Probabilistic Models for Fine-Grained Opinion Mining: Algorithms and Applications

BY

Arjun Mukherjee
B.Tech., Sikkim Manipal Institute of Technology, 2009

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2014

Chicago, Illinois

Defense Committee:

Bing Liu, Chair and Advisor, Department of Computer Science
Ugo Buy, Department of Computer Science
Chris Kanich, Department of Computer Science
Junhui Wang, Department of Mathematics, Statistics, and Computer Science
Philip S. Yu, Department of Computer Science

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my advisor Prof. Bing Liu for giving me the precious opportunity to pursue my doctoral studies under his guidance. His advice and instructions have been a pivotal element of my research preparation. He has been very supportive in also allowing me to explore my pet projects and ideas and been a constant source of encouragement and advice. It has been a very rewarding experience to be his Ph.D. student. I appreciate all his contributions of time, ideas, funding, and help to make my Ph.D. experience productive and stimulating, especially during tough times in the Ph.D. pursuit.

I also want to thank all my collaborators for their support. Discussions with Zhiyuan Chen, Malu Castellanos, Geli Fei, Michael Gamon, Riddhiman Ghosh, Natalie Glance, Meichun Hsu, Nitin Jindal, Abhinav Kumar, Huayi Li, Patrick Pantel, Sathish Ramados, Jianfeng Si, Vivek Venkataraman, and many others have been very helpful in shaping my research.

I would also like to thank my other thesis committee, Prof. Ugo Buy, Prof. Chris Kanich, Prof. Junhui Wang, and Prof. Philip S. Yu for their support and assistance.

Lastly, my greatest gratitude go to my own family without whom this thesis would not have been possible. The support system rendered by them to do along with the various troubles I gave them in the course of this PhD have been phenomenal.

AM

TABLE OF CONTENTS

CHAPTER **PAGE**

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The tremendous growth of the Web and the surge of social media render massive amounts of data that present new forms of actionable intelligence to corporate, government, and political sectors. The vast amount of user generated content in the Web today has valuable implications for the future. Public sentiments in online debates, discussions, news comments are crucial to governmental agencies for passing new bills/policy, gauging social unrest/upheaval, predicting elections, etc. However, to leverage the sentiments expressed in social opinions, we face two major challenges: (1) fine-grained opinion mining, and (2) filtering opinion spam to ensure credible opinion mining.

We first address the problem of mining opinions from social conversations. We focus on fine-grained sentiment dimensions like agreement (*I'd agree*, *point taken*), disagreement (*I refute*, *I don't buy your*), thumbs-up (*good critique*, *review convinced me*), thumbs-down (*biased review*, *seemingly honest*), question (*how do I*, *can you explain*), answer acknowledgement (*clears my*, *useful suggestion*). This is a major departure from the traditional polar (positive/negative) sentiments (e.g., *good*, *nice* vs. *poor*, *bad*) in standard opinion mining [Liu, 2012]. We focus on two domains: (1) debates and discussions, and (2) consumer review comments. In the domain of debates, joint topic and sentiment models are proposed to discover disagreement and agreement expressions, and contention points or topics both at the discussion level and also at the individual post level. Proposed models also encode social behaviors, i.e., interactions among discussants or debaters and topic sharing among posts through quoting and replying relations. Evaluation results using real-life discussion/debate posts from several domains demonstrate the effectiveness of the proposed models. In the domain of review comments, we additionally discover other dimensions of sentiments mentioned above.

Next, we address the problem of semantic incoherence in aspect extraction. Aspect extraction is a central problem in opinion mining which tries to group and extract semantically related aspects.

SUMMARY (Continued)

Although unsupervised models (e.g., LDA) are widely used, they suffer from semantic incoherence. We solve the problem by encoding knowledge in the modeling process using seed sets. Seeds are certain coarse groupings which may be provided by the user to guide the modeling process. Specifically, we build over topic models to propose aspect specific sentiment models guided by aspect seeds.

The later part of this thesis proposes solutions for detecting opinion spam. Opinion spam refers to "illegitimate" human activities (e.g., writing fake reviews) that try to mislead readers by giving undeserving opinions/ratings to some entities (e.g., hotels, products) to promote/demote them. We address two problems in opinion spam. First is the problem of group spam, i.e., a group of spammers working in collusion. Our experiments show that it is hard to detect group spammers using content features or even abnormal behaviors of individual spammers because the group context can enshroud its members who may no longer appear to behave abnormally. A novel relational ranking algorithm called GSRank is proposed for ranking spam groups based on mutual-reinforcement. GSRank is an instance of an eigenvalue problem and has robust theoretical guarantees on convergence. Experiments using real-world data from Amazon.com show that GSRank significantly outperforms a series of strong competitors like SVMRank and RankBoost and various baselines based on heuristics and review helpfulness metrics.

The second problem is opinion spam detection in the absence of labeled data. The situation is important as it is hard and erroneous to manually label fake reviews or reviewers. To address the problem, we leverage on the key hypothesis that spammers differ markedly from others on several behavioral dimensions which creates a distributional divergence between the populations of two (latent) clusters: spammers and non-spammers. Working in the Bayesian setting and modeling spamicity of users as "latent" with observed behavioral footprints, novel generative models are proposed for detecting opinion spam/fraud.

SUMMARY (Continued)

Experiments using real-world reviews from Amazon.com demonstrate the effectiveness of our models which outperformed several state-of-the-art competitors. To our knowledge, this is the *first* work to employ principled statistical modeling for detecting opinion spam in the unsupervised setting.

Thus, this work proposes statistical models and algorithms for opinion mining and opinion spam detection with an emphasis on generative joint models of language and user behaviors in social media. It lies at the confluence of web mining, computational linguistics, Bayesian inference, and probabilistic graphical models; and proposes novel models leveraging concepts and methodologies from all the aforementioned research domains.

CHAPTER 1

INTRODUCTION

Recent surveys have revealed that the opinion-rich resources offered by social media on the Web such as online reviews, comments, and discussions are having greater economic impact on both consumers and companies compared to traditional media [Pang and Lee, 2008].

Among other forms of social media, consumer reviews and comments have stood out to be important metrics of business intelligence, decision making, and market analysis. Also for assessing the pulse of people and their overall sentiment across a myriad of issues, forms of social media like Twitter, debate forum discussions, and newsgroup discussions are probably among the important media that the Web has to offer.

The core research area which governs this domain is opinion mining and sentiment analysis. However, for an end-to-end opinion system to be capable of the applications mentioned above, there are several challenges that need to be addressed which go beyond what the existing state-of-the art techniques can handle. The first challenge lies in building robust algorithms that can crunch Web scale data and draw inferences from them. For opinion mining in social media to be effective, which has vast amounts of user-generated content, it is also desirable for models to respect the human notions of semantics, extract fine-grained opinions and sentiments, and be noise/fault tolerant.

In the past decade, probabilistic topic models [Blei et al., 2003] have attracted a dramatic surge of interest in the field of information retrieval, data mining, computational linguistics, and web mining owing to their capability of discovering the hidden thematic structures in large archives of documents. In addition, probabilistic topic models are highly extensible and have proven to be effective for corpus exploration. Topic modeling provides a foundation for analyzing massive datasets and is a

powerful tool. However, on its own, topic models are not capable of capturing opinions and sentiments expressed is text. Further, most existing models being unsupervised are noisy and suffer from semantic incoherence. Hence, we need additional techniques to address our goal of fine-grained opinion mining.

The work in this thesis bridges the gap. It focuses on developing probabilistic topic and sentiment models for fine-grained opinion mining from unstructured social media. It also draws upon methods from linguistics, social behaviors, and proposes several joint models of topics, sentiments and social behaviors which render a deeper understanding, richer modeling, and analysis of opinions expressed in social media. Such analysis not only renders an insight into various complex social phenomenon (e.g., automatic gender attribution from weblogs [Mukherjee and Liu, 2010], mining user viewpoints and contentions in online debates [Mukherjee and Liu, 2012a], modeling communication tolerance [Mukherjee et al., 2013a], consumer comments [Mukherjee and Liu, 2012c], detecting deceptive opinion spam [Mukherjee et al., 2011; Mukherjee et al., 2012; Mukherjee et al., 2013b; Fei et al., 2013; Mukherjee et al., 2013c], modeling stock indices [Si et al., 2013]), but also paves the way for accurate prediction for such phenomenon. At the technical level, it proposes novel semi-supervised models for aspect extraction [Mukherjee and Liu, 2012b], a core problem in opinion mining and lays the foundation for more general knowledge induction in other opinion mining applications and knowledge based aspect extraction [Chen et al., 2013a; Chen et al., 2013b; Chen et al., 2013c]

1.1 Opinion Mining and Sentiment Analysis

Opinion mining or sentiment analysis analyzes people's opinions and attitudes towards entities such as products, events, topics, etc. Aspect and sentiment extraction are among the keys tasks. For example, the sentence "The iPhone's call quality is good, but its battery life is short" evaluates two aspects, call quality and battery life, of iPhone (entity). The sentiment on call quality is positive but the sentiment on battery life is negative.

The task aspect extraction in aspect-based opinion mining further consists of two sub-tasks. The first sub-task extracts *aspect terms* from an opinion corpus. The second sub-task clusters synonymous aspect terms into categories where each category represents a single aspect, which we call an *aspect category*. Existing research has proposed many methods for aspect extraction. They largely fall into two main types. The first type only extracts aspect terms without grouping them into categories. The second type uses statistical topic models to extract aspects and group them at the same time in an unsupervised manner. These are detailed in Chapter 3.

1.2 Research Challenges

In this section, we detail the research challenges mentioned before for fine-grained opinion mining and leveraging actionable knowledge from sentiments expressed in social media. Our scope is primarily opinion mining and knowledge based sentiment analysis but also encompasses user behaviors in social media and opinion spam. Opinion spam and social behaviors are tightly coupled with opinion mining as filtering opinion spam and fraudulent users and their behaviors on the Web is a precondition for reliable opinion mining. The solution methodologies are based on joint models of topics and sentiments, user behaviors and the nature of social dynamics which lay the foundation for our analysis and also motivate the development of novel models.

1.2.1 Modeling social conversations in debates and consumer discussions

A major part of social media involves blogs, comments, discussions and debates. Such kind of social media (e.g., newsgroup discussions, online debates, and forum discussions) have been widely mined for discovering various structures ranging from support-oppose classification [Murakami and Raymond, 2010], placing debate authors into for and against camps [Agarwal et al., 2003], and mining stances in online debates [Thomas et al., 2006]. These works mainly deal with classification of linguistic utterances into agree, disagree, and backchannel classes [Murakami and Raymond,

2010]; using reply networks to place debaters into support and oppose camps [Agarwal et al., 2003]; and mine debate stances using unsupervised classification [Somasundaran and Wiebe, 2009] and collective classification techniques [Burfoot et al., 2011].

However, the above works do not discover the linguistic expressions used for expressing agreement (e.g., "*I agree*," "*I think you're right*") or contention (e.g., "*I disagree*," "*you make no sense*") which are integral aspects of debates. Furthermore, existing works do not discover contention points (issues on which contentions are raised) expressed in debates or consumer discussions. Hence, we may regard the prior works as coarse-grained analysis as opposed to our fine-grained analysis.

The problem is both computationally challenging and potentially important because a large part of social media is about discussions/debates of contentious issues (e.g., social, political and religious issues) and discovering such issues is useful for many applications. For example, in a political election, contentious issues often separate voters into different camps and determine their political orientations. It is thus important for political candidates to know such issues. For contentious social topics, it is crucial for government agencies to be aware of them so that they can address the problems. Even for consumer products/services, contentions about them can be used to identify different types of customers and to make effective marketing and business decisions.

Extending further in the context of consumer reviews and comments, most of the existing work focuses on general sentiment analysis [Pang and Lee, 2008] feature based aspect extraction [Hu and Liu, 2004], and some others including utility, helpfulness assessment, and quality prediction in reviews [Kim et al., 2006; Zhang and Varadarajan, 2006]. However, reviews and their summarization using existing approaches only give the evaluations and experiences of the reviewers. Often a reviewer may not be an expert of the product and may misuse the product or make other mistakes. There may also be aspects of the product that the reviewer did not mention that a reader wants to know.

Thus, to improve the online review system and user experience, some review hosting sites (e.g., Amazon.com) allow readers to write comments about reviews (apart from just providing a feedback by clicking whether the review is helpful or not). Many reviews receive a large number of comments. It is difficult for a reader to read them to get a gist of them. This puts forth the research challenge of automatically mining the following six major kinds of comment expressions:

1. Thumbs-up (e.g., "review helped me")
2. Thumbs-down (e.g., "poor review")
3. Question (e.g., "how to")
4. Answer acknowledgement (e.g., "thank you for clarifying")[1].
5. Disagreement (contention) (e.g., "I disagree")
6. Agreement (e.g., "I agree").

The above problem belongs to the domain of information extraction and is particularly useful in not only generating fine-grained summaries of consumer comments and discovering key issues which commenters have trouble with, but also lays the basic building blocks for mining latent trust relationships that commenters have with reviewers. Additionally, this would facilitate discovering the experts among reviewers based on the nature of aggregate responses received by various commenters. Chapter 2 presents the models which address the problem of modeling review comments.

1.2.2 Fine grained opinion mining using knowledge induction

As mentioned previously, most existing research in the task of aspect extraction largely fall into two main types. The first type only extracts aspect terms without grouping them into categories [Hu and Liu, 2004]. The second type uses statistical topic models to extract aspects and group them at the same time in an unsupervised manner [Zhao et al., 2010]. While unsupervised probabilistic topic

[1] Note that we have no expressions for answers to questions as there are usually no specific phrases indicating that a post answers a question except starting with the name of the person who asked the question. However, there are typical phrases for acknowledging answers, thus answer acknowledgement expressions.

models for opinion mining (e.g., [Lin and He, 2009; Brody and Elhadad, 2010; Jo and Oh, 2011]) do have some edge over other methods (e.g., they can discover aspects and group them into semantic categories at the same time), there is one major drawback that these models suffer from. It is often the case that many discovered aspects are not understandable/meaningful to users. Chang et al., [Chang et al., 2009] stated that one possible reason for the phenomenon as the fact that the objective function of topic models does not always correlate well with human judgments. This renders a major room for improvement. Interestingly, in practical applications, it is often the case that the end users (who know their trade well enough) are capable of providing some "seeds" in relation to what their expectations are (from the output of the model). This puts forth the following research challenge: Using a handful of user provided seeds, how to generate more meaningful clustering of aspects than existing aspect and sentiment based topic models. Chapter 3 details with this problem and proposes two novel seed based aspect extraction models.

1.2.3 Detecting Deception in social media: Opinion Spam

With the usefulness of social media comes, it also brings about a curse – opinion spam! These days before buying any product, almost everyone reads reviews to make a decision. Positive opinions can result in significant financial gains and fames for organizations. This, unfortunately, gives strong incentives for opinion spamming, which refers to human activities (e.g., writing fake reviews, giving unfair ratings) that try to deliberately mislead readers by giving unfair reviews to some entities (e.g. products, hotels) in order to promote them or to damage their reputations. As more and more individuals and organizations are using reviews for their decision making, detecting such fake reviews becomes a pressing issue. The problem has also been widely reported in the news[2].

Opinion spam comes in various flavors, e.g., fake reviews, fake comments, fake blogs, fake social network postings, and deceptive messages. As all these can be encompassed into opinions, we refer to

[2] http://www.nytimes.com/2012/01/27/technology/for-2-a-star-a-retailer-gets-5-star-reviews.html

the general problem as opinion spam. Prior research in [Jindal and Liu, 2008] showed that manually spotting such spam opinions is very hard if not impossible. Hence, there have been attempts (e.g., [Jindal and Liu, 2008; Ott et al., 2011]) which try to deal with this problem using computational linguistics and machine learning classification algorithms.

There have been some priors works which aimed at analyzing *rating behaviors* [Lim et al., 2010] and *unusual reviewing patterns* using unexpected class association rules [Jindal and Liu, 2010]. However, these methods are good only for detecting the blatant cases of opinion spam (e.g., posting duplicate reviews, posting large number of reviews on a single day, etc.). A never ending battle where both spammers and combaters try their best in achieving their end means.

A novel angle in opinion spam is the detection of spammer groups. Group spamming is a crucial dimension of the problem where opinion spam soars to group scale involving group fake reviewer collusion. We call this *group opinion spam*. Group based deception/spamming refers to a group of reviewers (spammer group) writing fake reviews/deceptive opinions together to promote or to demote some target entities (e.g., products, hotels, etc.). Such groups can be highly damaging as it can take total control of the sentiment on a product because a group has many people to write fake reviews. The situation become worse than detecting individual deceptive opinion spam because a group has more manpower to post reviews and can enshroud the spam activities of its individual members, whereby each member may no longer appear to be behaving abnormally. Chapter 4 addresses the problem of group opinion spam by jointly modeling groups, members, and products in a mutual reinforcement ranking framework.

An important scenario in large-scale machine learning for anomaly detection is the *absence of ground truth or positive labeled data*. The issue intensifies in opinion spam because it is hard and erroneous to manually label fake reviews [Jindal and Liu, 2008; Ott et al., 2011]. This situation also arises in fields like *epidemiology* (accurate diagnosis of infected population), *credibility analysis*

(positive samples of rumors), *cyber-security* (positive cases of phishing, sockpuppets, email fraud, etc.). How to build large scale machine learning models in such real-world settings?

Distributional Divergence to the Rescue: Our models in [Mukherjee et al., 2012; Mukherjee et al., 2011] showed that spammers differ markedly from others on several behavioral dimensions. This creates a distributional divergence between the latent populations of two clusters: spammers and non-spammers. The behaviors can be exploited as the "footprints" left behind by spammers. Working in the fully Bayesian setting and modeling spamicity of users as "latent" with observed behavioral footprints, we developed generative models for deceptive opinion spam/fraud detection [Mukherjee et al., 2013b]. Experiments using real-world data from Amazon.com demonstrated the effectiveness of our models which outperformed several state-of-the-art competitors. To our knowledge, this is the *first* work to employ principled statistical modeling for detecting opinion spam without any supervision. Chapter 5 presents the details of models and algorithms.

1.3 Summary of Contributions

The major contributions of this thesis span across two research threads: (1) opinion mining, and (2) opinion spam detection.

In the thread of opinion mining, joint topic and sentiment models are proposed for modeling social conversations in debates and consumer review comments [Mukherjee and Liu, 2012a; Mukherjee and Liu, 2012c]. Specifically, these models are capable of mining fine-grained contentions in discussion/debate forums and consumer review comments. The models can also discover various dimensions of sentiment expressions (agreement, disagreement, thumbs up, thumbs down, question, answer acknowledgement, etc.). In the domain of debates, such indicator AD-expressions (Agreement and Disagreement expressions), and contention points or topics are jointly mined are both at the discussion collection level and also at each individual post level. To the best of our knowledge, limited work has been done on such detailed analysis. Additionally, the proposed models not only

model both contention/agreement expressions and discussion topics, but also, more importantly, model the intrinsic nature of discussions/debates, i.e., interactions among discussants or debaters and topic sharing among posts through quoting and replying relations. Thus, it proposes probabilistic topic and sentiment models which take into consideration the social behaviors of users. These are detailed in chapter 2.

In chapter 3, new knowledge based models are proposed which can address the issues of semantic incoherence in aspect extraction using a set of seed terms. Aspect extraction is a central problem in sentiment analysis. Most methods either extract aspects without categorizing them, or extract and categorize them using unsupervised topic modeling. By categorizing, we mean the synonymous aspects should be clustered into the same category. We solve the problem in a different setting where the user provides some seed words for a few aspect categories and the model extracts and clusters aspect terms into categories simultaneously. This setting is important because categorizing aspects is a subjective task. For different application purposes, different categorizations may be needed. Some form of user guidance is desired. We propose two statistical models to solve this seeded problem, which aim to discover aspects respecting user's semantic clustering notions provided by seeds. The proposed models also discover aspect specific sentiments guided by aspect seeds and improves clustering of non-seeded aspects.

In the thread of opinion spam, this work focuses on proposing solutions strategies for two problems (1) group opinion spam and (2) unsupervised opinion spam detection in the Bayesian setting.

Prior works on opinion spam focused on detecting fake reviews and individual fake reviewers. However, a fake reviewer group (a group of reviewers who work collaboratively to write fake reviews) is even more damaging as they can take total control of the sentiment on the target product due to its size. Chapter 4 studies spam detection in the collaborative setting, i.e., to discover fake reviewer groups. The proposed method first uses a frequent itemset mining method to find a set of candidate groups. It then uses several behavioral models derived from the collusion phenomenon

among fake reviewers and relation models based on the relationships among groups, individual reviewers, and products they reviewed to detect fake reviewer groups. Additionally, it also builds a labeled dataset of fake reviewer groups. The proposed technique departs from the traditional supervised learning approach for spam detection because of the inherent nature of the problem which makes the classic supervised learning approach less effective.

Chapter 5 deals with detecting opinion spam in the absence of labeled data using distributional divergence. Particularly, it proposes to model *spamicity* as latent. An unsupervised model, called Author Spamicity Model (ASM), is proposed. It works in the Bayesian setting, which facilitates modeling spamicity of authors as latent and allows us to exploit various observed behavioral footprints of reviewers. The intuition is that opinion spammers have different behavioral distributions than non-spammers. This creates a distributional divergence between the latent population distributions of two clusters: spammers and non-spammers. Model inference results in learning the population distributions of the two clusters. Several extensions of ASM are also considered leveraging from different priors. Chapter 6 concludes this thesis.

CHAPTER 2

MINING OPINIONS IN SOCIAL CONVERSATIONS

In this chapter, we build statistical models for mining opinions in social conversations for two domains: (1) Ideological discussions and debates in Volconvo.com, and (2) consumer review comments in Amazon.com.

2.1 Mining Contentions in Debates

A large part of discussions in social media is about social, political and religious issues. On such issues, there are often heated discussions/debates, i.e., people argue and agree or disagree with one another. In this work, we model this form of interactive social media. Given a set of discussion/debate posts, we aim to perform the following tasks:

1. Discover expressions that people often use to express agreement (e.g., "I agree", "I think you're right") and disagreement (e.g., "I disagree", "you make no sense"). We collectively call them AD-expressions.
2. Determine contentious topics. First discover discussion topics in the whole collection, and then for each disagreeing post, discover the contention points (or topics).

Although there is a large body of literature on social media analysis such as social network analysis [Easley and Kleinberg, 2010], sentiment analysis [Pang and Lee, 2008], and finding of different camps of people and stance mining in discussions/debates [Agarwal et al. 2003; Thomas et al. 2006; Somasundaran and Wiebe, 2009; Murakami and Raymond, 2010], to the best of our knowledge, limited research has been done on the fine-grained analysis of discussion/debate forums as proposed in this paper. This problem is important because a large part of social media is about discussions/debates of

contentious issues, and discovering such issues is useful for many applications. For example, in a political election, contentious issues often separate voters into different camps and determine their political orientations. It is thus important for political candidates to know such issues. For contentious social topics, it is crucial for government agencies to be aware of them so that they can address the problems. Even for consumer products/services, contentions and disagreements in discussions about them can be used to identify different types of customers and to make effective marketing and business decisions.

We use statistical modeling to perform the aforementioned tasks. Three novel statistical models are proposed. The first model, called JTE (Joint Topic-Expression model), jointly models both discussion topics and AD-expressions. It is a semi-supervised probabilistic graphical model which provides a general framework for discovering discussion topics and AD-expressions simultaneously. However, this model does not consider a key characteristic of discussions/debates, i.e., authors quote or mention the claims/views of other authors and express contention or agreement on those claims/views. That is, there are interactions among authors and topics through the reply-to relation, which is a salient feature of discussion/debate forums. We then extend the JTE model and propose two novel and more advanced models JTE-R and JTE-P which model the interactions of authors and topics in two different ways, based on reply-to relations and author-pair structures respectively.

