© 2014 by Stephen William Mayhew. All rights reserved.

TRUSTWORTHINESS AND THE IMPORTANCE OF GRAPH STRUCTURE

BY

STEPHEN WILLIAM MAYHEW

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Advisor:

Professor Dan Roth

Abstract

We begin by giving a comprehensive literature review that ties together many fields which have heretofore remained separate. We comment on the approaches from each field and show which algorithms are similar and which are different.

Then, starting from a concrete task, we extend traditional trustworthiness algorithms to deal with the more complex situation of multiclass list-valued trustworthiness. In addition, we introduce a learned predictive method based on standard classification algorithms.

In the last section, we explore the theory of trustworthiness and begin to make progress towards charting the space of all trustworthiness graphs. We address the commonly underestimated importance of the structure of a trustworthiness graph, and define a space in which to work as well as defining the solvability of a trustworthiness graph. Finally, we provide recommendations for future work.

To Heidi and Alan.

Acknowledgements

Many thanks to my advisor, Dan Roth.

Many thanks to my labmates in the Cognitive Computation Group for discussion and ideas, especially Jeff Pasternack, Vinod Vydiswaran, Daniel Khashabi, Subhro Roy, Christos Christodouloupoulos, and Chen-Tse Tsai.

Many thanks to the kind soul who put together the LaTeX template for this back in 2009 and put it online.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

> *On two occasions I have been asked, — "Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?" In one case a member of the Upper, and in the other a member of the Lower, House put this question. I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.*
>
> — Charles Babbage, Passages from the Life of a Philosopher (1864), ch. 5 "Difference Engine No. 1"

Babbage couldn't have known, but nearly 150 years after his incredulous quote, the "confusion of ideas" has become a legitimate research question. There are now research communities interested in making the right answers come out even though the wrong figures have been put in, and one name for this field of research is trustworthiness.

While the holy grail of trustworthiness perhaps looks like a browser extension that warns you when you are reading false news (even if it comes from a reputable source), or an application on your phone that (privately and securely) scans your email for incorrect assertions, or even an automated Snopes.com, we currently find ourselves studying problems of a much more abstract and low-level fashion. When the application-specific elements have been stripped away, we are left with the underlying bipartite trustworthiness graph, which is composed of abstract sources with edges going to abstract claims. Even with a formulation as simple as this, there are many questions left to answer. This thesis attempts to address some of the more pressing questions, and in the meantime brings up more questions.

We begin with a comprehensive literature review, which we find to be a timely addition to the field at this point. There have been several communities producing similar work, but not collaborating or citing each other. We hope to bring together some of them, or at least alert them of the presence of each other.

Then we present some simple new algorithms which are novel more for their existence than their techniques.

Finally, we point out a deficiency with most current trustworthiness techniques, and begin to suggest a solution. The deficiency lies in the fact that

trustworthiness algorithms are introduced and tested and published, but there is rarely any idea regarding the circumstances in which they will work. In learning theory, there are guarantees on the number of errors for a given input distribution, or guarantees of convergence to desirable results, but in trustworthiness, these are few and far between. The solution we begin to suggest is in examining some standard algorithms and learning when they succeed and when they fail. The hope is that with enough study, we can formally understand the strengths and weaknesses, as well as the semantics, of each trustworthiness algorithm.

Chapter 2

Literature Review

The field of trustworthiness has many names and many applications. Table 2.1 gives a partial list of some current lines of research which are pointed in more or less the same direction. For a researcher in this field, it is often difficult to discover the other lines of research because of the different names. Sometimes, a title will give no indication that the underlying algorithm falls into such a class.

In this chapter, we aim to provide a unification of several of these fields, a signpost in a fragmented land. We give an in-depth survey with commentary on each problem from a birds-eye view.

It is difficult to categorize the algorithms because there are several labels that apply to each one. In this chapter, we have chosen to categorize the algorithms by underlying algorithm type, and comment on the application of each algorithm along the way. We begin by looking at the simplest method, majority voting, then we consider eigenvector-based methods, and finally probabilistic methods.

Because of the many labels, and the many ways to slice the information, we provide Table 2.2. Although it is not exhaustive, it is intended to illustrate the wide variety of applications and methods. The upper section shows unsupervised algorithms while the bottom section shows supervised algorithm.

For some earlier surveys, see [Artz and Gil, 2007] and [Li et al., 2012]. The latter in particular provides information as well as experiments for each algorithm surveyed.

Trustworthiness
Factfinding
Reputation Management
Data Fusion
Crowdsourcing
Collective Classification
Structured Prediction
Rank Aggregation
Expertise Identification
Belief Propagation

Table 2.1: Other fields with similar algorithms

2.1 Preliminary Methods

The most obvious approach is to simply take the majority vote for each question. This is a naive approach, but is simple and intuitive. It can also be somewhat effective, as shown in [Sagae and Lavie, 2006]. Majority voting is a convenient baseline against which to measure all other techniques.

One of the earliest papers is [Gould, 1967], where the author, Peter Gould, a geographer, introduced a metric called the "accessibility index". The idea is to measure the "accessibility" of cities in a geographical network, where cities are nodes and roads are edges. Gould's observation was that, in certain instances, the principal eigenvector of the adjacency matrix of the geographical network corresponded to the "importance" or "accessibility" of the cities in the network.

Later, [Straffin, 1980] built on the idea by asking an important question:

> What do these numbers mean? Why is it that this manipulation through graphs, matrices, eigenvalues and eigenvectors should produce numbers entitled to the name of an "accessibility index"?

The paper goes on to provide some intuitions for why an eigenvector calculation is appropriate in this situation. The intuitions touch such subjects as the Perron-Frobenius theorem, the number of paths between two nodes, and rumor propagation.

This work is useful when we apply similar techniques to trustworthiness graphs, and we find that it is natural to ask the same question, modified to our purposes:

> Why is it that this manipulation through graphs, matrices, eigenvalues and eigenvectors should produce numbers entitles to the name of "trustworthiness"?

The final chapter of this thesis attempts to answer this question.

2.2 Eigenvector Methods

It only makes sense to start with the simplest of algorithms. We begin with Hubs and Authorities [Kleinberg, 1999] (alternately known as Sums [Pasternack and Roth, 2010], or Hyperlink Induced Topic Search, or HITS). This algorithm has a simple and appealing intuition: good sources make good claims, and good claims are made by good sources. It is iterative in nature.

We review the HITS algorithm, using the notation found in [Kleinberg, 1999] (this definition is also found in Section 4.3). For each node, we initialize the hub score, y_p, and the authority score, x_p, to 1. Then, we iterate until convergence (normalizing at each iteration).

$$x_p \leftarrow \sum_{q:(q,p)\in E} y_q$$

Application, Type	**Eigenvector**	**Probabilistic**	**Learned**
Geography	Accessibility Index		
Crowdsourcing	KOS		
IR	PageRank, HITS		
Trustworthiness, Factfinding	Sums, Avg. Log	LCA, Accu, PopAccu, TruthFinder	
Reputation	Eigentrust		
Classification			ICA, Gibbs Sampling
IE Validation			MTM

Table 2.2: Trustworthiness algorithms in an application by type matrix.

$$y_p \leftarrow \sum_{q:(p,q)\in E} x_q$$

Finally, having obtained a hub and authority score for each node, we order the claim nodes in each mutual exclusion set by the authority score. The claim with the largest authority is chosen to be the most trustworthy. (Note that in the specific case of using a bipartite graph, directed from left to right, the sources will have authority score 0 and the claims will have hub score 0).

