between defined members of R and their corresponding entries in $\hat{R}$ has been minimized. At the same time, estimates for every undefined entry in R are present in $\hat{R}$. A user i can then be recommended items where r_{ij} is undefined (the user has not yet rated the item) but $\hat{r}_{ij}$ is high (the user is predicted to rate the item highly).

This approach is particularly amenable to the recommendation task, as it makes the optimization far more tractable. In particular, rather than grappling with the $O(mn)$ user-item ratings directly, the $O(k(m + n))$ values in $\hat{U}$ and $\hat{V}$ are all that need to be optimized. This offers considerable performance improvements when $k << min(m, n)$ (in many applications there may be millions of users and items, but $10 \leq k \leq 100$ factors are sufficient for good modeling of the system [43]). Additionally, the running time of a single loop of the optimization equation is linear in the number of observed ratings (κ above).

Koren et al. [43] describe the intuition behind this procedure in illuminating way. For the task of recommending movies, we can imagine that each movie can be measured on k dimensions. For example: funniness, seriousness, amount of action, quirkiness, etc. Each user will have some level of preference for various dimensions of a movie. A user will enjoy a movie when the movie has high extension in the dimensions the user enjoys: this user prefers a mix of comedy and quirkiness, that user long dramas with character development. Rather than explicitly defining these k dimensions and laboriously categorizing each movie in this way, the SVD recommendation procedure infers factors directly from rating patterns. These so called "latent factors" may represent well known categories and generalizations such as those listed above, or qualities that defy description. By learning k-length vectors for users and movies respectively via error minimization over available data (such that $u_i^T v_j$ is close to r_{ij} as described in Equation 2.20), the preferences of users and qualities of movies across k latent factors is learned.

While the actual distance between R' and $\hat{R}$ is unknowable, this approach has been shown to perform well in numerous settings, including winning an international contest hosted by Netflix [43]. Further, minimizing Equation 2.20 has been shown to be equivalent to maximizing the probability of latent factor matrices U and V given R under a simple probabilistic model [63].

In Chapter 4 we will use a latent factor model, TrustMF [84], to test the accuracy of a trust link prediction algorithm. This system can be roughly characterized as combining the optimization described above with an optimization over a matrix of user-user trust links that shares a latent space with the user-item matrix. That is, in addition to learning matrices $\hat{U}$ and $\hat{V}$ describing user preferences and item factors respectively, a third matrix $\hat{W}$ representing user factors is learned. The matrix $\hat{U}$ now serves a dual purpose. Like before, the distance between r_{ij} and $u_i^T v_j$ is minimized, but in addition the distance between l_{ij} and $u_i^T w_j$ is minimized, where $l_{ij} > 0$ only when a trust link exists between users i and j. Conceptually, a user's preference for items shares a space with that user's preferences for trusting other users. Thus, the presence of trust links exerts an influence over the latent factors that are discovered, incorporating social trust into the recommendation process. The ability to incorporate social trust data makes this recommender system "trust-aware".

Chapter 3

Detecting Anti-Social Behaviour Based on Community Feedback

As argued in Section 2.2, one the the critical components of a trust model is the ability to interpret the outcomes of events that occur among agents. Specifically, one needs to know whether an interaction between agents was helpful or harmful to each of the agents in question. On e-marketplaces this is relatively straightforward, as the main interactions of interest between agents are the trading of real-world items, and feedback on these transactions is heavily encouraged by the site owners. On social networks, the situation is less clear. Most of the interactions on social networks are conversations represented as text, and when unambiguous feedback is elicited from agents, it is often in the form of up or down votes or likes. These indicators are useful for measuring popularity, but, as we will argue, they present a distorted picture of the relative quality of agent actions when they are taken as a sole measure of quality.

As the ability to quantify outcomes of interactions is key for the application of trust models to social networks, in this chapter we present novel methods for quantifying the outcome of an agent's action of posting a comment. First, we conjecture that anti-social behaviour (e.g. hate speech, bullying, trolling) is undesirable, and therefore agents that engage in this behaviour are producing negative outcomes for those that interact with them. Our hypothesis is that as anti-social behaviour in comments is expected to produce a reaction with certain patterns from the community (e.g. anger, name-calling, offense), it may be possible to predict whether an agent has breached social rules simply by observing the reaction to that agent's comment from the community. We test this hypothesis using data from the popular social link sharing site Reddit. We do this by evaluating the accuracy of a prediction task based on categorizing whether a conversation-starting comment contains

certain types of anti-social behaviour based solely on the reactions to that comment from other members of the community. Specifically, using a set of natural language and meta-data based features extracted from the replies to a comment, we evaluate the accuracy of predicting the presence of hate speech, profanity, negative sentiment and negative scores in the comment that prompted the replies.

3.1 Motivation

As discussed above, a substantial barrier for the application of trust models to social network is the problem of extracting data which quantifies the outcomes from the interactions between agents. In a social network, the main event of interest is the conversations between agents. Unlike in many online networks, agents in social networks are rarely encouraged to distill their feedback about a conversation into some set of ordinal or categorical variables - instead, the bulk of feedback is tied up in complex natural language replies. At best, agents can up or down vote a comment. In some cases (like on Facebook), they can only submit positive feedback. This up or down vote feedback system allows each agent to express a single bit of feedback towards each comment on the site.

On first examination, the quantity of positive feedback minus negative feedback (score) a comment received may seem to be a very rich source of information with respect to the quality of the comment. However, there are some caveats. On sites which implement this type of feedback system, the up or down votes typically affects the sort order of comments when they are displayed to other agents. Comments which have received large numbers of up-votes are displayed first, while those that are controversial or have received a large number of down-votes are shown last, or even hidden. This is supposed to enrich the experience of readers, but leads to a predictable "rich get richer" effect, where comments that are up-voted early get the most attention and can thus garner more up-votes. For this reason, the *magnitude* of a comment's score (score = up-votes - down-votes) is not a particularly useful statistic for measuring the quality of a post. Much of a post's score will be the result of timing, luck, and the compounding effects of popularity as it translates to visibility: a popular post's popularity will make it more visible, thus more popular! We argue a more useful statistic for detecting low-quality or anti-social comments is the sign of a post's score. Negative scores are rare - at least on the Reddit data we used, only 15% of comments in our data set had a negative score. Thus, at least for this site, a negative score may be a strong signal that a post was "rejected" by the community (remember, negative scores cause posts to be hidden).

We have argued that the most useful feedback one can get from the up-vote or down-

vote system is a binary variable representing the sign of the score of the comment. This is clearly not enough information for a trust model to aid in weeding out comments of agents who routinely post hate speech and misinformation. We'd like our trust models to predict things more interesting than simply "what is the probability this post will be well received". We'd like to know something about misinformation, hate speech, cruelty and argument-baiting. Clearly, more information is necessary to reason about any of these topics - the text of the comment clearly needs to be investigated.

Unfortunately, current NLP based techniques may be little help when it comes to detecting many types of anti-social behaviour. While acknowledging the advances in hate speech detection (e.g. [5, 44]) that have been made, we must also acknowledge the difficulties that arise, particularly from a purely NLP based approach. In particular, cruelty and hate often involve sarcasm and oblique references to some societal context, which are extremely difficult to capture in a hate speech detection system. While it is trivial for any automated system to detect the presence of loaded and hateful terms, it is similarly trivial for human authors to make the exact same point while avoiding the obvious "red flag" words such as racial slurs or extreme profanity.

We propose that the next logical step is to consider the entire set of comments in a discussion: specifically, to focus on the *reactions* to a comment. Anti-social behaviour, such as hate speech and cruelty, is expected to provoke a patterned reaction from a community - the very notion of anti-social behaviour can be thought of as that which is "disruptive" to others [33]. It is conceivable that the task of categorizing sarcastic, mocking, hateful text is *easier* to do by looking at the reactions of others than it is to do simply by looking at the text in isolation. This conforms to the expectation set by our lived experience: when in a group of individuals speaking a language one does not understand, one can still be made clearly aware of a socially unacceptable comment simply by observing the reactions of those who do understand the language.

We focus on "anti-social" behaviour in particular for two reasons:

- Anti-social behaviour is, practically by definition, undesirable. It is not controversial to propose that a comment which offends the community has resulted in a bad outcome for those agents exposed to the comment.
- It is reasonable to expect that there is some pattern to the responses to anti-social behaviour, that when a group is offended by the bad behaviour of an individual, they react with predictable pattern of outrage/anger/disgust.

Ultimately, in this work we want to reason about when an agent is disrupting the community. We approach this task by assuming that anti-social behaviour is always disruptive,

and try to recognize the pattern in the responses to this behaviour. Knowledge of which agents routinely disrupt the community can be useful for reasoning about which agents are untrustworthy.

3.2 Data Set and Filtering

Reddit is a social link sharing site that has been in operation since 2005. Users of the site can share links, comment on the links shared by others, and engage in conversations with each other.

Reddit consists of tens of thousands of subreddits - areas of the site devoted to the sharing and discussion of links related to some particular context. For example, `/r/cooking` is a subreddit where only links and conversations relating to cooking are allowed. Reddit can be seen as a federation of subreddits - the creation and moderation of subreddits is controlled by users of the site, while the site administrators mostly limit their executive power to restricting the set of allowable subreddits - for instance, subreddits which encourage bullying, harassment and the sharing of illegal information are frequently deleted or have their visibility reduced. As the power to moderate within subreddits is mostly given over to users, a failure to adequately moderate is also grounds for banning. That is, even if a subreddit is not explicitly devoted to the sharing of content which the site administrators find objectionable, the administrators may ban the subreddit if the moderators of that subreddit prove incapable of adequately policing the behaviour of the users therein. Reddit has frequently been a subject of controversy over the last decade, largely due to the mostly hands-off approach to moderation that the side administrators take, and the friction that occurs when the administrators take action against communities that perceive themselves as being independent. Nevertheless, Reddit is one of the top twenty most popular website on the Internet as of February 2020[1] - more popular than Netflix, Instagram or Twitter.

We downloaded all comments submitted to Reddit during the month of January 2016. This data is made available via the Reddit API, and is also aggregated on websites such as https://files.pushshift.io/reddit/comments/. We filtered the data to only consider comments submitted to the `/r/politics`, `/r/movies`, `/r/worldnews`, `/r/AskReddit`, and `/r/IAmA` subreddits. While the first three of these have obvious topics of interest (discussing politics, movies and world news respectively), the latter two bear some explanation. `/r/AskReddit` allows users to, instead of sharing links, share questions that readers can

[1] https://www.alexa.com/topsites

answer and discuss in the comments. For example: "Reddit, what is your favorite sandwich recipe?". Naturally this invokes lively and spirited debate. `/r/IAmA` is a subreddit for famous or interesting people to conduct informal interviews with users. The name of the subreddit is a pun, invoking the first and last parts of the phrase "I Am A person of interest, Ask Me Anything". There are a number of reasons why we chose these subreddits:

- Each is among the top one hundred most active subreddits[2]. With the exception of `/r/worldnews`, the others are in the top twenty.
- Each has a culture that promotes vigorous discussion and debate.
- None is overly politically or ideologically biased or extreme. That is, they attract a large and varied segment of the overall user base. In fact, all of them except `/r/politics` was formerly a "default" subreddit - one of the subreddits which a new Reddit user was automatically made a member of[3].

The last point is particularly important. As our approach will involve learning the pattern of reaction to anti-social behaviour in a community, it is important to focus on relatively mainstream communities. If this approach was targeted at communities that organize around extreme or racist ideologies, the response to what *those* communities perceived as antisocial would be learned. This could obviously differ significantly from what mainstream society deems acceptable or unacceptable. We will have more to say about this limitation, and how it fits in to the broader picture of determining what is and isn't acceptable behaviour by the end of this Chapter. After filtering for these subreddits, a total of 6,873,260 comments remained.

3.2.1 Discussion filtering

Further filtering was done based on language and the size and shape of discussions that a comment was involved in. In order to explain this step, we will first present a simple grammar for referring to elements of a discussion on Reddit.

A Link is a piece of content that is submitted by a user (human agent) to be shared within a subreddit. A Link is not always a hyperlink - plain text is allowed. A Comment is a piece of text submitted in response to either a Link or another Comment. Both Links and Comments can receive up or down votes, influencing the order in which they are presented

[2] http://redditlist.com/
[3] https://reddit.fandom.com/wiki/Default_subreddit

to users. All registered users can up or down vote each Link or Comment once. The Score of a Link or Comment is equal to up-votes received minus down-votes received.

Evidently, Links and Comments form trees of arbitrary branching factor and depth, where the root of the tree is always a Link and each node in the tree has a piece of text and a Score associated with it. This tree-like structure for discussions has become more popular in recent years online[4], replacing the simpler list structure of earlier online discussion forums.

We will use the following terminology to further distinguish elements of a conversation on Reddit.

- A **Parent** Comment is any Comment that has been replied to by another Comment.
- A **Child** Comment is any Comment which is replying to another Comment.
- The **Descendants** of a Comment are all the Comments in the subtree rooted at that Comment. For example, the Children of a Parent are part of its Descendants, as are the Children's Children, etc.
- A **Discussion** is a tuple formed by a Parent, its Children and its Descendants.

Just as in our common understanding of the terms, a Child and also be a Parent. This illustrates that we are interested in Comments at all levels of the tree, so long as that comment is at the root of a non-empty sub-tree (i.e. someone replied to them). Unlike our common understanding, not all Parents are Children: some Parents are submitted in reply to Links, which we do not consider to be a Child. Some Comments don't fit into any of the above categories. For example, some Comments are submitted in reply to a Link, but never have any Children. These Comments have likely been ignored or looked-over.

An illustrative Reddit conversation is presented in Figure 3.1, and a short excerpt from a real conversation on Reddit[5] showing a Parent, one of its Children and two of its Grand Children is presented in Figure 3.2.

We can now explicitly state that our work is interested specifically in Discussions rather than individual Comments. In particular, we wish to evaluate the accuracy of various prediction tasks aimed at predicting qualities of a Parent comment given only the features of its Descendants (as a reminder, this includes Children).

[4]Facebook, Youtube, and Twitter all use a similar structure

[5]This conversation occurred in the context of a user asking for interesting science facts to share with their daughter.

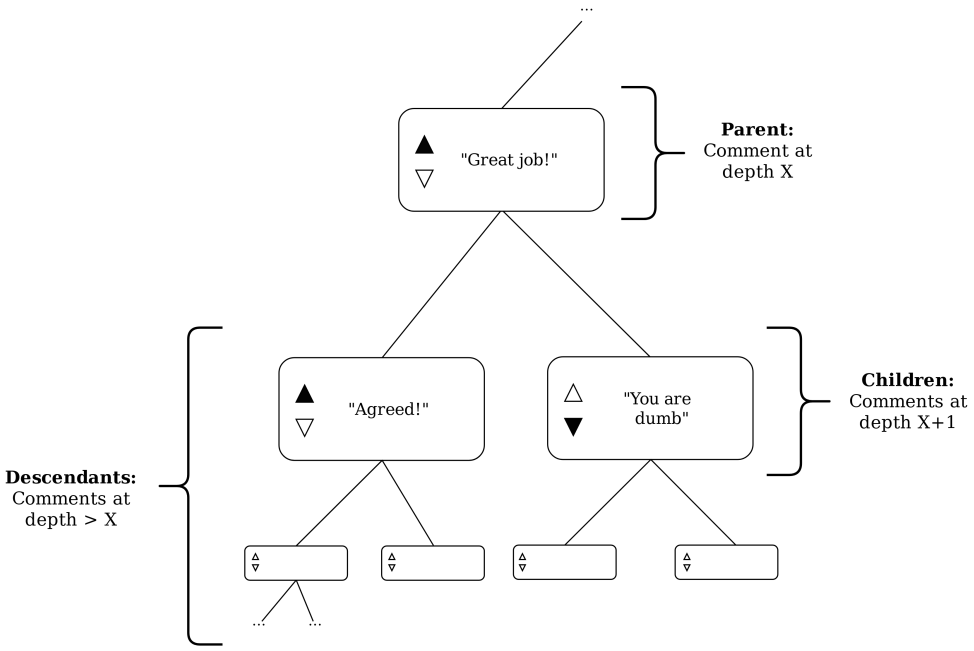


Figure 3.1: Illustrative Reddit Discussion

Based on this goal, we performed additional filtering in order to only consider comments embedded in *minimally interesting* discussions. As we've already mentioned, a single Comment which garners no replies is not interesting for the sake of this project - there are no Descendants from which to draw a notion of "community response". For this work, we only considered Discussions where the Parent of the Discussion had 1) at least two children and 2) at least one Descendant that was not a Child. That is, the tree rooted at the Parent had at least two subtrees, and at least one of those subtrees had height greater than one.

We call Discussions which pass this test *minimally interesting*, as Discussions with these properties have a number of useful qualities for our analysis. First, it is likely that at least two users other than the author of the Parent saw the Parent and were moved to reply to it directly (by 1). This is only untrue if the author of the Parent responded to themselves, of if a user authored two separate replies to the Parent (both unlikely given norms on Reddit). Second, for Discussions which satisfy these properties, at least one of the Child Comments was interesting enough to provoke a reply (by 2). In sum, these minimally interesting Discussions are those where it is highly probable that a significant conversation or interaction occurred between multiple agents. Discussions which were not minimally interesting were removed from the data set.

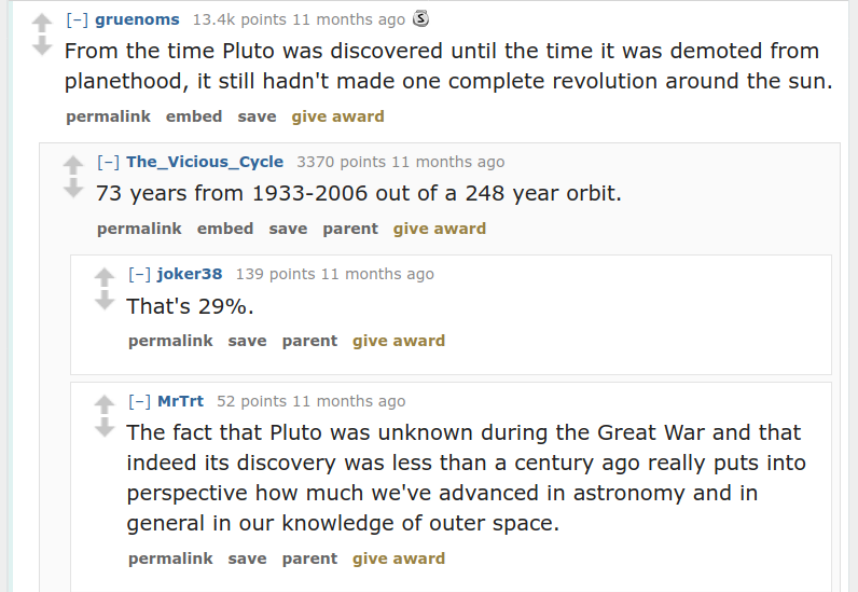


Figure 3.2: Example Reddit Discussion

We also removed from consideration any Discussions where any of the Comments in that Discussion did not appear to be in English. This was done to remove noise from the NLP features extracted from Comments. The PolyGot Python library was used to categorize Comment text language, which ultimately relies on a Naive Bayes classification based on tokens.

After this additional filtering, 448,611 Discussions remained, with a total of 3,658,384 comments.

3.3 Methodology

In order to process data in discussions into fixed length vectors that could be used to train machine learning classifiers, we extracted metadata and linguistic features on the basis of single comments, then used simple statistical methods of aggregation to compute features representing sets of comments. We then evaluated the accuracy of predicting features of a Parent given only the aggregated features of its Descendants.

In this section, we describe the list of features extracted, aggregation, prediction and evaluation procedures.

3.3.1 Feature extraction

The majority of features extracted were defined over single Comments. Our prediction methodologies all depended on fixed length inputs. Since the size of a Discussion is arbitrary, we relied on simple statistical methods to aggregate the features extracted on a per-comment basis when a group of Comments was being considered (e.g. like the Children of a comment). To do this, we simply calculated the minimum, maximum, mean, median and standard deviation of each of the features over all comments in the set. That is, for each of the features we describe that is defined on a single Comment, the feature was expanded into five features when a set of Comments was being considered. When generating feature vectors, we performed this aggregation both for the Children of a Parent and for the Descendants of a Parent. The logic for this was that these two sets offer different views of the effect that the Parent Comment had on the community. The features of the Children reflect the direct responses to the Parent, capturing the clearest indication of how the Comment itself was received. Meanwhile, the features of Descendants captures the general tone of the entire discussion that stemmed from the Parent Comment. We conjecture that both views are important, since the effect of the original comment does not completely end at the direct replies: the topic and tone set will persist through many levels of the discussion. The full list of features is presented below.

- **Word Count**: The number of space-separated words in the text of the comment.
- **Score**: The number of up-votes the comment received minus the number of down -votes.
- **Controversiality**: A boolean value distributed in the Reddit data, which is True when the ratio of up-votes to down-votes a comment received was close to 0.5. Only the Score of a comment and this boolean value are available, not the actual number of up-votes and down-votes.
- **First/Second/Third Pronoun Usage Ratio**: Actually three features, the ratio of pronouns of each degree to pronouns of other degrees in the text of the comment. This feature is inspired by [7], which found that a high number of first person pronouns (e.g. "I", "me") were more likely to appear in well received comments online, while high numbers of second person pronouns (e.g. "you") were associated with poorly received comments. This may be related to the reflective nature of first person pronouns and the accusatory nature of second person pronouns.
- **Punctuation Usage Ratio**: The ratio of tokens which were punctuation to non-punctuation tokens in the comment text.