Works related to ours are quite different both in application and in modeling. On application, the closely related work to ours is the finding of different camps of people and stance mining in discussions/debates [Agarwal et al. 2003; Thomas et al. 2006; Somasundaran and Wiebe, 2009; Murakami and Raymond, 2010]. This thread of research, however, does not discover AD-expressions or contention points, which are the objectives of this work. From a modeling point of view, our work is related to topic modeling in general and joint modeling of topics and certain other information in particular. Topic models are a principled way of mining topics from large text collections. There have

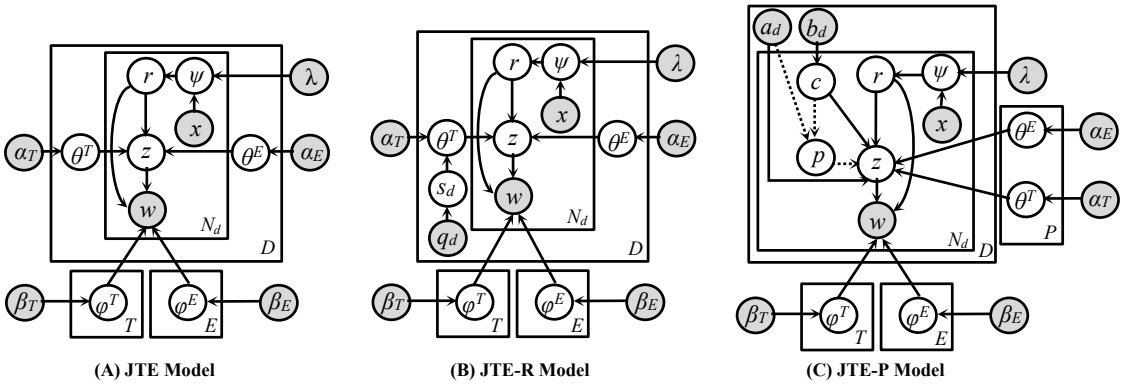


Figure 1: Plate notations of JTE variants (A) JTE, (B) JTE-R, (C) JTE-P models.

been many extensions [Blei and McAuliffe, 2007; Blei and Lafferty, 2009; Titov and McDonald, 2008] to the initial models, LDA (Latent Dirichlet Allocation) [Blei et al. 2003] and pLSA (Probabilistic latent semantic analysis) [Hofmann, 1999]. However, these models mine only topics, which are insufficient for our problem. In recent years, researchers have also proposed joint models of topics and sentiments [Jo and Oh, 2011; Lin and He, 2009; Zhao et al. 2010]. Our JTE model is related to these joint models. However, these models treat documents/posts independently, which fail to capture author and topic interactions in discussions/debates, i.e., authors reply to and quote each other's claims/views and express contentions or agreements. Due to such interactions, posts are clearly not independent of one another. The proposed JTE-R and JTE-P models capture such interactions. The next subsection presents the proposed JTE (Joint Topic Expression) model which lays the ground work for jointly modeling topics and AD-expressions.

2.1.1 Joint Topic Expression (JTE) Model

The JTE model belongs to the family of generative models for text where words and phrases (*n*-grams) are viewed as random variables, and a document is viewed as a bag of *n*-grams and each *n*-gram (word/phrase) takes one value from a predefined vocabulary. We use up to 4-grams, i.e., $n = 1, 2, 3, 4$, in this work. Note that topics in most topic models like LDA are usually unigram distributions over

words and assume words to be exchangeable at the word level. Arguably, this offers a great computational advantage over more complex models taking word order into account for discovering significant *n*-grams [Wallach, 2006]. Yet there exist other works which try to *post-process* discovered topical unigrams to form multi-word phrases using likelihood [Blei and Lafferty, 2009] scores. However, our goal in this work is to enhance the expressiveness of our JTE model (rather than modeling *n*-gram word order) by considering *n*-grams and preserving the advantages of exchangeable modeling. Thus, we consider both words and phrases as our vocabulary. For notational convenience, from now on we use *terms* to denote both *words* (unigrams) and *phrases* (*n*-grams). Since we are dealing with large corpora, for computational reasons, we only consider terms which appeared at least 30 times in the corpus[1].

We denote the entries in our vocabulary by $v_{1...V}$, where V is the number of unique terms in the vocabulary. The entire corpus (document collection) of study is comprised of $d_{1...D}$ documents. A document (e.g., discussion post) d is represented as a vector of terms $\boldsymbol{w_d}$ with N_d entries. W is the set of all observed terms in the corpus with cardinality, $|W| = \sum_d N_d$. The JTE model is motivated by the joint occurrence of AD-expression types (*agreement* and *disagreement*) and topics in discussion posts. A typical discussion/debate post mentions a few topics (using semantically related topical terms) and expresses some viewpoints with one or more AD-expression types (using semantically related agreement and/or disagreement expressions). This observation motivates the generative process of our model where documents (posts) are represented as random mixtures of latent topics and AD-expression types. Each topic or AD-expression type is characterized by a distribution over terms. Assume we have $t = 1,\ldots, T$ topics and $e = 1,\ldots, E$ expression types in our corpus. Note that in our case of discussion/debate forums, based on reading various posts, we hypothesize that $E = 2$ as in such forums,

[1] This is reasonable as our corpus contains about 100,000 documents. It is unlikely for a term with frequency less than 30 to show up as a top topical or AD-expression term.

we mostly find two expression types: agreement and disagreement (which we also statistically validate in Section 2.1.4.5). However, the proposed JTE and other models are general and can be used with any number of expression types. Let $\psi_{d,j}$ denote the probability of $w_{d,j}$ being a topical term with $r_{d,j} \in \{\hat{t}, \hat{e}\}$ denoting the binary indicator variable (topic or AD-expression) for the j^{th} term of d, $w_{d,j}$. $z_{d,j}$ denotes the appropriate topic or AD-expression type index to which $w_{d,j}$ belongs. We parameterize multinomials over topics using a matrix $\Theta^T_{D\times T}$ whose elements $\theta^T_{d,t}$ signify the probability of document d exhibiting topic t. For simplicity of notation, we will drop the latter subscript (t in this case) when convenient and use θ^T_d to stand for the d^{th} row of Θ^T. Similarly, we define multinomials over AD-expression types using a matrix $\Theta^E_{D\times E}$. The multinomials over terms associated with each topic are parameterized by a matrix $\Phi^T_{T\times V}$, whose elements $\varphi^T_{t,v}$ denote the probability of generating v from topic t. Likewise, multinomials over terms associated with each AD-expression type are parameterized by a matrix $\Phi^E_{E\times V}$. We now define the generative process of JTE (see Figure 1 (A) for plate notation).

1. For each AD-expression type e, draw $\varphi^E_e \sim Dir(\beta_E)$
2. For each topic t, draw $\varphi^T_t \sim Dir(\beta_T)$
3. For each forum discussion post $d \in \{1 \ldots D\}$:
   i. Draw $\theta^E_d \sim Dir(\alpha_E)$
   ii. Draw $\theta^T_d \sim Dir(\alpha_T)$
   iii. For each term $w_{d,j}$, $j \in \{1 \ldots N_d\}$:
      a. Set $\psi_{d,j} \leftarrow MaxEnt(\overrightarrow{x_{d,j}};\ \lambda)$
      b. Draw $r_{d,j} \sim Bernoulli(\psi_{d,j})$
      c. if $(r_{d,j} = \hat{e})$ // $w_{d,j}$ is a AD-expression term
         Draw $z_{d,j} \sim Mult(\theta^E_d)$
      else // $r_{d,j} = \hat{t}$, $w_{d,j}$ is a topical term
         Draw $z_{d,j} \sim Mult(\theta^T_d)$

d. Emit $w_{d,j} \sim Mult(\varphi_{z_{d,j}}^{r_{d,j}})$

We use Maximum Entropy (Max-Ent) model to set $\psi_{d,j}$. The Max-Ent parameters can be learned from a small number of labeled topical and AD-expression terms which can serve as good priors. The idea is motivated by the following observation: topical and AD-expression terms usually play different syntactic roles in a sentence. Topical terms (e.g. "U.S. senate", "sea level", "marriage", "income tax") tend to be noun and noun phrases while AD-expression terms ("I refute", "how can you say", "probably agree") usually contain pronouns, verbs, wh-determiners, and modals. In order to utilize the part-of-speech (POS) tag information, we place $\psi_{d,j}$ (the prior over the indicator variable $r_{d,j}$) in the word plate (see Figure 1 (A)) and draw it from a Max-Ent model conditioned on the observed feature vector $\overrightarrow{x_{d,j}}$ associated with $w_{d,j}$ and the learned Max-Ent parameters λ (see Section 2.4.2). $\overrightarrow{x_{d,j}}$ can encode arbitrary contextual features that may be discriminative. In this work, we encode both lexical and POS features of the previous, current and next POS tags/lexemes of the term $w_{d,j}$. Specifically, the feature vector, $\overrightarrow{x_{d,j}} = [POS_{w_{d,j}-1}, POS_{w_{d,j}}, POS_{w_{d,j}+1}, w_{d,j}-1, w_{d,j}, w_{d,j}+1]$. For phrasal terms (*n*-grams), all POS tags and lexemes of $w_{d,j}$ are considered as features. To learn the JTE model from data, exact inference is not possible. We thus resort to approximate inference using collapsed Gibbs sampling. We first derive the joint distribution below and then the Gibbs sampler.

To derive the joint distribution, we factor the joint according to the conditional distributions (causalities) governed by the Bayesian network of the proposed generative model.

$$P(W, Z, R) = P(W|Z, R) \times P(Z|R) \times P(R) \tag{2.1}$$

Since we employ a collapsed Gibbs sampler, we integrate out θ and φ and obtain the joint as follows.

$$P(W, Z, R) = \left[\prod_{t=1}^{T} \frac{B(n_t^{TV} + \beta_T)}{B(\beta_T)} \times \prod_{e=1}^{E} \frac{B(n_e^{EV} + \beta_E)}{B(\beta_E)}\right] \times$$

$$\left[\prod_{d=1}^{D} \left(\frac{B(n_d^{DT} + \alpha_T)}{B(\alpha_T)} \times \frac{B(n_d^{DE} + \alpha_E)}{B(\alpha_E)}\right)\right] \times \left[\prod_{d=1}^{D} \prod_{j=1}^{N_d} p(r_{d,j}|\psi_{d,j})\right] \tag{2.2}$$

where $p(r_{d,j}|\psi_{d,j}) = (\psi_{d,j,\hat{t}})^u(\psi_{d,j,\hat{e}})^{1-u}$; $u = \begin{cases} 1, r_{d,j} = \hat{t} \\ 0, r_{d,j} = \hat{e} \end{cases}$ and the outcome probabilities of the Max-Ent model are given by: $\psi_{d,j,\hat{t}} = p(y = \hat{t}|x_{d,j})$; $\psi_{d,j,\hat{e}} = p(y = \hat{e}|x_{d,j})$; $p(y|x_{d,j}) = \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))}{\sum_{y \in \{\hat{t}, \hat{e}\}} \exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))}$. $\lambda_{1...n}$ are the parameters of the learned Max-Ent model corresponding to the n binary feature functions $f_{1...n}$ from Max-Ent. $n_{t,v}^{TV}$ and $n_{e,v}^{EV}$ denote the number of times term v was assigned to topic t and expression type e respectively. $B(\cdot)$ is the multinomial Beta function $B(\vec{x}) = \frac{\prod_{i=1}^{\dim(\vec{x})} \Gamma(x_i)}{\Gamma(\sum_{i=1}^{\dim(\vec{x})} x_i)}$. $n_{d,t}^{DT}$ and $n_{d,e}^{DE}$ denote the number of terms in document d that were assigned to topic t and AD-expression type e respectively. n_t^{TV}, n_e^{EV}, n_d^{DT}, and n_d^{DE} denote the corresponding row vectors.

We use Gibbs sampling for posterior inference. Gibbs sampling is a form of Markov Chain Monte Carlo (MCMC) method where a markov chain is constructed to have a particular stationary distribution. In our case, we want to construct a markov chain which converges to the posterior distribution over R and Z conditioned on the observed data. We only need to sample z and r as we use collapsed Gibbs sampling and the dependencies of θ and φ have already been integrated out analytically in the joint. Denoting the random variables $\{w, z, r\}$ by singular subscripts $\{w_k, z_k, r_k\}$, $k_{1...K}$, $K = \sum_d N_d$, a single iteration consists of performing the following sampling:

$$p(z_k = t, r_k = \hat{t}|Z_{\neg k}, W_{\neg k}, R_{\neg k}, w_k = v) \propto \frac{n_{d,t\ \neg k}^{DT} + \alpha_T}{n_{d,(\cdot)\ \neg k}^{DT} + T\alpha_T} \times \frac{n_{t,v\ \neg k}^{TV} + \beta_T}{n_{t,(\cdot)\ \neg k}^{TV} + V\beta_T} \times \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, \hat{t}))}{\sum_{y \in \{\hat{t}, \hat{e}\}} \exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))} \quad (2.3)$$

$$p(z_k = e, r_k = \hat{e}|Z_{\neg k}, W_{\neg k}, R_{\neg k}, w_k = v) \propto \frac{n_{d,e\ \neg k}^{DE} + \alpha_E}{n_{d,(\cdot)\ \neg k}^{DE} + E\alpha_E} \times \frac{n_{e,v\ \neg k}^{EV} + \beta_E}{n_{e,(\cdot)\ \neg k}^{EV} + V\beta_E} \times \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, \hat{e}))}{\sum_{y \in \{\hat{t}, \hat{e}\}} \exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))} \quad (2.4)$$

where $k = (d, j)$ denotes the j^{th} term of document d and the subscript $\neg k$ denotes assignments excluding the term at k. Omission of a latter index denoted by $(\cdot)$ represents the marginalized sum over the latter index. The conditional probabilities in equations (2.3) and (2.4) were derived by applying the

chain rule on the joint distribution. We employ a blocked sampler where we sample r and z jointly, as this improves convergence and reduces autocorrelation of the Gibbs sampler. Here we only provide the final Gibbs update equations. For detailed derivation, please refer to the supplemental materials available here[2].

2.1.2 JTE-R: Encoding reply relations

To improve modeling we first augment JTE by encoding the reply relations. Reply relations are an important characteristic of debate and discussion forums where authors often reply to each other's viewpoints by explicitly mentioning the user name using @name, and/or by quoting others' posts. For easy presentation, we refer both cases by *quoting* from now on. Considering the reply-to relation, we call the new model JTE-R (Figure 1 (B)). This model is based on the following observation:

Observation: Whenever a post d replies to the viewpoints in some other posts by quoting them, d and the posts quoted by d should have similar topic distributions.

This observation indicates that the JTE-R model needs to depart from typical topic models where there is usually no topical interaction among documents, i.e., documents are treated as being independent of one another. Let q_d be the set of posts quoted by post d. Clearly, q_d is observed[3]. In order to encode this "reply-to" relation into our model, the key challenge is to somehow constrain the topic distribution of d, θ_d^T to be similar to the topic distributions of posts in q_d. Specifically, it is how to constrain θ_d^T to be similar to $\theta_{\acute{d}}^T$, where $\acute{d} \in q_d$ (i.e., constraining topic assignments to documents) during inference while the topic distributions of both θ_d^T and $\theta_{\acute{d}}^T$, $\acute{d} \in q_d$ are latent and unknown *apriori*. To solve our problem, we propose a novel solution, which exploits the following salient features of the Dirichlet distribution:

[2] http://www.cs.uic.edu/~amukherj/JTE.pdf
[3] We crawled the ids of posts quoted by each post.

1. Since $\theta_d^T \sim Dir(\alpha_T)$, we have $\sum_t \theta_{d,t}^T = 1$. Thus, it suffices that θ_d^T can act as a base measure for Dirichlet distributions of the same order.

2. Also, the expected probability mass associated with each dimension of the Dirichlet distribution is proportional to the corresponding component of its base measure[4].

Thus, to constrain a post d's topic distribution to be similar to the posts whom it replies/quotes (i.e., posts in q_d), we now need functional base measures as it is the base measure that governs the expected mass associated with each topical dimension in θ_d^T. One way to employ functional base measures is to draw $\theta_d^T \sim Dir(\alpha_T s_d)$, where $s_d = \sum_{d' \in q_d} \theta_{d'}^T / |q_d|$ (the expected topical distribution of posts in q_d). For posts which do not quote any other post, we simply draw $\theta_d^T \sim Dir(\alpha_T)$. For such a topic model with functional Dirichlet base measures, the sampling distribution is more complicated. Specifically, the document-topic distribution, θ_d^T is no longer a simple predictive distribution, i.e., when sampling z_n^d, the implication of each quoted document related to d by reply-to relations and their topic assignments must be considered because the sampling distribution for z_n^d in document d must consider its effect on the joint probability of the entire model. Unfortunately, this can be computationally expensive for large corpora. To circumvent this issue, we approximate the true Gibbs sampling distribution by updating the original smoothing parameter (α_T) to reflect the expected topic distributions of quoted documents $(s_{d,t}\alpha_T)$, where $s_{d,t}$ is the t^{th} component of the base measure s_d which is computed at runtime during sampling. Experimental results show that this approximation performs well empirically. The approximate Gibbs conditional distribution for JTE-R while sampling $z_n^d = t$ is given by:

$$p(z_k = t, r_k = \hat{t} \big| Z_{\neg k}, W_{\neg k}, R_{\neg k}, w_k = v) \propto \frac{n_{d,t\ \neg k}^{DT} + s_{d,t}\alpha_T}{\sum_{t=1}^{T}\left(n_{d,t\ \neg k}^{DT} + s_{d,t}\alpha_T\right)} \times \frac{n_{t,v\ \neg k}^{TV} + \beta_T}{n_{t,(\cdot)\ \neg k}^{TV} + V\beta_T} \times \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, \hat{t}))}{\sum_{y \in \{\hat{t}, \hat{e}\}} \exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))} \quad (2.5)$$

[4] Taking moments on $(X_1 \ldots X_n) \sim Dir(\alpha_1 \ldots \alpha_n)$, we get $E[X_i] = \frac{\alpha_i}{\Sigma \alpha_i}$. Thus, $E[X_i] \propto \alpha_i$

2.1.3 JTE-P: Encoding pair interactions

JTE-R builds over JTE by encoding reply-to relations to constrain a post to have similar topic distributions to those it quotes. An alternative strategy is to make θ^T and θ^E author-pair specific. The idea is motivated by the following observation.

Observation: When authors reply to others' viewpoints (by @name or quoting other authors' posts), they typically direct their own topical viewpoints with agreeing or disagreeing expressions to those authors. Such exchanges can go back and forth between pairs of authors. The discussion topics and AD-expressions emitted are thus caused by the author-pairs' shared topical interests and their nature of interactions.

Let a_d be the author of a post d, and $b_d = [b_{1...n}]$ be the list of *target authors* (we will also call them *targets* for short) to whom a_d replies to or quotes in d. The pairs of the form $p = (a_d, c), c \in b_d$ essentially shape both the topics and AD-expressions emitted in d as agreement or disagreement on topical viewpoints are almost always directed towards certain target authors. For example, if c claims something, a_d quotes the claim in his post d and then contends/agrees by emitting AD-expressions like "you have no clue", "yes, I agree", "I don't think," etc. Clearly, this pair structure is a crucial feature of discussions/debate forums. Each pair has its unique and shared topical interests and interaction nature (agreement or disagreement). Thus, it is appropriate to condition θ^T and θ^E over author-pairs. We will see in Section 2.1.4.5 that this model fits the discussion data better. Standard topic models do not consider this key piece of information.

We extend the JTE model to incorporate the pair structure. We call the new model JTE-P, which conditions the multinomial distributions over topics and AD-expression types (θ^T, θ^E) on authors and targets as pairs rather than on documents as in JTE and JTE-R. In its generative process, for each post, the author a_d and the set of targets b_d are observed. To generate each term $w_{d,j}$, a target, $c \sim Uni(b_d)$, is chosen at uniform from b_d forming a pair $p = (a_d, c)$. Then, depending on the switch variable $r_{d,j}$,

a topic or an expression type index z is chosen from a multinomial over topic distribution θ_p^T or AD-expression type distribution θ_p^E, where the subscript p denotes the fact that the distributions are specific to the author-target pair p which shape topics and AD-expressions. Finally, the term is emitted by sampling from topic or AD-expression specific multinomial distribution $\varphi_{z_{d,j}}^{r_{d,j}}$.

The graphical model in plate notation corresponding to the above process is shown in Figure 1 (C). Clearly, in JTE-P, the discovery of topics and AD-expressions are guided by the pair structure of reply-to relations in which the collection of posts was generated. For posterior inference, we again use Gibbs sampling. Note that as a_d is observed, sampling *c* is equivalent to sampling the pair $p = (a_d, c)$. Its Gibbs sampler is given by:

$$p(z_k = t, p_k = p, r_k = \hat{t} | \dots, w_k = v) \propto \frac{1}{|b_d|} \times \frac{n_{p,t}^{PT}{}_{\neg k} + \alpha_T}{n_{p,(\cdot)}^{PT}{}_{\neg k} + T\alpha_T} \times \frac{n_{t,v}^{TV}{}_{\neg k} + \beta_T}{n_{t,(\cdot)}^{TV}{}_{\neg k} + V\beta_T} \times \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, \hat{t}))}{\sum_{y \in \{\hat{t}, \hat{e}\}} \exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))} \quad (2.6)$$

$$p(z_k = e, p_k = p, r_k = \hat{e} | \dots, w_k = v) \propto \frac{1}{|b_d|} \times \frac{n_{p,e}^{PE}{}_{\neg k} + \alpha_E}{n_{p,(\cdot)}^{PE}{}_{\neg k} + E\alpha_E} \times \frac{n_{e,v}^{EV}{}_{\neg k} + \beta_E}{n_{e,(\cdot)}^{EV}{}_{\neg k} + V\beta_E} \times \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, \hat{e}))}{\sum_{y \in \{\hat{t}, \hat{e}\}} \exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))} \quad (2.7)$$

where $n_{p,t}^{PT}$ and $n_{p,e}^{PE}$ denote the number of times the pair p was assigned to topic t and expression type e respectively. As JTE-P assumes that each pair has a specific topic and expression distribution, we see that equations (2.6, 2.7) condition topics and expression types assignments over pairs. It is also worthwhile to note that given A authors, there are $\binom{A}{2}$ possible pairs. However, the actual number of pairs (i.e., where the authors have communicated at least once) is much less than $\binom{A}{2}$. Our experimental data consists of 1824 authors and 7684 actual pairs. Hence we are only modeling 7684 pairs instead of $\binom{1824}{2} \approx 4$ million pairs.

2.1.4 Experimental evaluation

We now evaluate the proposed models and compare with baselines. We first qualitatively show the AD-expressions and topics discovered by the models. We then evaluate the models quantitatively based on the two objectives of this work:

1. Discovering agreement and disagreement expressions (or AD-expressions).
2. Finding contention points or topics in each contentious post.

Before proceeding further, we detail our dataset used for empirical evaluation.

(a) JTE

t_1: Spirituality	t_2: Burka/Veil	t_3: Homo-sexuality	t_4: Evolution of Life	t_5: 9/11 Attacks	t_6:Theism/Atheism	t_7: Global warming	t_8: Vegetarianism	t_9: IRS/Taxes	t_{10}: U.S. Politics
spirituality	burka	marriage	evolution	9/11	god	earth	animals	tax	vote
life	burqa	gay	species	september 11	belief	co2	meat	government	president
soul	suffering	couples	theory	terrorism	**existence**	warming	egg	irs	democrats
wrong	hijab	straight	dna	**fox**	atheist	weather	beef	state	politics
self	women	homosexuality	humans	news	**evidence**	pollution	slaughter	money	electoral
death	islam	trans	homo sapiens	terror attacks	faith	global warming	kill	pay	us government
karma	conceals	individual right	darwin's theory	plane	irrational	floods	life	federal	obama
suffering	**death**	gay marriages	life	cia	jesus	ice	diet	state taxes	**policy**
afterlife	tyranny	heterosexual	intelligence	conspiracy theory	supreme being	nuclear waste	meat industry	services	elections
self realization mortal	terrorism	legal	mendel's theory	al qaeda	creationism	sea level	vegan	social security	senate
self knowledge	muslim bigots	**law**	human dna	bin laden	big bang	climate change	vegetables	income tax	libertarian
	sexist	sex	theory of evolution	bush	omnipotent	arctic ice	meat consumption	budget	left wing

(b) LDA

t_1: Spirituality	t_2: Burka/Veil	t_3: Homo-sexuality	t_4: Evolution of Life	t_5: 9/11 Attacks	t_6:Theism/Atheism	t_7: Global warming	t_8: Vegetarianism	t_9: IRS/Taxes	t_{10}: U.S. Politics
life	burka	gay	life	9/11	god	earth	kill	tax	vote
evil	**man**	couples	**god**	laden	religion	planet	meat	pay	electoral
live	women	straight	evolution	bush	jesus	ice	animal	agent	obama
knowledge	immoral	sex	**religion**	terror	**desire**	weather	cow	income	house
purpose	muslim	dating	intelligence	dead	**bang**	warming	egg	revenue	political
values	islam	family	human	**twin**	faith	floods	cooking	us	american
natural	sadistic	**funny**	beings	plane	life	level	**earth**	irs	democrats
existence	terrorism	trans	theory	obl	man	change	diet	american	party
goal	bigot	love	sea	**new**	adam	**clean**	milk	**draft**	bill
self	sexist	children	biology	crash	exists	polar	**fur**	fund	bush
sex	**beautiful**	**law**	earth	tower	**being**	climate	**herbivorous**	funding	senate
spirit	conceals	**ancient**	dna	**york**	omnipotent	**waste**	vegan	state	presidential

(c) SLDA

t_1: Spirituality	t_2: Burka/Veil	t_3: Homo-sexuality	t_4: Evolution of Life	t_5: 9/11 Attacks	t_6:Theism/Atheism	t_7: Global warming	t_8: Vegetarianism	t_9: IRS/Taxes	t_{10}: U.S. Politics
life	burka	gay	life	9/11	god	planet	kill	tax	vote
evil	muslim	couples	theory	laden	religion	ice	meat	pay	electoral
spirit	hijab	dating	evolution	plane	theist	earth	chicken	income	senate
knowledge	women	sex	**homo**	*nonsense*	life	level	egg	*valid*	house
values	*incorrect*	**cannot**	earth	terror	islam	*agree*	beef	state	*correct*
agree	bigot	family	human	bush	**cannot**	change	**fur**	irs	american
existence	sexist	*disagree*	**argument**	*incorrect*	belief	floods	diet	*agree*	democrats
live	terrorism	**don't**	sea	crash	**argument**	**point**	**claim**	**think**	*definitely*
correct	burqa	trans	biology	obl	adam	arctic	milk	us	bush
don't	**man**	children	*prove*	*bogus*	creationism	polar	*disagree*	revenue	elections
goal	*nonsense*	**funny**	dna	cia	**your**	*indeed*	**strength**	budget	obama
purpose	**beautiful**	**law**	darwin	tower	jesus	**clean**	cow	**draft**	*disagree*

Table I: Top terms of 10 topics discovered by (a) JTE, (b) LDA, and (c) SLDA. **Red (bold)** denotes errors and *blue (italics)* denotes contention/agreement terms.

