Journal of Universal Computer Science, vol. 18, no. 4 (2012), 532-553
submitted: 30/9/11, accepted: 14/12/11, appeared: 28/2/12 © *J.UCS*

Social Network Based Reputation Computation and Document Classification

Joo Young Lee, Yue Duan, Jae C. Oh, Wenliang Du, Howard Blair
Lusha Wang, Xing Jin
(Syracuse University, Syracuse, USA
jlee150, yuduan, jcoh, wedu, blair, lwang40, xjin05@syr.edu)

Abstract We develop two social network based algorithms that automatically compute *author reputation* from a collection of textual documents. We first extract *keyword reference behaviors* of the authors to construct a social network, which represents relationships among the authors in terms of *information reference behavior*. With this network, we apply the two algorithms: the first computes each author's reputation value considering only *direct reference* and the second utilizes *indirect reference* recursively. We compare the reputation values computed by the two algorithms and reputation ratings given by a human domain expert. We further evaluate the algorithms in email categorization tasks by comparing them with machine learning techniques. Finally, we analyse the social network through a community detection algorithm and other analysis techniques. We observed several interesting phenomena including the network being scale-free and having a negative assortativity.

Key Words: social network, reputation management, community analysis, computer security
Category: H.2, H.3.7, H.5.4

1 Introduction

Reputation management plays an important role in online communities that include e-commerce web sites, such as *e-bay* and *amazon.com*, peer-to-peer computing environments [Jung 2009], and online social networks [Jung 2010, Jung 2012]. Existing reputation management schemes often require users to explicitly rate each other to compute reputations. For example, the simplest way of computing reputation is to average all the ratings a user receives from other users as in *amazon.com*'s 5-star rating system. However, in general, these rating systems have the following weaknesses: (1) systems cannot force users to rate each other, and (2) consequently, not all user interactions contribute to ratings, resulting inaccurate calculation of reputation. In this paper, we show that it is possible to compute reputations of authors by analysing their reference behaviors in a social network that is built by extracting key contents of documents. Our method can extract reputation based on users' interactions manifested in the constructed social network.

Reputation management can also be useful in rating documents in their importance. In processing a large amount of unstructured data such as web documents and emails, identifying author's reputation can help in extracting important information. For example, an *automatic document summarization* technique can extract key phrases and return a shortened version of the original text. Before automatically summarizing documents, one could filter out less important documents with the additional aid of *author reputation*. Visualization is another example of presenting textual data in a schematically abstracted graphical form [Friendly 2009]. When visualizing a network of relationships among texts in a graphical form, one could associate quantitative measures of reputation with nodes and edges of the graph.

In this paper, we develop two automatic reputation computation schemes using knowledge extraction from unstructured emails through constructing a social network of authors. The first algorithm computes reputation only considering direct reference behaviors of authors. The second algorithm goes one step further and incorporates indirect references in the computation. The social network based algorithms are also tested on classification of emails. Finally, we conducted network analysis on the resulting network. We show that the network is scale-free as many social networks are, but does not have positive assortativity [Newman et.al. 2003]; rather, each community in the network asymptotically satisfies a power law distribution.

This paper is organized as follows. In Section 2, we discuss related work. Section 3 describes the steps in processing raw email data into tagged and separated format. In Section 4, we explain the two algorithms for automatic reputation computation. Section 5 presents experimental results on automatic reputation computation and email classification tasks. Section 5 also discusses network analysis on the social network found. Finally, Section 6 presents conclusions.

2 Background

Massive amount of information available on the Internet is motivating researchers to find efficient and effective ways to extract useful knowledge from massive quantities of unstructured data. Researchers have been working to transform such information into structured formats so that the data can be further processed and presented in forms that can be easily understood by humans. In order to process such massive amounts of data, various automation tools have been developed to extract information of interest.

Automatically computing the importance of documents has been well studied. Hummon and Doreian [Hummon et.al. 1989] proposed three indices representing weights of arcs to automatically identify the important sequences of links and nodes in the citation network. Later, Vladimir Batagelj [Batageli 2002] made a progress to efficiently compute the Hummon and Doreian's weights so that they can be used for analysis of very large citation networks with several thousands of vertices. They are interested in finding communities of papers that play important roles in citation networks by applying search algorithms. They focus on connectivity of search paths to identify strong communities. On the other hand, we focus on authors reference behavior and analyse how communities form in relation to the reference behavior. Google's *PageRank* algorithm, which has been influenced by citation analysis, also computes the importance of each page, called *PageRank*, based on the the number of pages supporting it as well as the *PageRank* of supporting pages [Page et. al. 1999]. In these examples, the referring documents' importance is considered, but the reputation of the authors of those referring documents is not.

Many methods for computing reputation usually require human users to provide ratings. Sporas [Zacharia 2000] is an example of how one can compute the reputation of individuals of interest in a network. Each user has one reputation value and the value is updated iteratively according to ratings given by other users.

We combine the above two approaches to automatically compute reputation values from documents using social network. The reputation values can be used to help data mining tasks for cyber security, which is our test application domain. In the following sections, we describe our approach in detail.

3 Document Preparation for Social Network Analysis

In this paper, we focus on email data to construct social network and compute reputation. We use emails from three mailing lists over a one month period for a total of $2,415$ individual emails. In order to construct a social network for reputation computation, we pre-process email data. *Fig. 1* shows the main steps of preprocessing. First, we start with a very long text file containing a series of raw emails. Then we separate the file into individual emails, one file per email, using our separator program, written in Java. These individual emails are one of the two inputs to reputation computation. Next, we remove the header and signature information from emails using a MIME (Multipurpose Internet Mail Extensions) reader to eliminate noise. We then use UIMA (Unstructured Information Management Architecture) to tag entities such as IP, URL and DOMAIN in the emails. *Fig. 2* shows one of the tagged email by UIMA. [1] UIMA is an open source architecture that analyses unstructured documents, video and audio. We wrote a UIMA descriptor that identifies IP, URL and DOMAIN as well as email addresses of authors. We then convert the UIMA annotated outputs into tagged emails. We construct a social network using both individual tagged emails and emails with headers. The purpose of emails with headers is to extract Date and Time of the emails sent to extract reference behaviors among authors.