- **Dot/Intero/Bang Usage Ratio**: Actually three features, the ratio of punctuation of each type to other punctuation tokens in the comment text.
- **Sentiment and Subjectivity**: Actually two features, the predicted sentiment (in the range of -1 to 1) and (subjectivity in the range of 0 to 1) of the comment text. We used the python library TextBlob[6] to compute these scores, which uses a naive Bayes regression learned on a corpus of English movie reviews for prediction.
- **Profanity**: The predicted probability that the comment text contains profanity. We used the python library profanity-check[7] to compute this score, which makes predictions based on a linear SVM trained on 200k human-labeled instances of English profanity.
- **Hate and Offensiveness Probability**: Actually two features, the outputs representing the predicted probability that a text is hateful or offensive from the hatesonar [21] project, which makes predictions based on a logistic regression trained on the text of vectorized hateful tweets.
- **Hate Count**: The number of n-grams in the comment text that are part of a corpus of n-grams[8] which appear frequently in human-labeled hateful texts produced by [21].
- **Hedge Count**: The number of n-grams in the comment text that are part of a corpus of "hedge" n-grams (e.g. "from my perspective", "apparently") produced by [20].
- **Deleted**: Simply whether or not a comment is deleted. When a comment on Reddit is deleted, its text is replaced by the string "[deleted]".
- **Child and Descendant Counts**: Actually two features, the size of the sets of Children and Descendants a comment has.
- **Score Disagreement**: Defined over a set of comments, the ratio of negatively scored to positively scored comments in the set. This feature is intended to quantify the extent to which mutual negativity pervades a discussion. We conjecture that the best Discussions contain mutual positivity (i.e. most Comments have positive scores), while the worst and most inflammatory Discussions contain mutual negativity.

[6] https://github.com/sloria/TextBlob
[7] https://github.com/vzhou842/profanity-check
[8] https://github.com/t-davidson/hate-speech-and-offensive-language/

Thus, a total of 21 relatively simple features where extracted, mostly on the loneComment level. After expansion via aggregation over the sets of Children and Descendants, a total of 154 features was used to predict target features of the Parent comment.

3.3.2 Experiments

We evaluated the accuracy of four binary classification tasks of Parent comments given the features of its Children and Descendants: 1) the presence of hate speech in the Parent 2) a negative sentiment in the Parent 3) the presence of profanity in the Parent 4) a negative score sign on the Parent. Each of these target variables captures some aspect of "bad behaviour" in the Parent comment that we hypothesis may be detectable from the reactions to the comment. The last target, negative score, is a sort of catch all, that captures any sort of behaviour that the community found it suitable to reject (as low scores cause comments to be hidden).

Each of the experiments described above required slightly different procedures to binarize the target variable. We used some domain knowledge to guide this process rather than a completely data driven approach. For example, when binarizing the scores of comments, we split the comments into two groups: those that received scores < 1 and those that received scores ≥ 1. We split the classes at this point, even though the median score in the data set was 11. This is because the default score of a Comment on Reddit is 1. While this leads to a more heavily unbalanced data set, it better captures what our classes aim to represent: those comments which were "rejected"(i.e. received more down-votes than up-votes) and those which were not. To counteract the class imbalance in training data we used adaptively weighted loss functions for all our machine learning methods and report evaluation scores on a balanced test set.

- **Parent Score Prediction**: We binarized the scores of Parents into classes representing comments that have score < 1 and score ≥ 1. Thus, our classifier tries to answer the question: "did the parent comment receive a score below 1?" For our data set, 68403 were in the first class, leaving 380208 in the second class.
- **Parent Sentiment**: We binarized sentiment of Parents into classes representing comments that have sentiment < 0 and sentiment ≥ 0. Thus, our classifier tries to answer the question: "did the parent comment have a more negative than positive sentiment?" For our data set, 260207 were in the first class and 188404 were in the second class.

- **Parent Hate Speech**: We binarized the count of detected hateful terms in Parents into classes representing comments with > 0 hateful n-grams and those with 0 hateful n-grams. Thus, our classifier tried to answer the question: "did the parent comment text contain at least one hateful n-gram?" For our data set, 9251 were in the first class and 439360 were in the second class.

- **Parent Profanity**: We binarized the probability of the Parent comment containing profanity into classes based representing > 0.5 chance of profanity and ≤ 0.5 chance of profanity. Thus, our classifier tried to answer the question: "was the parent comment text more likely than not to contain profanity?" For our data set, 57198 were in the first class and 391413 were in the second.

3.3.3 Predictive models

After testing multiple models, we settled on a random forest (RF) and feed-forward neural network (NN) architecture to evaluate the experiments above. We used both these models, as the neural network produced consistently high performance while the random forest offered superior explainability. We wished to find the best performing model, but also desired insight into what kind of features in a community's response are the most predictive.

We used a random forest with 200 estimators, each of a max depth of 5, and $\lfloor\sqrt{154}\rfloor = 12$ max features per estimator (note, 154 is the total number of features in an input vector). Random forests are briefly described in Section 2.3.4.

We used a simple feed forward neural network consisting of 2 hidden layers of width 64 with ReLu activation, a 25% dropout between layers and a softmax on the output layer. Feed forward neural networks are briefly described in Section 2.3.3.

Both classifiers were trained with balanced class weights in order to adapt to the class imbalances in training data. We used the scikit-learn [57] random forest implementation and built the neural network using the python library Keras [16].

3.4 Results

Results for the prediction tasks described above are summarized in Table 3.1. All results reported are based on a 5-fold cross validation procedures as follows: The data set was split into 5 stratified folds (i.e. each test fold has a proportion of elements of each class equivalent to the proportions in the overall data set). Test sets were then balanced by undersampling

	Accuracy	Precision	Recall	F1-Score
	Score			
RF	0.73	0.72	0.77	0.74
NN	0.77	0.77	0.78	0.78
	Hate			
RF	0.70	0.78	0.54	0.64
NN	0.69	0.74	0.59	0.64
	Profanity			
RF	0.63	0.65	0.56	0.60
NN	0.62	0.63	0.58	0.60
	Sentiment			
RF	0.57	0.57	0.55	0.56
NN	0.57	0.57	0.56	0.57

Table 3.1: Results for binary classification on balanced test sets across four tasks using a random forest (RF) and neural network (NN).

(i.e. randomly removing examples) from whatever class was over represented, if one was over represented. The end result is a 5 way split, where each vector appears in at most one test set, and each test set is class balanced. The results reported are the average over the 5 folds. In each case, the positive class (important for the precision and recall measures) is the "anti-social" class (negative score, hateful n-gram, etc.). Because we did not need to tune our methods for any of these tasks specifically, and in some cases the number of examples of a class were low, we preferred this cross validation method to reporting on reserved test sets.

The importance of the top 15 most important features for each of the prediction tasks according to the Gini importance on the Random Forest we trained (described in Section 2.3.4) are presented in Tables 3.2 and 3.3. Higher values indicate more importance. Bold face has been used to highlight interesting features. The naming scheme for features is {set}_{aggregate}_{feature}. Thus, for example, child_std_score is the standard deviation of scores in the Children of the Parent, and desc_med_off_prob is the median probability that a Comment in the Descendants of the Parents is offensive.

Overall, results for our experiments ranged from encouraging to disappointing, with binary classification accuracies ranging from 77% to 57%: all classifiers do better than random at identifying bad behaviour. No experiment was a complete failure, and we consider the Score and Hate prediction experiments successful.

Score		Sentiment	
Importance	Feature	Importance	Feature
.127	**desc_score_disag**	.153	desc_med_sent
.103	desc_max_score	.114	child_avg_sent
.092	desc_avg_score	.097	child_med_sent
.082	child_std_score	.096	desc_avg_sent
.082	desc_min_score	.057	**desc_med_profanity**
.060	desc_std_score	.043	**desc_max_profanity**
.042	child_avg_score	.038	desc_med_subj
.038	child_count	.033	child_avg_subj
.023	**desc_std_punc_bang**	.032	**desc_min_profanity**
.023	**desc_med_hate_prob**	.029	**child_min_profanity**
.022	**child_max_hate_prob**	.027	child_min_sent
.021	child_max_score	.025	**child_max_profanity**
.019	child_min_score	.022	child_max_sent
.019	child_med_score	.017	**desc_avg_punc_dot**
.018	**child_med_prp_second**	.016	child_med_subj

Table 3.2: Top 15 most important features for Score and Sentiment prediction tasks.

Score prediction had the higher overall accuracy. One reason for this may be that the labels for score prediction had no added noise - we simply took the scores reported in the data - while the other experiments had labels generated by the application of noisy NLP based techniques. Another reason may be that the scores of a Parent are highly correlated with the scores of its Descendants: this could be the case because a popular parent with a high score will lend higher visibility to its Descendants (as higher scoring comments are shown more readily on Reddit). Indeed, in Table 3.2, it can be seen that the most important features for score prediction are mostly based on the scores of the descendants and children, and the number of children (i.e. popularity). It's also interesting that the *most* impactful feature is the one we developed specifically to gauge the mutual positivity or negativity of a Discussion, the score disagreement. In addition, the importance of the variability of exclamation mark usage (desc_std_punc_bang) and probability of containing hate (desc_med_hate_prob and child_max_hate_prob) among Descendants are interesting, as one can speculate that highly enthusiastic Comments with low probabilities of hate are highly correlated with well received Parent comments. Finally, the importance of second degree pronouns in Children (child_med_prp_second) is interesting, as it can be seen as corroborating the findings of Brennan et al. [7], who showed that pronoun degree usage

Hate		Profanity	
Importance	Feature	Importance	Feature
.175	desc_avg_hate_count	.176	desc_avg_off_prob
.175	desc_std_hate_count	.176	child_avg_off_prob
.143	child_avg_hate_count	.087	desc_med_off_prob
.087	child_std_hate_count	.080	desc_avg_profanity
.067	child_med_hate_count	.067	desc_std_profanity
.057	desc_avg_hate_prob	.060	child_avg_profanity
.031	child_avg_hate_prob	.060	desc_std_off_prob
.031	desc_std_hate_prob	.052	child_std_off_prob
.028	child_std_hate_prob	.052	child_med_off_prob
.024	desc_med_hate_prob	.039	desc_med_profanity
.021	child_med_hate_prob	.033	child_std_profanity
.017	desc_med_hate_count	.020	child_min_off_prob
.013	**desc_avg_profanity**	.018	child_med_profanity
.012	**desc_std_profanity**	.016	child_max_off_prob
.009	**desc_score_disag**	.008	desc_min_off_prob

Table 3.3: Top 15 most important features for Hate and Profanity prediction tasks.

was correlated with the negative or positive reception of comments.

Prediction of the presence of hateful terms was relatively successful, with an accuracy of 70% and a high precision of 78%, indicating that 78% of the comments the classifier predicted to be hateful actually contained some hateful n-gram. However, recall is somewhat low at 59%. On a hate speech prediction task, high recall is especially desirable, as predictions of this type can be used in a pipeline of filtering comments for down stream human reviewers. The low recall may be because many of the comment which contained a hateful n-gram were actually quoting, or using the terms in a context that did not inspire a strong reaction. The most important features for predicting hate in a Parent, as shown in Table 3.3, are evidence of hate in the Descendants (the *_hate_count and *_hate_prob features). However, some importance is given to profanity in the Descendants (desc_avg_profanity and desc_std_profanity) and score disagreement (desc_score_disag), inviting the speculation that hateful comments may provoke profane and argumentative comments in the ensuing discussion.

The profanity and negative sentiment prediction tasks were less successful, with relatively unimpressive (although still better than random) scores on all evaluations. One

reason why this may be is that these classes stretch the definition of "anti-social" or "bad behaviour". In fact, profanity and negative sentiment are relatively common online, and are unlikely to provoke especially unique reactions from a community. In both cases, as shown in Tables 3.2 and 3.3, the most important features for prediction were mostly the presence of similar features in Descendants. Profanity in replies has some notable importance in the prediction of Sentiment, but as Sentiment prediction accuracy did not surpass 60%, it is not likely useful to speculate on any connection between these domains.

These results reveal interesting connections between multiple facets of natural language feedback in a threaded conversation system. While classification accuracy on most tasks was below what would be needed for a truly reliable system of classification, the results show that in principle, it's possible to predict certain qualities of a discussion starting comment simply be examining the discussion that follows: in particular, the tasks of predicting "rejected" (score < 1) comments and the presence of hate speech in a comment shows promise (with accuracy of 77% and 70% respectively shown in Table 3.1).

3.5 Conclusion

3.5.1 Summary

In this work, we explained why it is difficult to quantify outcomes for discussion starting events in the social network domain. In particular, we argued that binary scoring systems (i.e. up and down votes) offer only a glimpse of the effect agents are having on the social health of a community, and that the magnitude of score should likely be ignored except in cases where specifically gauging popularity is important. We argued that new methods for quantifying outcome should be developed which focus on the response to a comment by the community. This approach can be analogized to shifting the attention of diagnostic focus away from detecting an ailment directly and towards detecting the symptoms of an ailment (i.e. the "immune response" to hateful dialogue in a community). Specifically, we aim to measure the effect a comment has on a community directly, rather than comparing the comment to a pre-defined measure of goodness or badness.

While sophisticated NLP systems may be flummoxed by sarcasm and calls to context, the humans who read and respond to comments have a much better chance of noticing bad behaviour and responding to it - potentially in detectable ways. We argued that specifically for actions which can be described as anti-social, it is likely that a pattern exists in community responses that could be used to detect the presence of bad behaviour

which is difficult to detect directly. To the best of our knowledge, this is a novel approach to the problem, specifically as it relates to social networks.

We showed that even with a simple set of features and procedures for aggregating those features, a focus on community response yields surprisingly accurate classifiers, especially in the score polarity prediction task. Moreover, we examined which features held the most predictive power for specific tasks, highlighting the crossover between domains when they appeared, and offering speculative explanations for the observed results.

3.5.2 Discussion

There is a clear path to extend this work into a model of agents' trustworthiness on a social network. Given a classifier like the one described above, for example, one that predicts hateful terms in comments given the Descendants of that comment, $H(c)$, an agent's reputation with respect to likelihood to use hateful terms might be calculated as the average, maximum, or a weighted average of H evaluated over all of that agents' submitted comments. Such a metric could then be integrated into a multi-faceted trust system, like the ones described in Section 2.2.2 or in Chapter 4. In fact, multiple classifiers for multiple different types of bad behaviour and negative community reactions could all serve as distinct sources of evidence for such a model. A multi-faceted approach to trust modeling would be particularly useful, as the importance of these metrics with respect to some trust context should be learned in a data-driven manner.

One distinct advantage of hate speech detection via analysis of community response which we have alluded to multiple times is that such a detector effectively side steps many of the confounding issues in detecting hate speech directly, such as sarcasm, context, and specific attempts to avoid detection. So long as the bad behaviour is noticed and responded to as such by the community, the signals embedded in the response could be easier to detect. In a sense, we leverage the power of human-level hate speech detectors (i.e. actual humans) and posit that they respond to this behaviour in predictable ways. This may be because the respondents to this anti-social behaviour have no incentive to try to hide their reactions, for example, by being sarcastic or alluding slyly to some exterior context in order to make their point. In Appendix B, we have included a number of illustrative examples from our data set that were rated as being highly likely to contain hateful speech by the random forest classifier we trained and reported on above. This appendix contains a number of examples where complex objectionable ideas are expressed, which our classifier is highly confident about.

One potential disadvantage of this approach which we have already alluded to is the

completely "bottom-up" or *a posteriori* nature of the process. No "top-down" descriptions of what is and is not objectionable behaviour are coded in to the system, beyond, for example, a lexicon of hateful terms. But this lexicon only influences the labels that are generated, not the association that is learned between features of responses and these labels. We speculate that our classifiers are detecting hateful terms by noticing the angry responses to this behaviour in the community, but this is not necessarily the case. As we mentioned, were this process trained on data from communities where hate speech is accepted and lauded, the type of reaction to hate speech that is learned could be very different. As such, it is difficult to see how this system could be used across communities, especially when trained on moderate communities and deployed on radical communities. This disadvantage is a symptom of the pure bottom-up approach - the designers of the system do not dictate what is and isn't desirable, only what is to be detected. We rely on the wisdom and good nature of the community to effectively flag comments which cross a boundary - the boundary that is defined by the community itself. Such a system gives no power to define *a priori* where the boundary is between pro-social and anti-social behaviour, only to detect where the community in aggregate has placed that boundary. We will discuss more about the tension between bottom-up and top-down approaches in Chapter 5.

Another disadvantage is that this approach has nothing to say about comments that were simply ignored. A comment could be replete with anti-social content, but if no one ever notices it and replies to it, there is no basis for this model to make a prediction on. This disadvantage also relates to the bottom-up approach - our methods' definition of goodness and badness is entirely *a posteriori* - some sort of reaction from the community is necessary to have any clue what is happening.

Finally, we offer an additional perspective on the connection of this specific research to the larger topic of modeling trust in social networks. As we will discuss further in Section 5.1, modeling the trustworthiness of an author of a comment in a social network has proven to be rather elusive. Some researchers have put effort into modeling crediblity [72] or in demonstrating how credibility ratings can help to divert misleading majority opinion [65, 66], but have largely set aside the process of deriving a value for an author's trustworthiness. The approach outlined in this chapter suggests that valuable insights into the quality of an author can be gained by carefully analyzing the effect that agent is having on others. The ultimate aim may be to imagine users whose comments provoke bad reactions (thus disrupting a community) as having a bad reputation, which can be a useful indicator for reasoning about trustworthiness. Once this reputation or trustworthiness has been assessed, then it could potentially be used while performing message recommendation to decide whether that author's post should be shown to a user or not.

Chapter 4

Personalized Multi-Faceted Trust Modeling

In this chapter we describe an experiment which explores the influence of personalization and context on a multi-faceted trust model. In particular, we hypothesize that the propensity of agents to form trust relationships may vary on an individual basis, and that increasing the resolution of trust prediction by clustering agents and learning trust formulation behaviours at the level of these clusters, rather than on the entire population of agents, may have a positive impact on the performance of a trust-aware recommender system. We argue that this increase in resolution constitutes a form of personalization (albeit, performed at a group level rather than at an individual level). In addition, we explore the impact of considering differing contexts of trust by testing the effect of predicting two types of trust links.

At the heart of our solution is an effort to predict novel trust links in a social network by using machine learning methods to determine how to weight feature importance, and to approximate trust formulation procedures among groups of similar agents. Our approach makes use of the flexibility of MFTM, which we demonstrate by combining features drawn from two existing proposals with our own novel features. Evaluation is performed by measuring the error rates on a recommendation task that incorporates trust information. This is performed on a data set collected from Yelp[1], a content rating site with social network features.

[1] https://www.yelp.com/

Figure 4.1: Example Yelp review with rating

4.1 Motivation

As we have argued in Section 2.2.2, multi-faceted trust modeling provides a highly flexible and general framework for the application of trust models to various domains. In this work, we argue specifically for the application of MFTM to social networks, as these domains are replete with possible trust-relevant indicators.

4.1.1 Choice of data set

We used data from Yelp, a popular social-network and item-rating service to train and evaluate our models. On Yelp, users can indicate binary social trust towards other users and submit ratings for products/businesses/websites (taken together, and following the trend in recommender systems literature, these entities are called "items") that they have experienced, indicating their satisfaction with that product/business/website. These ratings are integers in the range $[1, 5]$, illustrated as stars, where higher numbers indicate a stronger recommendation. An example 5-star review for Schwartz's Deli[2] is presented in Figure 4.1.

We used this data set particularly because it is amenable to validation of trust model effectiveness via a downstream item recommendation task.

To expand on this point: one of the recurring challenges in the development of trust models is finding grounds for the validation of the accuracy of the models [14]. Since trust models aim to predict new trust links, trust is subjective, and the predictions of trust models can be followed or ignored by independent agents, it is difficult to truly evaluate the effectiveness of models without deploying a system on an active service and measuring the real effects of trust link prediction on active agents. This is difficult, expensive and

[2]Note, this user has mistaken smoked meat for corned beef.

requires the cooperation of an active social network service, so many models validate their effectiveness on data generated by an agent simulation instead (e.g. [38, 87, 59, 12]). While this is a useful approach for contrasting the effectiveness of various models and gives the researcher a large amount of control for simulating specific types of agent behavior, it clearly adds a layer of ambiguity between the reported effectiveness of the model and its potential for real world application. In some cases, merely changing simulation parameters can defeat systems that had performed well on the simulations their creators had designed [41].

A rising trend in this field is to validate models by applying their predictions to a downstream recommendation task (e.g. [22, 50]): that is, using the trust model to predict novel trust links, $\hat{\Gamma}$, in a MAS, then feeding those predicted links into a trust-aware item recommendation system. These trust-aware recommender systems incorporate both user-item rating behavior and user-user social/trust connections to better recommend items by leveraging the fact that social/trust connections exert influence on the preferences of agents (e.g. you are more likely to watch/enjoy a film a trusted friend recommends)[3]. The logic of this two part process is that when a trust model is able to accurately predict trust links in the context of peer to peer item recommendation, then the resulting accuracy of the recommender system trained with those links will improve.

More information about the Yelp data set will be presented below in Section 4.2.

4.1.2 Personalization

The rationale for testing the effect of personalization is simple: we expect trust formulation procedures to vary from person to person, therefore learning approximations of trust formulation procedures may be more accurate on a more personalized scale.

We have already explained that trust is a subjective phenomenon, and from personal experience we can verify that trust formulation procedures among humans vary. For example, some of us place a heavy importance on shared history and community involvement when choosing a car mechanic to trust, while others place importance on popularity and creative radio ads. As trust is inherently subjective, we do not impose the belief that one individual's trust formulation procedure is more correct than another's.

The inherent subjectivity of trust has important implication from a machine learning perspective. In particular, it implies that trust predictors trained on large data sets representing the behavior of many individuals are not necessarily more correct than those

[3]See Section 2.3.6.

trained on smaller groups. While the predictor trained in the former case will likely have a higher accuracy across the broad population, it is essentially learning the "average" trust formulation procedure, potentially disadvantaging agent's whose preferences are not aligned with the population at large.

The approach to personalization we take is to cluster users based on item and social preference similarity, then learn distinct predictors based on the data associated with each cluster of users. This approach was suggested in Fang et al. [22], but was not pursued. While our approach is useful for capturing the variance of trust formulation in smaller groups, it does not attempt to learn the preferences of individual agents. We will discuss possible avenues for truly individual personalization of trust modeling and other approaches in Section 6.2.2.