2.1.4.1 Dataset

For our experiments, we used debate/discussion posts from Volconvo[5]. The forum is divided into various domains: Politics, Religion, Society, Science, etc. Each domain consists of multiple threads. Each thread has a set of posts. For each post, we extracted the post id, author, time, domain, ids of all posts to which it replies/quotes, and the post content. In all, we extracted 26137, 34986, 22354 and 16525 posts from Politics, Religion, Society and Science domains respectively. Our final data consists of 5978357 tokens, 1824 authors with an average of 346 words per post, and 7684 author-target pairs.

2.1.4.2 Topic and AD-expression discovery

To set the background for our quantitative evaluation of the two tasks in the next two subsections, we first show the topics and the AD-expressions discovered by our models, and also compare them with topics found by LDA [Blei et al. 2003] and its variant SLDA (sentence-LDA) [Jo and Oh, 2011]. We choose LDA as it is the best-known topic model. We use SLDA as it constrains words in a sentence to be generated from a single topic. Since AD-expressions may appear with topics in the same sentence, SLDA can serve as a baseline although SLDA is unable to separate topical terms and AD-expressions.

For all our experiments here and the subsequent ones, the hyper-parameters for LDA and SLDA were set to the heuristic values $\alpha = 50/T$, $\beta = 0.1$ as suggested in [Griffiths and Steyvers, 2004]. Similarly, for the proposed models, we set $\alpha_T = 50/T$, $\alpha_E = 50/E$, $\beta_T = \beta_E = 0.1$. To learn the Max-Ent parameters λ, we randomly sampled 500 terms from our corpus appearing at least 50 times[6] and labeled them as topical (372) or AD-expressions (128) and used the corresponding feature vector of each term (in the context of posts where it occurs) to train the Max-Ent model. We set the number of topics, T = 100 and the number of AD-expression types, E = 2 (agreement and disagreement) as in

[5] http://www.volconvo.com/forums/forum.php

[6] A minimum frequency count of 50 ensures that the training data is reasonably representative of the corpus.

(a) JTE Disagreement expressions, $\Phi^{E}_{Disagreement}$
I, disagree, **don't**, I don't, **claim**, **you**, oppose, **debate**, I disagree, **argument**, reject, I reject, I refute, **your**, I refuse, doubt, nonsense, *I contest*, dispute, **I think**, completely disagree, don't accept, don't agree, *your claim isn't*, incorrect, *hogwash*, ridiculous, *I would disagree*, false, *I don't buy your*, **really**, *I really doubt*, your nonsense, **true**, *can you prove*, argument fails, *you fail to*, **sense**, your assertions, *bullshit*, *sheer nonsense*, **cannot**, *doesn't make sense*, why do you, *you have no clue*, *how can you say*, *do you even*, absolute nonsense, *contradict yourself*, absolutely not, *you don't understand*, …
(b) JTE Agreement expressions, $\Phi^{E}_{Agreement}$
agree, **I**, correct, yes, true, **do**, accept, **you**, I agree, right, **indeed**, indeed correct, I accept, **claim**, **your**, **point**, are right, **don't**, valid, *I concede*, *is valid*, **your claim**, *you are right*, **not really**, *would agree*, **might**, *agree completely*, **very**, yes indeed, **mean**, you're correct, **completely**, *valid point*, **argument**, proves, *do accept*, support, **said**, agree with you, *I do support*, *rightly said*, **personally**, absolutely, *completely agree*, well put, *very true*, *well said*, *personally agree*, **doesn't necessarily**, exactly, *very well put*, absolutely correct, **probably**, *kudos*, acknowledge, *point taken, partially agree*, agree entirely, ...

Table II: Top terms (comma delimited) of two expression types estimated by JTE model. **Red (bold)** terms denote possible errors. *Blue (italics)* terms are newly discovered; rest (black) were used in Max-Ent training.

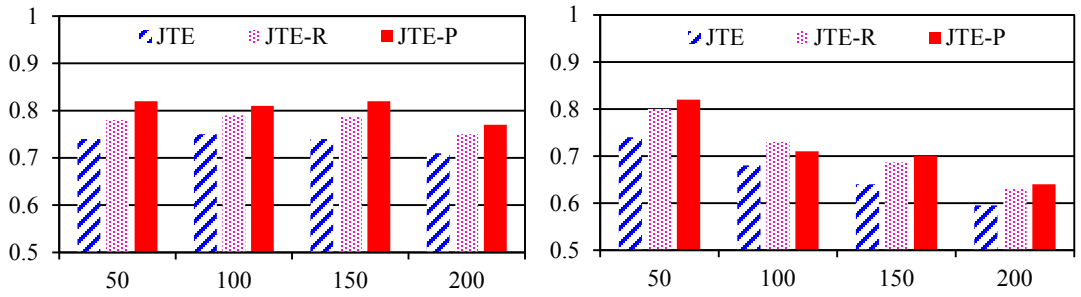


Figure 2: Precision @ top 50, 100, 150, 200 rank positions for $\Phi^{E}_{Disagreement}$ (left) and $\Phi^{E}_{Agreement}$ (right) expressions discovered by various methods.

discussion/debate forums, there are usually two expression types (This hypothesis is statistically supported by perplexity experiments in Section 2.1.4.5).

As JTE is the basic model (others build over it) and is also closest to LDA and SLDA, we compare the top terms for 10 topics discovered by JTE, LDA and SLDA in Table I. The top topical terms by other models are not so different. However, we will evaluate all the proposed models quantitatively

later using the task of identifying topics (or "points") of contention in each contentious post, which is one of our objectives. From Table I, we can observe that JTE is quite effective at discovering topics. Its topical terms are more specific and contain fewer semantic clustering errors (marked red in bold) than LDA and SLDA. For example, owing to the generative process of JTE, it is able to cluster phrases like "homo sapiens", "darwin's theory," and "theory of evolution" in t_4 (Evolution of Life), which makes the topic more specific. It is important to note that both LDA and SLDA cannot separate topics and AD-expressions because they find only topics. That is why we need joint modeling of topics and AD-expressions. We can see that the topics of SLDA do contain some AD-expressions (marked blue in italics) because SLDA constrains all words in a sentence to be generated from a single topic. Since AD-expressions can co-occur with topical words in a sentence, they are clustered with topics, which is undesirable. Our proposed models solve this problem based on a topic/AD-expression switch formulation.

Next we look at the discovered AD-expressions. We first list some top AD-expressions found by JTE in Table II for qualitative inspection. Since AD-expressions found by JTE-R and JTE-P were quite similar to those of JTE among the top 30 terms, they are omitted here. However, all three models are quantitatively evaluated in the next subsection. From Table II, we see that JTE can discover and cluster many correct AD-expressions, e.g., "I disagree," "I refute" and "completely disagree" in disagreement; and "I accept," "I agree," and "you're correct" in agreement. It additionally discovers more distinctive expressions beyond those observed in the training data of Max-Ent. For example, we find phrases like "I don't buy your", "I really doubt", "can you prove", "you fail to", and "you have no clue" being clustered in disagreement and phrases like "valid point", "rightly said", "I do support", and "very well put" clustered in agreement. These newly discovered phrases are marked *blue* (in italics) in Table II.

Lastly, we note that AD-expressions of JTE do contain some errors marked **red** (in bold). However, this is a common issue with all unsupervised topic models for text as the objective function of topic

models does not always correlate well with human judgments [Chang et al. 2009]. In our case, the issue is mainly due to unigram AD-expressions like "I", "your", "do", etc., which by itself do not signify agreement or disagreement but show up due to higher frequencies in the corpus. There are also phrase errors like "doesn't necessarily", "not really", etc. A plausible approach to deal with this is to discover significant *n*-grams based on multi-way contingency tables and statistical tests along with linguistic clues to pre-process and filter such terms. These issues are worth investigating and we defer them to our future work.

2.1.4.3 AD-expression evaluation

We now quantitatively evaluate the discovered AD-expressions by all three proposed models in two additional ways. We first evaluate them directly using rank precision and then evaluate them indirectly through a classification task.

2.1.4.3.1 Evaluating AD-expression ranking

Since AD-expressions (according to top terms in Φ^E) produced by JTE, JTE-R, and JTE-P are rankings, we evaluate them using *precision @ n* (*p@n*), which gives the precision at different rank positions. This measure is commonly used to evaluate a ranking when the number of correct items is unknown, which is our case. For computing *p@n*, we also investigated multi-rater agreement. Three judges independently labeled the top *n* terms as correct or incorrect for the estimated language models $\Phi^E_{Agreement}$ and $\Phi^E_{Disagreement}$. Then, we marked a term to be correct if all the judges deemed it so which was then used to compute *p@n*. Multi-rater agreement using Fleiss kappa was greater than 0.80 for all *p@n*, which imply perfect agreement [Landis and Koch, 1977]. This is understandable because one can almost certainly make out whether a term expresses agreement, disagreement, or none.

Figure 2 shows the precisions of estimated agreement and disagreement expressions for the top 50, 100, 150, 200 rank positions (i.e., *p@* 50, 100, 150, 200) in the two rankings $\Phi^E_{Agreement}$ and

Features	SVM	NB	LR
W+POS 1-gram	69.37	68.20	68.84
W+POS 1-2 gram	70.33	68.94	69.90
W+POS, 1-3 gram	70.86	69.16	70.44
W+POS, 1-4 gram	70.97	69.26	70.54
W+POS, 1-4 gram + IG	75.67	74.01	75.34
W+POS, 1-4 gram + χ^2	76.21	75.11	76.09
AD-Expr. Φ^E, JTE	80.79	78.55	79.30
AD-Expr. Φ^E, JTE-R	82.18	79.19	80.15
AD-Expr. Φ^E, M-JTE-P	83.88	79.30	81.43

Table III: Accuracies of post classification. The improvements of our models are significant (p<0.001) over two tailed t-test across 10-fold cross validation.

$\Phi^E_{Disagreement}$. We observe that both JTE-R and JTE-P are much better than JTE. JTE-P produces the best results. We believe the reason is that JTE-P's expression models being pair specific (θ^E_p) can capture AD-expressions better as agreement/disagreement expressed by an author is almost always directed to some other authors forming author-pairs.

2.1.4.3.2 Post Classification

We now use the task of classifying a post as being contentious or agreeing to evaluate the discovered AD-expressions. This classification task is also interesting in its own right. However, we should note that our purpose here is not to build the best classifier to classify posts but to indirectly show that the discovered AD-expressions are of high quality as they help to perform the classification better than the standard word *n*-grams and part-of-speech (POS) *n*-gram features for text classification.

To perform this experiment, we randomly sampled 1000 posts from our database and asked our human judges (3 graduate students well versed in English) to classify each of those posts as *agreeing*, *disagreeing*, or *other*. Judges were made to work in isolation to prevent any bias. We then labeled a post as agreeing or disagreeing if all judges deemed it so. In this way, 532 posts were classified as

disagreeing, 405 as agreeing. We inspected the rest 63 posts which had disagreements. We found that 18 of them were the first posts of threads. We removed them as the first posts of threads usually start the discussions and do not express agreement or disagreement. For the remaining 45 of them, 13 posts were partly disagreeing and partly agreeing, and the rest were mostly statements of views without agreement/disagreement. Since the number of these posts is small (only 45), we did not use them in classification. That is, we considered two mutually exclusive classes (agreeing vs. disagreeing) for post classification.

For supervised learning, a challenging issue is the choice of features. Word and POS *n*-grams are traditional features. We now compare such features with AD-expressions discovered by the proposed models. We used the top 1000 terms from the AD-expression rankings as features. Using classification learning algorithms, we compare a learner trained on all word and POS *n*-grams with those trained on AD-expressions induced by our models. We used SVM (linear kernel), Naïve Bayes (NB), and Logistic Regression (LR). Table III reports the accuracy results. Accuracy is appropriate metric here as the two classes are not skewed and we are interested in both classes. All results were obtained through 10-fold cross-validation. As the major advantage of AD-expressions arise from dimensionality reduction and feature selection, we also compared with two popular feature selection schemes: Information Gain (IG) and Chi-Square test (χ^2). We can observe that SVM performed the best among all learners. The accuracy dramatically increases with AD-expression (Φ^E) features. JTE, JTE-R, and JTE-P progressively improve the accuracy beyond those obtained by traditional *n*-gram features. JTE-P performed the best. Feature selection schemes also improved performance but the proposed models outperform feature selection schemes as well. All accuracy improvements are significant (p<0.001) using a two tailed *t*-test over 10-fold CV. This clearly shows that AD-expressions are of high quality and effective for downstream applications such as post classification.

D	Φ^E + Noun/Noun Phrase						JTE						JTE-R						JTE-P					
	J_1			J_2			J_1			J_2			J_1			J_2			J_1			J_2		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
D1	0.60	0.70	0.65	0.55	0.62	0.58	0.75	0.79	0.77	0.68	0.74	0.71	0.78	0.82	0.80	0.74	0.75	0.74	0.79	0.84	0.81	0.74	0.76	0.75
D2	0.61	0.68	0.64	0.57	0.61	0.59	0.74	0.77	0.75	0.67	0.70	0.68	0.76	0.80	0.78	0.70	0.70	0.70	0.77	0.82	0.79	0.70	0.72	0.71
D3	0.62	0.69	0.65	0.54	0.60	0.57	0.73	0.80	0.76	0.65	0.67	0.66	0.77	0.81	0.79	0.71	0.69	0.70	0.77	0.84	0.80	0.72	0.70	0.71
D4	0.63	0.71	0.67	0.53	0.58	0.55	0.72	0.81	0.76	0.63	0.71	0.67	0.74	0.83	0.78	0.67	0.72	0.69	0.73	0.82	0.77	0.68	0.72	0.69
Avg.	0.62	0.70	0.65	0.55	0.60	0.57	0.74	0.79	0.76	0.66	0.71	0.68	0.76	0.82	0.79	0.71	0.72	0.71	0.77	0.83	0.79	0.71	0.73	0.72

Table IV: Evaluation of topics or "points" of contention expressed in posts. For each method, we report the precision (P) and recall (R) for discovering points of contention in posts belonging to a particular domain. The experiments were performed on four domains D1: Politics, D2: Religion, D3: Society, D4: Science. The precision and recall for each domain are the average precision and recall over 125 posts in that domain.

Statistical significance: Differences between Nearest Noun Phrase and JTE for both judges (J_1, J_2) across all domains were significant at 98% confidence level (p<0.02). Differences among JTE, JTE-R and JTE-P for both judges (J_1, J_2) across all domains were significant at 95% confidence level (p<0.05). A two tailed t-test was used for testing significance.

Iteration	LDA	SLDA	JTE	JTE (E=3)	JTE (E=4)	JTE-R	JTE-P
1000	1795	1745	1475	1489	1497	1467	1442
2000	1684	1568	1381	1397	1423	1343	1326
3000	1575	1474	1318	1331	1348	1273	1257
4000	1561	1421	1248	1266	1278	1223	1208

(A) Perplexity vs. Gibbs Iteration

KL-Div.	LDA	SLDA	JTE	JTE-R	JTE-P
Θ^T	3.4	3.3	9.2	9.9	13.6
Θ^E	-	-	14.1	15.1	17.8
Φ^T	16.8	17.7	20.1	21.8	22.2
Φ^E	-	-	10.9	11.3	11.8

(B) Avg. KL-Div. between models

Table V: (A) Perplexity comparison of models across Gibbs iterations. The number of topics and AD-expression types were fixed at T = 100, E = 2. All differences are statistically significant (p<0.001) over two tailed t-test across samples from different chains for each group. (B) Average KL-Divergence of topics and AD-expressions, $D_{KL}(\varphi_z||\varphi_{\acute{z}})$ and per document distributions of topics and AD-expressions, $D_{KL}(\theta_d||\theta_{\acute{d}})$. For JTE-P we report the average per pair distributions of topics and AD-expressions $D_{KL}(\theta_p||\theta_{\acute{p}})$. All differences are significant (p<0.01) over two tailed t-test.

2.1.4.4 Discovering Contention Points

We now turn to the task of automatically discovering points of contention in contentious/disagreeing posts. By "points", we mean the topical terms on which the disagreement has been expressed. We employ the JTE and JTE-R models in the following manner using estimated θ_d^T. Note that JTE-P cannot be directly used for this task because it has θ^T placed in the author pair plate so its topics are pair specific (θ_p^T) rather than post specific. However, since we know the posterior topic assignments of z_n^d, we can get a posterior estimate of θ_d^T for JTE-P using $\theta_{d,t}^T = \frac{\left|\{j|z_j^d = t, 1 \leq j \leq N_d\}\right|}{\left|\{j|r_{d,j} = \hat{t}, 1 \leq j \leq N_d\}\right|}$. Given a contentious/disagreeing post d, we first select the top k topics that are mentioned in d according to its topic distribution, θ_d^T. Let T_d denote the set of these top k topics in d. Then, for each disagreement expression $e \in d \cap \varphi_{Disagreement}^E$, we emit the topical terms of topics in T_d which appear within a word window of v from e in d. More precisely, we emit the set $H = \{w|w \in d \cap \varphi_t^T, t \in T_d, |posi(w) - posi(e)| \leq v\}$, where $posi(\cdot)$ returns the position index of the word/phrase in a document d. To compute the intersection $w \in d \cap \varphi_t^T$, we need a threshold. This is so because the Dirichlet distribution has a smoothing effect which assigns some non-zero probability mass to every term in the vocabulary for each topic t. So for computing the intersection, we considered only terms in φ_t^T which have $p(v|t) = \varphi_{t,v}^T > 0.001$ as probability masses lower than 0.001 are more due to the smoothing effect of the Dirichlet distribution than true correlation. In an actual application, the values for k and v can be set according to the user's need. In this experiment, we use $k = 3$ and $v = 5$, which are reasonable because a post normally does not talk about many topics (k), and the contention points (topical terms) appear close to the contentious expressions.

For comparison, we also designed a baseline. For each disagreeing expression, $e \in d \cap \varphi_{Disagreement}^E$, we emit the nouns and noun phrases within the same window v as the points of

contention in d. This baseline is reasonable because topical terms are usually nouns and noun phrases and are near contentious expressions. But we should note that this baseline cannot standalone as it has to rely on the expression models Φ^E of JTE-P.

Next, to evaluate the performance of these methods in discovering points of contention, we randomly selected 125 contentious posts from each domain in our dataset and employed the aforementioned methods on the posts to discover the points of contention in each post. Then we asked two human judges (graduate students fluent in English) to manually judge the results produced by each method for each post. We asked them to report the precision (% of terms discovered by a method which are indeed valid points of contention in a post) and recall (% of all valid points of contention which were discovered) for each post. In Table IV, we report the average precision and recall for 125 posts in each domain by the two human judges J_1 and J_2 for different methods. Since this judging task is subjective, the differences in the results from the two judges are not surprising. We observe that across all domains, JTE, JTE-R and JTE-P progressively improve performance over the baseline. Note that it is difficult to compute agreement of two judges using kappa because although the models identify topic terms (which are the same for both judges), the judges also identify additional terms (for recall calculation) which are not found by the models.

2.1.4.5 Assessing Predictive Strength using Perplexity

To measure the ability of JTE, JTE-R and JTE-P to act as good "generative" models, we computed the test-set (see below) perplexity under estimated parameters and also compared with the resulting values of LDA and SLDA models.

Perplexity, widely used in the statistical language modeling community to assess the predictive power of a model, is algebraically equivalent to the inverse of the geometric mean per-word likelihood. A lower perplexity score indicates a better generalization performance. As perplexity monotonically

decreases with increase in log-likelihood (by definition), it implies that lower perplexity is better since higher log-likelihood on training data means that the model "fits" the data better and a higher log-likelihood on the test set implies that the model can "explain" the unseen data better. Thus, lower perplexity implies that the words are "less surprising" to the model. In our experiments we used 15% of our data (in Section 2.1.4.1) as our held out test set. As JTE-P requires pair structures, for proper comparison across all models, the corpus was restricted to posts which have at least one quotation. This is reasonable as quoting/replying is an integral part of debates/discussions and we found that about 77% of all posts have quoted/replied-to at least one other post (this count excludes the first posts of threads as they start the discussions and usually have nobody to quote/reply-to). The perplexity (*PPX*) of JTE given the learned model parameters Φ^T, Φ^E and the state of the Markov chain $\mu = \{\vec{z}, \vec{r}, \vec{w}\}$ is given by:

$$PPX = exp\left[-\frac{1}{\overline{W}}\sum_{\tilde{d}=1}^{D_{test}}\frac{1}{S}\sum_{s=1}^{S} logP(\tilde{d}|\mu^s)\right] \tag{2.8}$$

where $\overline{W}$ denotes the total number of terms in D_{test} . To obtain a better estimate, we average the per-word log-likelihood over *S* different chains. μ^s denotes the Markov state corresponding to chain *s*. From the generative process of JTE, we get:

$$logP\left(\tilde{d}\middle|\mu^s\right) = \sum_{v=1}^{V}\left(n_{\tilde{d}}^{v}\log\left(\sum_{t=1}^{T}\varphi_{t,v}^{s}\theta_{\tilde{d},t}^{s} + \sum_{e=1}^{E}\varphi_{e,v}^{s}\theta_{\tilde{d},e}^{s}\right)\right) \tag{2.9}$$

where $n_{\tilde{d}}^{v}$ denotes the number of times term $v \in V$ occurred in $\tilde{d} \in D_{test}$ and $\theta_{\tilde{d},t}^{s}$ and $\theta_{\tilde{d},e}^{s}$ are estimated by querying the model according to the query sampler. In a similar way, the perplexities of the other three models can also be derived.

We compare model perplexities across Gibbs iterations with $T = 100$ and $E = 2$ in Table V (A). We note the following observations:

1. *Noise Reduction*: The proposed models attain significantly (see caption of Table V (A)) lower perplexities with fewer iterations than LDA and SLDA showing that the new models fit the debate/discussion forum data better and clustering using the new framework contains less noise. This improvement is attributed to the capabilities of the framework to separate and account for AD-expressions using Φ^E.

2. *The number of AD-expression types*, *E*: In Sections 2.1 and 2.1.4.2, we hypothesized that for discussions/debates we mostly have two expression types: agreement and disagreement. To test this hypothesis, we ran JTE with $E > 2$ (Table V (A), columns 5, 6). We find that the test-set perplexity slightly increases (than $E = 2$) showing performance degradation. It is interesting to note that the number of AD-expression types *E* impacts perplexity differently than the number of topics *T* (as the decrease in perplexity usually slows in inverse proportions with increase in the number of topics [Blei et al. 2003]). We also tried increasing E to 5, 10, etc. However, the performance deteriorated with increase in model perplexity. This result supports our prior hypothesis of $E = 2$ in debate forums.

2.1.4.6 Evaluating Topic Distinctiveness using KL-Divergence

Another important measure for topic models is topic distinctiveness [Kawamae, 2010]. Here, we want to assess how distinctive the discovered topics and AD-expressions are. To measure topic and AD-expression distinctiveness, we computed the average topic and AD-expression distribution (φ_z^T and φ_z^E) separations between all pairs of latent topics and AD-expression types. To measure separations, we choose KL-Divergence as our metric as suggested in [Kawamae, 2010]. Clearly, for more distinctive topic and AD-expression discovery, it is desirable to have higher average KL-Divergence. Table V (B) shows the comparison results. Again, as JTE-P requires pair structures, for proper comparison, all models were run on the restricted corpus where posts have at least one quotation. We observe that topics

discovered by the new models are more distinctive than LDA and SLDA. For both topics and AD-expressions, JTE-R performed better than JTE showing that reply relations are highly beneficial. JTE-P with pair structures performed the best. Table V (B) also reports the average separations of per document distribution of topics and expressions (θ_d^T and θ_d^E). For models JTE and JTE-R, having higher average KL-Divergence for Θ^T and Θ^E implies that documents are well separated based on the estimated topics and two AD-expression types exhibited. We see that both JTE and JTE-R obtain higher average KL-Divergence for Θ^T than LDA and SLDA. Such separations are particularly useful when topic models are used for performing information retrieval.