In this situation, each claim node is an embodiment of a particular answer to a particular question. The question, which can only have one correct answer, corresponds to the mutual exclusion set. For example, in a binary situation, a claim node might be correspond to "Answer to Question 55 is TRUE", and the other node in the mutual exclusion set might be correspond to "Answer to Question 55 is FALSE."

This is classed with the eigenvector methods because the results of this computation, the hub and authority vectors, are none other than the principal eigenvectors of AA^T and A^TA respectively, where A is the adjacency matrix. One may recognize the above as being the power iteration algorithm.

The original conception of HITS was for ranking of webpages in the early days of internet search. Nodes were webpages, edges were hyperlinks, and the graph was not necessarily bipartite. As such, most nodes had both non-zero hub and non-zero authority scores. The process was to conduct a standard search with query terms, receive some large set S of pages (in the paper, $|S| = 200$), incremement it by all pages that point into the set, and all pages that are pointed to by elements of the set, run the iterative algorithm, sort by hub score then

by authority score, and finally report the top so many results of each sorting. The qualitative results of the paper as reported are satisfactory – the algorithm appears to do it's job.

We know from [Ding et al., 2004] that if a graph is a fixed-degree sequence random graph, then the ranking induced by the authority scores is identical to the ranking induced by degree. This is to say that if the graph can be accurately described as a fixed-degree random sequence graph, then all of the known conclusions regarding the successfulness of majority voting apply.

Another algorithm which accomplishes much the same thing is the famous PageRank [Brin and Page, 1998]. This algorithm uses the random surfer as the model. The result is also the principal eigenvector of some matrix, but in this case, the matrix is the adjacency matrix normalized by column. PageRank is known to be more stable than HITS [Ng et al., 2001].

[Ng et al., 2001] describe how HITS is very unstable. They note how using just the principal eigenvector leads to instability, and they show how the eigengap (difference between largest and next largest eigenvalue) is a good indicator of stability. Finally, they recommend an algorithm that uses some combination of all eigenvectors.

The next piece of work we examine is some extensions to Sums, namely AverageLog, Investment, and PooledInvestment, given in [Pasternack and Roth, 2010]. These three algorithms are very similar to HITS in structure (initialization and iteration), but with different update rules. AverageLog seeks to differentiate between trustworthy sources with a small amount of claims and trustworthy sources with a large amount of claims. Intuitively, the latter should be trusted more. Investment adds to the linear iterative process a non-linear growth factor for each claim. A good source "invests" credibility into it's claims and as time passes, this investment grows. Finally, PooledInvestment is similar to Investment, but with a linear scaling term added.

This paper (still [Pasternack and Roth, 2010]) then adds a further dimension by incorporating real-world knowledge in the form of constraints, written in terms of propositional logic. The framework works as follows:

1. Run trustworthiness algorithm to convergence on G
2. Repair inconsistencies in the graph (specifically in the claims) using the constraints.

They formulate a linear program using the results from the trustworthiness algorithm and the given constraints. Upon solving the linear program, the claims are reweighted, and the algorithm is run with these new claim weights as input.

Intuitively, this solves the problem where the trustworthiness algorithm suggests that two contradictory claims are true.

It is now convenient to make a point that is central to this thesis. None of the above algorithms have any notion of when they work, and when they will

not work. They provide some intuitions for why they should work, but these are not validated in data, and it is not even clear how to validate them.

We now describe an eigenvector-based algorithm from the field of crowdsourcing. In this domain, the situation is that we have a set of questions which we want answers to. For each question, we give it to multiple labelers to get redundancy of labeling, and now the question is, how accurate is each labeler? We look at labelers as sources and the labels as claims, and try to infer the correct answers.

[Karger et al., 2011] introduced an algorithm for this problem and binary labels, and derived an error bound for the algorithm. In their situation, they require that all workers be above a certain threshold for quality and that the graph be (l, r)-regular and generated by a random process. Since each question asked costs a certain amount of money for the asker, they wanted to ask as few questions as possible to get the best reliability score. This is called a budget-optimal algorithm.

In [Karger et al., 2013], they update the model to allow answers to take one of K distinct values (i.e. the multiclass problem).

This domain is different from the general trustworthiness domain in one major way: graphs are created by the task-master, and so can be constrained to follow a certain structure. In the general trustworthiness scheme, we want to address the situation where a graph is given, and may have any structure.

Finally, we consider Eigentrust [Kamvar et al., 2003], an algorithm from the field of reputation management. In Eigentrust, the nodes are users, and there are connections betwen users based on interactions, which can be positive or negative. Ultimately, trust is computed transitively (I will trust you as much as my friends trust you), and is calculated as being the principal eigenvector of matrix C, where c_{ij} represents the normalized difference between satisfactory and unsatisfactory transations between node i and node j. In essence, this can be understood as transitive trust through different length paths. In other words, I trust you as much as my friends trust you plus as much as my friends' friends trust you, and so on.

There is no notion of sources making claims in this scenario, although the C matrix is very similar the matrices AA^T and A^TA seen in [Kleinberg, 1999]. Note that Eigentrust, unlike most other algorithms, is able to handle collusion between malicious sources.

2.3 Probabilistic Methods

Latent Credibility Analysis (LCA) [Pasternack and Roth, 2013] is a probabilistic method that considers the credibility of each claim to be a latent variable, and uses the EM algorithm to uncover it. There are several variations: MistakeLCA, GuessLCA, and LieLCA. A further analysis of LCA is given in Section 4.4.

TruthFinder [Yin et al., 2008] is a pseudoprobabilistic method the trustworthiness of a source is the average truth value of it's claims. The confidence of a fact f is defined as one minus the probability that all systems making this claim are wrong (since each system is making the exact same claim f, they are either all right or all wrong). In further sections of the paper, they also account for such situations as: one fact influences another fact, and systems are not independent.

Data Fusion is another field that has several probabilistic methods. This term is often used for applications where you have multiple types of sensors producing readings on the same phenomenon, and you want to aggregate the readings into a cohesive (and high quality) single stream. For example, [Hazen et al., 2004] improve on standard speech recognition algorithms by fusing results obtained from visual features of the lips. Essentially, this is lip reading. Another speech recognition project puts a sensor inside the subject's mouth and uses the readings as an additional data source [Heracleous et al., 2010]. Another similar idea is to combine multiple types of video streams (for example, thermal imaging, depth, and standard image) to get a more accurate overall representation of a scene. Or, we could combine multiple video images from slightly different times. The idea is that most objects don't change significantly frame by frame, so a perspective of an object at time t can be compared with a different perspective of the same object at time $t + 1$.

[Dong et al., 2012] work on the problem of data integration. They define a cost/recall tradeoff where they have a set of sources of unknown accuracy, and they want to select the minimum number of sources (because each source has an associated cost) with the maximum recall. They define algorithms called Accu and PopAccu. The latter is interesting in particular because they show that under certain conditions it is monotonic, which is defined as follows. For each source that PopAccu adds (integrates) the overall accuracy will not go down. We note that the average source quality is assumed to be very high – on the order of 80% or higher.