4.1.3 Context

As we argued in Section 2.2, context is a critical semantic of trust. To illustrate, context is important as one may trust a peer to recommend a movie or restaurant, but not to fix their car. In this section, we consider specifically the context of recommending items. This is the case where agent a_i encourages agent a_j to invest resources into accessing or consuming item k based on their own experience with it. This particular context, while slightly generic (i.e. recommending movies and recommending restaurants could be considered different contexts), is appropriate for the recommendation task outlined above. In the next section, we will outline how data was filtered to help restrict the broadness of the context.

Given this context, it is unclear if predicting friendship or trust links is the most relevant target of prediction. For example, while it is well known that recommendations and influence from friends are impactful under diverse contexts (e.g. [3, 28]), it has also been shown that in multiple data sets from online environments that friendship only correlates weakly with similarity in reviewing behavior [32].

Therefore, keeping context in mind, it may be more effective to predict positive review score correlation between agents directly. Accordingly, in this work we conduct tests where both explicit links (declarations of friendship/trust) and implicit links (positive review score correlation) are the target of prediction in order to test whether the latter is more relevant than the former.

4.2 Data Set and Data Filtering

Yelp is a product review site and social network of crowd-sourced reviews targeting brick-and-mortar businesses. The product review side of the site involves businesses and services such as schools, restaurants and plumbing services, which are presented as individual items for users to review on a one to five star scale. Users can also write text reviews outlining their opinion of a service. The owners of businesses can also interact with the site by uploading meta data describing their business, such as hours of operation, location, and menus. In addition to writing reviews of services, users of the site can form mutual friendships and follow other users in order to receive the recommendations of these trusted users first.

The friendship relation on the site is important to some users to filter out the opinions of unknown and untrusted users. As one user put it[4]:

> For me I like it when I am looking at a new restaurant, etc., my yelp friends will come up first on the reviews. I have come to trust their opinions and insights.

To which another user replied:

> Yep, they nailed it. I don't need to see the reviews of the moron majority of Yelp any more. I just see trusted friends.

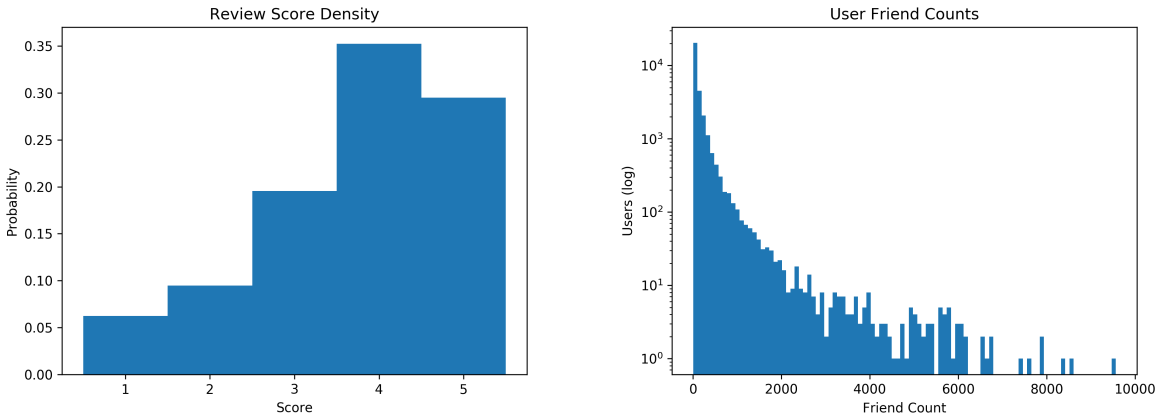
As these quotes outline, because of the feature where the reviews of friends are given priority, the friendship relation can be used as a tool for expressing mutual trust in the context of item recommendation.

Data describing users, reviews and businesses is made public by Yelp on a regular basis[5]. The full data set from 2019 contained descriptions of 1,637,138 users, 192,609 businesses, and 6,685,900 reviews.

We filtered the data set both to reduce the massive amount of data and to narrow down the context of trust in focus. Specifically, we only considered users who had reviewed at least 20 businesses that were tagged as restaurants. This narrows the context of trust from

[4]This conversation can be retrieved at: https://www.yelp.com/topic/glenside-whats-the-purpose-of-having-friends-on-yelp. An archived version of the page is hosted at: https://web.archive.org/web/20200204194254/https://www.yelp.com/topic/glenside-whats-the-purpose-of-having-friends-on-yelp

[5]https://www.yelp.com/dataset



(a) Review score distribution

(b) Friend counts

Figure 4.2: Review scores and friend counts

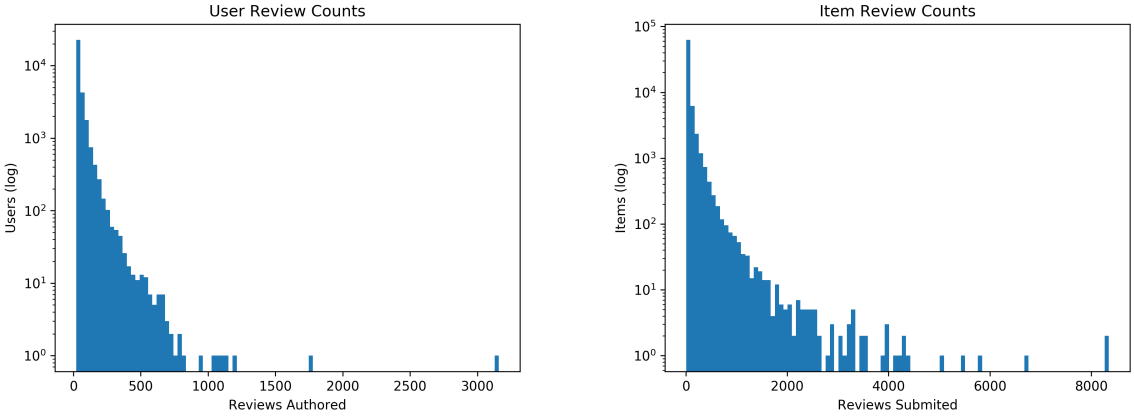
"recommending businesses or services" to "recommending restaurants" and reduces the data set down to 30,721 users and 4,432,064 reviews concerning 74,560 businesses. This filtering procedure was inspired by [50].

Statistics from the filtered data set are presented in the Table 4.1 and in the histograms in Figures 4.2 and 4.3. As can be seen, there is a relatively well spread out distribution of scores given to items[6], centered around 4 stars. This leads to a relatively difficult prediction task, as predicting the median review score is only correct in 35% of cases. Counts of friends and reviews are plotted on a logarithmic scale, and show a "long tail" distribution that is common in online phenomenon [70].

	Mean	Median	Mode	Min	Max
Friends Per User	153.23	45	1	1	9564
Review Per User	49.66	33	20	20	3159
Average User Rating	3.74	3.77	4	1.33	5
Reviews Per Item	59.33	17	3	3	8349
Global Review Scores	3.72	4	4	1	5

Table 4.1: Yelp Filtered Data Statistics

[6]Compared to the popular Epinions data set, where nearly 80% of reviews are 5 stars.



(a) Number of reviews submitted per user. (b) Number of reviews received per item.

Figure 4.3: Reviews submitted and received counts

4.3 Methodology

In this work we test the effect of personalization or trust link prediction on an item recommendation task. In particular, we test whether clustering agents and learning trust link predictors on the basis of clusters of similar agents, as opposed to learning a single trust link predictor for the entire population of agents, can increase the accuracy of trust-aware item recommendation systems. We also test the effect of altering the type of trust link prediction by either attempting to predict the presence of an explicit friend/trust link or predicting positive correlation in item review scores.

Our final analysis will report the recommendation accuracy of 7 configurations, where each configuration uses an identical set of agents and recommendation procedures, but a different procedure for predicting the trust links between agents. Each configuration is given a name reflecting which (if any) type of clustering was performed, and which type of trust link was predicted. The complete list is presented in Table 4.2.

When the first step is skipped, no personalization is performed (i.e. the first three items in the list above). When the second step is skipped, no trust link prediction is performed (i.e. the first item in the list above). The rationale of skipping certain steps is to compare and contrast the effect applying these steps has on the final accuracy of the recommendation task.

The entire procedure can be described sequentially, as follows. Each step will be briefly

Experiment Name	Experiment Description
RealLinks	Perform no prediction of trust links whatsoever. Use the real, explicit trust/friend links in the data set.
FriendPredict	Predict trust/friendship links with no personalization step (i.e. learn one trust predictor for the entire population of agents).
PrefPredict	Predict positive review score correlation with no personalization.
PrefCluster-PrefPredict	Determine clusters of agents with similar preferences for items (i.e. positive item review score correlation) and predict positive review score correlation links for each cluster.
PrefCluster-FriendPredict	Determine clusters of agents with similar preferences for items (i.e. positive item review score correlation) and predict trust/friendship links for each cluster.
SocialCluster-PrefPredict	Determine clusters of agents with high overlaps in their social circles and predict positive review score correlation links for each cluster.
SocialCluster-FriendPredict	Determine clusters of agents with high overlaps in their social circles and predict trust/friendship links for each cluster.

Table 4.2: Experiment descriptions

explained, noting its inputs and outputs, then will be more carefully considered in subsections below.

- **Clustering**
  - **Input:** All agents A and an agent-agent similarity matrix, S.
  - **Output:** An assignment of every agent to a cluster, C
  - **Description:** Partition the agents into groups of highly similar agents. We used social circle overlap (Jaccard Similarity) and review score correlation (Pearson Correlation Coefficient) as similarity measures. We developed two clustering methods for this step.

- **Trust Link Prediction**
  - **Input:** Clusters of agents, C, and trust indicator function $\Psi(a, b)$.
  - **Output:** A matrix of trust link predictions, $\hat{\Gamma}$
  - **Description:** For each cluster c_l of agents a logistic regression learns a distinct MFTM trust prediction function for that cluster. We experimented with predicting friendship links and positive review score correlation. Output a $|A| \times |A|$ matrix, $\hat{\Gamma}$, where, $\hat{\Gamma}_{ij} = 1$ if the classifier for the i'th agent's cluster predicts a trust link between agents i and j and 0 otherwise.
- **Recommendation Evaluation**
  - **Input:** Agent-item rating matrix R, trust link prediction matrix, $\hat{\Gamma}$.
  - **Output:** A agent-item matrix of predicted review scores, $\hat{R}$.
  - **Description:** Given reviews present in the original data set and the predictions from the previous step train a trust-aware recommender system to predict review scores. After training, we evaluate the correctness of the recommender on a reserved testing set using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics.

4.3.1 Clustering

Our main hypothesis involves exploring the effect that personalization can have on the effectiveness of trust link prediction. However, currently data sets make it difficult to test truly individual personalization. This is because explicit elicitation of factors which influence trust on a personal level is rare on most services, and most agents have not participated in enough activity in order to accurately measure the patterns of their preferences implicit in their behavior. Therefore, we focus on clusters of similar agents rather than considering each agent distinctly. We posit that if personalization at this level of granularity is sufficient to increase the accuracy of our trust models, then we will have found evidence that some level of personalization is indeed useful for the trust modeling task, and will have motivated further research in the area.

Clustering procedures generally rely on the definition of a distance or closeness (alternatively, similarity) metric between elements to be clustered [61]. In this work, we tested two separate similarity functions. Specifically, we tested clustering agents on the basis

of preference similarity and social circle similarity, as defined in Equations 4.1 and 4.2 respectively:

$$prefSim'(a_i, a_j) = \frac{\sum_{k \in R_{i,j}} (r_{ik} - \bar{r}_k)(r_{jk} - \bar{r}_k)}{\sqrt{\sum_{k \in R_{i,j}} (r_{ik} - \bar{r}_k)^2} \sqrt{\sum_{k \in R_{i,j}} (r_{jk} - \bar{r}_k)^2}} \tag{4.1}$$

$$socialSim'(a_i, a_j) = \frac{|friends(a_i) \cap friends(a_j)|}{|friends(a_i) \cup friends(a_j)|} \tag{4.2}$$

where $R_{i,j}$ is the set of items which both agents a_i and a_j have reviewed, r_{ik} is the rating given by agent a_i to item k, $\bar{r}_k$ is the average rating for item k, and $friends(a_i)$ is the set of agents a_i has entered into mutual friendship with. Put otherwise, we clustered agents on the basis of the Pearson Correlation Coefficient of scores they had given in reviews to items and on the basis of the Jaccard Similarity of their friend groups.

The choice of both metrics was motivated by a desire to extract metrics from our data which:

1. Are relatively generic (i.e. could likely be applied to similar data sets).
2. Could plausibly be argued to constitute a basis for determining which agents are similar enough that we might expect their trust formulation procedures to also be similar.

Since our context of trust is based on recommending items, we argue that both criteria are met. For 1., we argue it is reasonable to assume that on any online service with an item review and recommendation component, it will be possible to calculate Equation 4.1. Similarly, it is reasonable to assume that Equation 4.2 will often be computable on these services, as it is widely believed that friend relationships are a useful tool for expressing preference alignment among agents in such domains [32]. For 2., we argue that $socialSim'$ directly satisfies this criteria by its definition, as $socialSim'$ measures the observed similarity in the output of a trust-like relationship formation procedure (friendship). For $prefSim'$, we argue that if two agents a and b have demonstrated a strong preference for similar items, then it is reasonable to conclude that their procedures for choosing who to trust for new recommendations should be similar. Thus, it is reasonable to cluster them under this context.

While we have argued that these similarity metrics are relevant for our goals, they do present challenges as metrics for clustering algorithms. Specifically:

- Both metrics violate the triangle inequality.
- Both metrics can sometimes be undefined (when the denominator is 0).

As many clustering algorithms are defined over Euclidean spaces, these caveats represent significant restrictions of possible approaches. It is somewhat helpful that the second caveat can be addressed by simply substituting default values in the case where division by zero would occur. Accordingly, we used the following metrics in our final procedure:

$$prefSim(a_i, a_j) = \begin{cases} 1 & \text{if } |R_{i,j}| < 4 \\ & \text{or } \sum_{k \in R_{i,j}} (r_{i,k} - \bar{r}_k)^2 = 0 \\ & \text{or } \sum_{k \in R_{i,j}} (r_{j,k} - \bar{r}_k)^2 = 0 \\ 1 + prefSim'(a_i, a_j) & \text{otherwise} \end{cases} \tag{4.3}$$

$$socialSim(a_i, a_j) = \begin{cases} 0 & \text{if } |friends(a_i) \cup friends(a_j)| = 0 \\ socialSim'(a_i, a_j) & \text{otherwise} \end{cases} \tag{4.4}$$

Note that $0 \leq prefSim(a_i, a_j) \leq 2$, where values below 1 indicate a negative correlation. Therefore, the most appropriate default value is 1. Similarly, $0 \leq socialSim(a_i, a_j) \leq 1$, where values near 0 indicate very few common friends between a_i and a_j, thus, the most appropriate default value when neither agent has any friends is 0. In addition, in Equation 4.3 we have also substituted a default value when $|R_{i,j}| < 4$. This is because correlation tests produce noisy results with small data sets, making it prudent to choose a cutoff point under which no correlation metrics are considered. Meanwhile, if this cutoff is too high, then potentially useful data is ignored to avoid error. We chose the cutoff at 4 arbitrarily.

While this at least leaves the similarity functions well defined, it also creates a situation where the vast number of pairs of agents have a default distance between them, as any two randomly picked agents in a large enough environment will be unlikely to have any interaction history. This is a potential issue as it may causes clusters to appear significantly less cohesive than they actually are, e.g. in the case where agents a and b have "default" distance between them, but are both close to agent c. In this case, a and b should likely be in the same cluster as c, even if they don't themselves appear to share any relationship. Note how this bears conceptual similarity to the concept of trust transitivity as introduced in Section 2.2.3.

In addition to the challenges described above, our clustering task had the additional goal of finding relatively large clusters. This is because our "downstream" goal was to learn personalized classifiers for each cluster of agents. If clusters are too small, then the accuracy of classifiers will suffer.

Given these goals and constraints, our first attempt at a clustering was a simple, non-iterative greedy algorithm, shown in Algorithm 1. This algorithm takes as input the set of agents to be clustered, A, the similarity matrix between agents S (where $S_{i,j} = sim(a_i, a_j)$ for some similarity function) and the desired size of clusters η.

Algorithm 1: Greedy non-iterative clustering

Data: agents: A, similarity matrix: S, cluster size: η
Result: assignment of agents to clusters: C

$C \leftarrow \emptyset$;
while $|freeAgents(A, C)| > \eta$ **do**
  $a \leftarrow pickcentroid(A, C, S)$;
  $c \leftarrow \{a\}$;
  while $|c| < \eta$ **do**
    $next \leftarrow pickNext(c, A, C, S)$;
    $c \leftarrow c \cup \{next\}$;
  $C \leftarrow C \cup c$;
$C \leftarrow C \cup freeAgents(A, C)$;
return C;

In the above, $freeAgents(A, C)$ returns the set of agents not yet assigned to a cluster in C (unassigned agents), $pickCentroid(A, C, S)$ returns the unassigned agent with the greatest mean similarity to all other agents, and $pickNext(c, A, C, S)$ returns the unassigned agent with the greatest mean similarity to the agents in c.

Roughly, Algorithm 1 partitions the set of agents into at least $\lfloor |A|/\eta \rfloor$ clusters of size η. It does this by picking the most central unassigned agent as the core of a new cluster c_i, then adding agents to that cluster in order of greatest mean similarity to agents already in cluster c_i until $|c_i| = \eta$. The process repeats for c_{i+1}, except only agents not already assigned to a cluster are considered. This continues until less than η agents remain unassigned, at which point all unassigned agents are added to a final cluster of unspecified size.

Clearly this algorithm is quite simple, but it is appropriate for the constraints outlined above. Firstly, all clusters of agents except for one will have a guaranteed minimum size η, allowing control over the minimum training data size for the downstream prediction task. More importantly, it handles the non-Euclidean nature of the data by using the mean distance of all points in a cluster as a similarity metric, rather than a geometric center[7].

[7]This is inspired by the average linkage criterion used in hierarchical clustering algorithms [61]. We tested clustering this data hierarchically, but had little success producing clusters of reasonable size.

We improved this algorithm by transforming it into an iterative version listed below (Algorithm 2).

Algorithm 2: Modified k-means clustering

Data: agents: A, similarity matrix: S, cluster count: k, max iterations: m
Result: assignment of agents to clusters: C

$C \leftarrow greedilyPartition(A, S, \lfloor |A|/k \rfloor)$;
for $i \in [0...m]$ **do**
    $S' \leftarrow computeClusterSims(A, C, S)$;
    for $a_i \in A$ **do**
        $C \leftarrow assignToNearestCluster(a_i, S', C)$;
return C;

In the above, $greedilyPartition(A, C, S)$ assigns each agent to a cluster using the procedure outlined above in Algorithm 1. $computeClusterSims(A, C, S)$ computes a new similarity matrix, S', between agents and clusters, where $S'_{i,j}$ is the average similarity between agent i and all agents in the j'th cluster (other than themselves):

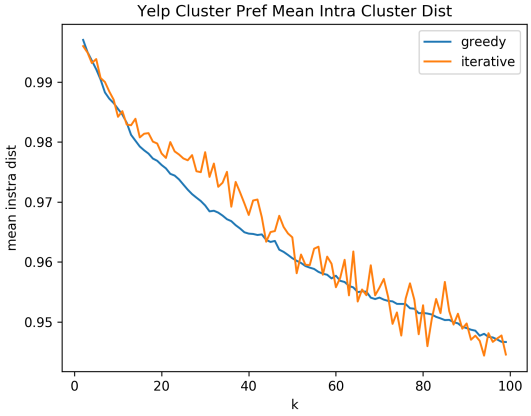
$$S'_{i,j} = \frac{1}{|c_j| - \mathbb{1}(a_i \in c_j)} \sum_{a_k \in c_j} \mathbb{1}(i \neq k) S_{k,i}$$

where $\mathbb{1}(cond)$ is the function which is equal to 1 when $cond$ is true and 0 otherwise. $assignToNearestCluster(a_i, S', C)$ computes a modification of the current set of clusters C by moving agent a_i to the cluster c_j that maximizes $S'_{i,j}$, that is, the cluster for which they have the highest average similarity with other cluster members, (with ties broken randomly). This process repeats for a predetermined maximum number of iterations m.

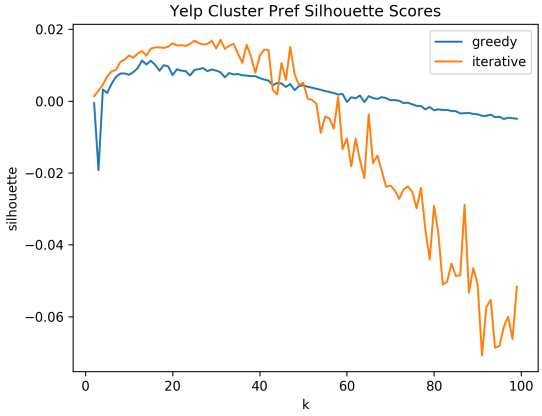
This process is much closer to classic k-means clustering, again with the modification that distances between clusters and points must be calculated on the basis of mean distances rather than distances to the cluster's geometric center. In addition, rather than picking random points to serve as initial cluster centers, the initial clusters are determined by a greedy partitioning method. These modifications result in an algorithm that, in our experiments, tended to produce relatively large and cohesive clusters.

Notably, a minimum cluster size is no longer guaranteed by this clustering method, which we have pointed out is a desirable feature for our later prediction task. We will describe our procedure for dealing with this in Section 4.3.2.