[1] Details of the email are deleted for confidentiality.

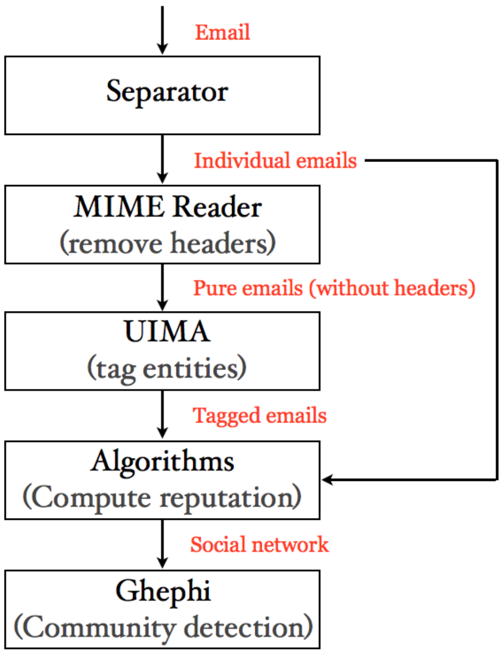


Figure 1: Original data is one long text file containing many emails. We separate emails using a separator program to individual files. We remove unwanted information from individual emails using simple MIME Reader to get Non-MIME formatted emails. These emails are further processed by UIMA to get tagged emails for the social network algorithms. We build a social network of authors based on their reference behavior and use Gephi [Bastian et. al. 2009] to visualize the network information.

4 Constructing Social Networks and Computing Reputations

This section discusses the method for building a social network from email data. We also discuss the two reputation computation algorithms.

4.1 Building a network

An author reputation social network is a weighted digraph where vertices represent authors and weighted edges represent reference behaviors. The weighted edges are computed as described in *4.2* and *4.3*. Such a network is built from a time-stamped collection of *documents*. Next, in order to present the network

Hi All,
As you may be aware the domains <Domain>mypremierfutbol.com</Domain>
and <Domain>todaysfutbol.com</Domain>=
=20
were being used by Stuxnet. These are now pointing to our sinkhole which i=
s=20
hosted at <IP>XXX.YY.ZZZ.220</IP>.=20=20

Cheers,
XXXXX.=20

Figure 2: An example of a tagged email using UIMA; this figure shows two DOMAIN tags and an IP tag.

building algorithms we define several convenient functions. Following these definitions we give an example of applying the functions to a set of documents.

In the application at hand, where we build a network from a collection of email text files, a *document* is an email, where we assume for each document d that a single *author*, denoted by $\mathsf{author}(d)$, and a single time stamp, which we represent by a real number and denote by $\mathsf{time}(d)$, is extractible from each email. Define a function $\mathsf{authors}$ that maps sets of documents to sets of authors by

$$\mathsf{authors}(D) = \bigcup_{d \in D} \{\mathsf{author}(d)\}$$

Also, where D is a set of documents, let

$$\mathsf{times}(D) = \bigcup_{d \in D} \{\mathsf{time}(d)\}$$

Assume that each document contains one or more *entities*, the nature of which may be left as a parameter to be instantiated later. We also assume that the set of entities, $\mathsf{entity}(d)$ contained in a document d is extractable from d. Again, where D is a set of documents, let

$$\mathsf{entities}(D) = \bigcup_{d \in D} \mathsf{entity}(d)$$

(Note that in the above definitions, $\mathsf{author}(d)$ and $\mathsf{time}(d)$ are not sets, whereas $\mathsf{entity}(d)$ is a set.)

While a document d uniquely determines a time t and an author a, the converse determination, a time/author pair (t, a) uniquely determines a document, is also true, assuming an author can generate only one document at a time (although the time stamps associated with any particular author may be in rapid succession.) Therefore, a document is a partial function of a time/author pair. (The function is partial because an author a may not have generated a document at a particular time t). We will, in the sequel, regard time/author pairs (t, a) as documents. Think of (t, a) as the *undefined document*, if there is no document in the set D of documents input to our algorithms with both time stamp t and author a.

The following are three email headers extracted from the real separated email documents:

email header1

Email Subject: additional information from First Citizens
Email creation time: Wed Mar 09 18:20:18 GMT 2011
Sender email:owner@xxx.net

email header 2

Email Subject: Money Mule and cards
Email creation time: Tue Mar 08 20:34:22 GMT 2011
Sender email:owner@yyy.net

email header 3

Email Subject: Mule account!
Email creation time: Fri Mar 04 23:14:00 GMT 2011
Sender email:owner@xxx.net

From the previous email headers, we can extract:

$$\begin{aligned}\mathsf{authors}(D) &= \{\text{owner@xxx.net}\} \bigcup \{\text{owner@yyy.net}\} \bigcup \{\text{owner@xxx.net}\} \\ &= \{\text{owner@yyy.net, owner@xxx.net}\} \\ \mathsf{times}(D) &= \{\text{"Wed Mar 09 18:20:18 GMT 2011"}, \} \\ &\quad \text{"Tue Mar 08 20:34:22 GMT 2011"}, \\ &\quad \text{"Fri Mar 04 23:14:00 GMT 2011"}\}\end{aligned}$$

Therefore, we get six pairs of (t, a), three (in *italic*) of which correspond to undefined documents:
(Wed Mar 09 18:20:18 GMT 2011, owner@xxx.net),
(Fri Mar 04 23:14:00 GMT 2011, owner@xxx.net),
(Tue Mar 08 20:34:22 GMT 2011, owner@yyy.net),
(Tue Mar 08 20:34:22 GMT 2011, owner@xxx.net),
(Wed Mar 09 18:20:18 GMT 2011, owner@yyy.net),
(Fri Mar 04 23:14:00 GMT 2011, owner@yyy.net).