[Klementiev et al., 2007] worked on a similar problem, but under the name rank aggregation (they also used the term "data fusion"). They solved the problem of combining multiple rankings of a list into a single, high-quality ranked list. The problem is an optimization which finds a linear combination of rankers such that the average distance of each ranking from the mean ranking is minimized. The intuition is that the correct rankers will agree while the incorrect rankers will choose randomly.

2.4 Supervised Approaches

Now we examine some supervised approaches. The first field to consider is Collective Classification, which is a term for classification of objects within a network structure. The key observation is that while a piece of data can be

classified using its content (the standard machine learning approach), if it is part of a network, then it can be useful to exploit the connections (the context). The intuition is that a node is likely to be similar to it's neighbors, or it is likely that it can be predicted by the labels of it's neighbors.

[Neville and Jensen, 2000] is one of the earlier papers in this field. They introduce Iterative Classification Algorithm (ICA), which is a simple algorithm for classifying nodes in a network structure. In short, the algorithm bootstraps labels from existing labels, and then classifies each node using it's connections as features. Because a node's label may change, then the features for that node's neighbors will change, and they may change their labels also when their time comes. The hope is that eventually the algorithm converges, labels stop being reassigned, and an optimal labeling is achieved.

Note that this approach is very similar to the field called structured prediction, or statistical relation learning. [Taskar et al., 2002] is one work that bridges this gap.

How is Collective Classification different from sructured prediction? Here are some differences:

- Collective Classification algorithms typically assign labels **sequentially** while structured prediction algorithms assign labels **globally**.
- Collective Classification has a **single network structure** while structured prediction problems have a **different network structure for each instance**.

[Sen et al., 2008] is a survey of different collective classification techniques, with experiments and results. They bring up ICA, along with Gibbs sampling and several other

[Tamang et al., 2012] created a system for information extraction validation which gets state of the art results on the TAC-KBP2013 Slot Filling Validation data set. Their method is a standard log-linear model which predicts true or false, and uses many diverse features, most notably in-depth content features for each instance.

In Section 3.2 we introduce a supervised approach that is somewhat different from those mentioned above. The main difference is that the nodes we want to classify have no connection to each other, as is supposed in this literature. That is, our claims do not connect to each other, and so they do not influence the decisions on each other. Also, we are interested in using features from the graph instead of just label propagation features.

2.5 Other Methods

Although eigenvector and probabilistic methods make up most of the existing algorithms, there are some other techniques extant.

Advogato's trust metric [Levien,] is a network flow-based algorithm. The algorithm measures trust between users of an online social network. Each user is a node, and each user has the power to certify other users, which comprises an edge between nodes in the graph. The idea is that communities of "bad" or untrustworthy users may exist, and it is likely that a few of them would be able to convince "good" users to certify them as trustworthy. However, the network flow will see these connections as a bottleneck and will cut off the "bad" nodes from the rest of them.

Chapter 3

New Methods for Trustworthiness

3.1 Multiclass List-valued Extensions

Typically when defining a particular trustworthiness problem, one specifies two parameters: the number of values an answer may take (either two or many), and the number of correct answers allowed (either one or many). For example, the simplest case is when the answers are binary (i.e. they only take two values) and only a single answer can be correct. We call this a binary single-valued trustworthiness problem. Although most algorithms deal with multiclass single-valued problems, to the best of our knowledge, no one has directly addressed the problem of multiclass list-valued trustworthiness.

For example, a query in a multiclass list-valued setting might be "list the children of Ogden Nash." Then, each response to the query is a claim, and more than one claim can be correct. For example, the responses may be "Isabel Eberstadt", "Linell Nash Smith", and "South Korea". The first two are correct, but the last is clearly false (it is not even a person).

With this change in the problem definition, we note several points. First, we see that in this case, the idea of mutual exclusion sets loses it's importance. The mutual exclusion set is largely important for when only one answer is correct, because then all other answers are incorrect. Second, we see that although we motivate the problem as being either single-valued or list-valued, we allow that it can also be mixed. This makes no difference on the outcome of the experiments.

Finally, while we are changing the definitions, we may as well also allow that there is no correct answer. In essence, we have removed all restrictions on number of correct answers.

Related Work

This problem is similar to the rank aggregation problem described in [Klementiev et al., 2007], but with several notable differences. In our formulation, each system makes claims about which elements are in the set (and implicitly, about which elements are not in the set) - it does not give a ranking over elements in the set. When we run our trustworthiness algorithm over the data, we are given a score for each claim. This now means that we have a single ranking, not multiple rankings.

We notice that this problem is analogous to the multi-label problem in classification [Har-Peled et al., 2003]. In the multi-label problem, an instance is given multiple labels, as opposed to just one. In our case, a query has more than one correct answer.

Approach	Precision	Recall	F1
Voting: topn=1	42	18	25
Voting: topn=10	30	45	36
Voting: topn=25	27	57	37
Voting: topn=5%	40	30	34
Voting: topn=25%	29	59	39
HITS: topn=1	**44**	19	26
HITS: topn=10	32	53	40
HITS: topn=25	29	69	41
HITS: topn=5%	39	26	31
HITS: topn=25%	33	49	45
HITS: topn=20% and top 2 SV	35	**69**	**47**

Table 3.1: Results from List-valued experiments

Methods

Most trustworthiness algorithms boil down to giving a score to each claim. The score assigned by majority voting, for example, is just the number of sources asserting this claim. When there is a single correct answer, the task is easy, we just choose the claim with the largest score. With multiple correct answers, there are several different ways to assign correct answers.

- Take the top k claims from the mutual exclusion set.
- Take the top p% of claims from the set.
- Use statistical outlier detection to remove clearly spurious answers.
- Set a numerical threshold and remove all answers below the threshold.

Most of these methods assume that whatever the scoring function is, it assigns scores such that a single threshold is able separate the true and false claims.

We experiment with the top two methods.

Experiments

We used data from the TAC-KPB2013 Slot Filling Validation (SFV) task, which we briefly explain. This task works on the output given by all entrants to the

TAC-KBP2013 Slot Filling (SF) task. The SF task is as follows: a system is given a query, which is either a person or an organization, and is requested to fill in several predetermined slots regarding that query. For example, if the query is "John Smith", the slots might be "date of birth", or "spouse", or "cause of death", etc. Now, once the Slot Filling task has been done, and many systems to solve the task have been submitted, we have a set of potentially conflicting answers. For example, source 1 may say that John Smith died in 1655, while source 21 may say that he died in August, 1990. We have sources making multiple claims about the same set of subjects, and we have ground truth for each claim.

In particular, the SFV task defines several list-valued slots. List-valued slots are slots for which a list of answers is acceptable. For example, "children", "places lived", "title" are all list-valued slots.

For each method, we had the following protocol: if the slot is single-valued, we selected the single most popular answer (ties broken randomly) to be the answer. Every individual claim that was the same as this answer was written to file (we decide sameness between claims by using "document + answer"). For list-valued claims, we kept *topn* claims. We vary this value in our experiments. When it is a percentage, this is the percentage of answers in the mutual exclusion set. Intuitively, a large number of claims in a mutual exclusion set suggests a proportionally large number of correct answers.