Both Algorithm 1 and 2 require a parameter used to control the number of clusters (η and k respectively). When performing our experiments, we determined values for these



(a) Intra-cluster distance (lower is better)



(b) Cluster silhouette scores (higher is better)

Figure 4.4: Yelp data preference clustering results

parameters by running the clustering step multiple times over a range of parameters with a relatively low maximum iteration setting, then choosing the best performing parameter to proceed with.

In Figures 4.4 and 4.5 we illustrate the performance of clustering techniques as the number of clusters (k) is altered. We measure cluster cohesiveness using two metrics, mean-intra cluster distance and silhouette score. In the below, $dist(i, j)$ is the appropriate distance measure for the similarity function chosen, i.e. if $sim(i, j)$ is high when i and j are similar, then $dist(i, j)$ is low when i and j are similar. Mean intra cluster distance is defined as follows:

$$meanintra(C) = \frac{1}{|C|} \sum_{c_i \in C} \sum_{j \in c_i} \sum_{k \in c_i} \frac{dist(i, j)}{|c_i|} \tag{4.5}$$

That is, the average distance between all elements in a cluster and the other elements in that cluster, averaged over all clusters.

Silhouette score $s(j)$ for a single clustered point j which has been assigned to cluster

(a) Intra-cluster distance (lower is better) (b) Cluster silhouette scores (higher is better)

Figure 4.5: Yelp data social clustering results

c_i is defined as follows:

$$a(j) = \frac{1}{|c_i| - 1} \sum_{k \in c_i, j \neq k} dist(j, k) \tag{4.6}$$

$$b(j) = \min_{\ell \neq i} \frac{1}{c_\ell} \sum_{k \in c_\ell} dist(j, k) \tag{4.7}$$

$$s(j) = \begin{cases} \frac{b(j) - a(j)}{max(a(j), b(j))} & if |c_i| > 1 \\ 0 & otherwise \end{cases} \tag{4.8}$$

That is, $a(j)$ is the average distance point j has to other points in its cluster. $b(j)$ is the minimum distance from j to to any other point not in the same cluster as j. $s(j)$ is the silhouette score for the point j. When a point exists which is not in the same cluster as j but is closer to j than the average point in j's cluster, then the score is negative. When the closest point to j outside of its cluster is farther away than the average distance of point in j's cluster, then the score is positive. The silhouette score for a set of clusters C is calculated by taking the average of a random sample of points from different clusters.

Both metrics capture a sense of the cohesiveness of a set of clusters, and can be used to judge the relative merits of different clustering schemes and parameter settings. While both metrics are interesting, there are a few caveats to consider. First, for this data set and clustering algorithm, it is expected that intra-cluster distance will decrease as cluster

count increases. This is shown to be true in the results reported. Thus, this metric is better suited for showing the difference in performance between clustering methods than it is for choosing a value of k. The silhouette metric does a better job of outlining the tradeoff for k values, as it will punish a method for assigning two close points to separate clusters. Therefore, a value of k that maximizes the silhouette score over a range is a more appropriate guide to choosing a value of k.

Noticeably, performance on both scores is low in an absolute sense. Silhouette score ranges from $[-1, 1]$, where positive scores are good and negative bad. Our clustering algorithms achieve scores in the range of $[-0.06, 0.02]$ - a tiny portion of the possible range near 0. Similarly, the scores for intra cluster dist should range from $[0, 2]$, where small scores are good, and our algorithms find scores in the $[0.95, 0.99]$ range. Why is this the case? The answer is related to the sparsity of defined links between agents when all $|A|^2$ possible pairs of agents are considered. As most agents do not know most other agents, and there is no basis for determining the similarity (distance) between them, in the vast majority of cases $sim(a_i, a_j)$ is equal to a default value for randomly picked i and j. Therefore, as both metrics take some kind of average of the distances between pairs of agents, the metrics will always be close to the default distance.

Should these low absolute scores deter us from this method? We argue they should not. First, as we have briefly argued above, the nature of this data implies that average measures of cluster cohesiveness will always be close to a default. Secondly, the trend lines show that appropriate choice of k and cluster methodology can effect the sign and magnitude of silhouette scores in consistent ways - for example, when k goes above 60 in Figure 4.4 (b). We take this as an indication that positive results are not simply a coincidence.

4.3.2 Trust link prediction

Our trust link prediction procedure was intended to combine what we perceived to be the best traits of the work of Fang et al. [22] and Mauro et al. [50]. Both works tested the effects of predicting trust links using Multi Faceted Trust Modeling (MFTM) on an item recommendation task.

In [50], a relatively large number of domain specific trust indicators are proposed for the Yelp data set; however, the importance of each trust indicator is not learned in a data driven way. Instead, they selectively enable and disable indicators for each performance test, combining their values by simply taking the average of the enabled indicators. In [22], a relatively small number of generic trust indicators are proposed for an Epinions data set.

The importance of each indicator is learned via logistic regression and performance is tested under a number of different sparsity conditions[8].

We will compare our work more closely to the works of Fang et al. and Mauro et al. in Chapter 5. In our work, we combined indicators proposed in both work and adopted a data-driven indicator importance weighting procedure. In addition, personalization for MFTM via clustering was proposed (but not tested) in [22], an approach which we evaluate here.

Trust indicator list

To quickly restate the goals of MFTM[9], we wish to define a vector of trust indicators over every pair of agents, $\Psi(\overrightarrow{a_i}, \overleftarrow{a_j})$ then use machine learning to approximate the function $f : \Psi(\overrightarrow{a_i}, \overleftarrow{a_j}) \rightarrow y$, where y is some type of trust link.

In Table 4.3 we have listed all trust indicators that we calculated for Yelp data. When an indicator was proposed in the works of either Fang et al. [22] or Mauro et al. [50], we have indicated this in the last column (although some were also adjusted by us, as clarified below). Some indicators are defined specifically for pairs of agents (e.g. the similarity of rating behavior for two agents), while others are defined on the basis of a single agent. In Yelp data, the only explicit trust link present is mutual friendship.

Here, we describe some of these indicators in full detail, in order to illustrate some less obvious indicators. Complete descriptions of the indicators not described here are available in [22] and [50].

Benevolence: Already described in Equation 4.3. In [22], $\bar{r}_j$, the average rating given to item j was replaced with $\bar{r}_i$, the average score agent i gave to items. We made this replacement as a common rating behavior in the Yelp data set was for an agent to only submit 5 star reviews, causing frequent divisions by zero. By comparing to the global average rating of an item, this behavior is no longer an issue. Intuitively, when $benevolence(a_i, a_j)$ is high, agents a_i and a_j may be inclined to trust each other's reviews, as they have reviewed items similarly in the past.

Competence: A threshold value ϵ is used to determine how often the trustee's ratings where "close enough" to the ratings of other agents who had also rated those items to be

[8] We use logistic regression because of its simplicity and interpretability. Note that while this model can only learn a linear boundary between classes, using a non-linear model is also possible. However simply using a non-linear model without clustering would not by itself lead to more personalized recommendations.

[9] A full treatment is given above in Section 2.2.2.

considered "correct".

$$competence(a_i) = \frac{\sum_{j \in R_i} \sum_{k \in I_j} \mathbb{1}(|r_{ij} - r_{kj}| < \epsilon)}{\sum_{j \in R_i} |I_j|} \tag{4.9}$$

Where R_i is all the items the i'th agent has rated and I_j is the set of all agents who have rated item j and r_{ij} is the rating agent i gave to item j. Competence is high when an agent's rating behavior is similar to the plurality of agents. Since ratings on Yelp use a 5-star scale, we used the threshold value 0.5. Intuitively, when $competence(a_j)$ is high, a_j may be trustworthy for agents who consider agreement with popular consensus to be important.

Predictability: A threshold value θ is used to determine how often a trustee's preferences are consistently higher, lower, or similar to the truster's :

$$n_u = \sum_{k \in R_{i,j}} \mathbb{1}(|r_{ik} - r_{jk}| \leq \theta) \tag{4.10}$$

$$n_n = \sum_{k \in R_{i,j}} \mathbb{1}(r_{ik} - r_{jk} < \theta) \tag{4.11}$$

$$n_p = \sum_{k \in R_{i,j}} \mathbb{1}(r_{ik} - r_{jk} < -\theta) \tag{4.12}$$

$$predictability(a_i, a_j) = \frac{max(n_u, n_n, n_p) - min(n_u, n_p, n_n)}{|R_{i,j}|} \tag{4.13}$$

where n_u, n_n, and n_p count how many times the trustee rated an item about the same as the truster, lower than the truster, and higher than the truster respectively. Accordingly, predictability is lowest when $n_u = n_n = n_p$, meaning the trustee rates items better, worse, and equivalent to the truster in equal amounts. This would mean there isn't a justification to expect that the trustee has a bias in any particular direction, relative to the truster. Similar to Competence, we used a threshold value of 0.5. Intuitively, $predictability(a_i, a_j)$ may be important to a_i deciding whether or not to trust a_j, as it is useful to know whether a_j's ratings have a consistent bias compared to a_i.

Visibility: The relative popularity of agent, taking into consideration how much content the agent has produced and the popularity of the most popular agent.

$$visibility(a_i) = \frac{appr(i)}{max_{a_j \in A}(appr(j) \times contr(i))} \tag{4.14}$$

Where $appr(i)$ is the number of public "appreciations" an agent has received from other agents (e.g. likes) and $contr(i)$ is the number of contributions an agent has made (e.g. posts, reviews). Intuitively, when $visibility(a_j)$ is high, a_j may be trustworthy to agents who consider consistent popularity important.

Name	**Description**	**Source**
Benevolence	Equation 4.3, the similarity in rating behavior between truster and trustee.	Fang
Integrity	How similar the trustee's ratings are to the global average.	Fang
Competence	How often the trustee's ratings are within an acceptable range of other agents' ratings	Fang
Predictability	How consistently the trustee's ratings are more/less positive than the truster's	Fang
SocialJacc	rel_{ab}, Equation 4.4, the Jaccard similarity in the truster and trustees friend sets	Mauro
EliteYears	$elite_a$, the number of elite years the trustee has	Mauro
ProfileUp	lup_a, the number of compliments on the trustee's profile	Mauro
Fans	$opLeader_a$, the number of fans the trustee has	Mauro
Visibility	vis_a, the ratio of compliments received to amount of content produced by the trustee	Mauro
GlobalFeedback	fb_a, the number of compliments the trustee's content has received	Mauro
EliteNorm	EliteYears divided by trustee account age in years	
ProfileNorm	ProfileUp divided by trustee account age in years	
FansNorm	Fans divided by trustee account age in years	
FeedbackNorm	GlobalFeesback divided by trustee account age in years	
ItemJacc	Jaccard similarity relative to items reviewed.	
CategoryJacc	Jaccard similarity relative to categories of items reviewed	
AreFriends	Are truster and trustee friends	
AreFoF	Are truster and trustee friends of friends	

Table 4.3: Trust indicators used for Yelp data.

Some of the indicators listed in Table 4.3 were developed by us. For example, we normalized a number of the indicators proposed by [50] by dividing by how many years the

target user had been on the site. This is useful for giving newer users a chance to compete with older users on certain attributes (e.g. how many "fans" they've accrued). We also computed the Jaccard similarity between users with respect to the sets of items they had reviewed and the categories of items they had reviewed, reasoning that these indicators would help to outline the case when users has similar areas of interest. Finally, we checked to see if pairs of users were friends of friends, a potentially useful feature for integrating trust transitivity into reasoning.

When computing these trust indicators, it appears necessary to consider all ordered pairs of agents in the environment, as trust is directed and can occur between any two agents. This presents a significant computational bound on the number of agents that can be considered. Discussion of this issue, and our approach to minimizing this impact is presented in Appendix C. In brief, only pairs of agents where there is significant evidence that the pair have an overlap in interests / social circle are actually considered as candidates for novel trust link prediction.

Classification process

The trust indicators listed in Table 4.3 were used to predict two types of trust links 1) whether the truster had explicitly expressed trust in the trustee (friendship), and 2) whether the truster and trustee had a positive correlation in review scores (e.g. Equation 4.3). Note, in the case where expressed trust was the target of prediction, review score correlation *was* considered as evidence (e.g. included in $\Psi(a_i, a_j)$) and vice versa, although the target of prediction was obviously not considered as evidence.

Following the example set by Fang et al [22], we use logistic regression to learn functions that predict the presence of statistically likely trust links based on the vector of trust indicators computed between pairs of agents. We refer to these functions as "trust link classifiers", as once learned, they classify each ordered pair of agents as either being linked by trust (the former should trust the latter) or not. We used the SAGA solver logistic regression classifier included in the sklearn Python package [57] to learn these functions.

In the case where no clustering was performed, a single classifier was learned for all agents. When clustering was performed, a classifier was learned for each cluster of agents. Thus, each cluster specific classifier learns how the agents in the cluster form trust links in their role as trusters. This makes obvious a substantial tradeoff to this approach to personalization: the more clusters are found (increasing cluster cohesiveness up to a point), the less data available to train machine-learning classifiers (decreasing prediction accuracy). We will discuss other potential approaches to personalization in Chapter 5. For

our purposes, we only learned cluster specific classifiers for clusters that had at least 100 agents and at least 1000 positive outgoing trust links. When a cluster failed to meet these standards, it was assigned a generic classifier, trained on examples from a random sample of users across all clusters. We implemented this strategy in order to avoid training wildly inaccurate classifiers. When training the cluster-specific classifiers, all available data relevant to each cluster was used for training. This is because we are not directly interested in how well each classifier is able to fit each cluster, only on whether this personalization process increases the accuracy of the downstream recommendation task. Therefore, it is not necessary to reserve a test/validation set for any trust link classifier. Note, at this point, test sets of ratings data *are already* reserved for the recommendation task outlined in the next section.

A common problem in link prediction generally is the large class imbalance between positive and negative examples. Put simply, the number of negative examples of trust links in a community of agents (agents that have nothing to do with each other, have never communicated, or who genuinely do not trust each other) grows with $O(|A|^2)$, while positive examples (friends, trust) have much more conservative linear growth. This can be either because humans have an upper limit on how many others they will trust, or, like on Yelp, technical limitations are imposed on the number of allowed friends. Compounding the problem is that there are two kinds of negative examples, which are often difficult to distinguish between. On the one hand, agents a_i and a_j may *not* be friends simply because they have never met. On the other hand, they may have interacted and prefer not to do so again in the future.

Therefore, it is necessary to devise a strategy for training classifiers to deal with this imbalance and ambiguity. One popular method, which we have used here, is to construct balanced training sets by including a random negative link for every positive link. This method has the advantage of requiring no further tinkering to classifiers in order to accommodate a class imbalance. The ambiguity in negative links is ignored as best as possible by simply sampling negative links randomly.

After training classifiers for each cluster, trust link prediction is performed by feeding the trust indicator vector $\Psi(a_i, a_j)$ to the appropriate classifier for the cluster of agent a_i. Ultimately a matrix of trust link predictions $\hat{\Gamma}$ is produced, where $\hat{\Gamma}_{ij} = 1$ if the classifier for the i'th agent's cluster predicts a trust link from a_i to a_j with probability greater than 0.5. It is noted once again that predictions were only made for pairs of agents that were considered to be in the same neighborhood, as described in Appendix C.

4.3.3 Recommender evaluation

Two trust-aware recommender systems were used to measure the accuracy of the trust links predicted in the previous step: TrustMF and MTR. Two systems were evaluated in order to reduce the risk (e.g. of using a a flawed implementation that skews results). TrustMF leverages matrix factorization and gradient descent to optimize predictions of user-item ratings, as described above in Section 2.3.6. We used the Librec implementation of TrustMF for our experiments [31]. MTR is a trust aware modification of a similarity based KNN recommendation model proposed by Mauro et al. [50]. Under this system, the predicted rating for an agent i for an item j is:

$$\hat{r}_{ij} = \bar{r}_i + \frac{\sum_{k \in N_j^\kappa(i)} inf_{ki}(r_{kj} - \bar{r}_k)}{\sum_{k \in N_j^\kappa(i)} |inf_{ki}|} \tag{4.15}$$

Where $\bar{r}_i$ is the mean score agent i has given in ratings, $N_j^\kappa(i)$ is the set of the top κ most influential agents on i who have also rated item j, and inf_{ki} is the influence agent k's recommendation exerts on agent i: a linear combination of the similarity between k and i's past rating behaviour and a trust metric, in our case the probability that k is trustworthy for i according to the predictions of the trust model in the previous step.

$$inf_{ki} = \beta \cdot \sigma(ik) + (1 - \beta) \cdot \hat{\Gamma}_{ik} \tag{4.16}$$

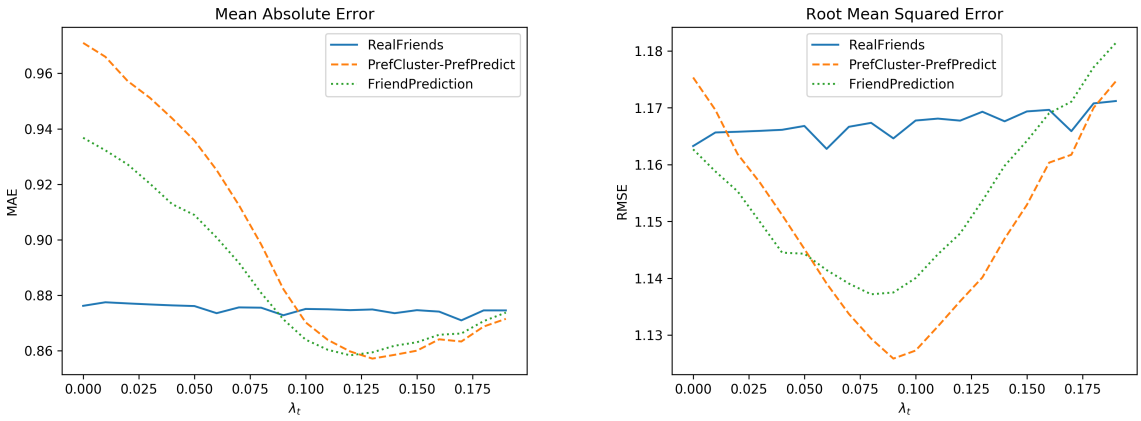
where β is simply a parameter for controlling the weight of trust modeling on the recommendation process. When $\hat{\Gamma}_{ik}$ was undefined (e.g. in the case where i and k are not in the same neighborhood, see Appendix C), a value of 0 was substituted. We modified an implementation of a KNN based recommender system distributed in the Surprise library [34] to test this method.

Accuracy of recommendation was measured by Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE):

$$MAE(\hat{R}) = \frac{1}{|\hat{R}|} \sum_{r_{ij} \in \hat{R}} |\hat{r}_{ij} - r_{ij}| \tag{4.17}$$

$$RMSE(\hat{R}) = \sqrt{\frac{1}{|\hat{R}|} \sum_{r_{ij} \in \hat{R}} (\hat{r}_{ij} - r_{ij})^2} \tag{4.18}$$

where R and $\hat{R}$ are a set of real agent-item ratings and predicted agent-item ratings respectively, and r_{ij} is the rating given by user i to item j. MAE simply captures the average



(a) MAE as λ_t increases.

(b) RMSE as λ_t increases.

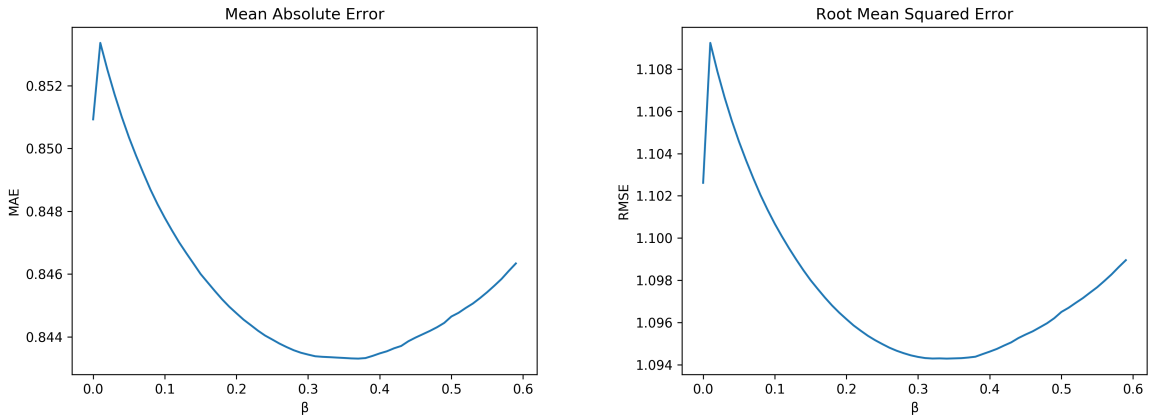
Figure 4.6: Tuning λ_t for TrustMF

unsigned error in predictions across all ratings, while RMSE is more heavily penalized for gratuitously erroneous ratings and less penalized for very nearly correct ratings. These measures can be analogized to the mean and variance of a distribution over prediction error. As measures of error, we prefer recommendations that minimize these measures. Thus for all of the following graphs lower values are better. The sensitivity of RMSE to outliers is a useful property for this application, as grossly inaccurate predictions can erode user trust in future recommendations.

TrustMF has the following significant hyperparameters: the regulation penalty λ, the weight given to fitting the user-user trust matrix (as opposed to the user-item rating matrix), λ_t, and the number of dimensions of the latent space d. We kept the number of dimensions at the default of 10 and the regulation penalty at 0.01. In order to determine an appropriate setting for λ_t, we sampled 10000 users from the filtered Yelp data set and plotted MAE and RMSE over the change in λ_t. Results of this tuning are presented in Figure 4.6. For readability we have only plotted the best performing experiments from each of the main groups[10] (RealFriends as a baseline, FriendPrediction as a non-personalized (MFTM) example, and PrefCluster-PredPredict as a personalized (PMFTM) example). Each data point is the average of three runs with different random seeds, and an iteration limit of 200 epochs. For these preliminary tests, we set the number of clusters at 10.

Recommendation accuracy changes little for the actual trust links in the data set (Re-

[10]See Table 4.2 for the complete list of experiments.



(a) MAE as β increases.