A document uniquely determines an author and time but the reverse doesn't hold. We consider authors as nodes when constructing the network later on, so the reverse need not hold (we don't need to know which documents entities come from as long as they belong to the same author).

The algorithm Build_Social_Network takes as input a finite sequence of 4-tuples, each of which is a *well-formed Information entity*. An *Information entity* is a 4-tuple (e, t, a, b), where e is an entity, t is a time, a is an author and b is a boolean. (e, t, a, b) is *well-formed* iff (t, a) is a defined document, $e \in \mathsf{entity}(d)$ and $b = \mathsf{initial}(t, a)$ where

$$\mathsf{initial}(t, a) = \begin{cases} \mathit{false}, \text{ if } (t, a) \text{ is a reply/forward} \\ \mathit{true}, \text{ otherwise} \end{cases}$$

Pseudo code for the network building algorithm is given in *Algorithm 1*. Again, the input to the algorithm is a finite sequence of information entities, I, and the output is the social network. (Once entities are extracted, we need not know which documents they are coming from since *information entities* have all the information we need.)

An example of an instance of a sequence of three *Information entities* is given below. The example extracts entities from the email shown in *Fig. 2*. The email has 3 entities; *mypremierfutbol.com*, *todaysfutbol.com* and *XXX.YY.ZZZ.220*. The timestamps, author and boolean value for all three are the same since they belong to the same email.

(mypremierfutbol.com, 22 Jul 2010 13:52:08, s@X.com, FALSE)
(todaysfutbol.com, 22 Jul 2010 13:52:08, s@X.com, FALSE)
(XXX.YY.ZZZ.220, 22 Jul 2010 13:52:08, s@X.com, FALSE)

When multiple authors have a common entity in any of their emails, a directed edge exists *to* the source author, whose email precedes others in terms of the time sent, *from* a destination author whose email has the same entity with the source email. More specifically, as shown in *Fig. 3*, when more than one author has a common entity, e_1, and if it is the case where author a_1 mentioned the entity e_1 and author a_2 also mentioned e_1 as a reply or forward, a_1 gets an incoming edge from a_2 with weight 1. Otherwise, if e_1 was mentioned by another author a_3 not as a reply or forward, a_1 will get another incoming edge

Algorithm 1 Build_Social_Network(I)

```
for each element (e_i, t_i, a_i, b_i) in I do
  a ← the first author who mentioned e_i
  if a = a_i then
    {self–referencing, next element in I}
    break
  end if
  if b_i =FALSE then
    {e_i is in a reply or forward document}
    make a connection from a_i to a with weight 1
  end if
  if b_i =TRUE then
    {e_i is in an original document}
    make a bidirectional connection between a_i and a with weight 2
  end if
end for
```

from a_3 with weight 2 and a_3 will also get an incoming edge from a_1 with weight 2. Authors mentioning common entities supports the fact that those entities are hot issues and the author who brought the issue first gets the credit. We only give a positive weight to directed edges because regardless of the context, either positive or negative, authors mentioning a common entity increases the popularity of the entity. The rationale for independent reference getting twice the weight is that since all the authors involved in independent reference are originals, their importance is identified as originators, unlike the authors of replies and forwarded messages. These weights are the basis for computing reputations of authors in algorithms described in *4.2* and *4.3*.

We have developed two algorithms to calculate the reputation of each author; one uses only direct references and the other uses indirect references as well. Both algorithms run on the network built from the previous algorithm.

4.2 A Sporas-based algorithm (Direct reference)

The first algorithm we propose is based on Sporas, a reputation mechanism for loosely connected online communities [Zacharia 2000]. Sporas updates user's reputation upon each rating given by another user. Ratings given by users with high reputation are weighted more. Since our application doesn't assume centralized environment where the system can ask users to rate each other whenever they have interactions, we adopted reference behavior as a way of giving ratings to other users. Therefore, from the social network we built, a node, which represents an author, has a reputation based on incoming edges it gets.

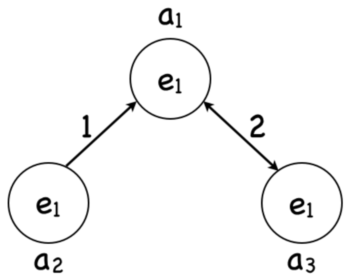


Figure 3: A social network of three authors; a_1, a_2, and a_3. All three of the authors mention the same entity, e_1. a_2 references a_1 as a reply or forward while a_1 and a_3 reference each other independently.

The reputation value for each author is computed as follows:

$$R_{t+1} = \frac{1}{\theta}\sum_{i=1}^{t} \Phi(R_i) \times R_{i+1}^{other} \times \frac{W_{i+1}}{2}$$

$$\Phi(R) = 1 - \frac{1}{1 + e^{\frac{-(R-D)}{\sigma}}}$$

Where,
t is the number of references the author has received so far,
θ is a constant integer greater than 1,
W_i represents the rating given by the user at time i,
R^{other} is reputation of the author who is referencing R,
D is the range of the reputation values,
σ is the acceleration factor of the damping function Φ.
The smaller the value of σ, the steeper the damping factor $\Phi(R)$.