Lastly, we look at the average per pair separation of topics and AD-expressions for JTE-P. Clearly, the KL-Divergence values indicate good separations. This information may be further used to mine contending author-pairs or classify these pairs according to interests, i.e., the posterior on θ_p^E can be used to make predictions on the interaction nature of any two authors. Detailed studies on such tasks appear in [Mukherjee and Liu, 2013].

This concludes our discussion for the first problem of modeling social conversations in debates. We now take a departure from modeling fine-grained sentiments in debates and throw light on the second problem of modeling review comments.

2.2 Modeling review comments

Online reviews enable consumers to evaluate the products and services that they have used. These reviews are also used by other consumers and businesses as a valuable source of opinions.

However, reviews only give the evaluations and experiences of the reviewers. Often a reviewer may not be an expert of the product and may misuse the product or make other mistakes. There may also be aspects of the product that the reviewer did not mention but a reader wants to know. Some reviewers may even write fake reviews to promote some products, which is called *opinion spamming* [Jindal and

Liu, 2008]. To improve the online review system and user experience, some review hosting sites allow readers to write comments about reviews (apart from just providing a feedback by clicking whether the review is helpful or not). Many reviews receive a large number of comments. It is difficult for a reader to read them to get a gist of them. An automated comment analysis would be very helpful. In this section, we propose two new generative models are proposed. The first model is called the Topic and Multi-Expression model (TME). It models topics and different types of expressions, which represent different types of comment posts:

1. *Thumbs-up* (e.g., "review helped me")
2. *Thumbs-down* (e.g., "poor review")
3. *Question* (e.g., "how to")
4. *Answer acknowledgement* (e.g., "thank you for clarifying"). Note that we have no expressions for answers to questions as there are usually no specific phrases indicating that a post answers a question except starting with the name of the person who asked the question. However, there are typical phrases for acknowledging answers, thus *answer acknowledgement* expressions.
5. *Disagreement* (*contention*) (e.g., "I disagree")
6. *Agreement* (e.g., "I agree").

For ease of presentation, we call these expressions the *comment expressions* (or *C-expressions*). TME provides a basic model for extracting these pieces of information and topics. Its generative process separates topics and C-expression types using a switch variable and treats posts as random mixtures over latent topics and C-expression types. The second model, called ME-TME, improves TME by using Maximum-Entropy priors to guide topic/expression switching. In short, the two models provide a principled and integrated approach to simultaneously discover topics and C-expressions, which is the goal of this work. Note that topics are usually product aspects in this work.

The extracted C-expressions and topics from review comments are very useful in practice. First of all, C-expressions enable us to perform more accurate classification of comments, which can give us a good evaluation of the review quality and credibility. For example, a review with many *Disagreeing* and *Thumbs-down* comments is dubious. Second, the extracted C-expressions and topics help identify the key product aspects that people are troubled with in disagreements and in questions. Our experimental results in Section 2.2.3 will demonstrate these capabilities of our models.

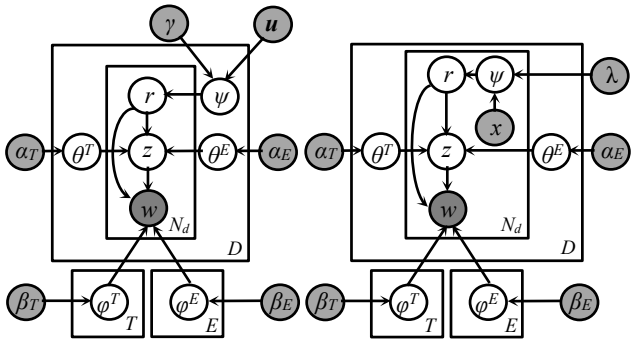
With these pieces of information, comments for a review can be summarized. The summary may include, but not limited to, the following: (1) percent of people who give the review thumbs-up or thumbs-down; (2) percent of people who agree or disagree (or contend) with the reviewer; (3) contentious (disagreed) aspects (or topics); (4) aspects about which people often have questions.

To the best of our knowledge, there is no reported work on such a fine-grained modeling of review comments. The related works are mainly in sentiment analysis [Pang and Lee, 2008; Liu 2012], e.g., topic and sentiment modeling, review quality prediction and review spam detection.

The proposed models have been evaluated both qualitatively and quantitatively using a large number of review comments from Amazon.com. Experimental results show that both TME and ME-TME are effective in performing their tasks. ME-TME also outperforms TME significantly.

2.2.1 Topic and Multi-Expression (TME) Model

This section discusses TME. The next section discusses ME-TME, which improves TME. These models belong to the family of generative models for text where words and phrases (*n*-grams) are viewed as random variables, and a document is viewed as a bag of *n*-grams and each *n*-gram takes a value from a predefined vocabulary. In this work, we use up to 4-grams, i.e., n = 1, 2, 3, 4. For simplicity, we use *terms* to denote both *words* (unigrams or 1-grams) and *phrases* (*n*-grams). We denote the entries in our vocabulary by $v_{1...V}$ where V is the number of unique terms in the vocabulary. The



(A) TME Model (B) ME-TME Model

Figure 3: Plate notations of (A) TME and (B) ME-TME models.

entire corpus contains $d_{1...D}$ documents. A document (e.g., comment post) d is represented as a vector of terms $\boldsymbol{w_d}$ with N_d entries. W is the set of all observed terms with cardinality, $|W| = \sum_d N_d$.

The TME (Topic and Multi-Expression) model is a hierarchical generative model motivated by the joint occurrence of various types of expressions indicating *Thumbs-up*, *Thumbs-down*, *Question*, *Answer acknowledgement*, *Agreement,* and *Disagreement* and topics in comment posts. As before, these expressions are collectively called C-expressions. A typical comment post mentions a few topics (using semantically related topical terms) and expresses some viewpoints with one or more C-expression types (using semantically related expressions). This observation motivates the generative process of our model where documents (posts) are represented as random mixtures of latent topics and C-expression types. Each topic or C-expression type is characterized by a distribution over terms (words/phrases). Assume we have $t_{1...T}$ topics and $e_{1...E}$ expression types in our corpus. Note that in our case of Amazon review comments, based on reading various posts, we hypothesize that E = 6 as in such review discussions, we mostly find 6 expression types (more details in Section 2.2.3.1). Let ψ_d denote the distribution of topics and C-expressions in a document d with $r_{d,j} \in \{\hat{t}, \hat{e}\}$ denoting the binary indicator variable (topic or C-expression) for the j^{th} term of d, $w_{d,j}$. $z_{d,j}$denotes the appropriate topic or C-expression type index to which $w_{d,j}$ belongs. We parameterize multinomials over topics using a matrix

$\Theta^T_{D\times T}$whose elements $\theta^T_{d,t}$ signify the probability of document d exhibiting topic t. For simplicity of notation, we will drop the latter subscript (t in this case) when convenient and use θ^T_d to stand for the d^{th} row of Θ^T. Similarly, we define multinomials over C-expression types using a matrix $\Theta^E_{D\times E}$. The multinomials over terms associated with each topic are parameterized by a matrix $\Phi^T_{T\times V}$, whose elements $\varphi^T_{t,v}$ denote the probability of generating v from topic t. Likewise, multinomials over terms associated with each C-expression type are parameterized by a matrix $\Phi^E_{E\times V}$. We now define the generative process of TME (see Figure 3 (A)).

1. For each C-expression type e, draw $\varphi^E_e \sim Dir(\beta_E)$
2. For each topic *t*, draw $\varphi^T_t \sim Dir(\beta_T)$
3. For each comment post $d \in \{1 \dots D\}$:
   - i. Draw $\psi_d \sim Beta(\gamma \boldsymbol{u})$
   - ii. Draw $\theta^E_d \sim Dir(\alpha_E)$
   - iii. Draw $\theta^T_d \sim Dir(\alpha_T)$
   - iv. For each term $w_{d,j}$, $j \in \{1 \dots N_d\}$:
     a. Draw $r_{d,j} \sim Bernoulli(\psi_d)$
     b. if ($r_{d,j} = \hat{e}$ // $w_{d,j}$is a C-expression term
        Draw $z_{d,j} \sim Mult(\theta^E_d)$
        else // $r_{d,j} = \hat{t}$, $w_{d,j}$is a topical term
        Draw $z_{d,j} \sim Mult(\theta^T_d)$
     c. Emit $w_{d,j} \sim Mult(\varphi^{r_{d,j}}_{z_{d,j}})$

To learn the TME model from data, as exact inference is not possible, we resort to approximate inference using *collapsed Gibbs sampling* [Griffiths and Steyvers, 2004]. Gibbs sampling is a form of Markov Chain Monte Carlo method where a Markov chain is constructed to have a particular stationary distribution. In our case, we want to construct a Markov chain which converges to the posterior distribution over R and Z conditioned on the data. We only need to sample z and r as we use collapsed Gibbs sampling and the dependencies of θ and φ have been integrated out analytically in the joint. Denoting the random variables $\{w, z, r\}$ by singular subscripts$\{w_k, z_k, r_k\}, k_{1...K}$, where $K = \sum_d N_d$, a single iteration consists of performing the following sampling:

$$p(z_k = t, r_k = \hat{t} | W_{\neg k}, Z_{\neg k}, R_{\neg k}, w_k = v) \propto \frac{n^T_{d\ \neg k} + \gamma_a}{n^T_{d\ \neg k} + n^E_{d\ \neg k} + \gamma_a + \gamma_b} \times \frac{n^{DT}_{d,t\ \neg k} + \alpha_T}{n^{DT}_{d,(\cdot)\ \neg k} + T\alpha_T} \times \frac{n^{CT}_{t,v\ \neg k} + \beta_T}{n^{CT}_{t,(\cdot)\ \neg k} + V\beta_T} \tag{2.10}$$

$$p(z_k = e, r_k = \hat{e} | W_{\neg k}, Z_{\neg k}, R_{\neg k}, w_k = v) \propto \frac{n^E_{d\ \neg k} + \gamma_b}{n^T_{d\ \neg k} + n^E_{d\ \neg k} + \gamma_a + \gamma_b} \times \frac{n^{DE}_{d,e\ \neg k} + \alpha_E}{n^{DE}_{d,(\cdot)\ \neg k} + E\alpha_E} \times \frac{n^{CE}_{e,v\ \neg k} + \beta_E}{n^{CE}_{e,(\cdot)\ \neg k} + V\beta_E} \tag{2.11}$$

where $k = (d, j)$ denotes the j^{th} term of document d and the subscript $\neg k$ denotes assignments excluding the term at (d, j). Counts$n^{CT}_{t,v}$ and $n^{CE}_{e,v}$ denote the number of times term v was assigned to topic t and expression type e respectively. $n^{DT}_{d,t}$ and $n^{DE}_{d,e}$ denote the number of terms in document d that were assigned to topic t and C-expression type e respectively. Lastly, n^T_d and n^E_d are the number of terms in d that were assigned to topics and C-expression types respectively. Omission of the latter index denoted by $(\cdot)$ represents the marginalized sum over the latter index. We employ a blocked sampler jointly sampling *r* and *z* as this improves convergence and reduces autocorrelation of the Gibbs sampler [Rosen-Zvi et al., 2004].

Asymmetric Beta priors: Based on our initial experiments with TME, we found that properly setting the smoothing hyper-parameter $\gamma\boldsymbol{u}$ is crucial as it governs the topic/expression switch. According to the generative process, ψ_d is the (success) probability (of the Bernoulli distribution) of emitting a

topical/aspect term in a comment post d and$1-\psi_d$, the probability of emitting a C-expression term in d. Without loss of generality, we draw ψ_d~$Beta(\gamma \boldsymbol{u})$ where γ is the concentration parameter and $\boldsymbol{u} = [u_a, u_b]$ is the base measure. Without any prior belief, one resorts to uniform base measure $u_a = u_b$ =0.5 (i.e., assumes that both topical and C-expression terms are equally likely to be emitted in a comment post). This results in symmetric Beta priors ψ_d~$Beta(\gamma_a, \gamma_b)$ where $\gamma_a = \gamma u_a, \gamma_b = \gamma u_b$ and $\gamma_a = \gamma_b = \gamma/2$. However, knowing the fact that topics are more likely to be emitted than expressions in a post *apriori* motivates us to take guidance from asymmetric priors (i.e., we now have a non-uniform base measure$\boldsymbol{u}$). This asymmetric setting of γ ensures that samples of ψ_d are more close to the actual distribution of topical terms in posts based on some domain knowledge. Symmetric γ cannot utilize any prior knowledge. In [Lin and He, 2009], a method was proposed to incorporate domain knowledge during Gibbs sampling initialization, but its effect becomes weak as the sampling progresses [Jo and Oh, 2011].

For asymmetric priors, we estimate the hyper-parameters from labeled data. Given a labeled set D_L, where we know the per post probability of C-expression emission $(1-\psi_d)$, we use the method of moments to estimate $\gamma = [\gamma_a, \gamma_b]$ as follows:

$$\gamma_a = \mu\Big(\tfrac{\mu(1-\mu)}{\sigma} - 1\Big), \gamma_b = \gamma_a\Big(\tfrac{1}{\mu} - 1\Big);\ \mu = E[\psi_d], \sigma = Var[\psi_d] \tag{2.12}$$

2.2.2 Guidance using Max-Ent Priors: ME-TME Model

The guidance of Beta priors, although helps, is still relatively coarse and weak. We can do better to produce clearer separation of topical and C-expression terms. An alternative strategy is to employ *Maximum-Entropy* (Max-Ent) priors instead of Beta priors. The Max-Ent parameters can be learned from a small number of labeled topical and C-expression terms (words and phrases) which can serve as good priors. The idea is motivated by the following observation: topical and C-expression terms

typically play different syntactic roles in a sentence. Topical terms (e.g. “ipod” “cell phone”, “macro lens”, “kindle”, etc.) tend to be noun and noun phrases while expression terms (“I refute”, “how can you say”, “great review”) usually contain pronouns, verbs, wh-determiners, adjectives, and modals. In order to utilize the part-of-speech (POS) tag information, we move the topic/C-expression distribution ψ_d (the prior over the indicator variable $r_{d,j}$) from the document plate to the word plate (see Figure 3 (B)) and draw it from a Max-Ent model conditioned on the observed feature vector $\overrightarrow{x_{d,j}}$ associated with $w_{d,j}$ and the learned Max-Ent parameters λ. $x_{d,j}$ can encode arbitrary contextual features for learning. With Max-Ent priors, we have the new model ME-TME. In this work, we encode both lexical and POS features of the previous, current and next POS tags/lexemes of the term, $w_{d,j}$. More specifically,

$$\overrightarrow{x_{d,j}} = \left[POS_{w_{d,j}-1}, POS_{w_{d,j}}, POS_{w_{d,j}+1}, w_{d,j}-1, w_{d,j}, w_{d,j}+1\right]$$

For phrasal terms (n-grams), all POS tags and lexemes of $w_{d,j}$are considered as features. Incorporating Max-Ent priors, the Gibbs sampler of ME-TME is given by:

$$p(z_k = t, r_k = \hat{t} | W_{\neg k}, Z_{\neg k}, R_{\neg k}, w_k = v) \propto \frac{exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, \hat{t}))}{\sum_{y \in \{\hat{e},\hat{t}\}} exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))} \times \frac{n^{DT}_{d,t\ \neg k} + \alpha_T}{n^{DT}_{d,(\cdot)\ \neg k} + T\alpha_T} \times \frac{n^{CT}_{t,v\ \neg k} + \beta_T}{n^{CT}_{t,(\cdot)\ \neg k} + V\beta_T} \quad (2.13)$$

$$p(z_k = e, r_k = \hat{e} | W_{\neg k}, Z_{\neg k}, R_{\neg k}, w_k = v) \propto \frac{exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, \hat{e}))}{\sum_{y \in \{\hat{e},\hat{t}\}} exp(\sum_{i=1}^{n} \lambda_i f_i(x_{d,j}, y))} \times \frac{n^{DE}_{d,e\ \neg k} + \alpha_E}{n^{DE}_{d,(\cdot)\ \neg k} + E\alpha_E} \times \frac{n^{CE}_{e,v\ \neg k} + \beta_E}{n^{CE}_{e,(\cdot)\ \neg k} + V\beta_E} \quad (2.14)$$

where $\lambda_{1...n}$ are the parameters of the learned Max-Ent model corresponding to the n binary feature functions $f_{1...n}$ from Max-Ent.

2.2.3 Experimental evaluation

We now evaluate the proposed TME and ME-TME models. Specifically, we evaluate the discovered C-expressions, contentious aspects, and aspects often mentioned in questions.

2.2.3.1 Dataset and experiment settings

We crawled comments of reviews in Amazon.com for a variety of products. For each comment we extracted its id, the comment author id, the review id on which it commented, and the review author id. Our database consisted of 21,316 authors, 37,548 reviews, and 88,345 comments with an average of 124 words per comment post.

For all our experiments, the hyper-parameters for TME and ME-TME were set to the heuristic values $\alpha_T = 50/T$, $\alpha_E = 50/E$, $\beta_T = \beta_E = 0.1$ as suggested in [Griffiths and Steyvers, 2004]. For γ, we estimated the asymmetric Beta priors using the method of moments discussed in Section 3. We sampled 1000 random posts and for each post we identified the C-expressions emitted. We thus computed the per-post probability of C-expression emission $(1 - \psi_d)$ and used equation (2.12) to get the final estimates, γ_a = 3.66, γ_b= 1.21. To learn the Max-Ent parameters λ, we randomly sampled 500 terms from our corpus appearing at least 10 times and labeled them as topical (332) or C-expressions (168) and used the corresponding feature vector of each term (in the context of posts where it occurs) to train the Max-Ent model. We set the number of topics, T = 100 and the number of C-expression types, E = 6 (*Thumbs-up*, *Thumbs-down*, *Question*, *Answer acknowledgement*, *Agreement* and *Disagreement*) as in review comments, we usually find these six dominant expression types. Note that knowing the exact number of topics, T and expression types, E in a corpus is difficult. While non-parametric Bayesian approaches [Teh et al., 2006] aim to estimate T from the corpus, in this work the heuristic values obtained from our initial experiments produced good results. We also tried increasing E to 7, 8, etc. However, it did not produce any new dominant expression type. Instead, the expression types became less specific as the expression term space became sparser.

Expression type	Top terms of TME model
Thumbs-up (e1)	**review**, **thanks**, great review, nice review, **time**, best review, **appreciate**, **you**, your review helped, nice, terrific, review helped me, good critique, **very**, **assert**, **wrong**, useful review, **don't**, **misleading**, thanks a lot, …
Thumbs-down (e2)	**review**, **no**, poor review, imprecise, **you**, **complaint**, **very**, suspicious, bogus review, **absolutely**, **credible**, very unfair review, criticisms, **true**, disregard this review, disagree with, **judgment**, without owning, …
Question (e3)	question, **my**, **I**, how do I, why isn't, please explain, **good answer**, clarify, don't understand, my doubts, I'm confused, **does not**, **understand**, help me decide, how to, **yes**, **answer**, how can I**, can't explain**, …
Answer Acknowledgement (e4)	**my**, **informative**, answer, good reply, thank you for clarifying, answer doesn't, good answer, **vague**, helped me choose, useful suggestion, **don't understand**, **cannot explain**, your answer, **doubts**, answer isn't, …
Disagreement (e5)	disagree, **I**, **don't**, I disagree, argument claim, I reject, I refute, I refuse, oppose, debate, **accept**, don't agree, **quote**, **sense**, would disagree, **assertions**, I doubt, **right**, your, **really**, you, I'd disagree, cannot, nonsense,...
Agreement (e6)	yes, **do**, correct, indeed, **no**, right, I agree, **you**, agree, I accept, **very**, yes indeed, true in fact, indeed correct, I'd agree, **completely**, true, **but**, **doesn't**, **don't**, definitely, **false**, completely agree, agree with your, true, …

Table VI: Top terms (comma delimited) of six expression types using TME model.

Expression type	Top terms of ME-TME model
Thumbs-up (e1)	**review**, **you**, great review, *I'm glad I read,* best review, *review convinced me*, review helped me, good review, *terrific review*, **job**, *thoughtful review*, awesome review, *level headed review*, *good critique*, good job, *video review*,...
Thumbs-down (e2)	**review**, **you**, bogus review, con, useless review, *ridiculous*, *biased review*, very unfair review, *is flawed*, **completely**, *skeptical*, *badmouth*, *misleading review*, cynical review, wrong, disregard this review, *seemingly honest*, …
Question (e3)	question, **I**, how do I, why isn't, please explain, clarify, *any clues*, **answer**, please explain, help me decide, **vague**, how to, how do I, where can I, *how to set*, *I was wondering how*, *could you explain*, how can I, *can I use*, …
Answer Acknowledgement (e4)	**my**, good reply, , answer, reply, helped me choose, *clears my,* *valid answer*, answer doesn't, *satisfactory answer*, **can you clarify**, *informative answer*, useful suggestion, *perfect answer*, thanks for your reply, **doubts**, …
Disagreement (e5)	disagree, **I**, **don't**, I disagree, doesn't, *I don't buy your*, **credible**, I reject, I doubt, I refuse, I *oppose*, *sheer nonsense*, **hardly**, don't agree, *can you prove*, *you have no clue*, *how do you say*, **sense**, *you fail*, *contradiction*, …
Agreement (e6)	**I**, **do**, agree, **point**, yes, **really**, *would agree*, **you**, agree, I accept, **claim**, *agree completely*, *personally agree*, true in fact, indeed correct, *well said*, *valid point*, correct, **never meant**, **might not**, *definitely agree*,…

Table VII: Top terms (comma delimited) of six expression types using ME-TME model. **Red** (**bold**) terms denote possible errors. *Blue* (*italics*) terms denote those newly discovered by the model; rest (black) were used in Max-Ent training.

2.2.3.2 C-Expression evaluation

We now evaluate the discovered C-expressions. We first evaluate them qualitatively in Tables VI and VII. Table VI shows the top terms of all expression types using the TME model. We find that TME can discover and cluster many correct C-expressions, e.g., "great review", "review helped me" in *Thumbs-up*; "poor review", "very unfair review" in *Thumbs-down*; "how do I", "help me decide" in *Question*; "good reply", "thank you for clarifying" in *Answer Acknowledgement*; "I disagree", "I refute" in *Disagreement*; and "I agree", "true in fact" in *Agreement*. However, with the guidance of Max-Ent priors, ME-TME did much better (Table VII). For example, we find "level headed review", "review convinced me" in *Thumbs-up*; "biased review", "is flawed" in *Thumbs-down*; "any clues", "I was wondering how" in *Question*; "clears my", "valid answer" in *Answer-acknowledgement*; "I don't buy your", "sheer nonsense" in *Disagreement*; "agree completely", "well said" in *Agreement*. These newly discovered phrases by ME-TME are marked in *blue* in Table VII. ME-TME also has fewer errors.

Next, we evaluate them quantitatively using the metric *precision @ n*, which gives the precision at different rank positions. This metric is appropriate here because the C-expressions (according to top terms in Φ^{E}) produced by TME and ME-TME are rankings. Table VIII reports the precisions @ top 25, 50, 75, and 100 rank positions for all six expression types across both models. We evaluated till top 100 positions because it is usually important to see whether a model can discover and rank those major expressions of a type at the top. We believe that top 100 are sufficient for most applications. From Table VIII, we observe that ME-TME consistently outperforms TME in precisions across all expression types and all rank positions. This shows that Max-Ent priors are more effective in discovering expressions than Beta priors. Note that we couldn't compare with an existing baseline because there is no reported study on this problem.

C-Expression Type	P@25		P@50		P@75		P@100	
	TME	ME-TME	TME	ME-TME	TME	ME-TME	TME	ME-TME
Thumbs-up	0.60	0.80	0.66	0.78	0.60	0.69	0.55	0.64
Thumbs-down	0.68	0.84	0.70	0.80	0.63	0.67	0.60	0.65
Question	0.64	0.80	0.68	0.76	0.65	0.72	0.61	0.67
Answer-Acknowledgement	0.68	0.76	0.62	0.72	0.57	0.64	0.54	0.58
Disagreement	0.76	0.88	0.74	0.80	0.68	0.73	0.65	0.70
Agreement	0.72	0.80	0.64	0.74	0.61	0.70	0.60	0.69

Table VIII: Precision @ top 25, 50, 75, and 100 rank positions for all C-expression types.