All scores are reported in Table 3.1.

The best score is obtained when we use HITS, and take 20% of the top list-valued claims, and the top 2 single-valued claims, which allows that the correct answer may be the second answer if it is not the first.

These scores are no major surprise. Because of the disparity of the number of answers to different types of slot, using a percentage of claims only makes sense.

3.2 Predictive Methods

Most of the trustworthiness algorithms described so far have been either unsupervised, or semi-supervised. We now present a supervised method which involves application of standard machine learning algorithms to the problem of trustworthiness. Although the techniques are not new, to the best of our knowledge, there has been no work in applying techniques in this way.

With this approach, we see each claim as an instance x which is given a label $y \in \{\text{true}, \text{false}\}$. We have labeled training instances, and we extract features from each instance, and train a standard classifier. Then, in testing, each instance is classified as either true or false according to the learned model.

The motivation for this approach is to notice that there are features of the bipartite trustworthiness graph that may have predictive power. Further, we

aren't restricted to features on the graph - we can also include problem specific features to make a richer model.

Related Work

This is most similar to collective classification [Neville and Jensen, 2003], where the graph is homogeneous (only one type of node) and the nodes are classified into categories based on their neighbors. Also, many of the collective classification algorithms have an iterative component because as the neighbors of a node n are classified, the classification for n itself may change. In our situation, we have heterogeneous bipartite graphs (source nodes are distinct from claim nodes), and there are no edges between the nodes to be classified.

Experiments

As our classifier, we used the LBJava [Rizzolo and Roth, 2010] implementation of SparseAveragedPerceptron with parameters learningRate=0.1 and thickness=3.5.

The features we used are described in Table 3.2. These features are not particularly expressive, but we wanted to know how far we could get using just features from the graph (source degree and claim degree).

Source Degree	Each claim is made by a source, and this captures the degree of that source (i.e. The number of claims that this source made). We binned this value into bins of size 500.
Claim Degree	Each claim also has a degree, which is just the number of sources that have asserted the claim. We binned this value into bins of size 5.
Confidence	In our data, every claim was associated with a confidence. It was noisy, but seemed to help. This is an example of a feature which can be added to a richer model.

Table 3.2: Features used in this model

The simplest version of this system enforces no mutual exclusion constraints. Such a setup naively allows, for example, multiple single-valued answers to be correct. To solve this problem, we experimented with using a linear program to correct the answers, much in the same way as in [Pasternack and Roth, 2010]. The results are not reported here, but we found that this made little difference on the F1 score, although it often boosted the precision at the cost of the recall.

We trained on TAC-KBP2012 SFV data, and tested on TAC-KBP2013 SFV data. We used the scorer from Heng Ji at Blender Lab, RPI.

In our experiments, we altered the decision threshold to get different results. This notated as thresh=X in Table 3.4. The default value for the threshold is 0.

ClaimDegreeFeatures:(cdegree:1)	-3.992281
ClaimDegreeFeatures:(cdegree:2)	-1.065480
ClaimDegreeFeatures:(cdegree:3)	1.935768
ClaimDegreeFeatures:(cdegree:4)	3.608687
ConfFeatures:(conf<= 0.5)	-0.330847
ConfFeatures:(conf > 0.5)	0.817541
SourceDegreeFeatures:(sdegree:1)	1.944433
SourceDegreeFeatures:(sdegree:2)	0.953255
SourceDegreeFeatures:(sdegree:3)	-0.967613
SourceDegreeFeatures:(sdegree:4)	0.075643
SourceDegreeFeatures:(sdegree:6)	-1.519025

Table 3.3: Feature Weights

Features	Precision	Recall	F1
All features	63	48	55
All features (thresh=-1)	57	57	**57**
All features (thresh=-2)	44	74	55
All features (thresh=-3)	34	87	49
All features (thresh=1)	65	42	51
All features (thresh=2)	70	34	46
All features (thresh=3)	71	26	39
ClaimDegree	72	30	42
ClaimDegree + SourceDegree	61	50	55
SourceDegree	33	34	33
MTM	50	71	59

Table 3.4: Results from predictive experiments

These results, shown in Table 3.4, are exciting because they are nearly state-of-the-art, and they are using such simple features. We found it interesting that the combination of source and claim degree could be so powerful. It is interesting that in general a lower source degree is better while a higher claim degree is better. This seems to have learned that bad quality sources tend to spam all claims, while good quality sources tend to be selective about which claims to answer.

Chapter 4

Theory of Trustworthiness

4.1 Introduction

We have seen, in the previous chapters, the depth and breadth of work that has been done on trustworthiness. So much work has been done, in fact, that one paper asks in the title, "Is the Problem Solved?" [Li et al., 2012]. But one aspect of the problem that has been largely overlooked is the effect of graph structure on each algorithm. It is clear that certain algorithms will work only under strict graph conditions, but it is unclear what these conditions are.

In general, we would like to answer the question, "given an arbitrary bipartite graph of potentially conflicting claims, what is the algorithm that will produce the best results?"

One way to answer this question is to consider the space of all possible trustworthiness graphs. In this space, there are clearly some graphs for which, in the absence of any other information, no algorithm will perform well on (the complete graph, and the empty graph, for example). Now, on all potentially solvable (for some definition of "solvable") graphs, we can either suggest an existing algorithm, propose a new algorithm, or claim that it is unsolvable.

However, a more practical approach, and the one we adopt, is to start with the canonical algorithms. That is, we carefully examine existing algorithms and try to understand when they work and when they don't. In this research scenario, the above question becomes, "given an arbitrary bipartite graph of potentially conflicting claims, will algorithm X produce good results?"

In this chapter, we examine majority voting, and then two particular algorithms, HITS [Kleinberg, 1999], and Latent Credibility Analysis (LCA) [Pasternack and Roth, 2013].

Let us define a trustworthiness bipartite graphs as follows. We have a graph $G(V, E)$, with V partitioned into sets S and C of size m and n respectively. Intuitively, we understand that a source $s \in S$ makes a claim $c \in C$ and this is represented by edge $(s, c) \in E$. Further, we define a set of mutual exclusion sets called M which partitions C. For the sake of simplicity, we constrain our mutual exclusion sets to be of size two. Intuitively, this corresponds to the notion that the claims c and c' in m_i are mutually exclusive, or, only one can be true.

We see that there are $3^{\frac{|S||C|}{2}}$ such graphs (considering each mutual exclusion

set as a unit, there are three options: $c_1 = 1$, $c_2 = 1$, or neither, and there are $|C|/2$ mutual exclusion sets).

4.2 Majority Voting

It is worthwhile to examine majority voting because in many cases, it is indistinguishable from other algorithms.

We define majority voting to be the algorithm that selects the true claim in each mutual exclusion set as the one with the highest degree (the most votes).

Majority voting is the most naive possible algorithm, and yet there conditions under which it is a reasonable choice. we can say that voting is correct only when the following assumption holds universally: claims with high degree are more trustworthy than claims of low degree.

It seems clear that if we know the credibility of each worker *a priori*, then it is optimal to use a sum weighted by the credibility for each claim, and order claims within mutual exclusion sets by their score. This is, of course, the same process as majority voting, but with weighted votes. Thus, the majority voting approach assumes that all sources have uniform trustworthiness.