For experiments, we used $\theta = 3$ and $\sigma = 0.5$. The maximum value for the reputation is 5 and the default value is 1. The damping function $\Phi(R)$, ensures that the reputations of trustworthy persons are more robust against temporary malicious attacks. The value of θ determines how fast the reputation value of the user changes after each rating. The larger the value of θ, the longer the memory of the system. For detailed information on Sporas algorithms, refer to [Zacharia 2000].

4.3 The Indirect Referencing algorithm

The second algorithm is based on the TrustMail rating system [James et.al. 2004]. TrustMail calculates the reputations of incoming emails based on human ratings.

Since the original algorithm was not designed for distributed settings, authors do not have representative reputation values. In the TrustMail system, reputation is computed only when a user asks for another's reputation value. The reputation values depend on the relationship between all of the requester's neighbors and the destination node (i.e., the node being evaluated). Consequently, reputations are relative. Since we want authors to have objective reputation values–so that we can have results to compare with the first algorithm–we evaluate all the relative reputation values for each author. In other words, we run the indirect algorithm for each author node as if each node is asking for everyone else's reputation values. *Algorithm 2* describes how reputation can be inferred when the *source* is asking for *sink*'s reputation value. We then average out the reputation values since we accumulate all the reputation values from all the neighbors a node has.

Algorithm 2 getRating($source, sink$)

```
mark source as seen
if source has no rating for sink then
  denom = 0, num = 0
  for each j in neighbors(source) do
    if j has not been seen then
      denom++
      j2sink = min(rating(source, j), getRating(j, sink))
      num += rating(source, j) * j2sink
      mark j unseen
    end if
    rating(source, sink) = num/demon
  end for
  return rating(source, sink)
end if
```

The main idea that given source i and sink node s, if i has direct edge to s then no inference is necessary. If there is no direct edge between i and s, i forwards the query to all the neighbors, namely *j*. The algorithm calculates t_{is}, the relative reputation of the *sink* for the source *i*. The condition in this formula ensures that the source will never trust the sink more than any intermediate node.

$$t_i s = \frac{1}{n} \sum_{j=0}^{n} \begin{cases} (t_{js} \times t_{ij}), & \text{if } t_{ij} \geq t_{js} \\ t_i^2 j, & \text{if } t_{ij} < t_{js} \end{cases}$$

4.4 Applying time decaying function to the algorithms

There could be cases where multiple authors independently discuss the same entity. As discussed in section *4.1*, the dependency relationship, the direction of an edge, is determined by the timestamp. Whoever has mentioned the entity earliest gets credit for the originality, whereas in the independency relationship, all the involved authors get credit regardless of time in independency relationship. Consider the case in *Fig. 4*. In addition to a_1 and a_3, a_4 mentions the same entity, e_1, say, a week later. According to the *Sporas-based algorithm*, both a_4 and a_1 should get incoming edges with weight 2. However, if the latter independent reference, which is a_4 mentioning e_1, happens after a sufficient amount of time, it is reasonable to consider the latter reference as a new topic rather than relating it to the previous reference.

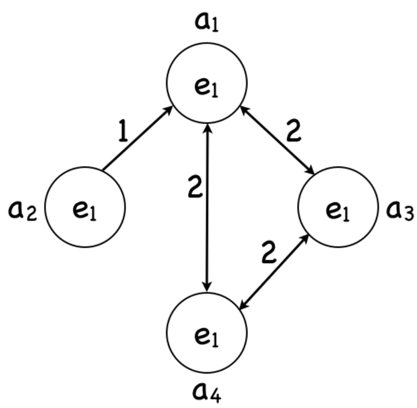


Figure 4: Independency relationship; a_1, a_3 and a_4 get bidirectional edges from each other since they all contain e_1, independently.

To accommodate this issue, we incorporated a *time decaying function* shown in *Fig. 5*. When a new entity is introduced by a source author and shortly referenced by others, there is a high chance that the references are related, but the relevance decreases over time. Therefore, after a sufficient amount of time has passed, we consider the entity to be independent from previous references. We use the *cosine* function to capture this idea. According to our *time decaying function*, when a new entity, e_2, is mentioned by the first author and independently referenced by another author immediately, they will both get incoming edges with weight 2 (technically, the latter author will get an edge with weight slightly less than 2 by the function). As time passes, authors who independently references e_2 will have incoming edges with weight less than 2 according to the *time decaying function*, at worst case with weight 1, when the topic has completely died out. Now, the edge weight of an independent reference is calculated as follows.

$$weight = 0.5 \times \cos(period \times (t_2 - t_1)) + 1.5$$

Where, t_1 is the time when original author introduce an entity, e, t_2 is the time when a new author independently references e, *period* is a 604,800,000 / $2\times$ PI.

604,800,000 is a week in milliseconds and *period* is set so that the period of the *cosine* function be a week.

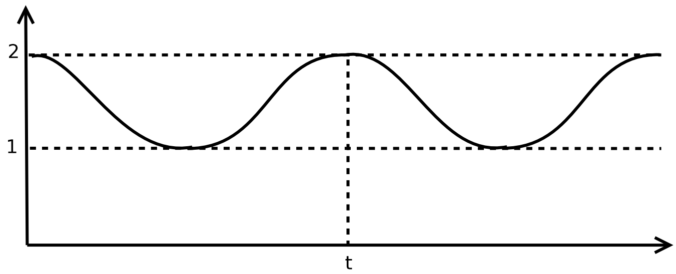


Figure 5: Time decaying function: $cos(period \times t)$

5 Analysis

5.1 Comparison of the Two Algorithms

From 2,415 emails, we have extracted 426 authors. In *Fig. 6*, we show the reputation values of all the authors, sorted in descending order from the perspective of the second algorithm. For the two algorithms, the reputation of authors in the top and bottom tiers tend to agree more than the middle ones.