Features	Thumbs-up			Thumbs-down			Question			Answer-Ack.			Disagreement			Agreement			Answer		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
W+POS 1-gram	0.68	0.66	0.67	0.65	0.65	0.65	0.71	0.68	0.69	0.64	0.61	0.62	0.73	0.72	0.72	0.67	0.65	0.66	0.58	0.57	0.57
W+POS 1-2 gram	0.72	0.69	0.70	0.68	0.67	0.67	0.74	0.69	0.71	0.69	0.63	0.65	0.76	0.75	0.75	0.71	0.69	0.70	0.60	0.57	0.58
W+POS, 1-3 gram	0.73	0.71	0.72	0.69	0.68	0.68	0.75	0.69	0.72	0.70	0.64	0.66	0.76	0.76	0.76	0.72	0.70	0.71	0.61	0.58	0.59
W+POS, 1-4 gram	0.74	0.72	0.73	0.71	0.68	0.69	0.75	0.70	0.72	0.70	0.65	0.67	0.77	0.76	0.76	0.73	0.70	0.71	0.61	0.58	0.59
C-Expr, Φ^E, TME	0.82	0.74	0.78	0.77	0.71	0.74	0.83	0.75	0.78	0.75	0.72	0.73	0.83	0.80	0.81	0.78	0.75	0.76	0.66	0.61	0.63
C-Expr, Φ^E, ME-TME	0.87	0.79	0.83	0.80	0.73	0.76	0.87	0.76	0.81	0.77	0.72	0.74	0.86	0.81	0.83	0.81	0.77	0.79	0.67	0.61	0.64

Table IX: Precision (P), Recall (R), and F_1 scores of binary classification using SVM and different features. The improvements of our models are significant (p<0.001) over paired t-test across 10-fold cross validation.

D	Φ^E + Noun/Noun Phrase						TME						ME-TME					
	J_1			J_2			J_1			J_2			J_1			J_2		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
D1	0.62	0.70	0.66	0.58	0.67	0.62	0.66	0.75	0.70	0.62	0.70	0.66	0.67	0.79	0.73	0.64	0.74	0.69
D2	0.61	0.67	0.64	0.57	0.63	0.60	0.66	0.72	0.69	0.62	0.67	0.64	0.68	0.75	0.71	0.64	0.71	0.67
D3	0.60	0.69	0.64	0.56	0.64	0.60	0.64	0.73	0.68	0.60	0.67	0.63	0.67	0.76	0.71	0.63	0.72	0.67
D4	0.59	0.68	0.63	0.55	0.65	0.60	0.63	0.71	0.67	0.59	0.68	0.63	0.65	0.73	0.69	0.62	0.71	0.66
Avg.	0.61	0.69	0.64	0.57	0.65	0.61	0.65	0.73	0.69	0.61	0.68	0.64	0.67	0.76	0.71	0.63	0.72	0.67

Table X (A)

D	Φ^E + Noun/Noun Phrase						TME						ME-TME					
	J_1			J_2			J_1			J_2			J_1			J_2		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
D1	0.57	0.65	0.61	0.54	0.63	0.58	0.61	0.69	0.65	0.58	0.66	0.62	0.64	0.73	0.68	0.61	0.70	0.65
D2	0.61	0.66	0.63	0.58	0.61	0.59	0.64	0.68	0.66	0.60	0.64	0.62	0.68	0.70	0.69	0.65	0.69	0.67
D3	0.60	0.68	0.64	0.57	0.64	0.60	0.64	0.71	0.67	0.62	0.68	0.65	0.67	0.72	0.69	0.64	0.69	0.66
D4	0.56	0.67	0.61	0.55	0.65	0.60	0.60	0.72	0.65	0.58	0.68	0.63	0.63	0.75	0.68	0.61	0.71	0.66
Avg.	0.59	0.67	0.62	0.56	0.63	0.59	0.62	0.70	0.66	0.60	0.67	0.63	0.66	0.73	0.69	0.63	0.70	0.66

Table X (B)

Table X (A, B): Evaluation of points of contention (A), questioned aspects (B). D1: Ipod, D2: Kindle, D3: Nikon, D4: Garmin. We report the average precision (P), recall (R), and F_1 score over 100 comments for each particular domain.

Statistical significance: Differences between Nearest Noun Phrase and TME for both judges (J_1, J_2) across all domains were significant at 97% confidence level (p<0.03). Differences among TME and ME-TME for both judges (J_1, J_2) across all domains were significant at 95% confidence level (p<0.05). A paired t-test was used for testing

2.2.3.3 Comment classification

Here we show that the discovered C-expressions can help comment classification. Note that since a comment can belong to one or more types (e.g., a comment can belong to both *Thumbs-up* and A*greement* types), this task is an instance of multi-label classification, i.e., an instance can have more than one class label. In order to evaluate all the expression types, we follow the binary approach which is an extension of one-against-all method for multi-label classification. Thus, for each label, we build a binary classification problem. Instances associated with that label are in one class and the rest are in the other class. To perform this task, we randomly sampled 2000 comments, and labeled each of them into one or more of the following 8 labels: *Thumbs-up*, *Thumbs-down*, *Disagreement*, *Agreement*, *Question*, *Answer-Acknowledgement*, *Answer*, and *None*, which have 432, 401, 309, 276, 305, 201, 228, and 18 comments respectively. We disregard the *None* category due to its small size. This labeling is a fairly easy task as one can almost certainly make out to which type a comment belongs. Thus we didn't use multiple labelers. The distribution reveals that the labels are overlapping. For instance, we found many comments belonging to both *Thumbs-down* and *Disagreement*, *Thumbs-up* with *Acknowledgement* and with *Question*.

For supervised classification, the choice of feature is a key issue. While word and POS n-grams are traditional features, such features may not be the best for our task. We now compare such features with the C-expressions discovered by the proposed models. We used the top 1000 terms from each of the 6 C-expression rankings as features. As comments in *Question* type mostly use the punctuation "?", we added it in our feature set. We use precision, recall and F_1 as our metric to compare classification performance using a trained SVM (linear kernel). All results (Table IX) were computed using 10-fold cross-validation (CV). We also tried Naïve Bayes and Logistic Regression classifiers, but they were poorer than SVM. Hence their results are not reported due to space constraints. As a separate experiment (not shown here also due to space constraints), we analyzed the classification performance

by varying the number of top terms from 200, 400,…, 1000, 1200, etc. and found that the F_1 scores stabilized after top 1000 terms. From Table IX, we see that F_1 scores dramatically increase with C-expression (Φ^E) features for all expression types. TME and ME-TME progressively improve the classification. Improvements of TME and ME-TME being significant (p<0.001) using a paired t-test across 10-fold cross validations shows that the discovered C-expressions are of high quality and useful.

We note that the annotation resulted in a new label "Answer" which consists of mostly replies to comments with questions. Since an "answer" to a question usually does not show any specific expression, it does not attain very good F_1 scores. Thus, to improve the performance of the *Answer* type comments, we added three binary features for each comment c on top of C-expression features:

1. Is the author of c the review author too? The idea here is that most of the times the reviewer answers the questions raised in comments.

2. Is there any comment posted before c by some author a which has been previously classified as a question post?

3. Is there any comment posted after c by author a that replies to c (using @name) and is an *Answer-Acknowledgement* comment (which again has been previously classified as such)?

Using these additional features, we obtained a precision of 0.78 and a recall of 0.73 yielding an F_1 score of 0.75 which is a dramatic increase beyond 0.64 achieved by ME-TME in Table IX.

2.2.3.4 Contention points and questioned aspects

We now turn to the task of discovering points of contention in disagreement comments and aspects (or topics) raised in questions. By "points", we mean the topical terms on which some contentions or disagreements have been expressed. Topics being the product aspects are also indirectly evaluated in this task. We employ the TME and ME-TME models in the following manner.

We only detail the approach for disagreement comments. The same method is applied to question comments. Given a disagreement comment post d, we first select the top *k* topics that are mentioned in *d* according to its topic distribution, θ_d^T. Let T_d be the set of these top k topics in d. Then, for each disagreement expression $w \in d \cap \varphi^E_{e=Disagreement}$, we emit the topical terms (words/phrases) of topics in $T_d$which appear within a word window of q from w in d. More precisely, we emit the set $A = \{w|w \in d \cap \varphi_t^T, t \in T_d, |posi(w) - posi(v)| \leq q\}$, where *posi*$(\cdot)$ returns the position index of the word or phrase in document d. To compute the intersection $w \in d \cap \varphi_t^T$, we need a threshold. This is so because the Dirichlet distribution has a smoothing effect which assigns some non-zero probability mass to every term in the vocabulary for each topic t. So for computing the intersection, we considered only terms in φ_t^T which have $p(v|t) = \varphi_{t,v}^T$ > 0.001 as probability masses lower than 0.001 are more due to the smoothing effect of the Dirichlet distribution than true correlation. In an actual application, the values for k and q can be set according to the user's need. In our experiment, we used k = 3 and q = 5, which are reasonable because a post normally does not talk about many topics (k), and the contention points (aspect terms) appear quite close to the disagreement expressions.

For comparison, we also designed a baseline. For each disagreement (or question) expression $w \in d \cap \varphi^E_{e=Disagreement}$($\varphi^E_{e=Question}$), we emit the nouns and noun phrases within the same window q as the points of contention (question) in d. This baseline is reasonable because topical terms are usually nouns and noun phrases and are near disagreement (question) expressions. We note that this baseline cannot stand alone because it has to rely on our expression models Φ^E of ME-TME.

Next, to evaluate the performance of these methods in discovering points of contention, we randomly selected 100 disagreement (contentious) (and 100 question) comment posts on reviews from each of the 4 product domains: Ipod, Kindle, Nikon Cameras, and Garmin GPS in our database and employed the aforementioned methods to discover the points of contention (question) in each post. Then we asked

two human judges (graduate students fluent in English) to manually judge the results produced by each method for each post. We asked them to report the precision of the discovered terms for a post by judging them as being indeed valid points of contention and report recall in a post by judging how many of actually contentious points in the post were discovered. In Table X (A), we report the average precision and recall for 100 posts in each domain by the two judges J_1 and J_2 for different methods on the task of discovering points (aspects) of contention. In Table X (B), similar results are reported for the task of discovering questioned aspects in 100 question comments for each product domain. Since this judging task is subjective, the differences in the results from the two judges are not surprising. Our judges were made to work in isolation to prevent any bias. We observe that across all domains, ME-TME again performs the best consistently. Note that agreement study using Kappa is not used here as our problem is not to label a fixed set of items categorically by the judges.

CHAPTER 3

KNOWLEDGE INDUCED ASPECT EXTRACTION

Aspect-based sentiment analysis is one of the main frameworks for sentiment analysis [Hu and Liu, 2004; Pang and Lee, 2008; Liu, 2012]. A key task of the framework is to extract aspects of entities that have been commented in opinion documents. The task consists of two sub-tasks. The first sub-task extracts *aspect terms* from an opinion corpus. The second sub-task clusters synonymous aspect terms into categories where each category represents a single aspect, which we call an *aspect category*. Existing research has proposed many methods for aspect extraction. They largely fall into two main types. The first type only extracts aspect terms without grouping them into categories. The second type uses statistical topic models to extract aspects and group them at the same time in an unsupervised manner. Both approaches are useful. However, in practice, one also encounters another setting, where grouping is not straightforward because for different applications the user may need different groupings to reflect the application needs. This problem was reported in [Zhai et al., 2010], which gave the following example. In car reviews, internal design and external design can be regarded as two separate aspects, but can also be regarded as one aspect, called "design", based on the level of details that the user wants to study. It is also possible that the same word may be put in different categories based on different needs. However, [Zhai et al., 2010] did not extract aspect terms. It only categorizes a set of given aspect terms.

To address the problem, we propose two novel statistical models to extract and categorize aspect terms automatically given some seeds in the user interested categories. It is thus able to best meet the user's specific needs. Our models also jointly model both aspects and aspect specific sentiments. The first model is called SAS and the second model is called ME-SAS. ME-SAS improves SAS by using

Maximum-Entropy (or Max-Ent for short) priors to help separate aspects and sentiment terms. However, to train Max-Ent, we do not need manually labeled training data.

In practical applications, asking users to provide some seeds is easy as they are normally experts in their trades and have a good knowledge what are important in their domains.

Our models are related to topic models in general [Blei et al., 2003] and joint models of aspects and sentiments in sentiment analysis in specific (e.g., [Zhao et al., 2010]). However, these current models are typically unsupervised. None of them can use seeds. With seeds, our models are thus semi-supervised and need a different formulation. Our models are also related to the DF-LDA model in [Andrzejewski et al., 2009], which allows the user to set must-link and cannot-link constraints. A must-link means that two terms must be in the same topic (aspect category), and a cannot-link means that two terms cannot be in the same topic. Seeds may be expressed with must-links and cannot-links constraints. However, our models are very different from DF-LDA. First of all, we jointly model aspect and sentiment, while DF-LDA is only for topics/aspects. Joint modeling ensures clear separation of aspects from sentiments producing better results. Second, our way of treating seeds is also different from DF-LDA.

The proposed models are evaluated using a large number of hotel reviews. They are also compared with two state-of-the-art baselines. Experimental results show that the proposed models outperform the two baselines by large margins.

3.1 Proposed seeded models

The standard LDA and existing aspect and sentiment models (ASMs) are mostly governed by the phenomenon called "higher-order co-occurrence" [Heinrich, 2009], i.e., based on how often terms co-occur in different contexts[1]. This unfortunately results in many "non-specific" terms being pulled and clustered. We employ seed sets to address this issue by "guiding" the model to group semantically

[1] w_1 co-occurring with w_2 which in turn co-occurs with w_3 denotes a second-order co-occurrence between w_1 and w_3.

related terms in the same aspect thus making the aspect more specific and related to the seeds (which reflect the user needs). For easy presentation, we will use *aspect* to mean *aspect category* from now on. We replace the multinomial distribution over words for each aspect (as in ASMs) with a special two-level tree structured distribution. The generative process of ASMs assumes that each vocabulary word is independently (i.e., not dependent upon other word-aspect association) and equally probable to be associated with any aspect. Due to higher-order co-occurrences, we find conceptually different terms yet related in contexts (e.g., in hotel domain terms like stain, shower, walls in aspect *Maintenance*; bed, linens, pillows in aspect *Cleanliness*) equally probable of emission for any aspect. Figure 4 (A) shows an example tree. Upon adding the seed sets {bed, linens, pillows} and {staff, service}, the prior structure now changes to the correlated distribution in Figure 4 (B). Thus, each aspect has a top level distribution over non-seed words and seed sets. Each seed set in each aspect further has a second level distribution over seeds in that seed set. The aspect term (word) emission now requires two steps: first sampling at level one to obtain a non-seed word or a seed set. If a non-seed word is sampled we emit it else we further sample at the second seed set level and emit a seed word. This ensures that seed words together have either all high or low aspect associations. Furthermore, seed sets preserve conjugacy between related concepts and also shape more specific aspects by clustering based on higher order co-occurrences with seeds rather than only with standard one level multinomial distribution over words (or terms) alone.

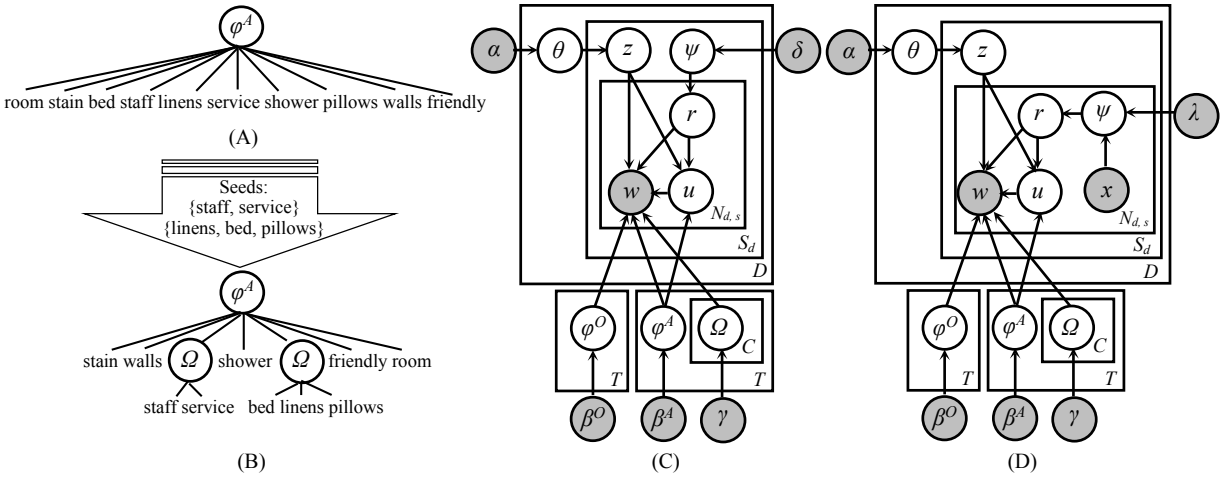


Figure 4: Prior structure: (A) Standard ASMs, (B) Two-level tree structured distribution. Graphical models in plate notation: (C) SAS and (D) ME-SAS.

3.1.1 SAS Model

We now present the proposed Seeded Aspect and Sentiment model (SAS). Let $v_{1...V}$ denote the entries in our vocabulary where V is the number of unique non-seed terms. Let there be C seed sets $Q_{l=1...C}$ where each seed set Q_l is a group of semantically related terms. Let $\varphi^A_{t=1...T}$, $\varphi^O_{t=1...T}$ denote T aspect and aspect specific sentiment models. Also let $\Omega_{t,l}$ denote the aspect specific distribution of seeds in the seed set Q_l. Following the approach of [Zhao et al., 2010], we too assume that a review sentence usually talks about one aspect. A review document $d_{1...D}$ comprises of S_d sentences and each sentence $s \in S_d$ has $N_{d,s}$words. Also, let $Sent^d_s$ denote the sentence s of document d. To distinguish between aspect and sentiment terms, we introduce an indicator (switch) variable $r_{d,s,j} \in \{\hat{a}, \hat{o}\}$ for the j^{th}term of $Sent^d_s$, $w_{d,s,j.}$ Further, let $\psi_{d,s}$ denote the distribution of aspects and sentiments in $Sent^d_s$. The generative process of the SAS model (see Figure 4 (C)) is given by:

1. For each aspect $t \in \{1, \ldots, T\}$:
   i. Draw $\varphi^O_t \sim Dir(\beta^O)$
   ii. Draw a distribution over terms and seed sets $\varphi^A_t \sim Dir(\beta^A)$

a) For each seed set $l \in \{Q_1, \dots, Q_C\}$

Draw a distribution over seeds $\Omega_{t,l} \sim Dir(\gamma)$

2. For each (review) document $d \in \{1, \dots, D\}$:

i. Draw $\theta_d \sim Dir(\alpha)$

ii. For each sentence $s \in \{1, \dots, S_d\}$:

a) Draw $z_{d,s} \sim Mult(\theta_d)$

b) Draw $\psi_{d,s} \sim Beta(\delta)$

c) For each term $w_{d,s,j}$ where $j \in \{1, \dots, N_{d,s}\}$:

I. Draw $r_{d,s,j} \sim Bernoulli(\psi_{d,s}), r_{d,s,j} \in \{\hat{a}, \hat{o}\}$

II. if $r_{d,s,j} = \hat{o}$ // $w_{d,s,j}$ is a sentiment

Emit $w_{d,s,j} \sim Mult(\varphi^O_{z_{d,s}})$

else // $r_{d,s,j} = \hat{a}$, $w_{d,s,j}$ is an aspect

A. Draw $u_{d,s,j} \sim Mult(\varphi^A_{z_{d,s}})$

B. if $u_{d,s,j} \in V$ // non-seed term

Emit $w_{d,s,j} = u_{d,s,j}$

else // $u_{d,s,j}$ is some seed set index say $l_{d,s,j}$

Emit $w_{d,s,j} \sim \Omega_{z_{d,s}, l_{d,s,j}}$

We employ collapsed Gibbs sampling [Griffiths and Steyvers, 2004] for posterior inference. As z and r are at different hierarchical levels, we derive their samplers separately as follows:

$$p(z_{d,s} = t \mid Z_{\neg d,s}, R_{\neg d,s}, W_{\neg d,s}, U_{\neg d,s}) \propto \frac{B(n^O_{t,[\ ]} + \beta^O)}{B(n^O_{t,[\ ]_{\neg d,s}} + \beta^O)} \times \frac{B(n^{U,A}_{t,[]} + \beta^A)}{B(n^{U,A}_{t,[]_{\neg d,s}} + \beta^A)} \times \prod_{l=1}^{C} \frac{B(n^{S,A}_{t,l,[]} + \gamma)}{B(n^{S,A}_{t,l,[]_{\neg d,s}} + \gamma)} \times \frac{n^{Sent.}_{d,t\ \neg d,s} + \alpha}{n^{Sent.}_{d,(\bullet)\ \neg d,s} + T\alpha} \tag{3.1}$$

$$p(r_{d,s,j} = \hat{o} \mid Z_{\neg d,s}, R_{\neg d,s,j}, W_{\neg d,s,j}, U_{\neg d,s,j}, z_{d,s} = t, w_{d,s,j} = w) \propto \frac{n^O_{t,w_{\neg d,s,j}} + \beta^O}{n^O_{t,(\bullet)_{\neg d,s,j}} + |V \cup \bigcup_l Q_l| \beta^O} \times \frac{n^O_{d,s_{\neg d,s,j}} + \delta_b}{n^A_{d,s_{\neg d,s,j}} + \delta_a + n^O_{d,s_{\neg d,s,j}} + \delta_b} \tag{3.2}$$

$$p(r_{d,s,j} = \hat{a} \big| \dots) \propto \begin{cases} \frac{n^{S,A}_{t,l,w_{\neg d,s,j}} + \gamma}{n^{S,A}_{t,l,(\bullet)_{\neg d,s,j}} + |Q_l|\gamma} \times \frac{n^{\Omega}_{t,l} + \beta^A}{n^{\Omega}_{t,(\bullet)} + (V+C)\beta^A} \times \\ \frac{n^{A}_{d,s_{\neg d,s,j}} + \delta_b}{n^{A}_{d,s_{\neg d,s,j}} + \delta_a + n^{O}_{d,s_{\neg d,s,j}} + \delta_b} \quad ; \quad w \in Q_l \\ \\ \frac{n^{U,A}_{t,w} + \beta^A}{n^{U,A}_{t,(\bullet)} + (V+C)\ \beta^A} \times \frac{n^{A}_{d,s_{\neg d,s,j}} + \delta_b}{n^{A}_{d,s_{\neg d,s,j}} + \delta_a + n^{O}_{d,s_{\neg d,s,j}} + \delta_b} ; \nexists l, w \in Q_l \end{cases} \tag{3.3}$$

where $B(\vec{x}) = \frac{\prod_{i=1}^{\dim(\vec{x})} \Gamma(x_i)}{\Gamma(\sum_{i=1}^{\dim(\vec{x})} x_i)}$ is the multinomial Beta function. $n^{O}_{t,v}$ is the number of times term v was assigned to aspect t as an opinion/sentiment word. $n^{U,A}_{t,v}$ is the number of times non-seed term $v \in V$ was assigned to aspect t as an aspect. $n^{S,A}_{t,l,v}$ is the number of times seed term $v \in V_l$ was assigned to aspect t as an aspect. $n^{Sent.}_{d,t}$ is the number of sentences in document d that were assigned to aspect t. $n^{A}_{d,s}$ and $n^{O}_{d,s}$ denote the number of terms in $Sent^d_s$ that were assigned to aspects and opinions respectively. $n^{\Omega}_{t,l}$ is the number of times any term of seed set Q_l was assigned to aspect t. Omission of a latter index denoted by [] in the above notation represents the corresponding row vector spanning over the latter index. For example, $n^{U,A}_{t,[]} = [n^{U,A}_{t,v=1}, \dots, n^{U,A}_{t,v=V}]$ and $(\cdot)$ denotes the marginalized sum over the latter index. The subscript $\neg d, s$ denotes the counts excluding assignments of all terms in $Sent^d_s$. $\neg d, s, j$ denotes counts excluding $w_{d,s,j.}$ We perform hierarchical sampling. First, an aspect is sampled for each sentence $z_{d,s}$ using equation (3.1). After sampling the aspect, we sample $r_{d,s,j}$. The probability of $w_{d,s,j}$ being an opinion or sentiment term, $p(r_{d,s,j} = \hat{o})$ is given by equation (3.2). However, for $p(r_{d,s,j} = \hat{a})$ we have two cases: (a) the observed term $w = w_{d,s,j} \in Q_l$ or (b) does not belong to any seed set, $\nexists l, w \in Q_l$, i.e., *w* is an non-seed term. These cases are dealt in equation (3.3).