(b) RMSE as β increases.

Figure 4.7: Tuning β for MTR

alFriends) as the importance of trust for recommendation increases, but the MFTM and PMFTM lines form roughly convex curves, reaching a range of minimal values around $0.8 < \lambda_t < 0.125$. For future experiments, we set $\lambda_t = 0.11$.

Figure 4.6 also serves as some encouraging early results, showing both that the impact of predicting trust links reduces recommendation error and a personalized approach can reduce this error further, given the correct weighting of trust importance.

MTR has two significant hyper parameters: the maximum neighborhood size for a user (e.g. the maximum number of peer recommendations that will be taken into consideration), κ, and a social weighting parameter, the value of β in Equation 4.16. In their original work, Mauro et al. [50] set β at 0.1, but did not report on how modifying this variable effects recommendation accuracy[11]. We used all Yelp users from the filtered data set and computed the recommendation accuracy as β changed using a set of predictions based on a single social classifier (i.e. the FriendPredict setup). Results are illustrated in Figure 4.7, showing a clear tradeoff between only considering user-user similarity and incorporating trust. Similar results were seen for the PrefPredict experiments. Accordingly, future experiment were run with a value of $\beta = 0.3$.

We ran experiments with κ set to 50, but also experimented later with modifying the value of κ (to simulate sparsity). As a reminder, this value is the maximum number of ratings that are considered when recommending to a user.

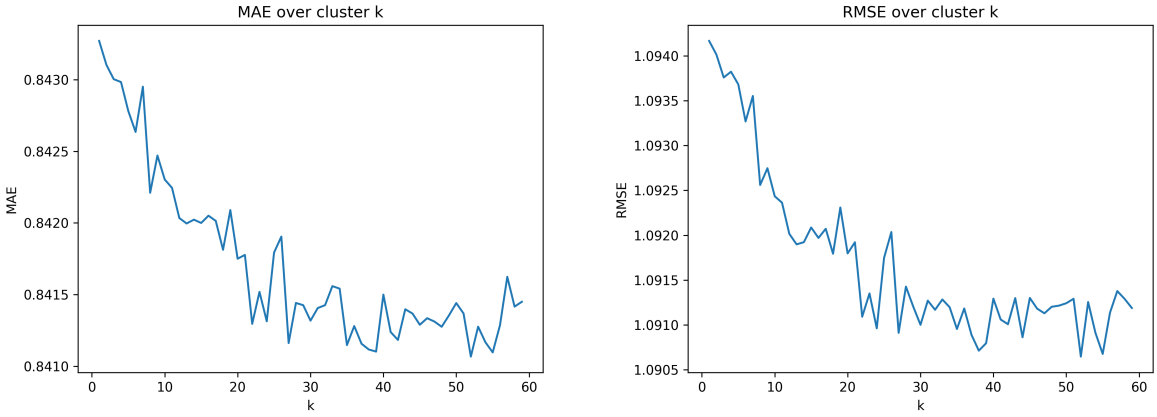
[11]Note, $0 \leq \beta \leq 1$.

Test sets were created by reserving 20% of each user's reviews. For all figures in this section, except in the results shown in Figure 4.6, these reviews were excluded from every step of the process[12], that is, the clustering and link prediction steps did not have access to these reviews. Due to the computation time required to generate and evaluate many of these experiments, only the results reported in Table 4.4 were cross validated. In this case, 5-fold cross validation was used with respect to users, so each user had a distinct 20% of their reviews reserved as a validation set for each of the folds. This validation set was hidden from every step in the pipeline. This validation approach is similar to the ones used in [50] and [22], where results were reported based on the average across folds of a 10-fold cross validation and the average across a complete leave-one-out cross validation, respectively.

4.4 Results

In our original tests, we attempted to set the number of clusters, k, by evaluating the silhouette scores of clusters in a large range for each data set, then simply choosing k based on whichever cluster count had the highest silhouette score. Unfortunately this method was fickle, as the silhouette evaluation is based on a random sample of the clusters, and a single outlier could achieve a minimal score even if other nearby values of k were not optimal. Further, it is basically a heuristic to use cluster cohesion to choose the number of clusters, when ultimately we are interested in improving the personalized trust links. Therefore, we iterated over a range of cluster values and repeated the entire experiment with each choice of cluster score, using the MTR recommender system. Results are illustrated in Figures 4.8 to 4.11. In general, results show that as the number of clusters searched for (k) increases, the error in the task follows a consistent trend of reduction. Note that when $k = 1$, the situation is equivalent to a non-personalized approach (as only searching for a single cluster is equivalent to doing no clustering), and as k increases the granularity of personalization increases. When predicting whether two users should be friends or not, MAE can be lowered by 0.003 points by adding personalization, while when predicting aligned preferences it is only lowered by 0.0005 points. These results are less impactful than the early results seen using the TrustMF classifier in Figure 4.6. That said, the results indicate a consistent trend of improvement as clustering based personalization is applied: a fairly consistent line of decrease in error can be observed in all lines.

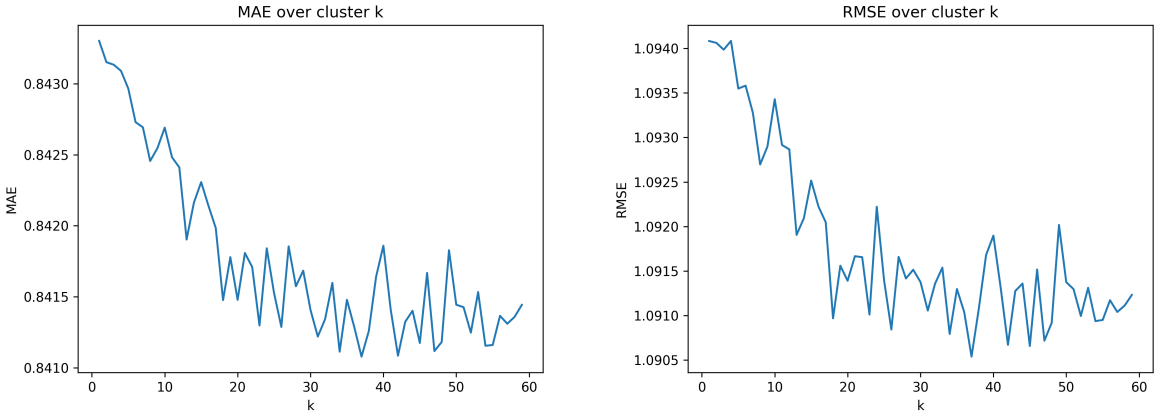
[12]In earlier versions of this work, we only split reviews into test and train sets at the last step (recommender evaluation). This would allow, for example, the clustering step to use data to form clusters which was later being tested on.



(a) MAE as k increases.

(b) RMSE as k increases.

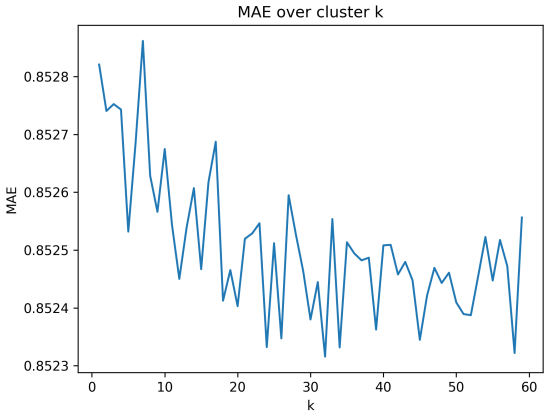
Figure 4.8: Effect of k on error for SocialCluster-FriendPredict with MTR



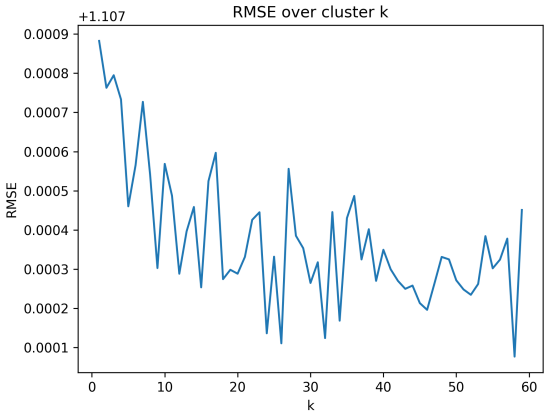
(a) MAE as k increases.

(b) RMSE as k increases.

Figure 4.9: Effect of k on error for PrefCluster-FriendPredict with MTR

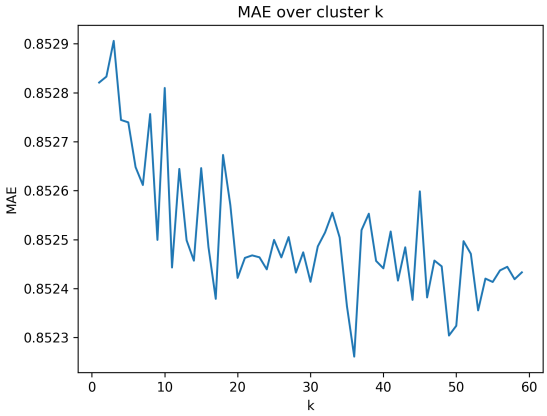


(a) MAE as k increases.

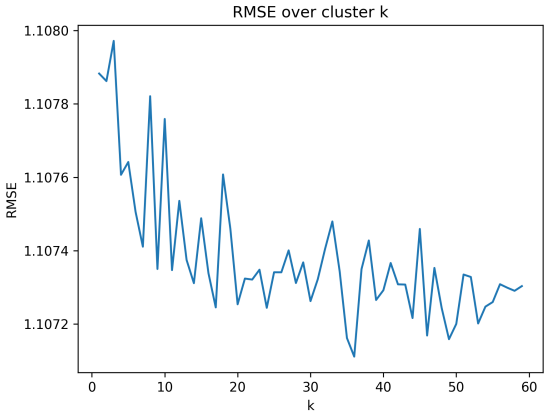


(b) RMSE as k increases.

Figure 4.10: Effect of k on error for SocialCluster-PrefPredict with MTR

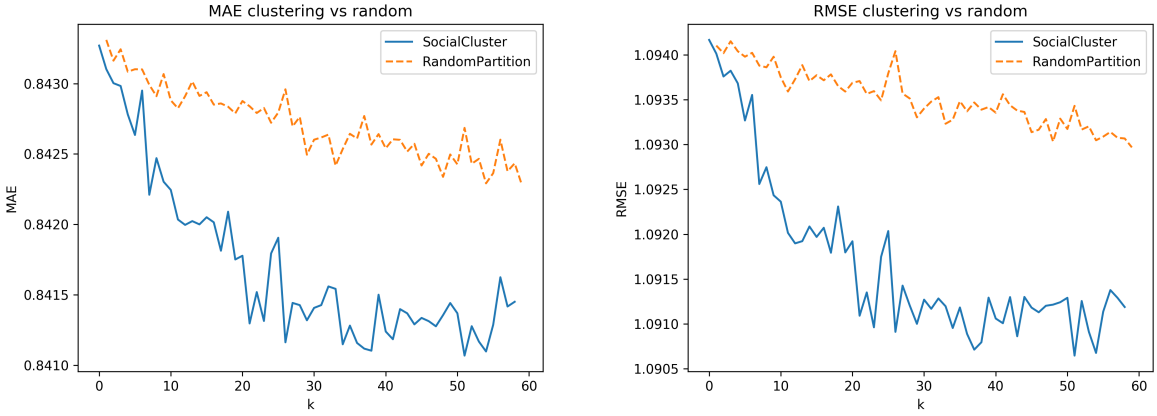


(a) MAE as k increases.



(b) RMSE as k increases.

Figure 4.11: Effect of k on error for PrefCluster-PrefPredict with MTR



(a) MAE as k increases.

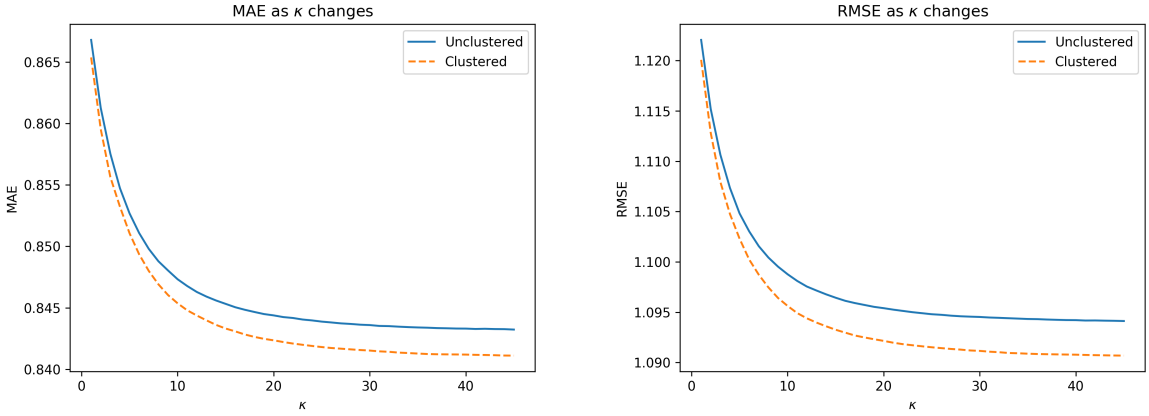
(b) RMSE as k increases.

Figure 4.12: SocialCluster-FriendPredict versus RandomCluster-FriendPredict.

We wished to verify that the results illustrated in these figures are indeed improving because our clustering technique was finding groups of similar users, which allowed the prediction techniques to learn more personalized classifiers for these groups. For instance, it is conceivable that splitting users into groups and learning multiple classifiers is helpful regardless of the groups picked, as this procedure would be similar to bootstrap aggregating [6], which allows simple classifiers to model multiple weak correlations in data. To test this, we repeated the FriendPredict experiment, but clustered agents into k clusters randomly[13]. The results illustrated in Figure 4.12 compare this random clustering to clustering by social circle overlap. This figure clearly shows that the reduction in error is largely due to the non-random clustering approach. The solid and dashed lines start off identically at $k = 1$ on the x-axis (no clustering) but as the numbers of clusters increases, the error decreases, for the case where social clusters are used. We take this as evidence that the clustering technique is improving accuracy because clustering genuinely enables more personalized predictions, not simply because the number of models being learned has increased.

We also experimented with modifying the κ variable on MTR, effectively simulating sparsity, as this variable controls how many peer advisers can be considered for a recommendation. Results illustrated in Figure 4.13 compare a accuracy on the SocialCluster-FriendPredict task, comparing the error rates for a single cluster (unclustered) and for 55 clusters (clustered). Overall, the gap in error rate is most dramatic when a larger κ value

[13]This random clustering essentially partitions the data set into k random samples (a close emulation of bootstrap aggregation).



(a) MAE as κ increases.

(b) RMSE as κ increases.

Figure 4.13: Effect of κ on accuracy for MTR.

is used, but the advantage for a clustered approach applies over the range of values.

Finally, in Table 4.4 we present the results of a 5-fold cross validation over user ratings for these tasks. Best results are bolded. There are a number of interesting results. First, the conceptually simple MTR system outperforms TrustMF across the board, despite the fact that the TrustMF system was allowed to run for a much greater period of time in order to reach convergence. This gap in performance is often dramatic, for example, in the best cases for each system, MTR has a MAE that is 5% lower than TrustMF, and a RMSE that is 1.6% lower.

	MTR		**TrustMF**	
	MAE	**RMSE**	**MAE**	**RMSE**
RealFriends	.8610	1.1196	**.8879**	1.1205
FriendPredict	.8453	1.0960	.9063	1.1179
SocialCluster-FriendPredict	**.8434**	**1.0932**	.9120	1.1179
PrefCluster-FriendPredict	.8436	1.0936	.9077	1.1189
PrefPredict	.8551	1.1105	.8984	**1.1109**
SocialCluster-PrefPredict	.8551	1.1103	.8987	1.1111
PrefCluster-PrefPredict	.8551	1.1103	.8987	1.1111

Table 4.4: Recommendation error results.

On the better performing MTR recommender system, the best results are achieved when predicting friendship links rather than predicting preference correlation. This makes, sense, as the MTR system already considers observable user preference (see Equation 4.16), thus predicting new instances of aligned preferences is not likely to add much new information. The best performing task, by a small margin, is the SocialCluster-FriendPredict task, which basically reconfirms the findings presented earlier in this section (with the added certainty of being averaged across folds). Clustering did not have an appreciable effect (at least not at the scale of 10^{-4}) for the preference alignment prediction task.

Note that, in the case where an improvement was seen, although the scale of the effect appears small it is clear from the previous graphs that this is not merely due to statistical variance. For example, Figure 4.8, clearly shows that the decrease in error between Friend-Predict and SocialCluster-FriendPredict is due to the increasing the number of clusters.

On TrustMF, results are more tightly grouped and there is quite little appreciable difference between experiments. As this system is more conceptually complex than MTR, it is difficult to interpret exactly why this might be the case. Interestingly, the best performing task for MTR is the *worst* performing task on TrustMF. Clustering does not have a positive effect in these experiments, and in the SocialCluster-FriendPredict task actually seems to harm the performance. Our early experiments with this recommender system (presented in Figure 4.6) suggested that there might be more interesting differences between approaches, but this was not the case in these final results. We speculate that because these earlier results were computed using different techniques to select Yelp users (randomly selecting 10000 users versus our final strategy of selecting all 30000 users with more than 20 reviews submitted), the underlying distributions of ratings may have been different enough to cause this change.

4.5 Conclusion

In this work, we evaluated the effect that personalization via clustering had on the accuracy of a trust link prediction task. We accomplished this by predicting novel trust links on a data set of Yelp users and measuring accuracy of these predicted trust links via a down-stream item recommendation task.

Our results confirm the results of earlier work, showing that the option of predicting novel trust links results in better performance than using the explicitly stated trust links for the recommendation task. Further, our results show a small but consistent improvement in recommendation accuracy when clustering is used to determine groups of similar agents

and distinct trust prediction models are learned for each group of agents with the MTR recommender system (e.g. Figures 4.8 to 4.11). We showed that this improvement was not simply the result of the fact that more classifiers were trained, as randomly splitting users into groups does not improve accuracy nearly as much as the clustering technique does (Figure 4.12). While our early results with TrustMF inspired confidence, and we hoped to see more dramatic improvement in recommender accuracy from these experiments, the final results show that, while consistent, the improvements in accuracy from the procedures outlined here are small. We will comment on ways these techniques could be improved, and potential avenues for future work, in Chapter 6.

In addition to the experiments with personalization (which explored multiple approaches for clustering users), we produced a comprehensive MFTM solution, combining techniques from the literature with novel features. We also make clear the applicability of MFTM to social networks. We experimented with predicting two types of trust links: explicit friendship (the FriendPredict experiments) and implicitly stated preference alignment (the PrefPredict experiments) and evaluated the utility of the predicted links derived by our methods, using two distinct trust-aware recommender systems. We found that the preferred target of trust link prediction can vary with the desired use-case: it was not clearly preferable to predict friendship links or preference alignment links. On the MTR system, which already strongly considers user preference alignment, our experiments performed better when predicting friend links, while on the TrustMF system predicting preference alignment between users produces (slightly) better recommendation accuracy.

Chapter 5

Discussion

In this chapter we compare the solutions we developed in Chapters 3 and 4 to similar works, drawing out the differences and similarities between them. We start by presenting a case study of a recent social network message recommendation system that inspired our work, explaining how our two works are related. Then, in an effort to widen the scope of consideration, we outline two fields of significant research aimed at applying trust models to social networks which differ significantly from our own: message recommendation and information diffusion analysis. We follow this by comparing the main topics of our two works (quantifying discussion outcomes, multi-faceted trust models and personalization) to similar works. Finally, we engage in a discussion regarding recurring challenges in the trust modeling research field.

5.1 Case Study of BayesTrust

BayesTrust [68] (see also [67, 64]) is a recent multiagent trust-based message recommendation system that filters messages for a particular agent based on the advice of other agents in the network. The goal of this system is to save the time and attention of social network readers by reasoning about which messages will be most beneficial to them, and filtering out or flagging messages which do not appear to be beneficial. This system is modeled as a Partially Observable Markov Decision Process (POMDP) that considers one message at a time in a purely episodic manner. The POMDP is formally defined by a tuple of parameters: $(S, A, O, T, \Omega, R, \gamma, h)$.

- S: The state space for a message, defined as {good, bad}

- A: The action space for the agent, defined as {accept, reject, elicit_advice}
- O: The set of observations associated with actions. In this work, the accept and reject actions always result in a nil observation, while requesting advice results in a tuple (r, m, c), where r is the rating a peer has given to the message, m is the similarity that peer has to the recomendee, and c is the credibility of the peer.
- $T = Pr(s'|s, a)$: The transition function for message state given the current state and the chosen action.
- $\Omega = Pr(o'|s', a)$: The probability of seeing observation o given action a is taken at t_i and the state at t_{i+1} is s'.
- $R : S \times A \to \mathbb{R}$: The reward function representing the utility of each state-action pair for a particular agent.
- γ: Reward discount factor.
- h: Horizon (finite or infinite).

For this work, T is an identity function on states (i.e. $T(s'|s, a) = 1$ iff $s' = s$ and 0 otherwise), as a message's potential utility for a recomendee (good/bad) is not expected to change as time goes on. R could be defined differently for each individual, allowing the system to implement some level or personalization by reasoning based on the recomendee's sensitivity to risk (e.g. for some users seeing a harmful message is more upsetting than for others). In actual experiments only one reward function was used, where requesting advice was always a positive utility action (so long as it was possible). While it is conceivable that some situations might warrant imposing a cost for soliciting advice (e.g. if advice is overwhelming and time is limited), the chosen reward function causes the agent to repeatedly request advice until all advice was exhausted before making a decision to show or hide a message. As one would expect, showing a good message or hiding a bad message resulted in positive reward, while hiding a good message or showing a bad message resulted in negative reward.