The comparison between the two algorithms with the *time decaying function* is shown in *Fig. 7*. *Fig. 8* shows differences in reputation values using the *time decaying function* or not using. We only picked 15 authors here, since visualizing all 426 authors would be messy. We picked five authors with high reputation, five authors with middle reputation, and five authors with low reputation. Some differences from authors with high reputation were zero and that's why some values are not shown. For authors with high and low reputations, the effect of *time decaying function* was minimal. Authors with the mid-reputation range

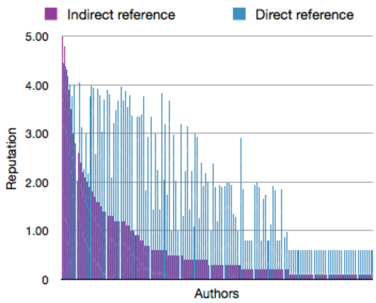


Figure 6: Reputation of authors before applying the *time decaying function*.

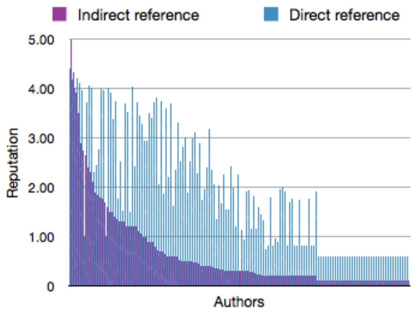


Figure 7: Reputation of authors after applying the *time decaying function*.

show that the difference between two algorithms is smaller in most cases after the *time decaying function* is applied.

We also compared human assigned reputations and reputations computed by the algorithms. Eleven authors were picked and assigned reputations by a human domain expert. *Fig. 9* shows how the reputations given by the domain expert compared to the reputations computed by the algorithms before applying the *time decaying function*. *Fig. 10* shows the result with the *time decaying function*. It is hard to conclude whether one is superior to the other since only eleven authors' reputation values are available from the domain expert. This difficulty motivated us to test our algorithms further; we compare the two algorithms with a decision-tree based machine learning algorithm in categorizing emails in Section 5.2.

In summary, we have shown that the two algorithms produce agreeing reputation values; but the reputations assigned by the domain expert shows small divergence from the reputations given by the algorithms. Possible explanations include the human expert may have assigned higher reputation values to recog-

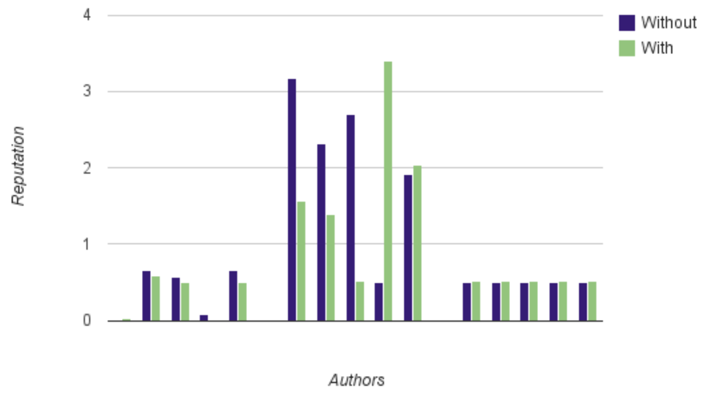


Figure 8: Effect of using time decaying function. We show differences of the reputations given by the two algorithms before and after applying the *time decaying function*. We pick 15 authors to compare; first 5 entries represent authors with high reputations, next 5 entries represent authors with middle reputations, and the last 5 entries represent authors with low reputations.

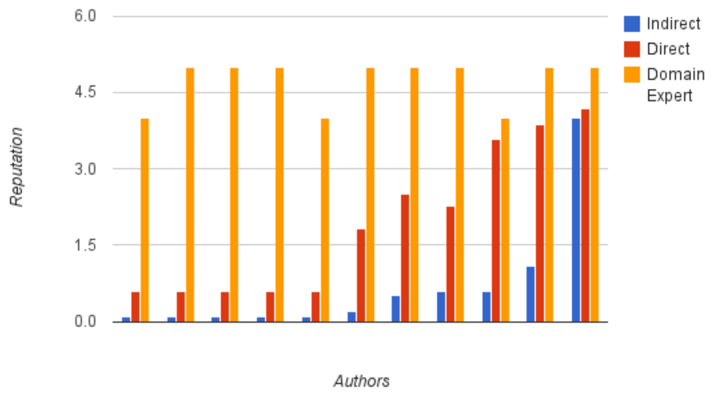


Figure 9: Human assigned reputations compared with the algorithm based reputations: without the time decaying function

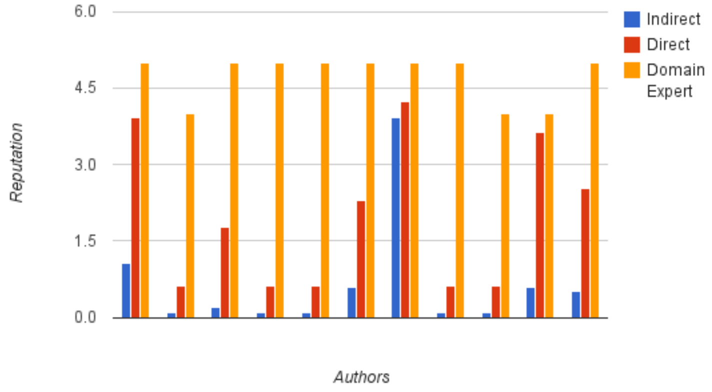


Figure 10: Human assigned reputations compared with the algorithm based reputations: with the time decaying function

nized authors or authors given high reputation by the domain expert were not active in writing important emails during the time the data were collected (one month). For future research, we plan to gather email data ranging over a year and study how experiment results change. It would be interesting if we divide the period into three so that each has four months of email data and compare how reputations of authors change over time.