Asymmetric Beta priors: Hyper-parameters α, β^O, β^A are not very sensitive and the heuristic values suggested in [Griffiths and Steyvers, 2004] usually hold well in practice [Wallach et al. 2009]. However, the smoothing hyper-parameter δ (Figure 4(C)) is crucial as it governs the aspect or sentiment

switch. Essentially, $\psi_{d,s}$~$Beta(\delta\vec{\xi})$ is the probability of emitting an aspect term[2] in $Sent_s^d$ with concentration parameter δ and base measure $\vec{\xi} = [\xi_a, \xi_b]$. Without any prior belief, uniform base measures $\xi_a = \xi_b = 0.5$ are used resulting in symmetric Beta priors. However, aspects are often more probable than sentiments in a sentence (e.g., "*The beds, sheets, and bedding were dirty*."). Thus, it is more principled to employ asymmetric priors. Using a labeled set of sentences, $S_{labeled}$, where we know the per sentence probability of aspect emission ($\psi_{d,s}$), we can employ the method of moments to estimate the smoothing hyper-parameter $\delta = [\delta_a, \delta_b]$:

$$\delta_a = \mu\left(\frac{\mu(1-\mu)}{\sigma} - 1\right), \delta_b = \delta_a\left(\frac{1}{\mu} - 1\right);\ \mu = E[\psi_{d,s}], \sigma = Var[\psi_{d,s}] \tag{3.4}$$

3.1.2 ME-SAS Model

We can further improve SAS by employing Maximum Entropy (Max-Ent) priors for aspect and sentiment switching. We call this new model ME-SAS. The motivation is that aspect and sentiment terms play different syntactic roles in a sentence. Aspect terms tend to be nouns or noun phrases while sentiment terms tend to be adjectives, adverbs, etc. POS tag information can be elegantly encoded by moving $\psi_{d,s}$ to the term plate (see Figure 4 (D)) and drawing it from a Max-Ent($x_{d,s,j}; \lambda$) model. Let $\overrightarrow{x_{d,s,j}} = [POS_{w_{d,s,j}-1}, POS_{w_{d,s,j}}, POS_{w_{d,s,j}+1}, w_{d,s,j} - 1, w_{d,s,j}, w_{d,s,j} + 1]$ denote the feature vector associated with $w_{d,s,j}$ encoding lexical and POS features of the previous, current and next term. Using a training data set, we can learn Max-Ent priors. Note that unlike traditional Max-Ent training, we do not need manually labeled data for training (see Section 3.2 for details). For ME-SAS, only the sampler for the switch variable *r* changes as follows:

$$p(r_{d,s,j} = \hat{o} \big| Z_{\neg d,s}, R_{\neg d,s,j}, W_{\neg d,s,j}, U_{\neg d,s,j}, z_{d,s} = t, w_{d,s,j} = w) \propto$$

[2] $r_{d,s,j}$~ $Bernoulli(\psi_{d,s})$. $\psi_{d,s}$, $1 - \psi_{d,s}$ are the success and failure probability of emitting an aspect/sentiment term.

$$\frac{n^{O}_{t,w_{\neg d,s,j}}+\beta^{O}}{n^{O}_{t,(\bullet)_{\neg d,s,j}}+|V\cup\bigcup_l Q_l|\beta^{O}} \times \frac{exp(\sum_{i=1}^{n}\lambda_i f_i(x_{d,s,j},\hat{o}))}{\sum_{y\in\{\hat{a},\hat{o}\}} exp(\sum_{i=1}^{n}\lambda_i f_i(x_{d,s,j},y))} \tag{3.5}$$

$$p(r_{d,s,j}=\hat{a}\big|\ldots)\propto \begin{cases} \frac{n^{S,A}_{t,l,w_{\neg d,s,j}}+\gamma}{n^{S,A}_{t,l,(\bullet)_{\neg d,s,j}}+|Q_l|\gamma} \times \frac{n^{\Omega}_{t,l}+\beta^{A}}{n^{\Omega}_{t,(\bullet)}+(V+C)\beta^{A}} \times \\ \frac{exp(\sum_{i=1}^{n}\lambda_i f_i(x_{d,s,j},\hat{a}))}{\sum_{y\in\{\hat{a},\hat{o}\}} exp(\sum_{i=1}^{n}\lambda_i f_i(x_{d,s,j},y))} \quad ; \quad w\in Q_l \\ \frac{n^{U,A}_{t,w}+\beta^{A}}{n^{U,A}_{t,(\bullet)}+(V+C)\ \beta^{A}} \times \frac{exp(\sum_{i=1}^{n}\lambda_i f_i(x_{d,s,j},\hat{a}))}{\sum_{y\in\{\hat{a},\hat{o}\}} exp(\sum_{i=1}^{n}\lambda_i f_i(x_{d,s,j},y))}; \nexists l, w\in Q_l \end{cases} \tag{3.6}$$

where $\lambda_{1\ldots n}$ are the parameters of the learned Max-Ent model corresponding to the n binary feature functions $f_{1\ldots n}$ of Max-Ent.

3.2 Experimental evaluation

This section evaluates the proposed models. Since the focus in this paper is to generate high quality aspects using seeds, we will not evaluate sentiments although both SAS and ME-SAS can also discover sentiments. To compare the performance with our models, we use two existing state-of-the-art models, ME-LDA [Zhao et al. 2010] and DF-LDA [Andrzejewski et al., 2009]. There are two main flavors of aspect and sentiment models. The first flavor does not separate aspect and sentiment, and the second flavor uses a switch to perform the separation. Since our models also perform a switch, it is natural to compare with the latter flavor, which is also more advanced. ME-LDA is the representative model in the latter flavor. DF-LDA adds constraints to LDA. We use our seeds to generate constraints for DF-LDA. While ME-LDA cannot consider constraints, DF-LDA does not separate sentiments and aspects. Apart from other modeling differences, our models can do both, which enable them to produce much better results.

Dataset and Settings: We used hotel reviews from tripadvisor.com. Our corpus consisted of 101,234 reviews and 692,783 sentences. Punctuations, stop words [3], and words appearing less than 5 times in the corpus were removed.

For all models, the posterior inference was drawn after 5000 Gibbs iterations with an initial burn-in of 1000 iterations. For SAS and ME-SAS, we set $\alpha = 50/T$, $\beta^A = \beta^O = 0.1$ as suggested in [Griffiths and Steyvers, 2004]. To make the seeds more effective, we set the seed set word-distribution hyper-parameter γ to be much larger than β^A, the hyper-parameter for the distribution over seed sets and aspect terms. This results in higher weights to seeded words which in turn guide the sampler to cluster relevant terms better. A more theoretical approach would involve performing hyper-parameter estimation [Wallach et al., 2009] which may reveal specific properties of the dataset like the estimate of α (indicating how different documents are in terms of their latent semantics), β (suggesting how large the groups of frequently appearing aspect and sentiment terms are) and γ (giving a sense of which and how large groupings of seeds are good). These are interesting questions and we defer it to our future work. In this work, we found that the setting $\gamma = 250$, a larger value compared to β^A, produced good results.

For SAS, the asymmetric Beta priors were estimated using the method of moments (Section 3.1.1). We sampled 500 random sentences from the corpus and for each sentence identified the aspects. We thus computed the per-sentence probability of aspect emission ($\psi_{d,s}$) and used equation (3.4) to compute the final estimates, which give $\delta_a = 2.35$, $\delta_b = 3.44$.

To learn the Max-Ent parameters λ of ME-SAS, we used the sentiment lexicon [4] of [Hu and Liu, 2004] to automatically generate training data (no manual labeling). We randomly sampled 1000 terms from the corpus which have appeared at least 20 times (to ensure that the training set is reasonably representative of the corpus). Of those 1000 terms if they appeared in the sentiment lexicon, they were treated as sentiment terms, else aspect terms. Clearly, labeling words not in the sentiment lexicon as

[3] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

[4] http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar

aspect terms may not always be correct. Even with this noisy automatically-labeled data, the proposed models can produce good results. Since ME-LDA used manually labeled training data for Max-Ent, we again randomly sampled 1000 terms from our corpus appearing at least 20 times and labeled them as aspect terms or sentiment terms, so this labeled data clearly has less noise than our automatically labeled data. For both ME-SAS and ME-LDA we used the corresponding feature vector of each labeled term (in the context of sentences where it occurs) to train the Max-Ent model. As DF-LDA requires must-link and cannot-link constraints, we used our seed sets to generate intra-seed set must-link and inter-seed set cannot-link constraints. For its hyper-parameters, we used the default values in the package[5] [Andrzejewski et al., 2009].

Setting the number of topics/aspects in topic models is often tricky as it is difficult to know the exact number of topics that a corpus has. While non-parametric Bayesian approaches [Teh et al., 2006] do exist for estimating the number of topics, T, they strongly depend on the hyper-parameters [Heinrich, 2009]. As we use fixed hyper-parameters, we do not learn T from Bayesian non-parametrics. We used 9 major aspects ($T = 9$) based on commonsense knowledge of what people usually talk about hotels and some experiments. These are *Dining*, *Staff*, *Maintenance*, *Check In*, *Cleanliness*, *Comfort*, *Amenities*, *Location* and *Value for Money* (VFM). However, it is important to note that the proposed models are flexible and do not need to have seeds for every aspect/topic. Our experiments simulate the real-life situation where the user may not know all aspects or have no seeds for some aspects. Thus, we provided seeds only to the first 6 of the 9 aspects/topics. We will see that without seeds for all aspects, our models not only can improve the seeded aspects but also improve the non-seeded aspects.

[5] http://pages.cs.wisc.edu/~andrzeje/research/df_lda.html

Aspect (seeds)	ME-SAS		SAS		ME-LDA		DF-LDA
	Aspect	**Sentiment**	**Aspect**	**Sentiment**	**Aspect**	**Sentiment**	**Topic**
Staff (staff service waiter hospitality upkeep)	attendant manager waitress maintenance bartender waiters housekeeping receptionist waitstaff janitor	friendly attentive polite nice **clean** pleasant **slow** courteous rude professional	attendant waiter waitress manager maintenance **helpful** waiters housekeeping receptionist **polite**	friendly nice dirty **comfortable** nice **clean** polite **extremely** courteous efficient	staff maintenance **room** upkeep **linens** room- service receptionist **wait** **pillow** waiters	friendly nice courteous **extremely** nice **clean** polite **little** helpful **better**	staff ***friendly*** ***helpful*** **beds** **front** room **comfortable** **large** receptionist housekeeping
Cleanliness (curtains restroom floor beds cleanliness)	carpets hall towels bathtub couch mattress linens wardrobe spa pillow	clean dirty comfortable fresh wet filthy extra stain **front** worn	hall carpets towels pillow stain mattress **filthy** linens **interior** bathtub	clean dirty fresh old nice good **enough** new **front** **friendly**	cleanliness floor carpets bed **lobby** bathroom **staff** closet spa **décor**	clean good dirty **hot** **large** nice **fresh** **thin** new **little**	clean pool **beach** carpets **parking** bed bathroom **nice** comfortable suite
Comfort (comfort mattress furniture couch pillows)	bedding bedcover sofa linens bedroom suites **décor** comforter blanket futon	comfortable clean soft nice uncomfortable spacious hard comfy dirty quiet	bed linens sofa bedcover **hard** bedroom **privacy** **double** **comfy** futon	nice dirty comfortable large clean **best** spacious **only** big **extra**	bed mattress suites furniture **lighting** **décor** room bedroom **hallway** **carpet**	great clean awesome dirty best comfortable soft nice **only** **extra**	bed mattress **nice** ***stay*** **lighting** lobby comfort room **dirty** sofa

Table XI: Top ranked aspect and sentiment words in three aspects (see the explanation in Section 3.2.1).

3.2.1 Qualitative results

This section shows some qualitative results to give an intuitive feeling of the results from different models. Table XI shows the aspect terms and sentiment terms discovered by the 4 models for three aspects. Due to space limitations, we are unable to show all 6 aspects for which we have seeds. Since DF-LDA cannot separate aspects and sentiments, we only show its topics (aspects). **Red** (**bold**) colored words show semantic clustering errors or inappropriate terms for different groups.

It is important to note that we judge the results based on how they are related to the user seeds (which represent the user need). The judgment is to some extent subjective. What we reported here are based on our judgments what are appropriate and what are not for each aspect. For SAS, ME-SAS and ME-

LDA, we mark sentiment terms as errors when they are grouped under aspects as these models are supposed to separate sentiments and aspects. For DF-LDA, the situation is different as it is not meant to separate sentiment and aspect terms, we use ***red*** italic font to indicate those adjectives which are aspect specific adjectives (see more discussion below). Our judgment may be slightly unfair to ME-LDA and DF-LDA as their results may make sense in some other ways. However, that is precisely the purpose of this work, to produce results that suit the user's need rather than something generic.

We can see from Table XI that ME-SAS performs the best. Next in order are SAS, ME-LDA, and DF-LDA. We see that only providing a handful of seeds (5) for the aspect *Staff*, ME-SAS can discover highly specific words like manager, attendant, bartender, and janitor. By specific, we mean they are highly related to the given seeds. While SAS also discovers specific words benefiting from seeds, relying on Beta priors for aspect and sentiment switching was less effective. Next in performance is ME-LDA which although produces reasonable results in general, several aspect terms are far from what the user wants based on the seeds, e.g., room, linens, wait, pillow. Finally, we observe that DF-LDA does not perform well either. One reason is that it is unable to separate aspects and sentiments. Although encoding the intra-seed set must-link and inter-seed set cannot-link constraints in DF-LDA discovers some specific words as ME-SAS, they are much lower in the ranked order and hence do not show up in the top 10 words in Table XI. As DF-LDA is not meant to perform extraction and to group both aspect and sentiment terms, we relax the errors of DF-LDA due to correct aspect specific sentiments (e.g., friendly, helpful for *Staff* are correct aspect specific sentiments, but still regard incorrect sentiments like front, comfortable, large as errors) placed in aspect models. We call this model DF-LDA-Relaxed.

No. of Seeds	DF-LDA			DF-LDA-Relaxed			SAS			ME-SAS		
	P@10	P@20	P@30	P@10	P@20	P@30	P@10	P@20	P@30	P@10	P@20	P@30
2	0.51	0.53	0.49	0.67	0.69	0.70	0.69	0.71	0.67	0.74	0.72	0.70
3	0.53	0.54	0.50	0.71	0.70	0.71	0.71	0.72	0.70	0.78	0.75	0.72
4	0.57	0.56	0.53	0.73	0.73	0.73	0.75	0.74	0.73	0.83	0.79	0.76
5	0.59	0.57	0.54	0.75	0.74	0.75	0.77	0.76	0.74	0.86	0.81	0.77

Table XII: Average *p@n* of the seeded aspects with the no. of seeds.

Aspect	ME-LDA			DF-LDA			DF-LDA-Relaxed			SAS			ME-SAS		
	P@10	P@20	P@30	P@10	P@20	P@30	P@10	P@20	P@30	P@10	P@20	P@30	P@10	P@20	P@30
Dining	0.70	0.65	0.67	0.50	0.60	0.63	0.70	0.70	0.70	0.80	0.75	0.73	0.90	0.85	0.80
Staff	0.60	0.70	0.67	0.40	0.65	0.60	0.60	0.75	0.67	0.80	0.80	0.70	1.00	0.90	0.77
Maintenance	0.80	0.75	0.73	0.40	0.55	0.56	0.60	0.70	0.73	0.70	0.75	0.76	0.90	0.85	0.80
Check In	0.70	0.70	0.67	0.50	0.65	0.60	0.80	0.75	0.70	0.80	0.70	0.73	0.90	0.80	0.76
Cleanliness	0.70	0.75	0.67	0.70	0.70	0.63	0.70	0.75	0.70	0.80	0.75	0.70	1.00	0.85	0.83
Comfort	0.60	0.70	0.63	0.60	0.65	0.50	0.70	0.75	0.63	0.60	0.75	0.67	0.90	0.80	0.73
Amenities	0.80	0.80	0.67	0.70	0.65	0.53	0.90	0.75	0.73	0.90	0.80	0.70	1.00	0.85	0.73
Location	0.60	0.70	0.63	0.50	0.60	0.56	0.70	0.70	0.67	0.60	0.70	0.63	0.70	0.75	0.67
VFM	0.50	0.55	0.50	0.40	0.50	0.46	0.60	0.60	0.60	0.50	0.50	0.50	0.60	0.55	0.53
Avg.	0.67	0.70	0.65	0.52	0.62	0.56	0.70	0.72	0.68	0.72	0.72	0.68	0.88	0.80	0.74

Table XIII: Effect of performance on seeded and non-seeded aspects (5 seeds were used for the 6 seeded aspects).

3.2.2 Quantitative results

Topic models are often evaluated quantitatively using perplexity and likelihood on held-out test data [Blei et al., 2003]. However, perplexity does not reflect our purpose since our aim is not to predict whether an unseen document is likely to be a review of some particular aspect. Nor are we trying to evaluate how well the unseen review data fits our seeded models. Instead our focus is to evaluate how well our learned aspects perform in clustering specific terms guided by seeds. So we directly evaluate the discovered aspect terms. Note again we do not evaluate sentiment terms as they are not the focus of this paper [6]. Since aspects produced by the models are rankings and we do not know the number of correct aspect terms, a natural way to evaluate these rankings is to use *precision @ n* (or *p@n*), where *n* is a rank position.

Varying number of seeds: Instead of a fixed number of seeds, we want to see the effect of the number of seeds on aspect discovery. Table XII reports the average *p@n* vs. the number of seeds. The average is a two-way averaging. The first average was taken over all combinations of actual seeds selected for each aspect, e.g., when the number of seeds is 3, out of the 5 seeds in each aspect, all $\binom{5}{3}$ combinations of seeds were tried and the results averaged. The results were further averaged over *p@n* for 6 aspects with seeds. We start with 2 seeds and progressively increase them to 5. Using only 1 seed per seed set (or per aspect) has practically no effect because the top level distribution φ^A encodes which seed sets (and non-seed words) to include; the lower-level distribution Ω constrains the probabilities of the seed words to be correlated for each of the seed sets. Thus, having only one seed per seed set will result in sampling that single word whenever that seed set is chosen which will not have the effect of correlating seed words so as to pull other words based on co-occurrence with constrained seed words. From Table

[6] A qualitative evaluation of sentiment extraction based on Table XI yields the following order: ME-SAS, SAS, ME-LDA.

XII, we can see that for all models *p@n* progressively improves as the number of seeds increases. Again ME-SAS performs the best followed by SAS and DF-LDA.

Effect of seeds on non-seeded aspects: Here we compare all models aspect wise and see the results of seeded models SAS and ME-SAS on non-seeded aspects (Table XIII). Shaded cells in Table XIII give the *p@n* values for DF-LDA, DF-LDA-Relaxed, SAS, and ME-SAS on three non-seeded aspects (Amenities, Location, and VFM)[7].

We see that across all the first 6 aspects with (5) seeds ME-SAS outperforms all other models by large margins in all top 3 ranked buckets *p*@10, *p*@20 and *p*@30. Next in order are SAS, ME-LDA and DF-LDA. For the last three aspects which did not have any seed guidance, we find something interesting. Seeded models SAS and especially ME-SAS result in improvements of non-seeded aspects too. This is because as seeds facilitate clustering specific and appropriate terms in seeded aspects, which in turn improves precision on non-seeded aspects. This phenomenon can be clearly seen in Table XI. In aspect *Staff* of ME-LDA, we find *pillow* and *linens* being clustered. This is not a "flaw" of the model per se, but the point here is *pillow* and *linens* happen to co-occur many times with other words like *maintenance*, *staff*, and *upkeep* because "room-service" generally includes staff members coming and replacing linens and pillow covers. Although *pillow* and *linens* are related to *Staff*, strictly speaking they are semantically incorrect because they do not represent the very concept "Staff" based on the seeds (which reflect the user need). Presence of seed sets in SAS and ME-SAS result in pulling such words as linens and pillow (due to seeds like beds and cleanliness in the aspect *Cleanliness*) and ranking them higher in the aspect *Cleanliness* (see Table XI) where they make more sense than *Staff*. Lastly, we also note that the improvements in non-seeded aspects are more pronounced for ME-SAS than SAS as SAS encounters more switching errors which counters the improvement gained by seeds.

[7] Note that Tables XII and XIII are different runs of the model. The variations in the results are due to the random initialization of the Gibbs sampler.

CHAPTER 4

DETECTING GROUP OPINION SPAM

This chapter takes a departure from opinion mining and addresses the issue of filtering fake/spam opinions on the Web. As noted before in chapter 1, detecting opinion spam is an important precondition for credible opinion mining on the Web.

Nowadays, if one wants to buy a product, most probably, one will first read reviews of the product. If he/she finds that most reviews are positive, he/she is very likely to buy it. However, if most reviews are negative, he/she will almost certainly choose another product. Positive opinions can result in significant financial gains and fames for organizations and individuals. This, unfortunately, gives strong incentives for *opinion spamming*, which refers to human activities (e.g., writing fake reviews) that try to deliberately mislead readers by giving unfair reviews to some entities (e.g. products) in order to promote them or to damage their reputations. As more and more individuals and organizations are using reviews for their decision making, detecting such fake reviews becomes a pressing issue. The problem has been widely reported in the news[1].

There are prior works [Jindal and Liu, 2008; Jindal et al., 2010; Li et al., 2011; Lim et al., 2010; Ott et al., 2011; Wang et al'. 2011] on detecting fake reviews and individual fake reviewers or spammers. However, limited research has been done to detect fake reviewer (or spammer) groups, which we also call *spammer groups*. Group spamming refers to a group of reviewers writing fake reviews together to promote or to demote some target products. A spammer group can be highly damaging as it can take total control of the sentiment on a product because a group has many people to write fake reviews. Our

[1] http://www.nytimes.com/2012/01/27/technology/for-2-a-star-a-retailer-gets-5-star-reviews.html
http://www.nytimes.com/2011/08/20/technology/finding-fake-reviews-online.html

1 of 1 people found the following review helpful: ★★★★★ **Practically FREE music**, December 4, 2004 **This review is from: Audio Xtract (CD-ROM)** I can't believe for $10 (after rebate) I got a program that gets me free unlimited music. I was hoping it did half what was ….	2 of 2 people found the following review helpful: ★★★★★ **Like a tape recorder…**, December 8, 2004 **This review is from: Audio Xtract (CD-ROM)** This software really rocks. I can set the program to record music all day long and just let it go. I come home and my ….	★★★★★ **Wow, internet music! …**, December 4, 2004 **This review is from: Audio Xtract (CD-ROM)** I looked forever for a way to record internet music. My way took a long time and many steps (frustrtaing). Then I found Audio Xtract. With more than 3,000 songs downloaded in …
3 of 8 people found the following review helpful: ★★★★★ **Yes – it really works**, Decembe 4, 2004 **This review is from: Audio Xtract Pro (CD-ROM)** See my review for Audio Xtract - this PRO is even better. This is the solution I've been looking for. After buying iTunes, ….	3 of 10 people found the following review helpful: ★★★★★ **This is even better than…**, December 8, 2004 **This review is from: Audio Xtract Pro (CD-ROM)** Let me tell you, this has to be one of the coolest products ever on the market. Record 8 internet radio stations at once, ….	2 of 9 people found the following review helpful: ★★★★★ **Best music just got …**, December 4, 2004 **This review is from: Audio Xtract Pro (CD-ROM)** The other day I upgraded to this TOP NOTCH product. Everyone who loves music needs to get it from Internet ….
5 of 5 people found the following review helpful: ★★★★★ **My kids love it**, December 4, 2004 **This review is from: Pond Aquarium 3D Deluxe Edition** This was a bargain at $20 - better than the other ones that have no above water scenes. My kids get a kick out of the ….	5 of 5 people found the following review helpful: ★★★★★ **For the price you…**, December 8, 2004 **This review is from: Pond Aquarium 3D Deluxe Edition** This is one of the coolest screensavers I have ever seen, the fish move realistically, the environments look real, and the ….	3 of 3 people found the following review helpful: ★★★★★ **Cool, looks great…**, Decembe 4, 2004 **This review is from: Pond Aquarium 3D Deluxe Edition** We have this set up on the PC at home and it looks GREAT. The fish and the scenes are really neat. Friends and family ….

(A): Big John's Profile (B): Cletus' Profile (C): Jake's Profile

Figure 5: Profile of Amazon reviewers.

experiments show that it is hard to detect spammer groups using review content features [Ott et al., 2011] or even indicators for detecting abnormal behaviors of individual spammers [Lim et al., 2010] because a group has more manpower to post reviews and thus, each member may no longer appear to behave abnormally. Note that by a group of reviewers, we mean a set of reviewer-ids. The actual reviewers behind the ids could be a single person with multiple ids (sockpuppet), multiple persons, or a combination of both. We do not distinguish them in this work.

Before proceeding further, let us see a spammer group found by our algorithm. Figures 5 (A, B, C), show the reviews of a group of three reviewers[2]. The following suspicious patterns can be noted about this group: (i) the group members all reviewed the same three products giving all 5 star ratings; (ii) they posted reviews within a small time window of 4 days (two of them posted in the same day); (iii) each

[2] http://www.amazon.com/gp/pdp/profile/A3URRTIZEE8R7W
http://www.amazon.com/gp/pdp/profile/A254LYRIZUYXZG
http://www.amazon.com/gp/cdp/member-reviews/A1O70AIHNS4EIY

of them only reviewed the three products (when our Amazon review data [14] was crawled); (iv) they were among the early reviewers for the products (to make a big impact). All these patterns occurring together strongly suggest suspicious activities. Notice also, none of the reviews themselves are similar to each other (i.e., not duplicates) or appear deceptive. If we only look at the three reviewers individually, they all appear genuine. In fact, 5 out of 9 reviews received 100% helpfulness votes by Amazon users indicating that the reviews are useful. Clearly, these three reviewers have taken total control of the sentiment on the set of reviewed products.