While the system as proposed is purely episodic, it is suggested that the observation function Ω could be learned offline with a supervised technique or online using a reinforcement learning technique (e.g. by eliciting the feedback of a user after a message is shown to them, determining if they really think the message is good or bad[1]). User specific reward

[1]Note, this would only allow feedback to be collected in the case where a message is shown.

functions could be implemented in a similar manner. It is suggested that O can be significantly expanded in future works, incorporating (for example) features of the underlying message and its author.

Given this POMDP formulation, the system reasons about the expected utility of showing a message to a user, by repeatedly requesting advice from the peer network, iteratively updating the belief over the state of the message.

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

Where b_t is the belief that the message is of the given state at time t. After advice from the peer network is exhausted, it is straightforward to reason about the expected utility of either showing (recommending) or hiding (rejecting) the message from the user:

$$EU_{show} = b(G) \cdot R(show, G) + b(B) \cdot R(show, B)$$
$$EU_{hide} = b(G) \cdot R(hide, G) + b(B) \cdot R(hide, B)$$

where b is the belief that the message is either G, good or B, bad (i.e. the product of the probabilities of seeing the observed sequence of advice given the underlying state, according to Ω) and R is the reward function that encodes the reward of showing or hiding good and bad messages respectively. A user will be shown the message only if $EU_{show} > EU_{hide}$.

We were heavily inspired by this work, and sought to replicate some of its best features while filling in some of its missing pieces. Sardana's principled Bayesian approach supports predictions of trustworthiness of messages in social networks based on any parameters chosen to be modeled, but he examined in most detail the scenario where the observations are ratings, credibililty of the rater and similarity of the rater to the user. The idea is that over time certain combinations of values for these observations either strengthen or weaken the belief that the message is worth showing to the user. Sardana mentioned two significant steps forward that could be taken from his work. He suggested that modeling the trustworthiness of authors of messages would be valuable to incorporate into the reasoning (perhaps helping to inform the priors of his Bayesian approach), but acknowledged that to date he had focused on modeling only the raters of those messages. He also believed that personalized recommendations could be achieved by allowing each user to have a distinct reward function: some users may have different sensitivities or tolerances with respect to trust and distrust. But in his experimentation, he opted to select a uniform reward function for all of the agents.

In our works, we have sought to provide tools to help fill in these gaps. The personalized trust approach we presented in Chapter 4 gives an example of how a trust model can be

applied to a social network and implement personalization, while the experiments with quantifying discussion outcomes in Chapter 3 provides a new tool for understanding the effects of an agent's interactions in a conversational social network; this then provides insights into the trustworthiness of the author.

Therefore, we see our work, while quite different, as inspired by and extending many of the ideas presented in this model.

5.2 Applying Trust Models to Social Networks

In this work we spoke at length about the difficulties that emerge in applying trust models (traditionally defined over marketplace-like domains) to social networks. In particular, the project described in Chapter 3 was specifically targeted at improving the applicability of trust models to social networks, by developing a new method of quantifying the outcome of a discussion. We also argued that, of the trust models we surveyed, multi-faceted trust modeling was among the best candidates for applying to social networks, as we expect the expression of trust in these complex environments to depend on many factors, and a solution which can flexibly integrate these factors and be adapted to the peculiarities of specific social networks is desirable.

However, our works are not the first to approach this topic, nor have the views we've expressed here summarized all avenues of approach towards this problem. In this section, we briefly summarize two popular approaches to the analysis of trust online: message recommendation and social graph analysis.

5.2.1 Message recommendation in social networks

In addition to BayesTrust, significant work has been done in a message-recommendation context (e.g. filtering the message feed on a user's home page) where inter agent trust is considered an important feature. Message (e.g. posts, comments, status updates, tweets) recommendation is a considerably different task than content (e.g. movies, music) recommendation, as the authors of a messages in social media contribute new messages extremely frequently, in quick reaction to each other, and the content of messages is expected to be highly correlated with the author's opinions.

For example, an approach specifically aimed at social networks is the Learning Object Annotation Recommender System, (LOAR) [12], which models which agents have good

histories of submitting quality comments on educational material online. This system combines global and local information (e.g. overall popularity and subjective opinion) to determine which messages are best to present to students. While the system relies primarily on the number of up-votes and down-votes a comment receives to measure its authors' success, the integration of personal rating history and weighting of votes by voter reputation makes it much more sophisticated than a mere measure of popularity. A notion of innate user credibility (e.g. applicable to professors and TAs in a learning environment) was added to LOAR in [65], enhancing the system's ability to deal with popular rumors and folklore. This notion gestures towards a tension between what the masses believe is true and what authorities define as true, a concept with connections to interesting philosophical and practical considerations, which we will discuss in Section 5.4.3.

5.2.2 Graph analysis and social ties

Many approaches to modeling trust in social networks have focused on the social connection graph and examining information diffusion dynamics, often excluding an analysis of other attributes of agents and their interactions.

For example, in Tong et al. [76], information defusal analysis is used to model the spread of rumors and misinformation online, a critical application for trust models in social networks. This work also proposes counter-measures in the form of an approximation algorithm for determining where in a network to seed factual information in order to increase the odds of halting the flow of misinformation. A similar approach is explored by Shao et al. [73]. These approaches limit their scope of attention to the flow of information through the network, and particularly which nodes certain information (e.g. known hoaxes) passes through.

In Seth et al. [72] the message recommendation task is approached with a significant social graph analysis, including clustering the social graph to find communities and estimating the credibility of communities. This work also implements insights from sociology, attempting to tag links in the social graph as either strong or weak ties (based on the extremely impactful 1977 paper by Granovetter [30]). This work performs well on real world data, implementing a recursive definition of author credibility that could be applied to a moderately sized data set.

In our work, little attention is given to the explicit social or implicit information defusal networks that exist in social networks. In Chapter 4, we focused on predicting novel links in the network, under the assumption that there are many possible trustworthy partners

for network participants that they are simply unaware of, while in Chapter 3 the network of social connections was not considered.

5.3 Similar Works

In this section, we compare our works in Chapters 3 and 4 to works that take on similar topics and with similar inspirations.

5.3.1 Outcomes of discussions

In Chapter 3 we proposed an approach to measure outcomes of discussion starting comments by attempting to quantify the qualities of a discussion starting comment based on the replies to that comment. Other works have been done to quantify the quality of online discussions, although not always in a trust modeling context, and rarely by examining entire discussions. We believe our work is the first that attempts to predict the presence of anti-social behaviour in a discussion starting comment solely on the basis of how that comment was replied to (for discussions of an arbitrary depth and branching factor) making it somewhat difficult compare the work directly.

In Brennan et al. [7] an SVM trained on lexical features of text comments is shown to perform better than chance at predicting well received comments on the social link sharing site Slashdot. While our approach does include analysis of lexical features, we take a significantly different route from this work by focusing on the text in reactions to a comment rather than the comment itself. Further, we expand our analysis to predict the presence of multiple types of attributes of a discussion starting comment.

Choi at al. [15] created a comprehensive statistical analysis of conversations on Reddit, invoking many of the same concepts as our works. They focused on discovering correlations between discussion size, text difficulty and document relevancy in discussions, revealing some correlations between them, and expanding their analysis to consider multiple communities. While this work is highly relevant to our own, its focus is largely on describing relationships between relatively anodyne qualities of comments, rather than attempting to predict the types of bad behaviour we are interested in. The authors propose that the results of their analysis are useful for network administrators to better describe the qualities of the discussions in communities.

Liang [46] build a regression model to predict the scores of comments on the /r/Techsupport subreddit, based on features such as discussion size and depth and various attributes of the

discussion participants. Like in our work, they consider features on the individual comment level and on the basis of sets of comments. Their analysis is more heavily focused on the social connections between the various discussion participants, and finds strong statistical evidence of an association between these features and comment score outcome. They do not, however, consider any features extracted from the text of comments.

5.3.2 Multi-faceted trust modeling

Our work in Chapter 4 was heavily inspired by the works of Mauro et al. [50] and Fang et al. [22]. All works have a similar structure: they propose a multi-faceted trust model and test it on a recommendation task on a data set harnessed from a site with item rating component. We sought to extend these works by combining the best features from each of them while testing the effects of a personalization step. In particular, Mauro's work developed a large set of trust indicators on the Yelp data set, while Fang's work proposed a smaller set of relatively generic indicators that could be used on the Epinion's data set. In our work, we combined these indicators when testing on the Yelp data set, with the goal of achieving a more comprehensive model of user to user trust formulation. While Mauro's work proposes a large number of trust indicators, it does not seek to weight the importance of those indicators in a data driven manner: they instead experiment by taking a non-weighted average of a subset of the indicators. Like in Fang's work, we have used a logistic regression to find weights for these indicators that fit the data set, believing this method to be a more principled approach to the problem. In Fang et al. [22], the authors included the modeling of distrust, using the distrust links in the Epinions data set. We did not attempt to replicate this; however we did some preliminary investigation of Epinions data in order to expand the environments examined under our approach. Conceptually, the approaches taken to personalize recommendations we undertook on the Yelp data would be easily transferable to this data set. See Section 6.2.2 for more details.

The concept of clustering users before learning weights for trust prediction, as a method of personalization, was proposed parenthetically in Fang et al. [22]. Our work in Chapter 4 was largely conceived as a test of whether or not this suggestion would indeed increase the accuracy of trust prediction.

We acknowledge as well that other trust modeling researchers have promoted the integration of different facets. For example, the trust-based reasoning for vehicular as-hoc network (VANET) environments [52] examines experience-based, priority-based, role-based and majority-based components, in order to provide a richer basis for decision making in this setting. With our work, we specifically focus on how to weight the different factors under consideration, and also on identifying a useful set of trust indicators.

Another work which has relevance is that of Gilbert and Karahalios [27]. While not an artificial intelligence paper, the authors present a multi-faceted statistical analysis of the factors which affect tie strength between pairs of users in social media. They found that a set of 74 variables collected from a the Facebook accounts of participants could be used to predict, with high accuracy, the answers these participants gave to survey questions designed to model social tie strength with their friends on Facebook (e.g. "How comfortable would you feel asking this person for a loan?"). This work presents strong evidence for the notion that trust (i.e. as an aspect of a strong social tie) can be predicted between agents based on relatively simple data extracted from interaction history on social media.

5.3.3 Personalization

Given the subjective nature of trust, it is clear that accurate trust models need to incorporate some level of agent-specific personalization. It is feasible to model reputation or popularity without such personalization (as these metrics do not depend on individual opinion[2]), but not trust. However, given the sparsity of data in most networks, the cold start problem[3], and computational limitations, it is not typically feasible to give each agent a completely distinct model.

Our approach to personalization in Chapter 4 was to determine clusters of similar users and learn trust link classifiers on the basis of these clusters. This approach was suggested by Fang et al. [22], but other approaches have been attempted in the trust modeling space.

The Personalized Trust Model developed by Zhang and Cohen [87] can be seen as an extension of the Beta Reputation System [38] that computes both a private and public trust factors for integrating the advice of some other agent. An agent's private trust factor is based on the similarity in advising behaviour between the trustee and the truster, while the public trust factor is based on how similar the trustee's advising behaviour is to average advising behaviour (similar, but not identical, to the Competence trust indicator proposed in [22]). The final trust prediction for some truster-trustee pair is then a weighted combination of the private and public trust factors. This weighting is based on the overlap in the number of common items the truster and trustee have advised on (rated), thus giving higher weight to the private trust factor when there is more basis for comparing the two agents directly with respect to past behaviour. Effectively, this system implements personalization and attempts to solve the cold-start and sparsity problems

[2] See Appendix A

[3] That is, the problem of giving personalized recommendations to a user who has just joined the network and has not expressed any beliefs, opinions or preferences.

by implementing generic predictions under uncertainty about individual preferences, and offering progressively more personalized predictions as more data becomes available. However, their formula for assigning weight to personal and private trust factors is basically a heuristic, as the settings for appropriate error and confidence bounds are not derived in a data driven manner. Our work attempts to implement personalization in a data-driven way, by identifying clusters of similar users and learning their trust formulation procedures at a cluster level[4].

Stereotype trust suggests that a group of users meeting the definition of some stereotype can be expected to behave similarly, and thus trust can be reasoned about with respect to the stereotypes rather than with respect to individuals. Stereotyping induces a partitioning of users, not dissimilar from our introduction of clustering in order to enable personalized solutions for users. Stereotype trust has often been used to help with cold start problems for trust-based recommendation [8] and some recent effort has focused specifically on how best to represent these stereotypes in order to perform efficient processing [23]. In contrast, we view the clusters that are identified as being the avenue for supporting differing recommendations to users, and derive these groupings based on data analysis through the network.

The usefulness of stereotypes towards improved trust modeling has been examined by other researchers who may also derive benefit from examining our data-driven methods. The StereoTrust Model developed by Liu et al. [48] implements personalization by allowing each agent to define its own grouping function for partitioning the set of other agents via stereotypes. For example, an agent may decide to stereotype based on stated interest, location, seniority, etc. This is intended to model the subjective assumptions humans apply in every day life. The agent then uses a trust estimation function (again, inspired by the Beta Reputation System [38]) to reason about their trust with respect to groups defined by stereotypes, rather than with respect to individuals. The trust an agent $\overrightarrow{a}$ has in another agent $\overleftarrow{b}$ is then computed as a weighed average of the trust $\overrightarrow{a}$ has in all the groups that $\overleftarrow{b}$ is a part of. Thus, by partitioning the set of agents in the environment into groups and reasoning about trust with respect to these groups, the data sparsity problem is reduced. However, it is unclear how the cold-start problem is alleviated by this system. This system implements personalization by allowing each agent to specify its own stereotypes, although in practice it's not clear how this information would be elicited from real users. This approach relies on the notion that members of a group will act similarly,

[4]Fleming [25] also proposes a progression in user modeling from assumptions about general users to ones about individuals but they also suggest an intermediate phase of learning more about groups, via stereotypes. Our consideration of clusters fits well within this vision.

but by allowing individual agents to define groups arbitrarily, the usefulness of this notion is under a certain strain. Without the ability to statistically analyze large amounts of data from the environment, it is unclear how individual agents could be expected to create stereotypes that define groups which actually have some cohesiveness of behaviour. In their actual implementation, stereotypes were implemented based on rating similarity, so no agents actually had an opportunity to specify their stereotypes, and in practice this solution turns out to be a complex approach to reach the same end goal as, for instance, clustering users based on rating behaviour (as we have done).

With respect to our clustering approach to personalization, it is worth noting clustering based approaches have been used in other contexts for trust modeling. For example, the StereoTrust model above uses stereotypes to ultimately clusters users such that they can be treated similarly [48]. Clustering has also been used as the basis for trust models, for example, clustering based on behaviour in order to distinguish between honest and dishonest reviewers [2, 18, 47].

5.4 Challenges in Trust Modeling

In this final discussion section, we change focus significantly, surveying a number of important challenges that concern the trust modeling field, and especially the application of trust models to social networks. Our hope is that this section will inspire reflection on what has been done and what is to be done.

5.4.1 Data availability

A persistent difficulty in applying trust models to social networks has been in finding appropriate data set and evaluation procedures. Most previous attempts to apply trust models to real social networks have relied on networks that included a significant content rating component, such as Yelp, Epinions, and FilmTrust. As explained earlier, these networks are attractive primarily because of the ease of harvesting objective test sets from the data extracted from them. How to tell if two agents should *really* trust each other? Simply check the correlation between the ratings they have given to content - if it's positive, they should trust each other.

This advantage has led to some undesirable effects: Epinions and FilmTrust have been defunct for over a decade, yet these data sets are still commonly used because of this advantage, and the most popular social networks of the day (Facebook, Reddit and Twitter)

are often ignored because of their lack of a significant content rating component. Of course, these modern networks share some of the blame: in the wake of the Cambridge Analytica scandal, many networks may be loathe to share data, even to academics, for fear of new scandals being unearthed.

This trend of using old and well-trodden data sets, which we have followed in Chapter 4, unfortunately represents a weakness which is common in the trust modeling and in many fields of AI more broadly: a lack of interest or resources for interrogating actual users of systems. As a foil to this trend, we note the work of Gilbert and Karahalios [27], where 35 participants were recruited for the experiment. The authors had access to the data that the participants agreed to share with them - it was not necessary to convince Facebook to produce a data set for the researchers. After the statistical analysis, a qualitative analysis was performed to help contextualize the errors in the system by interviewing the participants. Cooperating more closely with the users of online social networks in this way will likely be impactful for future trust modeling research.

5.4.2 Filtering vs flagging

In this thesis, we worked under the assumption that once trust models had identified agents whose content is trustworthy (e.g. they appear to produce content which induces good outcomes for the consumers of that content), then their content should be made more visible to those users who benefit from it. This is usually phrased positively, i.e., "if an agent is trustworthy then we might recommend their content", but it is interesting to also consider the negative phrasing: "if an agent is not trustworthy, then we might not recommend their content". While in a strict logical sense agreeing to the former does not imply agreeing to the latter, both properties are likely to be considered desirable in a message or content recommendation system: promoting "good" content and discouraging "bad" content. This negative formulation shows that, in essence, we plan to filter content from the view of users. Under a critical lens, what we call filtering here might just as easily be called censorship. This points towards an important question that needs to be considered by the trust modeling community, especially when these systems are applied to content recommendation tasks. To what extent should content that is expected to be untrustworthy be hidden from a user's attention? Does this hiding of content rob users of agency? If so, how can agency be put back in the hands of the users of the system? We call the unifying tension underlying these questions the "filtering vs flagging" problem: whether information that probabilistic models deem to be untrustworthy be completely hidden from a user (filtered), or presented with a warning or additional context (flagging).

While we have argued that some level of content filtering is basically necessary in large social networks in order to deal with content overload, there clearly exists a tradeoff between strict filtering and strict flagging. A strict approach to filtering potentially saves time and resources for users and protects them from objectionable content, but potentially leaves them with little control over what information they are shown and puts them in the a state of "unknown unknowing": not even knowing what content is out there that they don't know about, as the system does not make it clear which content is being removed from their attention. A strict approach to flagging potentially empowers users to make their own decisions about which content to trust and educates users about the true nature of discourse on the platform, but could lead to information overload and expose users to extremely objectionable content. In light of this tradeoff, it is worth further discussing potential approaches.

For example, consider hashtags on Twitter. Hashtags are short words or phrases preceded by a # character which are used by the authors of tweets (short messages on Twitter) to indicate that their tweet is referencing some topic or idea. Users can search for tweets that use a particular hashtag, effectively partitioning the tweets on the site into implicit topic forums. These hashtags often consist of multiple English words without spaces or upper case to delimit them, and can be difficult to parse unless one knows the context that they refer to. When a new hashtag begins to be used, it is not always clear what the hashtag refers to, even to human readers.

Since multiple hashtags are often affixed to a single tweet, and since multiple ideologically related hashtags are often used by the same author, it should be possible to find the closest neighbors of a relatively new hashtag by developing a distance measure based on tweet-level and author-level co-occurrence, especially for hashtags related to politically charged issues. This information could then be used in a number of ways. For users who express a dislike of an old and well known hashtag, e.g. #lockherup (a reference to the phrase "lock her up" and used to express dislike of Hilary Clinton during the 2016 American election), we could consider authors and tweets which make use of hashtags which are close neighbors of that old hashtag to also be less desirable to the user.

But should this new content which is closely related to undesired content be hidden from users? This would constitute a strict filtering approach. Perhaps simply exposing the relationships between hashtags to users could allow the users to make informed decisions about the content they are viewing. By informing a reader that a new hashtag appearing in a tweet, e.g. #qannon (a reference to a complex right-wing conspiracy theory), is a close neighbor of #lockherup, the user could make an informed decision about whether to trust the opinions of the author. This information could be exposed by simply allowing a user to see the top-n closest neighbors to a hashtag by hovering over it with their mouse, or by

appending the information to the tweet. This approach has the advantage of offering the user critical information in a timely manner that is only accessible via a complex statistical analysis. It is also empowering, giving the user the agency to engage critically with the content and decide for themselves whether to trust the content or not.

Whether or not users have the attention and desire to deal with such information is an open question, and it is likely that a balance between filtering and flagging will turn out to be the most useful. For example, information that is deemed likely to be untrustworthy for the *average* user could be presented to an individual user with a warning or extra context. Once the individual expresses their own feelings towards the content, this feedback could be incorporated into a personalized filtering algorithm. This personalized approach to filtering, which uses flagging to elicit individual preferences, could potentially strike a good balance: preserving the time and attention saving benefits of filtering while giving the user agency to choose which types of information are filtered out for them.

5.4.3 Top down vs bottom up

The potential for individual users to decide which information is trustworthy to them, which may be quite different from what is trustworthy to the average user, gestures towards another major tension in trust modeling: who decides what makes content/agents trustworthy?

We have stated multiple times that trust must be treated as a subjective phenomenon, and this poses little problem when the context of trust is limited to relatively private interpersonal interactions where subjective preferences impose no reasonably likely externalized consequences. For example, if a buyer in a marketplace *prefers* low quality goods and long shipping times, there is very little practical or moral ground on which to criticize their preference, and it is not too controversial to propose that a trust model should be able to learn which kinds of sellers are likely to be trustworthy for that buyer. In a market, we expect that agents have the right to express their preferences by engaging in mutually beneficial, private transactions, and getting in the way of this expression is contrary to the beliefs of many. The situation is more complex when the interactions between agents are public and have the potential to cause external consequences. For example, if a reader in a social network prefers content that presents racist and hateful rumors as fact, there are realistic reasons why outsiders might oppose the idea of a trust model learning this user's preferences and acquiescing to them. For example, third parties may believe that the very existence of racist rumors is immoral and damages social cohesion, or they might be concerned that the reader will become radicalized or desensitized by their media diet and that

racist ideation will creep into their every day behaviour. In this case, the preferences of the user are potentially in conflict with the public good, and it is unclear that the individual user has a right to express their preference.