A word of clarification may be needed about the tail ends of *Fig. 6* and *Fig. 7*. In *Fig. 6* and *Fig. 7*, the tail with euqal values represents authors with the default value. The difference only exists because author reputation given by the two algorithms were in different ranges before normalization. Since reputations given by the second algorithm go up to 255, even if an author has the same default reputation value from both algorithms, which is 1, when normalized, reputation given by the second algorithm appears to be smaller. The same explanation applies to *Fig. 9* and *Fig. 10* as well as other comparisons. For example, in *Fig. 8*, the rightmost short bars from authors with low reputation actually represent no difference between the two algorithms.

5.2 Email Categorization Experiments

Generally, reputation results are very hard to evaluate since there is no concrete values to compare with and the values are often subjective. We have used human expert's ratings to compare with outputs from our algorithms but the available

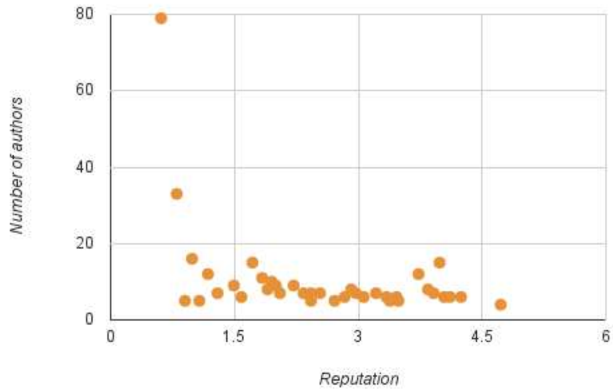


Figure 11: Distribution of the reputations of authors by the direct reference algorithm follows power law distribution

number of ratings was not enough. To further evaluate the efficacy of our algorithms, we test the algorithms in email categorization task and compare the results with machine learning algorithms.

We categorize emails into three groups: *useful*, *helpful*, and *useless*. We used 16 attributes to categorize emails. The details of each attribute is explained in *Table 1*. We use RapidMiner [Mierswa et. al. 2006], which is the most widely used open source data mining tool, to train the model. First, we manually categorize 100 samples of emails into the three groups by reading the contents of the emails, without relying on the attributes so that our manual categorization be independent from the machine learning of RapidMiner. Then we train the model with the training set. *Fig. 14* shows the decision-tree model trained.

To build a social network of emails, for each email, we counted the number of entities referenced by other emails. Analogous to the reference behaviors among authors, our intuition is that, if an email has higher *reputation* than others, (i.e., it has been referenced highly) then it is categorized as useful. The reputation of emails ranges from 0 to 10.

Among 709 emails, the decision tree model categorized 537 as *useless*, 67 as *helpful* and 105 as *useful*. As shown is *Fig. 15*, most of the emails categorized as *useless* has low *supported score*, i.e., less than 1, and only a few have *supported score* higher than 3. The emails categorized as *helpful* have consistent *supported score* between 2.5 and 3.5. The *useful* category, as expected, has the highest *supported score* on average, most of these emails having *supported score* of more

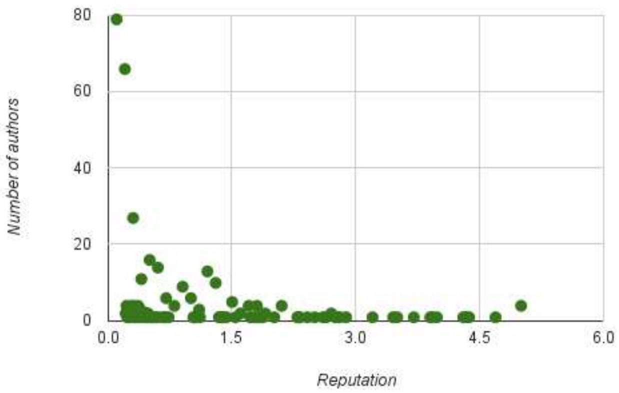


Figure 12: Distribution of the reputations of authors by the indirect reference algorithm follows power law distribution

than 3.5.

5.3 Social Network Analysis

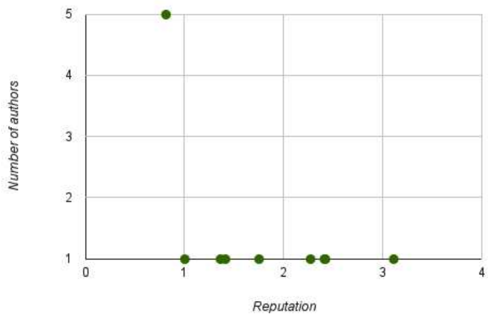
In this section, we analyze the social network constructed in Section 4. For the purpose of analysis, we used Gephi [Bastian et. al. 2009], an interactive visualization and exploration platform for networks and complex systems.