If a group of reviewers work together only once to promote or to demote a product, it is hard to detect them based on their collective behavior. They may be detected using the content of their reviews, e.g., copying each other. Then, the methods in [Jindal and Liu, 2008; Li et al., 2011; Ott et al., 2011; Wang et al., 2011] are applicable. However, over the years, opinion spamming has become a business. People get paid to write fake reviews. Such people cannot just write a single review as they would not make enough money that way. Instead, they write many reviews about many products. Such collective behaviors of a group working together on a number of products can give them away. This chapter focuses on detecting such groups. Since reviewers in the group write reviews on multiple products, the data mining technique *frequent itemset mining* (FIM) [Agrawal and Srikant, 1994] can be used to find them. However, so discovered groups are only group spam candidates because many groups may be coincidental as some reviewers happen to review the same set of products due to similar tastes and popularity of the products (e.g., many people review all 3 Apple products, iPod, iPhone, and iPad). Thus, our focus is to identify true spammer groups from the candidate set.

One key difficulty for opinion spam detection is that it is very hard to manually label fake reviews or reviewers for model building because it is almost impossible to recognize spam by just reading each individual review [Jindal and Liu, 2008]. In this work, multiple experts were employed to create a labeled group opinion spammer dataset. This research makes the following main contributions:

1. It produces a labeled group spam dataset. To the best of our knowledge, this is the first such dataset. What was surprising and also encouraging to us was that unlike judging individual fake reviews or reviewers, judging fake reviewer groups were considerably easier due to the group context and their collective behaviors. We will discuss this in Section 4.1.
2. It proposes a novel relation-based approach to detecting spammer groups. With the labeled dataset, the traditional approach of supervised learning can be applied [Jindal and Liu, 2008; Li et al., 2011; Ott et al., 2011]. However, we show that this approach can be inferior due to the inherent nature of our particular problem:
   (i) Traditional learning assumes that individual instances are independent of one another. However, in our case, groups are clearly not independent of one another as different groups may share members. One consequence of this is that if a group *i* is found to be a spammer group, then the other groups that share members with group *i* are likely to be spammer groups too. The reverse may also hold.
   (ii) It is hard for features used to represent each group in learning to consider each individual member's behavior on each individual product, i.e., a group can conceal a lot of internal details. This results in severe information loss, and consequently low accuracy.

   We discuss these and other issues in greater detail in Section 4.5. To exploit the relationships of groups, individual members, and products they reviewed, a novel relation-based approach is proposed, which we call GSRank (Group Spam Rank), to rank candidate groups based on their likelihoods for being spam.
3. A comprehensive evaluation has been conducted to evaluate GSRank. Experimental results show that it outperforms many strong baselines including the state-of-the-art learning to rank, supervised classification and regression algorithms.

4.1 Building a reference dataset

As mentioned earlier, there was no labeled dataset for group opinion spam before this project. To evaluate our method, we built a labeled dataset using expert human judges.

Opinion spam and labeling viability: Castillo et al., [2006] argues that classifying the concept "spam" is difficult. Research on Web [Wu et al., 2006], email [Chirita et al., 2006], blogs [Kolari et al., 2006], and even social spam [Markines et al., 2009] all rely on manually labeled data for detection. Due to this inherent nature of the problems, the closest that one can get to gold standards is by creating a manually labeled dataset using human experts [Castillo et al., 2006; Koutrika et al., 2007; Li et al., 2011; Markines et al., 2009]. We too built a group opinion spam dataset using human experts.

Amazon dataset: In this research, we used product reviews from Amazon [Jindal and Liu, 2008], which have also been used in [Jindal et al., 2010; Lim et al., 2010]. The original crawl was done in 2006. Updates were made in early 2010. For our study, we only used reviews of manufactured products, which are comprised of 53,469 reviewers, 109,518 reviews and 39,392 products. Each review consisted of a title, content, star rating, posting date, and number of helpful feedbacks.

Mining candidate spammer groups: We use frequent itemset mining (FIM) here. In our context, a set of items, I is the set of all reviewer ids in our database. Each transaction t_i ($t_i \subseteq I$) is the set of reviewer ids who have reviewed a particular product. Thus, each product generates a transaction of reviewer ids. By mining frequent itemsets, we find groups of reviewers who have reviewed multiple products together. We found 7052 candidate groups with *minsup_c* (minimum support count) = 3 and at least 2 items (reviewer ids) per itemset (group), i.e., each group must have worked together on at least 3 products. Itemsets (groups) with support lower than this (*minsup_c* =1, 2) are very likely to be due to random chance rather than true correlation, and very low support also causes combinatorial explosion because the number of frequent itemsets grows exponentially for FIM [Agrawal and Srikant, 1994].

FIM working on reviewer ids can also find *sockpuppeted* ids forming groups whenever the ids are used *minsup_c* times to post reviews.

Opinion spam signals: We reviewed prior research on opinion spam and guidelines on consumer sites such as consumerist.com, lifehacker.com and consumersearch.com[3], and collected from these sources a list of spamming indicators or signals, e.g., (i) having zero caveats, (ii) full of empty adjectives, (iii) purely glowing praises with no downsides, (iv) being left within a short period of time of each other, etc. These signals were given to our judges. We believe that these signals (and the additional information described below) enhance their judging rather than bias them because judging spam reviews and reviewers is very challenging. It is hard for anyone to know a large number of possible signals without substantial prior experiences. These signals on the Web and research papers have been compiled by experts with extensive experiences and domain knowledge. We also reminded our judges that these signals should be used at their discretion and encouraged them to use their own signals.

To reduce the judges' workload further, for each group we also provided 4 additional pieces of information as they are required by some of the above signals: reviews with posting dates of each individual group member, list of products reviewed by each member, reviews of products given by non-group members, and whether group reviews were tagged with AVP (Amazon Verified Purchase). Amazon tags each review with AVP if the reviewer actually bought the product. Judges were also given access to our database for querying based on their needs.

Labeling: We employed 8 expert judges: employees of Rediff Shopping (4) and eBay.in (4) for labeling our candidate groups. The judges had domain expertise in feedbacks and reviews of products due to their nature of work in online shopping. Since there were too many patterns (or candidate groups), our judges could only manage to label 2431 of them as being "spam", "non-spam", or "borderline". The judges were made to work in isolation to prevent any bias. The labeling took around 8 weeks. We did

[3] http://consumerist.com/2010/04/how-you-spot-fake-online-reviews.html
http://lifehacker.com/5511726/hone-your-eye-for-fake-online-reviews
http://www.consumersearch.com/blog/how-to-spot-fake-user-reviews

not use Amazon Mechanical Turk (MTurk) for this labeling task because MTurk is normally used to perform simple tasks which require human judgments. However, our task is highly challenging, time consuming, and also required the access to our database. Also, we needed judges with good knowledge of the review domain. Thus, we believe that MTurk was not suitable.

4.2 Labeling results

We now report the labeling results and analyze the agreements among the judges.

Spamicity: We calculated the "spamicity" (degree of spam) of each group by assigning 1 point for each *spam* judgment, 0.5 point for each *borderline* judgment and 0 point for each *non-spam* judgment a group received and took the average of all 8 labelers. We call this average the *spamicity* score of the group. Based on the spamicities, the groups can be ranked. In our evaluation, we will evaluate the proposed method to see whether it can rank similarly. In practice, one can also use a spamicity threshold to divide the candidate group set into two classes: *spam* and *non-spam* groups. Then supervised classification is applicable. We will discuss these in detail in the experiment section.

Agreement study: Previous studies have showed that labeling individual fake reviews and reviewers is hard [Jindal and Liu, 2008]. To study the feasibility of labeling groups and also the judging quality, we used Fleiss' multi-rater kappa [Fleiss, 1971] to measure the judges' agreements.

We obtained $\kappa = 0.79$ which indicates close to perfect agreement based on the scale[4] in [Landis and Koch, 1977]. This was very encouraging and also surprising, considering that judging opinion spam in general is hard. It tells us that labeling groups seems to be much easier than labeling individual fake reviews or reviewers. We believe the reason is that unlike a single reviewer or review, a group gives a good context for judging and comparison, and similar behaviors among members often reveal strong signals. This was confirmed by our judges who had domain expertise in reviews.

[4] No agreement ($\kappa<0$), slight agreement ($0 < \kappa \leq 0.2$), fair agreement ($0.2 < \kappa \leq 0.4$), moderate agreement ($0.4 < \kappa \leq 0.6$), substantial agreement ($0.6 < \kappa \leq 0.8$), and almost perfect agreement for $0.8 < \kappa \leq 1.0$.

4.3 Opinion spamming behavioral features

For modeling or learning, a set of effective spam indicators or features is needed. This section proposes two sets of such indicators or behaviors which may indicate spamming activities.

4.3.1 Group spam behavior indicators

Here we discuss group behaviors that may indicate spam.

1. ***Group Time Window* (*GTW*)**: Members in a spam group are likely to have worked together in posting reviews for the target products during a short time interval. We model the degree of active involvement of a group as its *group time window* (*GTW*):

$$GTW(g) = \max_{p \in P_g}(GTW_P(g,p)),$$

$$GTW_P(g,p) = \begin{cases} 0 & \text{if } L(g,p) - F(g,p) > \tau \\ 1 - \dfrac{L(g,p) - F(g,p)}{\tau} & otherwise \end{cases}, \tag{4.1}$$

where $L(g, p)$ and $F(g, p)$ are the latest and earliest dates of reviews posted for product $p \in P_g$ by reviewers of group g respectively. P_g is the set of all products reviewed by group g. Thus, $GTW_P(g, p)$ gives the time window information of group g on a single product p. This definition says that a group g of reviewers posting reviews on a product p within a short burst of time is more prone to be spamming (attaining a value close to 1). Groups working over a longer time interval than τ, get a value of 0 as they are unlikely to have worked together. τ is a parameter, which we will estimate later. The group *time window* $GTW(g)$ considers all products reviewed by the group taking *max* over p ($\in P_g$) so as to capture the worst behavior of the group. For subsequent behaviors, *max* is taken for the same reason.

2. ***Group Deviation* (*GD*)**: A highly damaging group spam occurs when the ratings of the group members deviate a great deal from those of other (genuine) reviewers so as to change the sentiment on a product. The larger the deviation, the worse the group is. This behavior is modeled by *group deviation* (*GD*) on a 5-star rating scale (with 4 being the maximum possible deviation):

$$GD(g) = \max_{p \in P_g}(D(g,p)),$$

$$D(g,p) = \frac{|r_{p,g} - \bar{r}_{p,g}|}{4}, \tag{4.2}$$

where $r_{p,g}$ and $\bar{r}_{p,g}$ are the average ratings for product *p* given by members of group *g* and by other reviewers not in *g* respectively. *D*(*g*, *p*) is the deviation of the group on a single product *p*. If there are no other reviewers who have reviewed the product *p*, $\bar{r}_{p,g} = 0$.

3. ***Group Content Similarity* (*GCS*)**: Group connivance is also exhibited by content similarity (duplicate or near duplicate reviews) when spammers copy reviews among themselves. So, the victimized products have many reviews with similar content. *Group content similarity* (*GCS*) models this behavior:

$$GCS(g) = \max_{p \in P_g}(CS_G(g,p)),$$

$$CS_G(g,p) = \underset{m_i, m_j \in g, i<j}{avg}\left(cosine(c(m_i,p), c(m_j,p))\right), \tag{4.3}$$

where $c(m_i, p)$ is the content of the review written by group member $m_i \in g$ for product *p*. $CS_G(g, p)$ captures the average pairwise similarity of review contents among group members for a product *p* by computing the cosine similarity.

4. ***Group Member Content Similarity* (*GMCS*)**: Another flavor of content similarity is exhibited when the members of a group *g* do not know one another (and are contacted by a contracting agency). Since writing a new review every time is taxing, a group member may copy or modify his/her own previous reviews for similar products. If multiple members of the group do this, the group is more likely to be spamming. This behavior can be expressed by *group member content similarity* (*GMCS*) as follows:

$$GMCS(g) = \frac{\sum_{m \in g} CS_M(g,m)}{|g|},$$

$$CS_M(g,m) = \underset{p_i, p_j \in P_g, i<j}{avg} \left(cosine(c(m,p_i), c(m,p_j)) \right) \tag{4.4}$$

The group attains a value ≈1 (indicating spam) on *GMCS* when all its members entirely copied their own reviews across different products in P_g. $CS_M(g, m)$ models the average pairwise content similarity of member $m \in g$ over all products in P_g .

5. **Group *Early Time Frame* (*GETF*)**: Lim et al., [2010] reports spammers usually review early to make the biggest impact. Similarly, when group members are among the very first people to review a product, they can totally hijack the sentiments on the products. The *group early time frame* (*GETF*) models this behavior:

$$GETF(g) = \max_{p \in P_g}(GTF(g,p)),\ GTF(g,p) = \begin{cases} 0 & \text{if } L(g,p) - A(p) > \beta \\ 1 - \dfrac{L(g,p) - A(p)}{\beta} & \text{otherwise} \end{cases}, \tag{4.5}$$

where $GTF(g, p)$ captures the time frame as how early a group g reviews a product p. $L(g, p)$ and $A(p)$ are the latest date of review posted for product $p \in P_g$ by group g and the date when p was made available for reviewing respectively. β is a threshold (say 6 months, later estimated) which means that after β months, *GTF* attains a value of 0 as reviews posted then are not considered to be early any more. Since our experimental dataset [Jindal and Liu, 2008] does not have the exact date when each product was launched, we use the date of the first review of the product as the value for $A(p)$.

6. ***Group Size Ratio* (*GSR*)**: The ratio of group size to the total number of reviewers for a product can also indicate spamming. At one extreme (worst case), the group members are the only reviewers of the product completely controlling the sentiment on the product. On the other hand, if the total number of reviewers of the product is very large, then the impact of the group is small.

$$GSR(g) = \underset{p \in P_g}{avg}(GSR_p(g,p)), \quad GSR_p(g,p) = \frac{|g|}{|M_p|}, \tag{4.6}$$

where $GSR_P(g, p)$ is the ratio of group size to M_p (the set of all reviewers of product p) for product p.

7. ***Group Size* (*GS*)**: Group collusion is also exhibited by its size. For large groups, the probability of members happening to be together by chance is small. Furthermore, the larger the group, the more damaging it is. *GS* is easy to model. We normalize it to [0, 1]. max($|g_i|$) is the largest group size of all discovered groups.

$$GS(g) = \frac{|g|}{\max(|g_i|)} \tag{4.7}$$

8. ***Group Support Count* (*GSUP*)**: Support count of a group is the total number of products towards which the group has worked together. Groups with high support counts are more likely to be spam groups as the probability of a group of random people happen to have reviewed many products together is small. *GSUP* is modeled as follows. We normalize it to [0, 1], with max($|P_{g_i}$) being the largest support count of all discovered groups:

$$GSUP(g) = \frac{|P_g|}{\max(|P_{g_i}|)} \tag{4.8}$$

These eight group behaviors can be seen as group spamming features for learning. From here on, we refer the 8 group behaviors as $f_1 \ldots f_8$ when used in the context of features.

It is important to note that by no means do we say that whenever a group attains a feature $f > 0$ or a threshold value, it is a spam group. It is possible that a group of reviewers due to similar tastes coincidently review some similar products (and form a coincidental group) in some short time frame, or may generate some deviation of ratings from the rest, or may even have modified some of the contents of their previous reviews to update their reviews producing similar reviews. The features just indicate the extent those group behaviors were exhibited. The final prediction of groups is done based on the learned models. As we will see in Section 4.4, all features $f_1 \ldots f_8$ are strongly correlated with spam groups and feature values attained by spam groups exceed those attained by other non-spam groups by a large margin.

4.3.2 Individual spam behavior indicators

Although group behaviors are important, they hide a lot of details about its members. Clearly, individual members' behaviors also give signals for group spamming. We now present the behaviors for individual members used in this work.

1. ***Individual Rating Deviation* (*IRD*)**: Like group deviation, we can model *IRD* as

$$IRD(m,p) = \frac{|r_{p,m} - \bar{r}_{p,m}|}{4}, \tag{4.9}$$

where $r_{p,m}$ and $\bar{r}_{p,m}$ are the rating for product *p* given by reviewer *m* and the average rating for *p* given by other reviewers respectively.

2. ***Individual Content Similarity* (*ICS*)**: Individual spammers may review a product multiple times posting duplicate or near duplicate reviews to increase the product popularity [Lim et al., 2010]. Similar to *GMCS*, we model *ICS* of a reviewer *m* across all its reviews towards a product *p* as follows:

$$ICS(m,p) = avg(cosine(c(m,p)) \tag{4.10}$$

The average is taken over all reviews on *p* posted by *m*.

3. ***Individual Early Time Frame* (*IETF*)**: Like *GETF*, we define *IETF* of a group member *m* as:

$$IETF(m,p) = \begin{cases} 0 & \text{if } L(m,p) - A(p) > \beta \\ 1 - \dfrac{L(m,p) - A(p)}{\beta} & \text{otherwise} \end{cases}, \tag{4.11}$$

where $L(m,p)$ denotes the latest date of review posted for a product *p* by member *m*.

4. ***Individual Member Coupling in a group* (*IMC*)**: This behavior measures how closely a member works with the other members of the group. If a member *m* almost posts at the same date as other group members, then *m* is said to be tightly coupled with the group. However, if *m* posts at a date that is far away from the posting dates of the other members, then *m* is not tightly coupled with the group. We find the difference between the posting date of member *m* for product *p* and the average posting date

of other members of the group for p. To compute time, we use the time when the first review was posted by the group for product p as the baseline. Individual member coupling (*IMC*) is thus modeled as:

$$IMC(g,m) = \underset{p \in P_g}{avg}\left(\frac{|(T(m,p) - F(g,p)) - avg(g,m)|}{L(g,p) - F(g,p)} \right), \quad avg(g,m) = \frac{\sum_{m_i \in G - \{m\}} (T(m_i,p) - F(g,p))}{|g| - 1}, \tag{4.12}$$

where $L(g, p)$ and $F(g, p)$ are the latest and earliest dates of reviews posted for product $p \in P_g$ by group g respectively, and $T(m, p)$ is the actual posting date of reviewer m on product p.

Note that IP addresses of reviewers may also be of use for group spam detection. However, IP information is privately held by proprietary firms and not publicly available. We believe if IP addresses are also available, additional features may be added, which will make our proposed approach even more accurate.

4.4 Empirical analysis

To ensure that the proposed behavioral features are good indicators of group spamming, this section analyzes them by statistically validating their correlation with group spam. For this study, we used the classification setting for spam detection. A spamicity threshold of 0.5 was employed to divide all candidate groups into two categories, i.e., those with spamicity greater than 0.5 as *spam* groups and others as *non-spam* groups. Using this scheme, we get 62% non-spam groups and 38% spam groups. In Section 4.7, we will see that these features work well in general (rather than just for this particular threshold). Note that the individual spam indicators in Section 4.5 are not analyzed as there is no suitable labeled data for that. However, these indicators are similar to their group counterparts and are thus indirectly validated through the group indicators. They also helped GSRank well (Section 4.7).

4.4.1 Statistical validation

For a given feature f, its effectiveness ($Eff(\cdot)$) is defined with:

$$Eff(f) \equiv P(f > 0 \mid Spam) - P(f > 0 \mid Non - spam), \tag{4.13}$$

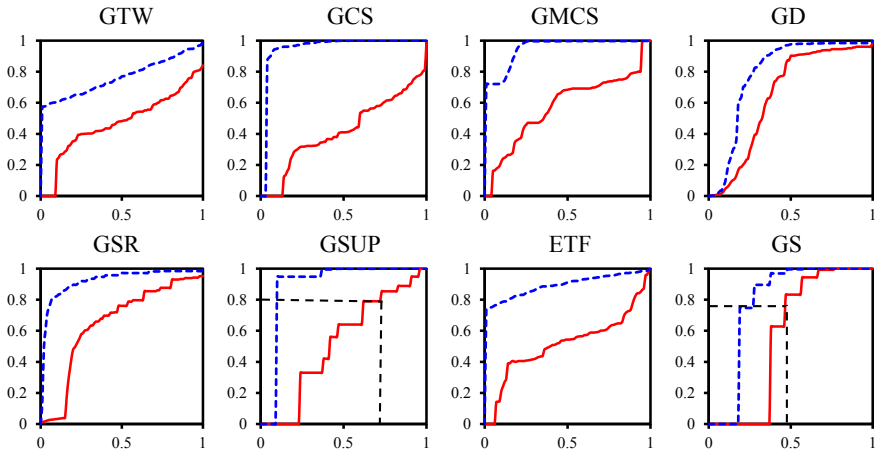


Figure 6: Behavioral Distribution. Cumulative % of spam (solid) and non-spam (dashed) groups vs. feature value.

where $f > 0$ is the event that the corresponding behavior is exhibited to some extent. Let the *null hypothesis* be: both spam and normal groups are equally likely to exhibit *f*, and the *alternate hypothesis*: spam groups are more likely to exhibit *f* than non-spam groups and are correlated with *f*. Thus, demonstrating that *f* is observed among spam groups and is correlated is reduced to show that $Eff(f) > 0$. We estimate the probabilities as follows:

$$P(f > 0 \mid Spam) = \frac{|\{g \mid f(g) > 0 \wedge g \in Spam\}|}{|\{g \mid g \in Spam\}|} \tag{4.14}$$

$$P(f > 0 \mid Non-spam) = \frac{|\{g \mid f(g) > 0 \wedge g \in Non-spam\}|}{|\{g \mid g \in Non-spam\}|} \tag{4.15}$$

We use Fisher's exact test to test the hypothesis. The test rejects the *null hypothesis* with $p<0.0001$ for each of the modeled behaviors. This shows that spam groups are indeed characterized by the modeled behaviors. Furthermore, since the modeled behaviors are all *anomalous*, and Fisher's exact test verifies strong correlation of those behaviors with groups labeled as "spam", it also indirectly gives us a strong confidence that the majority of the class labels in the reference dataset are trustworthy.

4.4.2 Behavioral distribution

We now analyze the underlying distribution of spam and non-spam groups across each behavioral feature dimension. Figure 6 shows the cumulative behavioral distribution (CBD). Against each value x attained by a feature f ($0 \leq x \leq 1$ as $f \in [0, 1]\ \forall f$), we plot the cumulative percentage of spam/non-spam groups having values of $f \leq x$. We note the following insights from the plots:

Position: CBD curves of non-spam groups lean more towards the left boundary of the graph than those for spam groups across all features. This implies that for a given cumulative percentage *cp*, the corresponding feature value x_n for non-spam groups is less than x_s for spam groups. For example, in CBD of the *GS* feature, if $cp = 0.75$, then 75% of the non-spam groups are bounded by $x_n = 0.18$ (i.e. $0.18 \times 11 \approx 2$ members)[5] while 75% of the spam groups are bounded by $x_s = 0.46$ (i.e. $0.46 \times 11 \approx 5$ members). As another example, we take CBD of *GSUP* with $cp = 0.8$. We see that 80% of the non-spam groups are bounded by $x_n = 0.15$ (i.e. $0.15 \times 13 \approx 2$ products)[6] while 80% of spam groups are bounded by $x_s = 0.76$ (i.e. $0.76 \times 13 \approx 10$ products). This shows that spamming groups usually work with more members and review more products. As non-spam groups are mostly coincidental, we find that their feature values remain low for most groups indicating benign behaviors. Also, we emphasize the term "bounded by" in the above description. By no means do we claim that every spam group in our database reviewed at least 10 products and is comprised of at least 5 members. Lastly, since $x_n < x_s\ \forall$ *cp* (due to the leftward positioning of the CBD curve for non-spam groups), spam groups obtain higher feature values than non-spam groups for each modeled behavior *f*.

Steep initial jumps: These indicate that very few groups obtain significant feature values before jump abscissa. For example, we find that there are very few spam groups with *GSUP* < 0.25, and for *GCS*,

[5] Dataset in Sec. 3 of all candidate groups with *minsup_c* =3, yielded $\max\{|g_i|\} = 11$ and max support = 13. We multiply feature values of *GS* and *GSUP* by these numbers to get the actual counts. See equations (4.7) and (4.8) for details.

we find a majority (≈ 90%) of non-spam groups in our database have minuscule content similarity[6] among their reviews.

Gaps: CBD curves of non-spam groups are higher than those of spam groups and the gap (separation margin) refers to the relative discriminative potency. *GCS* has the maximum gap and next in order are *GSUP* and *GETF*. This result is not surprising as a group of people having a lot of content similarity in their reviews is highly suspicious of being spamming and hence *GCS* has good discriminative strength.