By all accounts, this is an unusual situation. Sources generally are believed to have varying quality.

Now, if sources have varying quality, for majority voting to work, the high quality sources need to be prolific. This also is an unusual situation (in fact, certain proverbs suggest the opposite).

In short, majority voting is only reasonable under the following conditions:

- Sources are known to have uniform quality or,
- Sources have varying quality and the high-quality sources are prolific

In the space of all graphs, this corresponds to a relatively small subset.

For a formal discussion of the error bounds of majority voting on any graph, regular or irregular, we direct the reader to [Karger et al., 2011]. They show that the probability of error decays as l, the degree of the sources, rises, and also that the probability of error decays as q, the average expected quality of the sources, rises. In particular, the relationship is as follows:

$$\inf_{G \in \mathcal{G}(m,l)} \sup_{s,f \in \mathcal{F}(q)} d_m(s, \hat{s}_{MV}) = e^{-(C_1 l q^2 + O(l q^4 + 1))}$$

Where d_m is the average probability of error per task, and "$\mathcal{G}(m, l)$ is the set of all bipartite graphs, including irregular graphs, that have m [source] nodes and ml total number of edges." (Quote from [Karger et al., 2011]).

In words, this equation means that with the worst (because maximizing error is bad) selection of s and f, and the best selection of a graph G, the error is less than a certain amount. For our purposes, we assume a fixed graph. In such a

case, the equation still holds, but we see that the error has the potential to be very high, because it is unlikely that our fixed graph is the graph that minimizes error.

Now we consider HITS, which is a modest extension to majority voting.

4.3 HITS

The motivation for Hyperlink Induced Topic Search (HITS) [Kleinberg, 1999] is that majority voting is insufficient because it assigns equal weight to all sources, good and bad alike. As we noted in the previous section, the assumption of uniform quality sources is generally optimistic at best.

For completeness sake, we review the HITS algorithm, using the notation found in [Kleinberg, 1999]. For each node, we initialize the hub score, y_p, and the authority score, x_p, to 1. Then, we iterate until convergence (normalizing at each iteration).

$$x_p \leftarrow \sum_{q:(q,p)\in E} y_q$$

$$y_p \leftarrow \sum_{q:(p,q)\in E} x_q$$

Finally, having obtained a hub and authority score for each node, we order the claim nodes in each mutual exclusion set by the authority score. The claim with the largest authority is chosen to be the most trustworthy. (Note that in the specific case of using a bipartite graph, directed from left to right, the sources will have authority score 0 and the claims will have hub score 0).

Now it is relevant to repeat the question from [Straffin, 1980].

> What do these numbers mean? Why is it that this manipulation through graphs, matrices, eigenvalues and eigenvectors should produce numbers entitled to the name of an "[authority score]"?

A closely related question is, "under what conditions are these numbers entitled to the name of an authority score?"

To answer the first question, we look at the intentions and intuitions of the HITS algorithm. The oft-repeated intuition behind this algorithm is that "good sources make good claims, and good claims are made by good sources." One problem with this saying is that, by symmetry, we can say, "bad sources make bad claims, and bad claims are made by bad sources." Unfortunately, HITS has no notion of "good" or "bad" except for what is available in the graph. As such, HITS uses degree as a first approximation to "goodness."

We examine these intuitions in a concrete example, shown in Figure 4.1. This graph is contrived such that voting gets one question wrong, one question right, and has to guess on the last one. We might say that it gets 1.5 questions

correct. However, when we run HITS on this graph, it so happens that every answer is correct.

Now we make some observations on this graph.

- The graph is disconnected. In fact, it has four connected components, and each one either contains all wrong claims, or all correct claims.
- The claim with highest degree (Bush 7/6/46) is correct.

This leads us to some preliminary conclusions. HITS makes the following assumptions:

1. Nodes with high degrees must be generally correct
2. Correct nodes are closely connected to other correct nodes

Now, we look at the steps of the algorithm on this graph to examine these conclusions. The way HITS works is to create two graphs, AA^T and A^TA, to raise these to successively higher powers until convergence, and then sum across rows (multiply by a vector of ones) to get vectors interpreted as hub scores and authority scores respectively.

Effectively, this means that there are two separate problems being solved simultaneously: finding the principal eigenvector of AA^T and finding that of A^TA. This means that we can explore them separately.

We explore the situation involving A^TA as the power iteration progresses. First of all, A^TA by itself can be thought of as a coreference matrix. The value of $[A^TA]_{ij}$ simply measures the number of parents shared by node i and node j (for diagonals, this is trivially the indegree). As we take the powers of A^TA, we notice first of all that at each iteration the diagonals increase by a factor of at least the original indegree. If they increased by exactly a factor of the original indegree, the problem would be uninteresting. We are really interested in seeing how the ranking of nodes changes from the original indegree.

So we look at what drives such a change: it is values *not* on the diagonal. In A^TA, these represent number of common parents.

There are some things we can say about the structure of A^TA that follow from the way it is constructed.

- If the sum of the diagonals is larger than the number of sources, then, by the pigeonhole principle, at least one source must have degree > 1, which means that at least one pair of claims has common parents, which means at least two non-diagonals in A^TA are non-zero (two because of symmetry). In fact, let each element of A^TA represent the size of a set of nodes. Each set s_{ij} is understood to contain the common parents of nodes i and j. Now we look at a single row i, and we see:

$$|s_{ii}| = |\cup_{j:j\neq i} s_{ij}|$$

(A similar equation holds true for columns by symmetry). Further:

$$s_{ij} \subseteq s_{ii}$$

$$s_{ij} \subseteq s_{jj}$$

This is just a mathematical way to relate the diagonals to the non-diagonals.

(This brings up another interesting question: the non-diagonals of A^TA are only sizes of sets, which implies that it may be possible to have more than one graph that creates the same A^TA. It seems that this is not possible, because A is unique for a given graph.)

- We can also see that $a_{ij} \leq min(a_{ii}, a_{jj})$
- Since we are talking about trustworthiness, we can include the notion of mutual exclusion sets. But it is problematic to place restrictions on them (exactly one answer in each set must be true, in particular). This means that at best we can write heuristics such as: it is unlikely that a single source will be connected to more than half of the claims.

One thing we can say is: if a graph is connected (and for our purposes connected graphs are the only interesting ones) then this means that every row (and also every column, by symmetry) will have some non-zero value not on the diagonal. Further, most nodes will have more than one non-zero value not on the diagonal. This means that as n increases in $(A^TA)^n$, then every value in the matrix will become non-zero.

This introduces an important intuition. For the time being, let us ignore the size of each number in A^TA, and just think about zero vs. non-zero values. Remember that we can think of A^TA as being a matrix that records common parents. Now think of A^TA as being an adjacency graph (representing a connected graph), and think of $(A^TA)^2$ in the sense of $(A^TA)^T(A^TA)$. This is true because A^TA is symmetric. This operation can be thought of in the same light as A^TA : it finds common parents among nodes in the graph represented by A^TA. If we unroll the recursion, we see that common parents in this graph now means "common children of common parents as seen in A^TA." This is all very hand-wavy, but the recursion continues as more powers of A^TA are taken (eg "common parents of common children of common parents as seen in A^TA." This is also an informal proof of why $(A^TA)^n$ has no non-zero elements in the limit when A is connected).