5.3.1 Community Detection

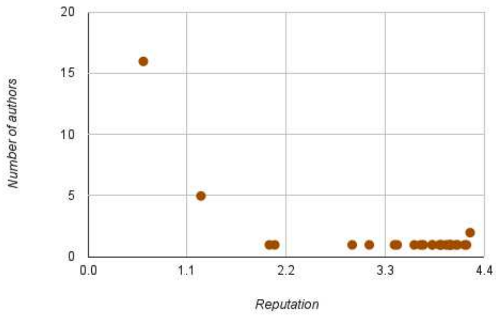
Community detection can reveal interesting facts about social networks. For example, the community structure of a social network can serve as a summary of the entire network, producing an easy to understand visualization of the network. *Fig. 16* is a visualization of our social network. Nodes and edges represent authors and reference behaviors, respectively, in the emails. Colors represent communities. The network has 36 communities and the modularity is between 0.46. A network with modularity of 0.4 or greater has meaningful community structures.

5.3.2 Average Path Length

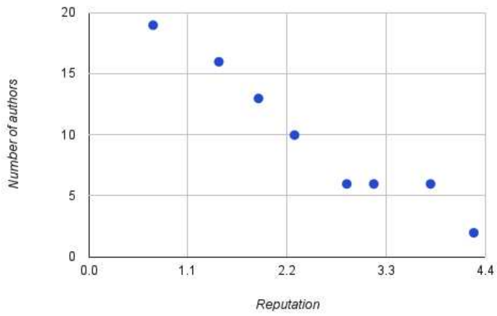
The network has the *average path length* of 4.1, which is shorter than the "e-mail network"of Ebel et al. [Ebel et. al. 2002]. Since email communication doesn't



(a) Modularity class 1



(b) Modularity class 9



(c) Modularity class 38

Figure 13: Each community follows power law distribution

require senders and receivers to closely share certain chracteristics, unlike other networks, such as co-authorship networks, email networks are believed to have lower value of *average path length*. This means that nodes in the network in general are more closely connected.

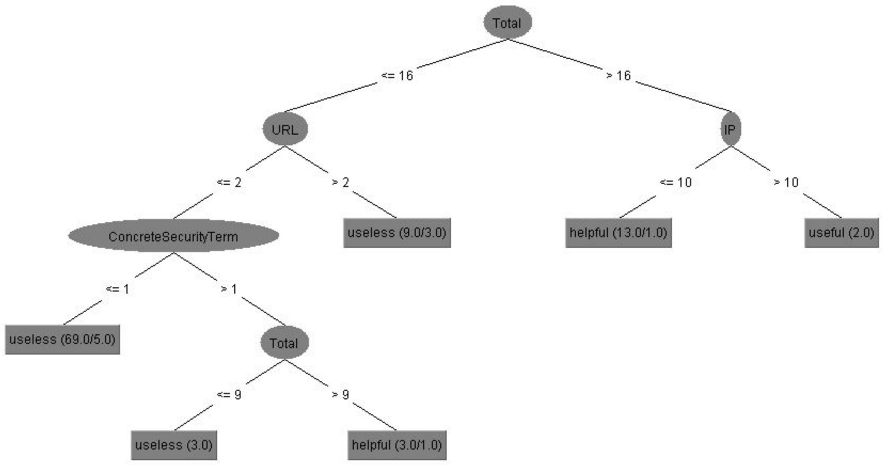
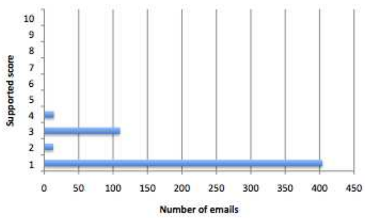
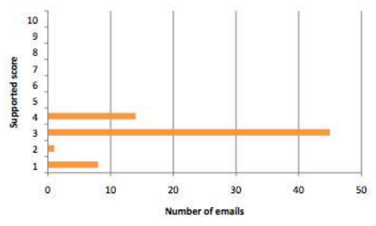


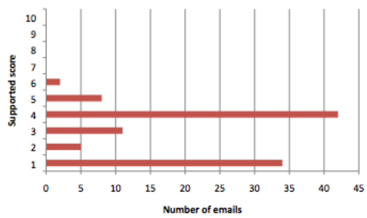
Figure 14: Trained tree model for categorization by Rapidminer



(a) categorized useless



(b) categorized helpful



(c) categorized useful

Figure 15: Comparing categorization result from the machine learning algorithm versus supported score of emails

5.3.3 Scale-free behavior

A scale-free network is a network with its degree distribution following power law, at least asymptotically. As shown in *Fig. 11* and *Fig. 12*, the distribution of

Attribute Name	Usage
ConcreteSecurityTerm	Concrete security terms in the email, such as root kit, Zeus
GenericSecurityTerm	Generic security terms in the email, such as threat, Malware
SpecialWords	Special interesting words, such as Russia, Iran
SecurityVerb	Security related verbs in the email, such as attack, hide
Length	Length of each email
RegistrantInfo	True, if the email contains system- generated registrant information
Request	True, if the email is requesting specific information
ReplyToRequest	True, if the email is a reply to any request
Attachment	True, if the emails contains attachment
List	True, if the email contains non- Natural Language formats such as a list
IP	IPs in the email
DOMAIN	DOMAINs in the email
URL	URLs in the email
EMAIL	EMAIL addresses in the email
WinRegistry	True, if the emails contains window registry information
Total	Sum of the number of attributes values (except the attributes that return boolean values)

Table 1: Sixteen attributes used for training

reputation in the overall network follows a power-law distribution. Interestingly, each community in the network also follows a power-law distribution as in *Fig. 13*. Within each community, there is a “super” author that the members of the community follows. This implies that the network has *negative assortativity* which will be discussed in the next section.

5.3.4 Assortativity

Assortativity is a preference for nodes in a network to attach to others that are similar or different in a metric. We calculated the *assortativity coefficient*, r, of the network found. The assortativity coefficient is essentially the *Pearson correlation coefficient* of degree between pairs of linked nodes [Newman 2002].