These examples are illustrative of the tension that emerges in the "top down vs bottom up" problem. We use the term "top down" to refer to the types of processes where law makers, administrators and public thought leaders are given the power to decide which content and behaviour is acceptable or not, and the term "bottom up" to refer to the types of processes where users, employees and citizens decide which content behaviour is acceptable. As in the last subsection, it is clear that there is a tradeoff that exists in prioritizing these approaches. Further, it is clear that both the "filtering vs flagging" and "top down vs bottom up" problems are issues which, while important concepts to grapple with in the design of trust modeling systems, have broad applicability and occur in many spheres of public life.

Both projects described in this thesis take a "bottom up" approach to trust modeling. We harvest data of user activity and use machine learning to approximate functions which describe, in aggregate, how users react to anti-social content and formulate trust. In general, we have not imposed notions of acceptability or factuality as determined by outside sources. In Chapter 3, we conjectured that it would be useful to detect when agents were engaging in anti-social activity, presuming that under a reasonable definition of anti-social activity this behaviour was undesirable and could be integrated into trust models in order to discourage this behaviour. However, were these detectors integrated into a trust model like the one proposed in Chapter 4, the decision of whether users who routinely engaged in anti-social behaviour would be more or less trustworthy than average users would be driven by data, and would not necessarily reflect our presumption.

Further, as we mentioned, in Chapter 3, the detectors for anti-social behaviour do not assume that the reactions from other users are always negative or offended. While we guess this may be the case on mainstream communities, in fact we are searching for any association between user reactions and anti-social behaviour. Therefore, were this process repeated on data harvested from extremist communities, it is entirely likely that the system would learn that, for example, hateful speech is associated with praise in the reactions from the community. We argue that this is not a specific weakness of our approach, but is in fact a weakness inherent in taking a predominately bottom up approach. Without an outside authority to define what kind of behaviour is and isn't acceptable (i.e. top down approach), the definition of unacceptable behaviour is entirely *a posteriori* and thus dependent on what sort of behaviour is in fact accepted in the community in question.

In Chapter 4 we investigated what effects personalization of trust modeling could have

on a recommendation task. It is worth mentioning that personalization is a *defacto* bottom up operation. Weighing the preferences of an individual against the good of the community (or the stated definitions of acceptable preferences by administrators), is an inevitable issue when taking any approach to personalization.

These issues acknowledged, we can briefly add that top down systems are also plagued with issues: the bureaucratic nightmares that have grown out of corporate HR departments, university administrations and national governments of all stripes can attest to this. While it may seem odd to attempt to deal with the concept of anti-social behaviour without explicit definitions, we note that the definitions of many controversial topics, including hate speech, pornography and racism, are so vague that human experts, judges and juries routinely spend weeks at a time attempting to determine whether particular instances fit the stated definition. Under this light, it seems absurd to expect that an automated reasoner, especially one bearing current technological limitations, could reason effectively based on top down definitions of these controversial topics.

An awareness of this tradeoff, and the tremendous difficulties which lie in both approaches, is a critical issues to be kept front of mind in trust modeling and content recommendation research. The methods we have developed in this thesis are designed to be self-contained algorithms which can be provided to any party which has the data at hand, in order to reason about trustworthiness and anti-social behaviour.

Chapter 6

Conclusions and Future Work

6.1 Summary

In this thesis we considered the problem of improving the experience of users on social networks, particularly with respect to content overload and the propagation of untrustworthy information. We argued that a trust modeling approach could be appropriate for social networks and could be used to enhance message recommendation systems. We then outlined some of the issues involved in applying these models as they currently exist. Firstly, the outcomes of interactions between agents are difficult to quantify on social networks, and secondly, the types of trust models that can be applied need to be highly flexible, capable of capturing many different kinds of data, and personalizable.

In response to the first problem, in Chapter 3 we proposed an approach to quantifying the outcome of online conversations by training machine learning models to recognize the responses to anti-social behaviour on discussions on Reddit. This system is unique, in that it completely ignores the observable behaviour of an agent when trying to reason about whether or not that behaviour was acceptable, instead focusing entirely on the reactions to that behaviour[1]. Our results show that even when only incorporating a modest amount of lexical features and metadata features from a discussion, negative score polarity and the presence of hateful terms in discussion starting text can be predicted with surprising accuracy. We then outlined how these trained models could be used to develop a metric

[1]We note again, that the purpose of ignoring the comment text is simply to verify the importance of responses - later in this chapter we will propose hybrid models that consider both comment text and responses.

signifying the effect a user was having on the community. This metric could then be implemented into a flexible trust model, like the one described in the next chapter.

In response to the second problem, we argued that a multi-faceted trust model was ideal for application to social networks. This is because the multi-faceted model can incorporate arbitrarily many signals from the agents and their environment into a data driven model of how trust is apportioned by agents in an environment. We argued that this flexibility was a key feature, as it allows the model to adapt to many different kinds of social networks. In Chapter 4, we designed a comprehensive MFTM and applied it to a large data set, including multiple new features and features proposed in previous works. We experimented with personalizing the predictions generated by a multi-faceted model, by clustering similar users and learning distinct models for each cluster of users. We argued that although this approach is not "truly individualized" personalization, a data driven model like MFTM imposes a tradeoff between the number of users a model is learned for, as smaller numbers of users will have less data available to train classifiers with. We showed that this approach can lower error rates in a downstream trust aware recommendation task.

Both works, although aimed at different problems, fit in to the general goal of making trust models more applicable to social networks. Indeed, these works could potentially be combined in a follow up work, as the outputs of the system developed in Chapter 3 could be used as a trust indicator in a personalized multi-faceted trust model.

Finally, we compared our works to similar works, and engaged in a discussion regarding ongoing difficulties in the trust modeling and message recommendation space.

Our contributions can be succinctly summarized as follows:

- Developed a novel syntax for describing the workings of trust models, and showed how a number of influential models can be described using this unifying syntax.
- Identified two critical challenges that emerge in applying trust models to social networks: quantifying event outcome and personalizing trust prediction.
- Proposed a novel method of quantifying discussion outcome, by relating the reactions of conversants to identifiable undesirable behaviour. Showed accuracy above 70% on two tasks: prediction of a negative score and prediction of hateful terms in a discussion starting comment.
- Evaluated a clustering based approach to personalization on a large data set, showing a consistent improvement in error rates when predictions were applied to a downstream recommendation task.

- Presented a discussion of three major issues of practical and philosophical importance to future trust modeling and message recommendation research: the data availability problem, the filtering vs flagging problem, and the top down vs bottom up problem.

6.2 Future Work

Finally, we present a number of avenues for future work based on our works.

6.2.1 Expanding on Chapter 3

The work we presented with respect to quantifying the outcome of discussions on Reddit can be extended in a number of ways.

For example, it would be useful to consider a larger data set. Our experiment used data collected from comments submitted to five popular subreddits over the course of a single month in 2016. While this was a large amount of data to filter through, much of it was removed from analysis for not being written in English, or for not having generated interesting discussions. Further, certain types of behaviour are simply rare - such as engaging in hateful dialogue. In our data, we only found about ten thousand comments that had at least one n-gram from a list of hateful speech n-grams, and many of these were probably not truly hateful when context was taken into account (see Appendix B). A larger scope of data collection would help to find more examples of such rare behaviour, and also to increase the accuracy of predictions.

Similarly, a hand labeled data set would likely be extremely helpful and increase accuracy on all tasks other than the score polarity prediction task (as the labels for this task were defined unambiguously in the data). As our labels for hateful content were generated through a process with a large degree of error, it is surely the case that many false positives entered the data set. It would be helpful to take the set of comments that had a likely hateful n-gram in them and have human annotators decide if these comments were really engaging in hateful dialogue. Similar considerations apply to the profanity and sentiment prediction tasks, which fared the worst of our experiments. The examples in Appendix B show that our approach can indeed identify many instances of complex hateful dialogue, but our recall and precision scores show that a large number of false positives are also flagged. A hand labeled data set would allow us to get a better picture of this systems effectiveness. An excellent starting point would be to consider the data set of human annotated Globe and Mail comments produced by the SFU discourse lab [42].

Another interesting modification to the data set would be to consider multiple Reddit communities distinctly, as well as other discussion sites. One advantage of this project is that it uses very little of the features which are specific to Reddit in its design. As we've mentioned, the tree-shaped discussion style is now common on many social media sites, meaning this procedure could be applied to these services. On Reddit itself, there are an extremely diverse set of communities, including some where extreme views are commonplace. It would be extremely interesting to train anti-social behaviour classifiers like the ones we've described here on mainstream and extreme communities, and examine the differences in feature importance and output.

Similarly, it would be useful to consider applying these techniques to communities other than those found on Reddit. As we've pointed out, we've based most of our features on the tree-shaped discussion structure (where the requirements are that discussions can be of arbitrary length and it is unambiguous who is replying to whom). This structure is supported by most large social networks now, including Facebook and Twitter. Both services also contain a notion of a score (Likes), although unlike on Reddit these scores can only be positive (thus some adjustments may be required when implementing our approach).

The concept underlying this project could remain unchanged while upgrading many aspects of the machine learning based approach. Besides making use of a multi-layer ("deep") neural network for prediction, this project did not make use of the recent advances in the field of deep learning to improve its feature extraction and embedding procedures - offering many avenues to remove researcher bias from the process and allow design decisions to be driven by data. For example, the set of features extracted from comments were created by hand, and some of them were based on the outputs of error prone classifiers. It would be useful to explore using a text embedding system to represent each comment based on a un-biased picture of its text content (something as simple as a bag of words approach could accomplish this), rather than on a hand-picked set of extracted features. Similarly, the procedures used to aggregate these features across sets of comments were very simple and no doubt lost a great deal of subtlety and structure present in the discussion. A machine learning model that can handle graph input would be ideal for learning aggregations of node features. For example, we are currently investigating applying a Graph Attention Network [80] to this problem - a type of neural network which can learn how to attend to the features of neighbors in a graph, enabling effective aggregations of features from neighbors to be learned in a data-driven way.

Another approach would be to consider a hybrid model, combining the outputs of a model that examines community response (like our own) with a purely NLP based model. Recent NLP based hate detection works have demonstrated very high accuracy on curated

data sets (e.g. [5]). Since the focus of our work was to demonstrate the effectiveness of examining community responses, we set aside a deeper analysis of the natural language. Having demonstrated this, we believe that hybrid models would represent a very promising path forward.

Finally, as we've stressed repeatedly, the outputs of the classifiers proposed in this section should be integrated into a trust model. We re-stress that the motivation for this project is to extract quantified outcomes of interactions between agents on social networks in order to empower trust models to reason about the behaviour of agents. While we have not taken this step yet, the outputs of our models can be integrated directly into a multi-faceted trust model. The outputs of the model may also be useful for other applications. For instance, this model could be integrated into a system which flags comments for review by administrators, surfacing those comments whose responses are indicating they may contain unacceptable behaviour. This system could also be useful in an active learning set up, finding the comments which seem to have disrupted the community and passing them to a set of human annotators.

6.2.2 Expanding on Chapter 4

There are a number of ways the project of personalizing multi-faceted trust predictions can be extended.

For example, we spent considerable time in this thesis explaining the difficulties involved in clustering points that represent agents in a social network. While the approach we took was ultimately geometrically inspired, graph clustering algorithms could potentially offer a better fit to this type of data. This is an especially attractive option, as the sparsity of defined similarities between agents when considered geometrically is a major issue for applying and accurately measuring performance of geometric clustering approaches. We briefly experimented with the Markov Clustering (MCL) algorithm [79], however performance (as we measured it) was not significantly improved on the social clustering task, and somewhat worse when clustering by preference similarity. This can be because the Pearson Correlation Coefficient's notions of similarity, dissimilarity and neutrality can be expressed easily in the geometric setting (i.e. as 1, -1, and 0 respectively), but can not be expressed well in the MCL formulation, where only similarity and neutrality can be expressed. We also experimented with hierarchical clustering methods, but found difficulty in tuning the parameters in order to produce groupings with many moderate sized clusters. While these particular approaches did not appear to be helpful in our experiments, we believe that methods more amenable to this type of data may exist, or at least be good subjects for future research.

Two other challenges related to the clustering aspect of this work are finding new methods of determining the optimal number of clusters (k), and considering other distance functions. In this work, we ran the entire experiment from beginning to end (cluster, predict, recommend) many times in order to measure the effect of cluster count changes. This approach would not be feasible on larger data sets. It may be that certain heuristics exist: for example, a minimum cluster size of 500 agents is enough to capture many distinct groups and will usually provide sufficient training data to train an accurate trust link predictor for that group of users. Searching for new methods of determining k which are more computationally tractable than an exhaustive search, or finding heuristics that can guide this search, would be a useful and interesting research project. Second, we clustered agents in this work on the basis of social circle overlap (Jaccard similarity of trusted users) and preference similarity (Pearson Correlation Coefficient observed in train set ratings). While we've argued that each of these are fairly natural metrics, it would be interesting to explore new metrics, including those based on implicit preferences (e.g. browsing behaviour), categories of interest (e.g. types of items enjoyed), and other biographical factors of the agents (e.g. geographic location, age). Each of these can plausibly be argued to be indicative of some facet of agent similarity, which in turn may be correlated to similarities in trust formulation procedures.

Our work does not consider dynamic changes in the network or agent preferences over time. For example, our method did not consider agents who had no preference data associated with them, that is, new agents joining the network. In practice, this could be handled by simply assigning generic predictions for agents who lacked sufficient preference data on which to cluster them. A periodic re-training of the models would also allow the system to account for changing preferences over time. This dynamic process of agents entering the network could be simulated for our experiments by leaving out a sample of users from the initial processing, then adding them after clusters have been created already. Our methodology for clustering should allow a moderate number of users to be added to existing clusters based on existing distance measures. This would likely degrade the performance of the overall solution over time, as the cohesiveness of clusters would suffer by greedily assigning new users to the best existing clusters. The periodic retraining mentioned above would then be applied.

Another area for possible expansion is in our use of the personalized cluster classifiers. We did not learn a classifier for a cluster when that cluster had less than 1000 positive examples of outgoing trust links and 100 agents in it. This step was taken to avoid learning very inaccurate classifiers, but some of the classifiers learned still fit the data related to the cluster significantly worse than a classifier trained on larger sample of random agents. Therefore, it is worth exploring better ways of combining the "local" (cluster specific)

predictions with the "global" predictions, similar to the procedure taken in the Personalized Trust Model [87]. Perhaps the weight given to a local trust model could be based on the difference in accuracy between the fit of that model to the agents it represents and the accuracy a generic classifier would achieve for those agents. This way, local irregularities could still be learned, but in cases where data is sparse, a little help from a generic classifier can nudge predictions towards a more accurate final outcome. This approach could also be taken to enable more "truly individual" personalization. For users with a large amount of activity (thousands of friends and other users to compare preferences with), a single-user classifier could be trained, and the results of this classifier combined linearly with a cluster or global classifier, allowing truly individualized personalization, and a gradual ramp up from generic to individual solutions as more data becomes available.

In our work, we excluded users from experimentation who had fewer than 20 reviews. This filtering procedure was inspired by Mauro et. al [50], but it imposes certain biases on the following evaluations. Under this procedure, only the opinions and activities of the most active users are taken into account (only about 2% of Yelp users have submitted at least 20 reviews). In our earlier experiments, we sampled users randomly, and the results from this time tended to show a more dramatic difference between personalized and non personalized approaches using TrustMF (e.g. in Figure 4.6). It would be valuable to experiment with different procedures for sampling users from this data set.

There is also merit in examining how our model operates in other social networking contexts. Epinions is a reasonable second case for us to explore, as it was also examined by [22]. We conducted a preliminary study of Epinions data sets and noticed that the chance of a randomly picked review score being 5 (the highest) is over 70%, while on Yelp the distribution is much more spread out, with the highest probability being only 35% on a score of 4. With this kind of bias in the data, we would expect even better score accuracy on the score prediction task when applying our methods. It is also interesting to note that Epinions users typically have fewer friends (trusted users) and that with Epinions users submitting ratings to written text (rating others' reviews), there is vastly more feedback to examine. All of these differences may provide greater insights into the conditions under which our model has the most value.

There may be additional challenges when examining other social networks. While many recent projects in trust modeling focus on data from social networks with a significant item rating component (as it is convenient to measure trust-aware recommendation accuracy on a set of reserved ratings as a proxy measure for the quality of novel predicted trust links), we acknowledge that many popular networks such as Twitter and Facebook lack a significant item rating component. In cases like these, it would likely be necessary to engage in a user study (like the one in [27]) and survey actual users whether the predicted

trust links appeal to them or not. This work would be useful, especially if a data set can be publicly released, as more data where preferences are explicitly indicated by users (rather than inferred) will be a boon to future trust modeling research.

6.2.3 Addressing digital misinformation

Our work aims to improve online experiences by supporting distinct presentation of content to differing users, achieved by reasoning about relationships with peers and the concept of trust. Our concern with trustworthiness of content relates well to companion efforts devoted to detect digital misinformation [17, 35, 85]. There is a spectrum of possible outcomes when messages which are of questionable quality are shown to users, including special attention in contexts such as healthcare where the consequences may be more troubling [54]. As we have discussed in Sections 5.4.2 and 5.4.3, there will still be various options for actions to take, once trust modeling has provided some insights into messages of concern. As our work has drawn out the value of personalized solutions, the models that we have presented should be flexible enough to support a variety of overall preferences with respect to final outcomes. Integrating our proposed approaches directly into the larger effort aimed at combating digital misinformation would be a rich area for future work.

6.3 Final Thoughts

We believe that, for better or for worse, the future of information sharing will be online and it will be social. Institutional accreditation and verification of information appears to be losing relevance in the minds of many online readers, and this trend may continue. We can see this trend in the recent growth of groups which oppose childhood vaccination, question the shape of the earth, engage in obsessive conspiratorial thinking, and reserve the highest scrutiny for long-standing and well respected mainstream institutions. In this thesis, we have suggested that trust modeling and message recommendation algorithms are key components in improving the trustworthiness of information online. We've proposed new methods to quantify the quality of discussions and experimented with methods of personalizing trust prediction, as we believe these and related approaches will be valuable for these ongoing efforts

Perhaps in the future, powerful governments and institutions will attempt to control the spread of viral information in the same way they attempt to control the spread of biological viruses today. This situation already exists to some extent in authoritarian states, but

may become increasingly common in Europe and North America over the coming decades. While this idea raises fears in many, it must be remembered that before the internet era, the vast majority of information flow was indeed controlled by governments and large institutions. Control over the flow of information has shifted dramatically over the past half century: today, a single individual may have as much ability to spread their ideas and opinions as a newspaper or radio station had 50 years ago, perhaps even more. While this may appeal to lovers of free speech and the "market of ideas", one must question to what extent this environment has led to the rampant denials of scientific consensus and the degradation of societal cohesion that appears to be increasing today. We suggest that not all instances of institutional influence are an attack on the liberty of individuals, and that open cooperation between groups (institutional and private) can indeed be productive in balancing the needs and desires of individuals with those of society at large.

It is our hope that the tools we and others are building today will be of genuine value to the good faith activities of responsible governments, individuals and institutions in detecting and removing untrustworthy information and hateful rhetoric from our shared information spaces.

References

[1] Ashutosh Adhikari and Robin Cohen. A pomdp-based context aware approach for trust modeling. In *Proceedings of the 21st International Workshop on Trust in Agent Systems*, 2019.

[2] Athirai Aravazhi Irissappane and Jie Zhang. Filtering unfair ratings from dishonest advisors in multi-criteria e-markets: a biclustering-based approach. *Autonomous Agents and Multi-Agent Systems*, 31(1):36–65, Jan 2017.

[3] Myrna L. Armstrong, Donna C. Owen, Alden E. Roberts, and Jerome R. Koch. College students and tattoos: Influence of image, identity, family, and friends. *Journal of psychosocial nursing and mental health services*, 40(10):20–29, 2002.

[4] Fatemeh Torabi Asr and Maite Taboada. Fake news detection. http://fakenews.research.sfu.ca/.

[5] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, 2017.

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[7] Michael R. Brennan, Stacy Wrazien, and Rachel Greenstadt. Learning to extract quality discourse in online communities. In *Proceedings of the 2nd AAAI Conference on Collaboratively-Built Knowledge Sources and Artificial Intelligence*, pages 4–9, 2010.

[8] Chris Burnett, Timothy Norman, and Katia Sycara. Bootstrapping trust evaluations through stereotypes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 241–248, 2010.

[9] Chris Burnett and Nir Oren. Position-based trust update in delegation chains. In *Proceedings of the 16th International Workshop on Trust in Agent Societies*, 2013.

[10] Christiano Castelfranchi and Rino Falcone. *Trust theory: A socio-cognitive and computational model*, volume 18. John Wiley & Sons, 2010.

[11] Pew Research Center. Internet use. https://www.pewresearch.org/internet/chart/internet-use/, 2019 (accessed March 23, 2020).

[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 the Nineteenth International Conference on User , Adaptation, and Personalization*, UMAP'11, pages 38–49. Springer, 2011.

[13] Shuo Chen, Athirai Irissappane, and Jie Zhang. Pomdp-based decision making for fast event handling in vanets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] Jin-Hee Cho, Kevin Chan, and Sibel Adali. A survey on trust modeling. *ACM Computing Surveys*, 48:1–40, 2015.

[15] Daejin Choi, Jinyoung Han, Taejoong Chung, Yong-Yeol Ahn, Byung-Gon Chun, and Ted Taekyoung Kwon. Characterizing conversation patterns in reddit: From the perspectives of content properties and user participation behaviors. In *Proceedings of the 2015 ACM on Conference on Online Social Networks*, COSN '15, pages 233–243, 2015.

[16] François Chollet et al. Keras. https://keras.io, 2015.

[17] Giovanni Luca Ciampaglia, Alexios Mantzarlis, Gregory Maus, and Filippo Menczer. Research challenges of digital misinformation: Toward a trustworthy web. *AI Magazine*, 39(1):65–74, Mar. 2018.

[18] Robin Cohen, Peng-Fei Wang, and Zehong Hu. Revisiting public reputation calculation in a personalized trust model. In *Proceedings of the 20th International Trust Workshop*, 2018.