Theorem 1. *Let A be the adjacency matrix of an undirected graph with n nodes. We allow that the graph can have any number of edges between two given nodes, and each edge has weight 1. Thus, $[A]_{i,j} \in \{0, 1, 2, 3, ...\}$. A is symmetric and has size $n \times n$. If A is connected and if the diagonals of A are all non-zero (i.e. every node in the graph has a self loop), then every element of A is non-zero in A^i where $i \leq n$.*

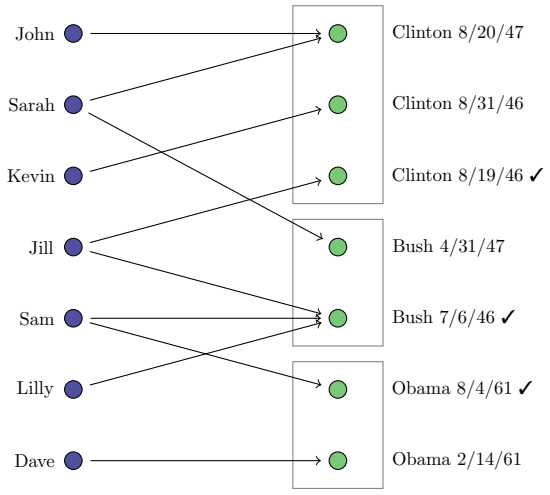

Figure 4.1: Graph contrived to work for HITS, but fail for voting (with thanks to Jeff Pasternack).

Proof. We need to show three things: that values become non-zero, values never become zero, and that all values have become non-zero before n powers.

The proof lies in the fact that $[A^k]_{i,j}$ represents the number of paths from node i to node j (a common understanding). If two nodes are distance $k+1$ apart, $[A^k]_{i,j} = 0$. Now, let $[A^{k+1}]_{i,j} = 1$. This means that there is one path between i and j of length $k+1$. Since every node (including i and j) has a self loop, there also exists at least two paths of length $k+2$. This is because we can follow the self loop on i and then the path of length $k+1$ to j, or follow the path of length $k+1$ from i to j and then follow the self-loop on j.

Since the graph is connected, the longest path between any two nodes has length $\leq n$. □

There is another intuition here: after enough iterations, influence flows from every node directly to every other node (influence is correlated with score - high influence means high score). However, if it takes several iterations for two nodes to finally be connected, it could be the situation that one node has grown so much relative to the other that it is too late to have much influence. This gives us the idea that influence is most effective at close proximity or graph distance, i.e. the closer you are to a high-degree node, the better chance you have of being affected by it.

Our conclusion: HITS works well when the true answer in each mutual exclusion set has a higher "distance score" than all the false answers. In all other situations, HITS acts as well as majority voting.

4.3.1 Graph Construction

If we properly understand the way that HITS exploits the structure in Figure 4.1, then we should be able to encode the assumptions in an algorithm that can automatically construct such graphs.

This algorithm is described in Algorithm 1. In English, the algorithm knows beforehand which claims are correct and which are incorrect. It selects a small number of correct claims to have high-degree. These are connected to sources in such a way as to encourage a relatively small number of "correct" sources. Then we add a single edge to each other correct claim. Again, we choose sources that are likely to be "correct" sources. Finally, we connect each bad claim to two different sources. This means, of course, that in most mutual exclusion sets, voting will give the wrong answer, which is the point of this exercise. We are careful to connect bad claims with low-scoring sources, according to some definition of scoring. For this graph, we define a score vector s of length n as follows:

$$s(i) = \sum_{j=1}^{n} deg(j) \cdot \exp(-dist(i,j))$$

where $deg(j)$ is the in-degree of claim j and $dist(i,j)$ is the distance between claim i and j, defined as the minimum number of edges needed to travel from i to j in an undirected graph. Notice that in a bipartite graph, this distance is always even.

We can intuitively understand this function by seeing that degree drives score, and my score can benefit from the degree of my connections, but that influence decays as they are farther from me.

Note also that at some point $i == j$. This is by design. This is how your own degree affects your score ($dist(i,i) == 0$).

Note: this algorithm is slow. It can be sped up, but that is not a major focus.

Limitations of the algorithm

One major problem with this algorithm is that it doesn't keep track of how scores of previously examined claims change when adding new edges. For example, when adding edges to bad claims, we make sure to connect the sources that result in the lowest score. However, by making this connection, we are increasing the score of every node connected to it (which are all bad nodes). This means that many of the nodes could increase in score until they are the nodes which appear to be right.

This is seen occasionally when HITS seems to get everything backwards.

Algorithm 1 Graph Construction

Input: N, M, *NumMaxDegree*, *MaxDegree*
Output: Graph g
// Assume: each mutual exclusion set has two elements, the first one is always correct and the second is always incorrect.

// Populate graph with vertices
g = DirectedGraph()
add N source nodes to g, naming them `s0` through `s{N-1}`.
add M claim nodes to g naming them `0` through `{M-1}`.
// good claims have even numbers, bad claims have odd numbers

// Add edges to high-degree correct claims
define a uniform distribution D_s over sources
highdegreeclaims = randomly select *NumMaxDegree* claims from the good claims list without replacement
for *claim* in *highdegreeclaims* **do**
    target = select *MaxDegree* sources from D_s without replacement
    g.add_edges(target, *claim*)
    update D_s by increasing probability of sources in target
end for

// Add edges to low-degree correct claims
for *claim* in *goodclaims* **do**
    g.add_edge(source selected from D_s, *claim*)
end for

// Add edges to low-degree incorrect claims
for *claim* in *badclaims* **do**
    for src in sources **do**
        g.add_edge(src, *claim*)
        get score of *claim* and store it
        g.remove_edge(src, *claim*)
    end for
    choose the two sources with smallest scores and add corresponding edges to g
end for

return g

Simulations with the Algorithm

Although this algorithm does not always work as intended, it is capable of producing good results. For our purposes, this is all we need - we want to show that a graph exists where HITS works well and voting fails.

We ran the algorithm with the following parameters:

- NumSources (N) = 30
- NumClaims (M) = 100
- NumMaxDegree = 15
- MaxDegree = 5

NumMaxDegree is 15, which means that 15 of the 50 mutual exclusion sets will be such that the correct claim has degree 15 and the incorrect claim has degree MaxDegree, which is 5 in this case. For every other mutual exclusion set, the incorrect claim had had degree 5, and the correct claim had degree 2.

This means that 15 of the 50 mutual exclusion sets will be correct with voting, but the rest will be wrong.

Our results were as follows:

- **Voting**: 15 correct / 50 mutual exclusion sets
- **HITS**: 50 correct / 50 mutual exclusion sets

These results are not proof that our graph construction algorithm is correct, but since we are able to create the graph with such good success, this is a good indication that it is correct.

4.3.2 Pairwise Stability of HITS

In a discussion of the success rate of HITS, we would be remiss not to cite certain other work. [Ng et al., 2001] showed that the stability of the authority scores given by HITS is correlated with the eigengap of $A^T A$ (where the eigengap is the difference between the two largest eigenvalues). Specifically, they showed that a global ranking induced by authority scores is generally unstable, but is more stable with a large eigengap.