The value of r is approximately *-0.2* in the network. This is interesting because, unlike many social networks, which have positive assortativity [Newman et.al. 2003], each community in our social network follows a power law distribution. This means that in each community, there is small number of authors with high reputations followed by a greater number of authors with lower reputations as shown in *Fig. 13.* This characteristic is shared with citation networks [Page et. al. 1999]. Another fact is that the network has two obvious clusters as shown in *Fig. 16.* The clusters have almost equal number of authors and the degree distributions of nodes for the two clusters are quite similar. The nodes connecting the two cluster may play special roles. We plan to investigate the roles in our future work.

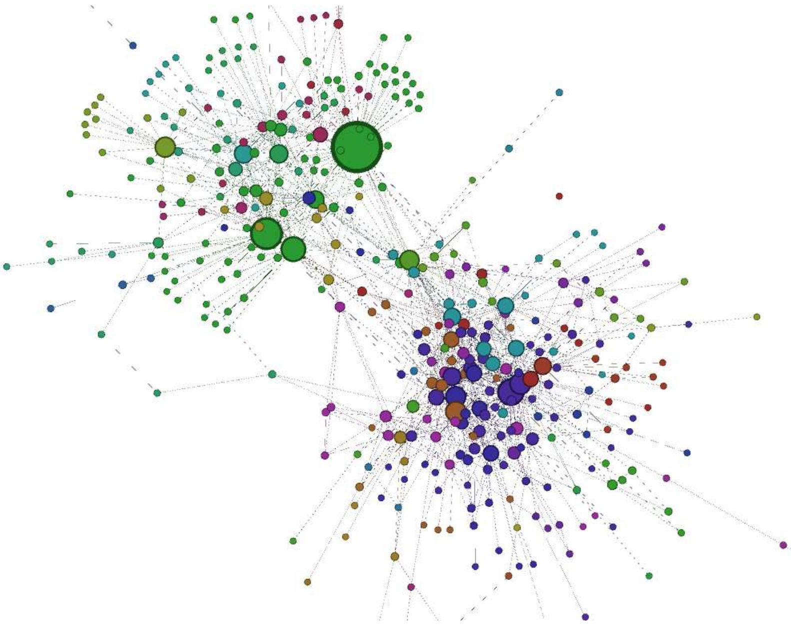


Figure 16: Social network of the authors; nodes are weighted with degrees, colors are partitioned by modularity classes.

6 Conclusions and Future Work

Document processing and social network reputation computation have been around for some years, but combining them to automatically compute reputations of authors and to categorize emails has rarely been done. We developed a method to extract references from contents of documents and to build author reputation network automatically. We also have developed two algorithms for calculating reputations of authors by traversing the network using direct references and indirect references. Our methods can be applied not only to emails but also to other unstructured data such as RSS and proxy logs, which will be our next step. We analyzed the reputation network generated using community detection. Some interesting properties are identified–such as power-law distribution of author reputations within each community as well as in the global network . In future research, we plan to evaluate importance of documents using automatically extracted author reputation and construct visualization tools that highlight the importance.

References

[Bastian et. al. 2009] Bastian, M., Heymann, S., and Jacomy, M.: "Gephi: An open source software for exploring and manipulating networks", 2009.

[Batageli 2002] Batagelj, V., and Batagelj, V.: "Efficient algorithms for citation network analysis", 2002.

[Ebel et. al. 2002] Ebel, H., Mielsch, L. I., and Bornholdt, S.: "Scale-free topology of e-mail networks"; *Physical Review E 66* (2002), 035103+.

[Friendly 2009] Friendly, M.: "Milestones in the history of thematic cartography, statistical graphics, and data visualization."; `http://datavis.ca/milestones/`.

[Hummon et.al. 1989] Hummon, N. P., and Doreian, P.: "Connectivity in a citation network: The development of DNA theory."; *Social Networks 11*(1989), 39-63.

[James et.al. 2004] James, J. G., and Hendler, J.: "Reputation network analysis for email filtering."; In *In Proc. of the Conference on Email and Anti-Spam (CEAS), Mountain View* (2004).

[Jung 2009] Jung, J. J.: Trustworthy knowledge diffusion model based on risk discovery on peer-to-peer networks. *Expert Systems with Applications*, 36(3):7123–7128, 2009.

[Jung 2010] Jung, J. J.: Integrating social networks for context fusion in mobile service platforms. *Journal of Universal Computer Science*, 16(15):2099–2110, 2010.

[Jung 2012] Jung, J. J.: Evolutionary Approach for Semantic-based Query Sampling in Large-scale Information Sources. *Information Sciences*, 182(1):30–39, 2012.

[Mierswa et. al. 2006] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., and Euler, T.: "Yale: Rapid prototyping for complex data mining tasks."; In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, August 2006), L. Ungar, M. Craven, D. Gunopulos, and T. Eliassi-Rad, Eds., ACM, pp. 935–940.

[Newman 2002] Newman, M. E. J.: "Assortative mixing in networks."; *PHYS. REV. LETT. 89* (2002), 208701.

[Newman et.al. 2003] Newman, M. E. J., and Park, J.: "Why social networks are different from other types of networks."; *Physical Review E - Statistical, Nonlinear and Soft Matter Physics 68*, 3 Pt 2 (2003), 036122.

[Page et. al. 1999] Page, L., Brin, S., Motwani, R., and Winograd, T.: "The pagerank citation ranking: Bringing order to the web, 1999."

[Zacharia 2000] Zacharia, G.: "Trust management through reputation mechanisms."; *Applied Artificial Intelligence 14* (2000), 881–907.