[19] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[20] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. A computational approach to politeness with application to social factors. *arXiv preprint arXiv:1306.6078*, 2013.

[21] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

[22] Hui Fang, Guibing Guo, and Jie Zhang. Multi-faceted trust and distrust prediction for recommender systems. *Decision Support Systems*, 71:37–47, 2015.

[23] Hui Fang, Jie Zhang, and Murat Şensoy. A generalized stereotype learning approach and its instantiation in trust modeling. *Electronic Commerce Research and Applications*, 30:149–158, 2018.

[24] Marc Fisher, John Woodrow Cox, and Peter Hermann. Pizzagate from rumor, to hashtag, to gunfire in d.c. *The Washington Post*, 2016, (accessed March 23, 2020). https://www.washingtonpost.com/local/pizzagate-from-rumor-to-hashtag-to-gunfire-in-dc/2016/12/06/4c7def50-bbd4-11e6-94ac-3d324840106c_story.html.

[25] Michael Fleming. The use of increasingly specific user models in the design of mixed-initiative systems. In Ahmed Y. Tawfik and Scott D. Goodwin, editors, *Advances in Artificial Intelligence*, pages 434–438, 2004.

[26] David Garcia, Pavlin Mavrodiev, Daniele Casati, and Frank Schweitzer. Understanding popularity, reputation, and social influence in the twitter society: Understanding the twitter society. *Policy & Internet*, 9, 04 2017.

[27] Eric Gilbert and Karrie Karahalios. Predicting tie strength with social media. In *Proceedings of the 27th international conference on Human factors in computing systems*, 2009.

[28] Richard Gitelson and Deborah Kerstetter. The influence of friends and relatives in travel decision-making. *Journal of Travel & Tourism Marketing*, 3(3):59–68, 1995.

[29] Jeffrey Gottfried, Michael Barthel, and Amy Mitchell. Internet use. *Pew Research Center*, 2017 (accessed March 23, 2020).

[30] Mark S Granovetter. The strength of weak ties. In *Social networks*, pages 347–367. Academic Press, 1977.

[31] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. Librec: A java library for recommender systems. In *UMAP Workshops*, volume 4, 2015.

[32] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 123–129, 2015.

[33] Clare Hanrahan. Antisocial behaviour. Encyclopedia.com, 2020 (accessed March 27, 2020). https://www.encyclopedia.com/medicine/psychology/psychology-and-psychiatry/antisocial-behavior.

[34] Nicolas Hug. Surprise, a Python library for recommender systems. http://surpriselib.com, 2017.

[35] Pik-Mai Hui, Chengcheng Shao, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. The hoaxy misinformation and fact-checking diffusion network. In *Proceedings of the 12th International AAAI Conference on Web and Social Media*, 2018.

[36] Dongyan Jia, Fuzhi Zhang, and Sai Liu. A robust collaborative filtering recommendation algorithm based on multidimensional trust model. *Journal of Software*, 8(1):11–18, 2013.

[37] Audun Josang, Ross Hayward, and Simon Pope. Trust network analysis with subjective logic. In *Proceedings of the 29th Australasian Computer Science Conference, CRPIT Volume 48, Hobart, Australia*, volume 48, 01 2006.

[38] Audun Jøsang and Roslan Ismail. The beta reputation system. In *Proceedings of the 15th Bled Conference on Electronic Commerce*, 2002.

[39] Audun Jøsang and Simon Pope. Semantic constraints for trust transitivity. In *Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling*, APCCM 05, page 5968, 2005.

[40] Reid Kerr and Robin Cohen. Smart cheaters do prosper: defeating trust and reputation systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 993–1000. Citeseer, 2009.

[41] Reid Kerr and Robin Cohen. Treet: the trust and reputation experimentation and evaluation testbed. *Electronic Commerce Research*, 10(3-4):271–290, 2010.

[42] Varada Kolhatkar, Hanhan Wu, Luca Cavasso, Emilie Francis, Kavan Shukla, and Maite Taboada. The sfu opinion and comments corpus: A corpus for the analysis of online news comments. *Corpus Pragmatics*, pages 1–36, 2019.

[43] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[44] Dhruv Kumar, Robin Cohen, and Lukasz Golab. Online abuse detection: the value of preprocessing and neural attention models. In *Proceedings of the 10th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 16–24, 01 2019.

[45] Kwiseok Kwon, Jinhyung , and Yongtae Park. Multidimensional credibility model for neighbor selection in collaborative recommendation. *Expert Systems with Applications*, 36(3):7114–7122, 2009.

[46] Yuyang Liang. Knowledge sharing in online discussion threads: What predicts the ratings? In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 146–154, 2017.

[47] Siyuan Liu, Jie Zhang, Chunyan Miao, Yin-Leng Theng, and Alex C. Kot. An integrated clustering-based approach to filtering unfair multi-nomial testimonies. *Computational Intelligence*, 30(2):316341, 2014.

[48] Xin Liu, Anwitaman Datta, Krzysztof Rzadca, and Ee-Peng Lim. StereoTrust: a group based personalized trust model. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 7–16, 2009.

[49] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[50] Noemi Mauro, Liliana Ardissono, and Zhongli Filippo Hu. Multi-faceted trust-based collaborative filtering. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '19, pages 216–224, 2019.

[51] Roger C. Mayer, James H. Davis, and F. David Schoorman. An integrative model of organizational trust. *The Academy of Management Review*, 20(3), 1995.

[52] Umar Farooq Minhas, Jie Zhang, Thomas Tran, and Robin Cohen. A multifaceted approach to modeling agent trust for effective communication in the application of mobile ad hoc vehicular networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(3):407–420, 2011.

[53] John O'Donovan and Barry Smyth. Trust in recommender systems. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 167–174, 01 2005.

[54] Daniel Ohashi, Robin Cohen, and Xiaotian Fu. The current state of online social networking for the health community: Where trust modeling research may be of value. In *Proceedings of the 2017 International Conference on Digital Health*, DH 17, page 2332, New York, NY, USA, 2017. Association for Computing Machinery.

[55] Alexandre Parmentier and Robin Cohen. Learning user reputation on reddit. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI 19, pages 242–247, 2019.

[56] Alexandre Parmentier and Robin Cohen. Personalized multi-faceted trust modeling in social networks. In *Proceedings of the 33rd Canadian Conference on Artificial Intelligence*, 2020.

[57] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Micheland Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[58] Andrew Perrin. Social media usage: 2005-2015. https://www.pewresearch.org/internet/2015/10/08/social-networking-usage-2005-2015/, 2015 (accessed March 23, 2020).

[59] Kevin Regan, Pascal Poupart, and Robin Cohen. Bayesian reputation modeling in e-marketplaces sensitive to subjectivity, deception and change. In *Proceedings of the National Conference on Artificial Intelligence*, volume 2, 01 2006.

[60] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 2000.

[61] Lior Rokach and Oded Maimon. *Clustering Methods*, pages 321–352. Springer, 2005.

[62] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[63] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, 2007.

[64] Noel Sardana. Recommending messages to users in participatory media environments: a bayesian credibility approach. Master's thesis, University of Waterloo, 2014.

[65] Noel Sardana and Robin Cohen. Demonstrating the value of credibility modeling for trust-based approaches to online message recommendation. In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pages 363–370. IEEE, 2014.

[66] Noel Sardana and Robin Cohen. Modeling agent trustworthiness with credibility for message recommendation in social networks. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems*, 2014.

[67] Noel Sardana and Robin Cohen. Validating trust models against realworld data sets. In *Twelfth Annual International Conference on Privacy, Security and Trust*, 2014.

[68] Noel Sardana, Robin Cohen, Jie Zhang, and Shuo Chen. A bayesian multiagent trust model for social networks. *IEEE Transactions on Computational Social Systems*, 5(4):995–1008, 2018.

[69] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[70] John Seely Brown and RP Adler. Open education, the long tail, and learning 2.0. *Educause review*, 43(1):16–20, 2008.

[71] Sandip Sen. A comprehensive approach to trust management. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS 13, pages 797–800, 2013.

[72] Aaditeshwar Seth, Jie Zhang, and Robin Cohen. Bayesian credibility modeling for personalized recommendation in participatory media. In *Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization*, UMAP'10, pages 279–290, 2010.

[73] Chengcheng Shao, Pik-Mai Hui, Lei Wang, Xinwen Jiang, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. Anatomy of an online misinformation network. *PloS one*, 13(4), 2018.

[74] Liam Stack. Over 1,000 hate groups are now active in united states, civil rights group says. *The New York Times*, 2019, (accessed March 23, 2020). https://www.nytimes.com/2019/02/20/us/hate-groups-rise.html.

[75] Alex Stamos. An update on information operations on facebook. https://about.fb.com/news/2017/09/information-operations-update/, 2017 (accessed March 23, 2020).

[76] Amo Tong, Ding-Zhu Du, and Weili Wu. On misinformation containment in online social networks. In *Advances in Neural Information Processing Systems 31*, pages 341–351, 2018.

[77] Thomas Tran and Robin Cohen. Improving user satisfaction in agent-based electronic marketplaces by reputation modelling and adjustable product quality. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 828–835, 2004.

[78] Thomas T. Tran, Robin Cohen, and Eric Langlois. Establishing trust in multiagent environments: Realizing the comprehensive trust management dream. In *Proceedings of the 17th International Workshop on Trust in Agent Societies*, 2014.

[79] Stijn Marinus Van Dongen. *Graph clustering by flow simulation*. PhD thesis, University of Utrecht, 2000.

[80] Petar Velikovi, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Li, and Yoshua Bengio. Graph attention networks. *arXiv preprint*, 2017.

[81] Omar Wahab, Jamal Bentahar, Robin Cohen, Hadi Otrok, and Azzam Mourad. A two-level solution to fight against dishonest opinions in recommendation-based trust systems. In *Proceedings of the 21st International Workshop on Trust in Agent Systems*, 2019.

[82] Gerhard Weiss. *Multiagent systems*. MIT press, 2013.

[83] Yue Xia, Alexandre Parmentier, and Robin Cohen. Trust modelling in dynamic environments. In *Proceedings of the 21st International Workshop on Trust in Agent Systems*, 2019.

[84] Bo Yang, Yu Lei, Daylon Liu, and Jiming Liu. Social collaborative filtering by trust. In *Proceedings of the Twenty-Third IJCAI International Joint Conference on Artificial Intelligence*, pages 2747–2753, 2013.

[85] Shuo Yang, Kai Shu, Suhang Wang, Renjie Gu, Fan Wu, and Huan Liu. Unsupervised fake news detection on social media: A generative approach. In *Proc. AAAI 2019*, 02 2019.

[86] Amy X. Zhang, Aditya Ranganathan, Sarah Emlen Metz, Scott Appling, Connie Moon Sehat, Norman Gilmore, Nick B. Adams, Emmanuel Vincent, Jennifer Lee, Martin Robbins, Ed Bice, Sandro Hawke, David Karger, and An Xiao Mina. A structured

response to misinformation: Defining and annotating credibility indicators in news articles. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 603–612, 2018.

[87] Jie Zhang and Robin Cohen. Evaluating the trustworthiness of advice about seller agents in e-marketplaces: A personalized approach. *Electronic Commerce Research and Applications*, 7:330–340, 2008.

APPENDICES

Appendix A

Reputation, Popularity, Trust and Credibility

In the trust modeling literature there is significant confusion over the use of the terms *reputation, popularity, trust,* and *credibility*. It is not uncommon for these terms to be used interchangeably (perhaps "popularity" less so) and to appear in papers without a principled definition. Since all qualities are desirable in some degree when choosing a partner for interaction in a multiagent system, the differences between these concepts often get lost. Here we adopt definitions of popularity and reputation inspired by Garcia at al. [26] and our definition of trust is inspired by Cho et al. [14]. It is useful to commit to a clear definition of these terms. For some properties, it is useful consider the social network in question as a directed graph, where nodes represent agents and edges represent trust/interest/approval (any of these can serve as an edge, depending on the system being modeled).

Popularity can be thought of as the in-degree of an agent's node. That is, an agent popularity is the raw count of how many other agents trust (or approve of, or are interested in) them. On an online social network, this might correspond to how many up votes their posts have gotten, to how many subscribers they have, or how many positive comments their submissions receive.

Reputation can be thought of as a kind of recursive popularity. That is, a reputable agent has the trust/interest/approval of a large amount of other reputable agents. While a popular agent is at the center of a "star" of interested (but not themselves interesting) agents, a reputable agent sits atop a pyramid of interest from other reputable agents [26]. Note how the notion of reputation is used differently in the terms "a reputable person"

and "a person with a reputation for x", where x is some behaviour. The former is the type of reputation described here, while the latter is a description of past behaviour.

Credibility is the propensity for an agent to produce reliable information. An agent is credible to the extent that the information they produce is not muddled by ignorance or warped by malice. In the physical world, in addition to considering past history when available, we apply heuristics and stereotypes to judge a stranger's credibility (including their education, social standing (popularity/reputation), job, personality traits, and similarity to ourselves). These heuristics are not available in semi anonymous networks, and measuring how frequently an agent produces reliable information is difficult. Nevertheless, the term credibility is often used interchangeably with the term trust, as it is quite natural that a highly credible agent is more deserving of trust. Since automated fact checking is difficult, is it difficult to measure credibility directly online [86]. However, measuring trust between agents under the context of producing credible information should be achievable, given the trustees subjective notions of what types of information are credible.

Trust is a context-bound, subjective, directed belief between two agents. Agent i's trust in agent j is an indication of the former's belief that the latter will act competently and in accordance with the norms and expectations of a good-faith partner under the context in question. Importantly, this trust usually implies i is willing to take on some risk in their interactions with j under the given context. The item at risk may be simply time and attention, but often financial transactions are considered as well. For instance, in an e-marketplace context, a buyer's trust in a seller corresponds to that buyer's belief that the seller will deliver the product on time, for the agreed upon price, in an acceptable condition, etc.

One may think of "trustworthiness" as an expectation on a distribution over trust. That is, an agent considered trustworthy under some context if a trust model (human or machine) estimates it is more likely than not that trusting an agent will have a good outcome.

While trust is subjective, being based on one agent's belief in another, in online social networks it can often be appropriate to model the network itself as an agent with goals defined by the rules and purpose of the network. Thus, while an agent participating in a social network does not have a 'global' trustworthiness, it is appropriate to model the trust that the network administrators have in that agent to conform to the norms of the network (e.g. to not post spam, to not share banned content, etc.).

Appendix B

Hateful Reddit Comments

In this appendix we present a number of comments from our data set in Chapter 3 which scored highly probable to contain hateful rhetoric based on the outputs of our random forest classifier. Note that the comments in this section are highly charged. Although none of them contain explicit racial slurs (such comments would likely be filtered out immediately by Reddit administrators), clearly all of these comments are provocative, and likely incited some kind of backlash. These comments show the interesting many types of provocative behaviour our classifier can detect, as it is not dependent on actually understanding the comment text, only the reaction to it. We compare the confidence that a comment is hateful reported by our random forest classifier to the confidence that HateSonar [21] reports that the comment is *either* hateful or offensive. Overall, these examples seem to show that while our classifier can identify complex instances of racist or hateful text with high confidence, it also tends to have high confidence about controversial topics. All comments printed here are selected from the top 40 highest confidence hateful comments according to our classifier.

> No thats racist. Blackpeoplemeet.com is empowering and fresh but whitepeoplemeet.com is racist and exclusionary. Theres always a double standard for white people. God forbid whites like whites just as much as blacks like blacks.

Random Forests Confidence: 93%. HateSonar Confidence: 68%

> White people do score higher on IQ tests than blacks and hispanics though, and OP referenced IQ, not intelligence.

Random Forests Confidence: 93%. HateSonar Confidence: 56%

> Of course I am. Any decent human being would be. Innocent people are being raped, robbed and murdered by Muslims the world over, and people like you are telling me to ignore it for the sake of politically correct nonsense ideas. No. Now it's time we finally acknowledge the problem. Now is the time we finally stop listening to that "religion of peace" smokescreen.

Random Forests Confidence: 92%. HateSonar Confidence: 41%

> As a us-born white, our opportunities are fairly limited. low access to social resources and higher education unless you can afford it. whites have the lowest social mobility.

Random Forests Confidence: 93%. HateSonar Confidence: 59%

> Europeans doesn't seem to care about antisemitic provocations in their states. This is getting ridiculous, and social justice warriors care more about Muslims and Homosexual people's rights. I guess, there is no hope left.

Random Forests Confidence: 93%. HateSonar Confidence: 36%

> Spending other peoples money is a human right! Women should earn the same as men regardless of their choices or qualifications! Homosexuality is a moral good! We should encourage our children to be homosexual! Spanking your child is abuse but cutting off his penis and putting him in a dress is progressive! Hormone blockers are a human right! Replacing white people in the nations they built is important! Overrepresentation of a demographic in university matters unless its women, jews, or black athletes! White people are inherently evil unless they have a vagina! Black people burning down historic American cities is good!

Random Forests Confidence: 92%. HateSonar Confidence: 79%

> I expect minorities do enjoy some advantages. Do you think they outweigh those of whites? Do you think they somehow prove 'white privilege' is nonsense?

Random Forests Confidence: 92%. HateSonar Confidence: 66%

Appendix C

Computation of Trust Indicators

One of the considerable challenges encountered in the implementation of the project described in Chapter 4 was the computation of trust indicators between pairs of agents. The trust indicator function $\Psi(a_i, a_j)$ is expected to be computed for all ordered pairs of agents. Of course, as there are $O(n^2)$ possible pairs of agents, this rapidly becomes a computational issue as the number of agents considered grows. In our experiments we worked with groups of agents where $|A| \approx 30000$, implying approximately 900,000,000 pairs - a large but tractable computation on modern consumer hardware. However, the unfiltered Yelp data set contains descriptions of 1,637,138 agents, and we can be assured that other large online environments contains many millions of users. At this scale, the $O(n^2)$ computation time becomes a serious barrier, and storing the trillions of resulting vectors for further processing would likely be extremely costly.

However, it is not necessary to consider *every* possible pair of agents. For example, if a_i and a_j have never interacted in any meaningful way and share no known interests – in sum, we have no evidence of any way they might know or be interested in each other – then it is likely safe to conclude, without any complex trust modeling, that they need not trust each other. Further, we can conclude that the lack of a trust link between them is most likely the result of ignorance rather than opinion. To analogize, the potential trust relationship between a university professor in China and a wheat farmer in Canada need not be explicitly modeled and computed if no evidence can be found that the two may in fact share a communication channel or desire to interact in the future.

Thus a solution to the computation barrier presents itself: simply defining a neighborhood function, $N(a)$, on individual agents and only computing trust indicators and trust predictions between pairs of agents in the same neighborhood. So long as computing $N(a)$

is efficient, the execution time of computing all *relevant* trust indicator pairs then becomes linear with a constant bounded by the maximum neighborhood size.

The definition of $N(a)$ can be very liberal and still result in a substantial speed up. For example, when computing trust indicators for the Yelp and Epinions data set, we used:

$$a_j \in N(a_i) \iff |R_{ij}| > 0 \vee friends(a_i, a_j) \vee friendOfFriend(a_i, a_j)$$

That is, a_j is in the neighborhood of a_i if they have both reviewed at least one item in common, if they are friends, if they are friends of friends, or if they are friends of friends.

Applying this neighborhood function drastically reduces the number of pairs of agents that need to be considered in the following stages.

Appendix D

List of Symbols

- A: The set of all agents in an environment where a trust model is deployed.
  - $a_i \in A$: The i'th agent in the environment. Actual agent ordering is undefined: subscripts are used simply to identify and differentiate agents.
  - $\overrightarrow{a}$: An agent acting as a truster.
  - $\overleftarrow{a}$: An agent acting as a trustee.
- E: The set of all events in an environment where a trust model is deployed.
  - $e_i \in E$: The i'th event in the history of the environment. An event, e, is a tuple composed of one or more agents, a context c and an outcome o. That is, $e_i = \langle a_1, a_2, ..., a_n, c, o \rangle$. An event represents an observable action or interaction among agents.
  - c or $\text{con}(e)$: The context of an event. As trust occurs in the scope of a context, it is necessary to distinguish between varying contexts of events. Contexts can be arbitrarily complex, and differ by domain. In many domains, there is only one relevant context.
  - o or $\text{out}(e)$: The outcome of the event. The space of possible outcomes differs by domain, but a ranking or ordering based on desirability is expected among elements in the space of o.
  - $\overleftarrow{\text{te}}(e)$: The agent(s) acting as a trustee in event e.
  - $\overrightarrow{\text{tr}}(e)$: The agent(s) acting as a truster in event e.

- T: The abstract "trustworthiness" function. This is ultimately what trust models aim to compute. T is used to rank the relative trustworthiness of trustees: If $T(\overrightarrow{a_i}, \overleftarrow{a_j}, c) > T(\overrightarrow{a_i}, \overleftarrow{a_k}, c)$, then $\overleftarrow{a_j}$ is more trustworthy than $\overleftarrow{a_k}$ for $\overrightarrow{a_j}$ under context c.
- T_c: An abbreviation for the trust function when context is set, static, or unitary. In particular: $T_c(\overrightarrow{a_i}, \overleftarrow{a_j}) = T(\overrightarrow{a_i}, \overleftarrow{a_j}, c)$.
- $\hat{f} : x \to \hat{y}$: The approximation of a function $f : x \to y$ learned by some machine learning based classifier.
- $\Psi(a_{1,2})$: an application specific vector of trust indicators describing the individual and relational evidence for and against trustworthiness between a_1 and a_2.
- Γ: the user-user trust matrix.
- R_i: the set of items that agent a_i has reviewed.
- $R_{i,j}$: the set of items that agents a_i and a_j have both reviewed.
- I_j: the set agents that have reviewed item j.
- r_{ij}: the score given to item j by agent i in their review.
- $\mathbb{1}(cond)$: The indicator function, equal to 1 when *cond* is true and 0 otherwise.