However, in the case of trustworthiness, we are not concerned with a global ranking. We care mostly about rankings in each mutual exclusion set. We do not care if a global ranking is perturbed as long as the local rankings are intact.

In summary, HITS will work only when the graph has the characteristics described above. One important result of this is to show that for at least one algorithm, it is nonsensical to presume that it works well across the space of all graphs.

Next we delve into the world of probabilistic algorithms and discuss when Latent Credibility Analysis (LCA) works and when it doesn't.

4.4 Latent Credibility Analysis

Latent Credibility Analysis (LCA) [Pasternack and Roth, 2013] is a probabilistic method for trustworthiness where the credibility of each claim is the latent variable. This latent variable is estimated using Expectation Maximization (EM) [Dempster et al., 1977].

This framework captures a useful formalism in the $w_{s,m}$ variable, which determines if source s makes a claim in mutual exclusion set m, regardless of what the claim is. However, the assumption is that the distribution of the $w_{s,m}$ variables is unimportant to the success of the algorithm. In this section, we show that even when a graph is created using the assumed model, this distribution can have a significant impact on the success or failure of any given run. That is, even when a graph is created according to the assumed model, the algorithm can fail. In this section, we aim to understand what forms of graph make a difference. The algorithms presented by LCA are intended for particular graph generation models (for example, LieLCA is clearly inappropriate in certain situations, where perhaps GuessLCA fits the assumptions better). However, we show that even when the correct modeling assumption is made, a graph can be created on which LCA fails.

In this section, we explore only SimpleLCA. The reason for this is that since the underlying mechanism is the same for each algorithm, the understanding we take from SimpleLCA should carry over to the more complex algorithms. We leave the study of the other models to future work. Hereafter, when we say LCA, we mean SimpleLCA.

Since LCA is based on EM, it is iterative, iterating between an expectation step that calculates claim probabilities and a maximization step that calculates source probabilities. As such, it has an immediate parallel to HITS, which is also iterative, with the same intuitions for each iteration. It is interesting to ask how the algorithms are the same and how they are different.

The HITS update step for the hubs is

$$y_p \leftarrow \sum_{q:(p,q)\in E} x_q \tag{4.1}$$

In words, this calculates the hub score (or trustworthiness) of each source by taking the sum of the scores of all it's neighbors (which are claims in our formulation). In LCA, this is equivalent to

$$H_s = \frac{\sum_m \sum_{y_m} P(y_m \mid X, \theta^t) w_{s,m} b_{s,y_m}}{\sum_m w_{s,m}} \tag{4.2}$$

We see that the $w_{s,m}$ and the b_{s,y_m} control the sum such that the only non-zero terms in the numerator come from those claims which s is connected to; in other words, the neighbors of s. In essence, this formula calculates the trustworthiness of a source as the sum of the trustworthiness of the claims it

gives, divided by the number of mutual exclusion sets it participates in (if we assume that a source gives no more than one claim per mutual exclusion set, then this number is equivalent to the number of claims made).

Right away, we see that this update is very similar to HITS, but different in a significant way. In the previous section, we made the point that HITS is largely driven by degree, and a source can get a good score simply by making many claims (this is a problem that is resolved one of the later models from [Pasternack and Roth, 2010]). But in this situation, the score of a source is the average of all the claims it makes: more claims won't give you better score. Only better claims will raise your score. This is good modeling.

Now we look at the authority score update in HITS:

$$x_p \leftarrow \sum_{q:(q,p)\in E} y_q$$

This calculates the authority score of a claim by taking the sum of the hub scores of all it's neighbors. In LCA, the equivalent is the following exposition (to lighten the notation, we write $P(c, ...)$ when we mean $P(c = \text{true}, ...)$). First we calculate the probability that a certain claim is true given a certain source s.

$$P(c, X \mid H_s) = P(c) \cdot \left[(H_s)^{b_{s,c}} \cdot \left(\frac{1 - Hs}{|m| - 1} \right)^{(1-b_{s,c})} \right]^{w_{s,m}}$$

This is just a single simple value, depending on if $b_{s,c}$ is 1 or 0. This will end up being either $P(c) \cdot H_s$, $P(c) \cdot (1 - H_s)/(|m| - 1)$, or $P(c)$.

Then we aggregate the opinions of all sources by taking the product

$$P(c, X \mid \theta) = \prod_s P(c, X \mid H_s)$$

And finally we normalize within the mutual exclusion set. (This step is not strictly necessary. Since we compare the resultant scores only within a mutual exclusion set the numerator alone is sufficient.)

$$P(c \mid X, \theta^t) = \frac{P(c, X \mid \theta^t)}{\sum_{v \in m} P(v, X \mid \theta^t)}$$

Interpreted, this means that the believability of a claim is the product of responses from each source (not just those that contribute a claim to the mutual exclusion set), where a response is either an assertion that this claim is true (with weight H_s), or an assertion that this claim is not true (because another claim is true) (with weight $(1 - H_s)/(|m| - 1)$), or an ignorant assertion (with weight $P(c)$).

When scoring a claim c, HITS takes the sum of the scores of sources asserting that claim. LCA is different in that it also takes into account the fact that if a source asserts a different claim c', this carries the implicit judgment that c is

false.

With these differences in mind, we can make the conclusion that the signal exploited by LCA is agreement between sources. That is, if it is going to work in an unsupervised situation (and simulations and experiments show that it does work in such situations), then it must exploit some signal, and we claim that the signal is agreement between sources.

We note that this intuition of exploiting agreement between sources is not a novel idea, but can also be found in [Karger et al., 2011] and [Klementiev et al., 2007].

But when does it work? This, and the sister question, when does it fail?, are of interest to us now.

First, we note that it is sensitive to initial conditions of the parameters (as EM-based algorithms are known to be). If we initialize with $H_s = \{1\}^m$, it never converges. Further, the following inequality must hold true for most sources.

$$H_s > \frac{1 - H_s}{|m| - 1} \tag{4.3}$$

If this is not true, then this means that source s believes that the claims it did not assert are more likely to be true than the one it asserted. From a modeling perspective, this makes no sense. If a source doesn't believe a claim is true, it shouldn't assert it.

Equation 4.3 is a nice way to relate the size of mutual exclusion sets and the distribution over H_s. In general, we see that larger mutual exclusion sets lead to better results. Intuitively, this makes sense: the common intuition for agreement-based systems is that correct answers are all the same, but incorrect answers are randomly spread out. If there is more space to spread out random answers, then there is less chance that they form a pattern that competes with the right answers.

We also note that if H_s is initialized to $1/|m|$, then this doesn't work. The reason is that we have

$$H_s = \frac{1}{|m|}, \quad \frac{1 - \frac{1}{|m|}}{|m| - 1} = \frac{1}{|m|} \tag{4.4}$$

This modeling means that all answers are equally likely, and the algorithm can never recover from that. This makes a lot of sense: we would be modeling a situation where any given source is just as likely to give the right answer as any wrong answer. There is no signal here, and we cannot expect an algorithm to exploit it in the absence of other information. If $H_s < 1/|m|$, this is an adversarial situation, and we don't consider this.

However, it is not enough that the inequality in 4.3 be satisfied. Imagine the simple situation where 2/3 of the sources have quality $1/|m|$, and 1/3 have quality 0.9. On average, 4.3 is satisfied.

Now imagine constructing a graph with these sources such that the probabil-

ity that a source makes a claim in a mutual exclusion set is $1 - H_s$. If it decides to make a claim, then it follows the standard probability H_s of being correct. In this situation, we have good sources which are quiet, but bad sources which are prolific.

In our simulations of LCA, the accuracy in such situations is usually around that of majority voting, or worse. Our conclusion from this is that the structure of the graph makes a difference on the result.

4.5 Future Work

These two algorithms only give us a small taste of how all algorithms work in the space of trustworthiness graph. There are obvious research paths in studying the variations on the models presented above, including AverageLog, Investment, PooledInvestment, and the different flavors of LCA, as well as all different algorithms. It seems likely that patterns will emerge, and we may find that most algorithms are accomplishing the same thing in slightly different ways. This would be an interesting result. Of course, the next question is how to cover the full space of trustworthiness graphs.

Related to that idea is the problem of knowing if a graph is even solvable. That is, there may not be any exploitable in the graph. If this is the case, then no algorithm can be successful. It would be valuable to be able to discover this just from the graph structure. It may be prudent to follow an information theoretic trail to solving this problem.

Another important direction is to know how to categorize a graph in terms of structure. How does the degree distribution over sources and over claims affect the result? Is it possible that two graphs with the same distribution over claim degrees and the same distribution over source degrees can be non-isomorphic (from a trustworthiness perspective)? In general, how can we succinctly describe a graph in such a way that divides the space of graphs into meaningful partitions (from a trustworthiness perspective)?

Algorithm 2 LCA algorithm

Input: B
Output: θ, estep

// Initialize
$H_s = \{0.5\}^n$
$estep = \{0\}^{m \times n}$

while true **do**

  // E-step
  for $s \in S$ **do**
    for $m \in M$ **do**
      for $c \in m$ **do**
        $\text{estep}(s,\, c) = P(c) \cdot \left[(H_s)^{b_{s,c}} \cdot \left(\frac{1-Hs}{|m|-1} \right)^{(1-b_{s,c})} \right]^{w_{s,m}}$
      end for
    end for
  end for

  if converged, break.

  // M-step
  for $s \in S$ **do**
    $num = denom = 0$
    for $m \in M$ **do**
      $denom \mathrel{+}= w_{s,c}$
      for $c \in m$ **do**
        $num \mathrel{+}= \frac{\prod_s estep(s,c)}{\sum_{c \in m} \prod_s estep(s,c)} \cdot w_{s,m} \cdot b_{s,c}$
      end for
    end for
    $H_s = num/denom$
  end for

end while
return $\theta = \{H_s\}, estep$

References

[Artz and Gil, 2007] Artz, D. and Gil, Y. (2007). A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., Rubin, D. B., et al. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38.

[Ding et al., 2004] Ding, C. H., Zha, H., He, X., Husbands, P., and Simon, H. D. (2004). Link analysis: hubs and authorities on the world wide web. *SIAM review*, 46(2):256–268.

[Dong et al., 2012] Dong, X. L., Saha, B., and Srivastava, D. (2012). Less is more: Selecting sources wisely for integration. In *Proceedings of the 39th international conference on Very Large Data Bases*, pages 37–48. VLDB Endowment.

[Gould, 1967] Gould, P. R. (1967). On the geographical interpretation of eigenvalues. *Transactions of the Institute of British Geographers*, (42):pp.53–86.

[Har-Peled et al., 2003] Har-Peled, S., Roth, D., and Zimak, D. (2003). Constraint classification for multiclass classification and ranking. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 785–792. MIT Press.

[Hazen et al., 2004] Hazen, T. J., Saenko, K., La, C.-H., and Glass, J. R. (2004). A segment-based audio-visual speech recognizer: Data collection, development, and initial experiments. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 235–242. ACM.

[Heracleous et al., 2010] Heracleous, P., Badin, P., Bailly, G., and Hagita, N. (2010). Exploiting multimodal data fusion in robust speech recognition. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 568–572. IEEE.

[Kamvar et al., 2003] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM.

[Karger et al., 2011] Karger, D. R., Oh, S., and Shah, D. (2011). Iterative learning for reliable crowdsourced systems. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*.

[Karger et al., 2013] Karger, D. R., Oh, S., and Shah, D. (2013). Efficient crowdsourcing for multi-class labeling. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 81–92. ACM.

[Kleinberg, 1999] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

[Klementiev et al., 2007] Klementiev, A., Roth, D., and Small, K. (2007). An unsupervised learning algorithm for rank aggregation. In *Proc. of the European Conference on Machine Learning (ECML)*.

[Levien,] Levien, R. Advogato's trust metric. http://www.advogato.org/trustmetric.html.

[Li et al., 2012] Li, X., Dong, X. L., Lyons, K., Meng, W., and Srivastava, D. (2012). Truth finding on the deep web: is the problem solved? In *Proceedings of the 39th international conference on Very Large Data Bases*, pages 97–108. VLDB Endowment.

[Neville and Jensen, 2000] Neville, J. and Jensen, D. (2000). Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20.

[Neville and Jensen, 2003] Neville, J. and Jensen, D. (2003). Collective classification with relational dependency networks. In *Proceedings of the Second International Workshop on Multi-Relational Data Mining*, pages 77–91. Citeseer.

[Ng et al., 2001] Ng, A. Y., Zheng, A. X., and Jordan, M. I. (2001). Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 903–910. LAWRENCE ERLBAUM ASSOCIATES LTD.

[Pasternack and Roth, 2010] Pasternack, J. and Roth, D. (2010). Knowing what to believe (when you already know something). In *Proceedings the International Conference on Computational Linguistics (COLING)*, Beijing, China.

[Pasternack and Roth, 2013] Pasternack, J. and Roth, D. (2013). Latent credibility analysis. In *The International World Wide Web Conference*.

[Rizzolo and Roth, 2010] Rizzolo, N. and Roth, D. (2010). Learning Based Java for Rapid Development of NLP Systems. In *Proceedings of the International Conference on Language Resources and Evaluation*, Valletta, Malta.

[Sagae and Lavie, 2006] Sagae, K. and Lavie, A. (2006). Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132. Association for Computational Linguistics.

[Sen et al., 2008] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93.

[Straffin, 1980] Straffin, P. D. (1980). Linear Algebra in Geography: Eigenvectors of Networks. *Mathematics Magazine*, 53(5):269–276.

[Tamang et al., 2012] Tamang, S., Chen, Z., and Ji, H. (2012). Cuny blender tac-kbp2012 entity linking system and slot filling validation system. proc. In *Text Analytics Conference (TAC2012)*.

[Taskar et al., 2002] Taskar, B., Abbeel, P., and Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 485–492. Morgan Kaufmann Publishers Inc.

[Yin et al., 2008] Yin, X., Han, J., and Yu, P. S. (2008). Truth discovery with multiple conflicting information providers on the web. *Knowledge and Data Engineering, IEEE Transactions on*, 20(6):796–808.