4.5 Modeling relations

With the 8 group behavioral features separating spam and non-spam groups by a large margin and the labeled data from Section 4.1, the classic approach to detect spammer groups is to employ a supervised classification, regression, or learning to rank algorithm to classify or rank candidate groups. All these existing methods are based on a set of features to represent each instance (group in our case). However, as we indicated in the introduction section, this feature-based approach has some shortcomings for our task:

1. They assume that training and testing instances are drawn independently and identically (iid) from some distribution. However, in our case, different groups (instances) can share members and may review some common products. Thus, our data does not follow the iid assumption because many instances are related, i.e., apart from group features, the spamicity of a group is also affected by the other groups sharing its members, the spamicity of the shared members, the extent to which the reviewed products are spammed, etc.
2. Group features ($f_1 \ldots f_8$) only summarize (e.g., by max/avg) group behaviors. This clearly leads to loss of information because spamicity contributions from members are not considered at each individual member level, but are summarized (max/avg) to represent the whole group behavior.

[6] Computed using LingPipe Java API available at http://alias-i.com/lingpipe

Due to different group sizes and complex relationships, it is not easy to design and include each individual member related features explicitly without some kind of summary.

3. It is also difficult to design features which can consider the extent to which each product is spammed by groups. Although our focus is on detecting spammer groups, the underlying products being reviewed are clearly related.

Below, we propose a more effective model which can consider the inter-relationship among products, groups, and group members in computing group spamicity. Specifically, we model three binary relations: Group Spam–Products, Member Spam–Products, and Group Spam–Member Spam. The overall idea is as follows:

We first model the three binary relations to account for how each entity affects the other. We then draw inference of one entity from the other entity based on the corresponding binary relation. For example, using the Group Spam–Member Spam relation, we infer the spamicity of a group based on the spamicity of its individual members and vice-versa. Our ranking method called GSRank (Group Spam Rank) is then presented to tie all these inferences, which solves an eigenvalue problem by aligning the group vector to the dominant eigenvector. Before going to the details, we first define some notations used in the following sub-sections.

Let $P = \{p_i\}$, $G = \{g_j\}$, and $M = \{m_k\}$ be the set of all products, groups and members. Let $s(g_j)$ and $s(m_k)$ be the "spamicity" of g_j and m_k graded over [0, 1] respectively, and let $s(p_i)$ be the "extent to which p_i is spammed" also graded over [0, 1]. Values close to 1 signify high spamicity for groups and members and greater extent to which products are spammed. Additionally, let $V_P = [s(p_i)]_{|P|}$, $V_G = [s(g_j)]_{|G|}$, and $V_M = [s(m_k)]_{|M|}$ be the corresponding product, group and member score vectors.

4.5.1 Group Spam–Product model

This model captures the relation among groups and products they target. The extent a product p is spammed by various groups is related to: (i) spam contribution to p by each group reviewing p and (ii)

"spamicity" of each such group. Also, spam contribution by a group with high spamicity counts more. Similarly, the spamicity of a group is associated with (i) its own spam contribution to various products and (ii) the extent those products were spammed. To express this relation, we first compute the spam contribution to a product p_i by a group g_j. From Section 4.3, we have GTW_P (time window of group g's activity over a product p), D (g's deviation of ratings for p), GTF (early time frame of g's spam infliction towards p), CS_G (g's content similarity of reviews on p) and GSR_P (ratio of group's size for p). We note that these behaviors are "symmetric" in the sense that higher values indicate that g's behavior on p is suspicious, and also indicate that spam contribution to p by g is high. Thus, the spam contribution by g_j to p_i can be expressed by the following function:

$$w_1(p_i, g_j) = \frac{1}{5}[GTW_P(g_j, p_i) + D(g_j, p_i) + GTF(g_j, p_i) + CS_G(g_j, p_i) + GSR_P(g_j, p_i)],$$

$$W_{PG} = [w_1(p_i, g_j)]_{|P| \times |G|} \tag{4.16}$$

$w_1(p_i, g_j) = 0$ when g_j did not review p_i. The sum captures the spam inflicted across various spamming dimensions and is normalized by 5 so that $w_1 \in [0, 1]$. For subsequent contribution functions too, summation and normalization are used for the same reason. W_{PG} denotes the corresponding contribution matrix.

Using (4.16), (4.17) computes the extent p_i is spammed by various groups. It sums the spam contribution by each group, $w_1(p_i, g_j)$, and weights it by the spamicity of that group, $s(g_j)$. Similarly, (4.18) updates the group's spamicity by summing its spam contribution on all products weighted by the extent those products were spammed. The relations can also be expressed as matrix equations.

$$s(p_i) = \sum_{j=1}^{|G|} w_1(p_i, g_j) s(g_j); \quad V_P = W_{PG} V_G, \tag{4.17}$$

$$s(g_j) = \sum_{i=1}^{|P|} w_1(p_i, g_j) s(p_i); \quad V_G = W_{PG}^T V_P \tag{4.18}$$

Since $s(g_j) \propto$ "spamicity of g_j", $s(p_i) \propto$ "extent to which p_i was spammed", and $w_1 \propto$ "degree of spam inflicted by g_j towards p_i", (4.17) and (4.18) employ a summation to compute $s(g_j)$ and $s(p_i)$. Further, as spam contribution by a group with higher "spamicity" is more damaging, the degree of spam contribution by a group is weighted by its spamicity in (4.17). Similarly, for (4.18), spam contribution, w_1 is weighted by $s(p_i)$ to account for the effective spamicity of the group. For subsequent models too, weighted summation is used for similar reasons. The matrix equation (4.17) also shows how the product vector can be inferred from the group vector using matrix W_{PG} and vice-versa using (4.18).

4.5.2 Member Spam–Product model

Spam by a group on a product is basically spam by individuals in the group. A group feature can only summarize spam of members in the group over the set of products they reviewed. Here we consider spam contributions of all group members exclusively. Like w_1, we employ $w_2 \in [0, 1]$ to compute the spam contribution by a member m_k towards product p_i. We model w_2 as follows:

$$w_2(m_k, p_i) = \frac{1}{3}[IRD(m_k, p_i) + ICS(m_k, p_i) + IETF(m_k, p_i)],$$

$$W_{MP} = [w_2(m_k, p_i)]_{|M| \times |P|} \tag{4.19}$$

$w_2(m_k, p_i) = 0$ if m_k did not review p_i. Similar to (36), w_2 captures individual member spam contribution over the spam dimensions: *IRD* (individual rating deviation of m towards p), *ICS* (individual content similarity of reviews on p by m), and *IETF* (individual early time frame of spam infliction by m towards p). Similar to (38), we compute the spamicity of m_k by summing its spam contributions towards various products, w_2 weighted by $s(p_i)$ (4.20). And like (4.17), we update p_i to reflect the extent it was spammed by members. We sum the individual contribution of each member w_2, weighted by its spamicity (4.21).

$$s(m_k) = \sum_{i=1}^{|P|} w_2(m_k, p_i) s(p_i); \quad V_M = W_{MP} V_P \tag{4.20}$$

$$s(p_i) = \sum_{k=1}^{|M|} w_2(m_k, p_i) s(m_k); \;\; V_P = W_{MP}^T V_M \tag{4.21}$$

4.5.3 Group Spam–Member Spam model

Clearly, the spamicity of a group is related to the spamicity of its members and vice-versa. If a group consists of members with high spamicities, then the group's spam infliction is likely to be high. Similarly, a member involved in spam groups of high spamicity affects its own spamicity. We first compute the contribution of a member *m* ($\in$ *g*) towards a group *g*. From Section 4.3, we see that the contribution is captured by *IMC* (degree of *m*'s coupling in *g*), *GS* (size of *g* with which *m* worked), and *GSUP* (number of products towards which *m* worked with *g*). We model it as follows:

$$w_3(g_j, m_k) = \frac{1}{3}[IMC(g_j, m_k) + (1 - GS(g_j)) + GSUP(g_j)],$$

$$W_{GM} = [w_3(g_j, m_k)]_{|G| \times |M|} \tag{4.22}$$

$w_3(g_j, m_k) = 0$ when $m_k \notin g_j$. As *GS* is normalized over [0, 1], for large groups, the individual contribution of a member diminishes. Hence we use 1-*GS*(g_j) to compute w_3.

Using (4.22), (4.23) computes the spamicity of a group by summing up the spamicities of all its members, $s(m_k)$; each weighted by his contribution to the group, $w_3(g_j, m_k)$. Since groups can share members, (4.24) updates the spamicity of a member by summing up the spamicities of all groups it worked with, each weighted by its own contribution to that group.

$$s(g_j) = \sum_{k=1}^{|M|} w_3(g_j, m_k) s(m_k); \;\; V_G = W_{GM} V_M, \tag{4.23}$$

$$s(m_k) = \sum_{j=1}^{|G|} w_3(g_j, m_k) s(g_j); \;\; V_M = W_{GM}^T V_G. \tag{4.24}$$

Algorithm 1: GSRank

Input: Weight matrices W_{PG}, W_{MP}, and W_{GM}

Output: Ranked list of candidate spam groups

1. Initialize $V_G^0 \leftarrow [0.5]_{|G|}$; $t \leftarrow 1$;
2. **Iterate:**
   - i. $V_P \leftarrow W_{PG}\, V_G^{(t-1)}$; $V_M \leftarrow W_{MP}\, V_P$;
   - ii. $V_G \leftarrow W_{GM}\, V_M$; $V_M \leftarrow W_{GM}^T\, V_G$;
   - iii. $V_P \leftarrow W_{MP}^T\, V_M$; $V_G^{(t)} \leftarrow W_{PG}^T\, V_P$;
   - iv. $V_G^{(t)} \leftarrow V_G^{(t)} / \| V_G^{(t)} \|_1$;

until $\| V_G^{(t)} - V_G^{(t-1)} \|_\infty < \delta$

3. Output the ranked list of groups in descending order of V_G*

4.6 GSRank: Ranking Group Spam

Using the relation models, each entity is inferred twice, once from each other entity. As the two inferences for each entity are conditioned on other two entities, they are thus complementary. For example, V_G is inferred once from V_P (4.18) and then from V_M (4.23). Both of these inferences complement each other because group spamicity is related to both its collective spam activities on products and also the spamicities of its members. Since the relations are circularly defined, to effectively rank the groups, GSRank uses the iterative algorithm detailed above.

In line 1, we first initialize all groups with spamicity of 0.5 over the spamicity scale [0, 1]. Next, we infer V_P from the current value of V_G; and then infer V_M from the so updated V_P (line 2-i). This completes the initial bootstrapping of vectors V_G, V_P, and V_M for the current iteration. Line 2-ii then draws inferences based on the Group Spam–Member Spam model. It first infers V_G from V_M because V_M contains the recent update from line 2-i and then infers V_M from so updated V_G. This ordering is used to guide the inference procedure across the iterations. Line 2-iii then updates V_P based on the Member

Spam–Product model first, and defers the inference of V_G from so updated V_P based on the Group Spam–Product model until the last update so that V_G gets the most updated value for the next iteration. Finally, line 2-iv performs normalization to maintain an invariant state (discussed later). Thus, as the iterations progress, the fact that each entity affects the other is taken into account as the score vectors V_G, V_M, and V_P are updated via the inferences drawn from each relation model. The iterations progress until V_G converges to the stable V_G*. Since V_G contains the spamicity scores of all groups, line 3 outputs the final ordering of spam groups according to the spamicity scores of the stable vector V_G*. GSRank has robust theoretical guarantees. It can be shown that it is an instance of an eigenvalue problem and converges. For proof details, please refer to [Mukherjee et al., 2012].

4.7 Experimental evaluation

We now evaluate the proposed GSRank method. We use the 2431 groups described in Section 4.1. We first split 2431 groups into the development set, D with 431 groups (randomly sampled) for parameter estimation and the validation set, V with 2000 groups for evaluation. All evaluation metrics are averaged over 10-fold cross validation (CV) on V. Below we first describe parameter estimation and then ranking and classification experiments.

4.7.1 Parameter estimation

Our proposed behavioral model has two parameters τ and β, which have to be estimated. τ is the parameter of *GTW*, i.e., the time interval beyond which members in a group are not likely to be working in collusion. β is the parameter of *GETF* which denotes the time interval beyond which reviews posted are not considered to be "early" anymore (Section 4.3.1). For this estimation, we again use the classification setting. The estimated parameters actually work well in general as we will see in the next two subsections.

Let $\boldsymbol{\theta}$ denote the two parameters. We learn $\boldsymbol{\theta}$ using a greedy hill climbing search to maximize the log likelihood of the set D:

$$\vec{\theta}_{estimated} = \arg\max_{\vec{\theta}} \sum_{j} \log P(g_j = spam \mid \vec{\theta}) \tag{4.25}$$

where $g_j \in D$. To compute $P(g_j = spam)$ we treat $g_j = [X_1 \ldots X_8]$ as a vector where each X_i takes the values attained by the i^{th} feature f_i. As each feature models a different behavior, we can assume the features to be independent and express $P(g_k{=}spam) = \Pi P(X_i^{g_k} = spam)$. To compute $P(X_i^{g_k} = spam)$, i.e., $P(X_i{=}spam)$ for g_k, we discretize the range of values obtained by f_i into a set of k intervals $\{{}^1f_i, \ldots, {}^kf_i\}$ such that $\cup\, {}^kf_i = [0, 1]$. $P(X_i = spam)$ then reduces to $P({}^kf_i = spam)$ whenever X_i lies in the interval ${}^kf_i$. And $P({}^kf_i = spam)$ is simply the fraction of spam groups whose value of f_i lies in interval ${}^kf_i$. We used the popular discretization algorithm in [Fayyad and Irani, 1993] to divide the value range of each feature into intervals. To bootstrap the hill climbing search, we used initial seeds $\tau_0 = 2$ months and $\beta_0 = 6$ months. The final estimated values were: $\tau = 2.87$, $\beta = 8.86$.

4.7.2 Ranking experiments

To compare group spam ranking of GSRank, we use regression and learning to rank [Liu, 2009] as our baselines. Regression is a natural choice because the spamicity score of each group from the judges is a real value over [0, 1] (Section 4.1). The problem of ranking spammer groups can be seen as optimizing the spamicity of each group as a regression target. That is, the learned function predicts the spamicity score of each test group. The test groups can then be ranked based on the values. For this set of experiments, we use the support vector regression (SVR) system in SVM^{light} [Joachims, 1999].

Learning to rank is our second baseline approach. Given the training samples $x_1 \ldots x_n$, a learning to rank system takes as input k different rankings $y_1 \ldots y_k$ of the samples generated by k queries $q_1 \ldots q_k$. Each ranking y_i is a permutation of $x_1 \ldots x_n$ based on q_i. The learning algorithm learns a ranking model

h which is then used to rank the test samples $u_1 \ldots u_m$ based on a query q. In our case of ranking spam groups, the desired information need q denotes the question: *Are these group spam*? To prepare training rankings, we treat each feature f as a ranking function (i.e. the groups are ranked in descending order of values attained by each $f_1 \ldots f_8$). This generates 8 training ranks. A learning algorithm then learns the optimal ranking function. Given no other knowledge, this is a reasonable approach since $f_1 \ldots f_8$ are strongly correlated with spam groups (Section 4.4). The rank produced by each feature is thus based on a certain spamicity dimension. None of the training ranks may be optimal. A learning to rank method basically learns an optimal ranking function using the combination of $f_1 \ldots f_8$. Each group is vectorized with (represented with a vector of) the 8 group spam features. We ran two widely used learning to rank algorithms [Li, 2009]: SVMRank [Joachims, 2002] and RankBoost [Freund, 2003]. For SVMRank, we used the system in [Joachims, 2002]. RankBoost was from RankLib[7]. For both systems, their default parameters were applied. We also experimented with RankNet in RankLib, but its results are significantly poorer on our data. Thus, its results are not included.

In addition, we also experimented with the following baselines:

1. Group Spam Feature Sum (GSFSum): As each group feature $f_1 \ldots f_8$ measures spam behavior on a specific spam dimension, an obvious baseline (although naïve) is to rank the groups in descending order of the sum of all feature values.
2. Helpfulness Score (HS): In many review sites, readers can provide helpfulness feedback to each review. It is reasonable to assume that spam reviews should get less helpfulness feedback. HS uses the mean helpfulness score (percentage of people who found a review helpful) of reviews of each group to rank groups in ascending order of the scores.

[7] http://www.cs.umass.edu/~vdang/ranklib.html

3. Heuristic training rankings (H): In our preliminary study [Mukherjee et al., 2011], three heuristic rankings using feature mixtures were proposed to generate the training ranks for learning to rank methods. We list them briefly here. For details, see [Mukherjee et al., 2011].
   i. $h_1(g) : G \rightarrow \boldsymbol{R}^+$, $h_1(g) = GCS(g) + GMCS(g)$
   ii. $h_2(g) : G \rightarrow \boldsymbol{R}^+$, $h_2(g) = GS(g) + GSUP(g) + GTW(g)$
   iii. $h_3(g) : G \rightarrow \boldsymbol{R}^+$, $h_3(g) = GSR(g) + GETF(g) + GD(g)$
4. Using these three functions to generate the training ranks, we ran the learning to rank methods. We denote these methods and their results with RankBoost_H, and SVMRank_H.

To compare rankings we first use Normalized Discounted Cumulative Gain (NDCG) as our evaluation metric. NDCG is commonly used to evaluate retrieval algorithms with respect to an ideal ranking based on relevance. It rewards rankings with the most relevant results at the top positions, which is also our objective, i.e., to rank those groups with the highest spamicities at the top. The spamicity score for each group computed from judges (Section 4.1) thus can be regarded as the relevance score to generate the "ideal" spam ranking. Let $R(m)$ be the relevance score of the m^{th} ranked item. NDCG @ k is defined as:

$$NDCG@k = \frac{DCG@k}{Z_k}; \qquad DCG@k = \sum_{m=1}^{k} \frac{2^{R(m)} - 1}{\log_2(1+m)}, \tag{4.26}$$

where Z_k is the discounted cumulative gain (DCG) of the ideal ranking of the top k results. We report NDCG scores at various top positions up to 100 in Figure 7. In our case, $R(m)$ refers to the score(g_m) computed by each ranking algorithm (normalization was applied if needed), where g is the group ranked at position m. To compute Z_k = DCG@k for the ideal ranking, we use the spamicity(g_m) from our expert judges.

From Figure 7, we observe that GSRank performs the best at all top rank positions except at the bottom, which are unimportant because they are most likely to be non-spam. Paired t-tests for rank

positions k = 20, 40, 60, 80 show that all the improvements of GSRank over other methods are significant at the confidence level of 95%. Although regression is suitable for the task, it did not perform as well as RankBoost and SVMRank. RankBoost_H and SVMRank_H behave similarly to RankBoost and SVMRank, but performed slightly poorer. GSFSum fared mediocrely as ranking based on summing all feature values is unable to balance the weights of features because not all features are equal in discriminative strength. HS performs poorly, which reveals that while many genuine reviews may not be helpful, spam reviews can be quite helpful (deceptive). Thus, helpfulness scores are not good for differentiating spam and non-spam groups.

Since in many applications, the user wants to investigate a certain number of highly likely spam groups and NDCG does not give any guidance on how many are very likely to be spam, we thus also use *precision @ n* to evaluate the rankings. In this case, we need to know which test groups are spam and non-spam. We can use a threshold ξ on the spamicity to decide that, which can reflect the user's strictness for spam. Since in different applications the user may want to use different thresholds, we use two thresholds in our experiments, ξ = 0.5 and ξ = 0.7. That is, if the spamicity value is $\geq \xi$, the group is regarded as *spam*, otherwise *non-spam*. Figure 8 (A) and (B) show the precisions @ 20, 40, 60, 80, and 100 top rank positions for ξ = 0.5 and ξ = 0.7 respectively. We can see that GSRank consistently outperforms all existing methods. RankBoost is the strongest among the existing methods.

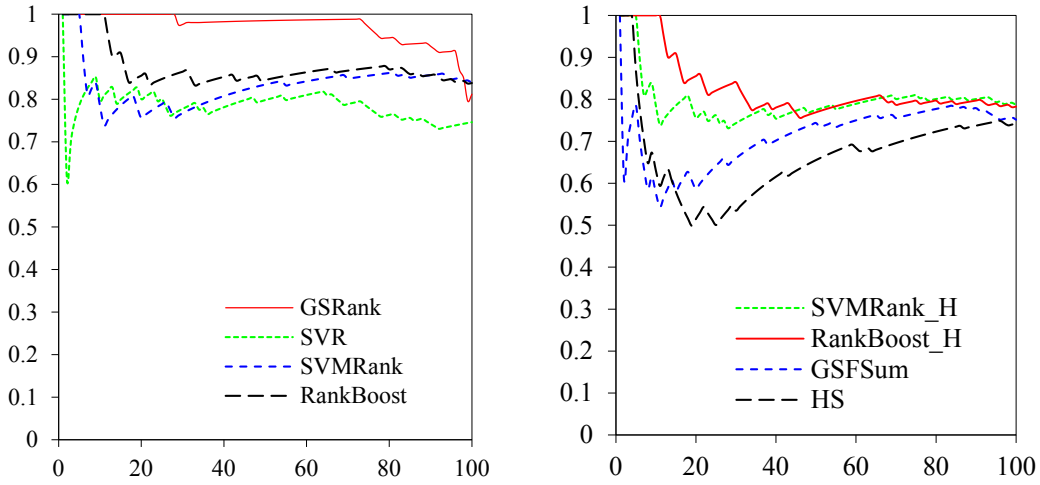
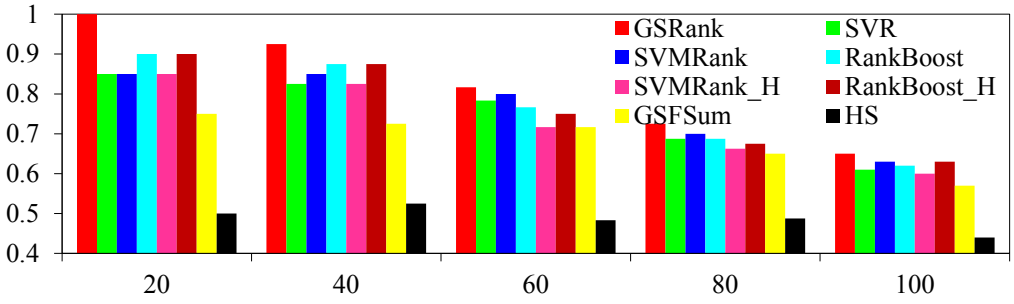
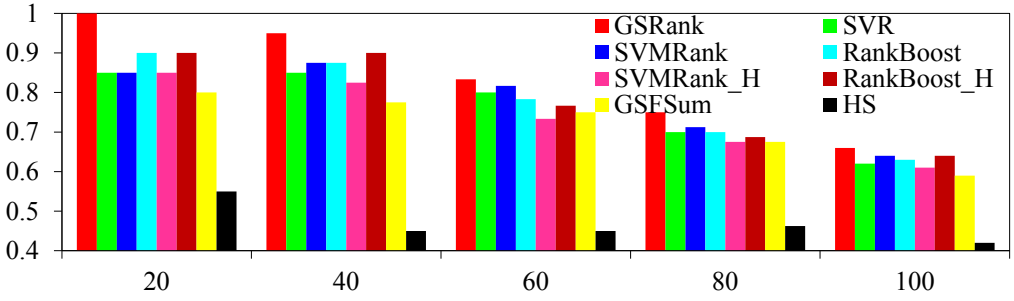


Figure 7: NDCG@*k* comparisons (NDCG for top 100 rank positions)



(A) The spamicity threshold of $\xi = 0.5$



(B) The spamicity threshold of $\xi = 0.7$

Figure 8: Precision @ n = 20, 40, 60, 80, 100 rank positions. All the improvements of GSRank over other methods are statistically significant at the confidence level of 95% based on paired *t*-test.

Feature Settings	SVM	LR	SVR	SVM Rank	Rank Boost	SVM Rank_H	Rank Boost_H	GS Rank
GSF	0.81	0.77	0.83	0.83	0.85	0.81	0.83	**0.93**
ISF	0.67	0.67	0.71	0.70	0.74	0.68	0.72	
LF	0.65	0.62	0.63	0.67	0.72	0.64	0.71	
GSF + ISF + LF	0.84	0.81	0.85	0.84	**0.86**	0.83	0.85	

(A) The spamicity threshold of $\xi = 0.5$

Feature Settings	SVM	LR	SVR	SVM Rank	Rank Boost	SVM Rank_H	Rank Boost_H	GS Rank
GSF	0.83	0.79	0.84	0.85	0.87	0.83	0.85	**0.95**
ISF	0.68	0.68	0.73	0.71	0.75	0.70	0.74	
LF	0.66	0.62	0.67	0.69	0.74	0.68	0.73	
GSF + ISF + LF	0.86	0.83	0.86	0.86	**0.88**	0.84	0.86	

(B) The spamicity threshold of $\xi = 0.7$

Table XIV: AUC results of different algorithms and feature sets. All the improvements of GSRank over other methods are statistically significant at the confidence level of 95% based on paired *t*-test.

4.7.3 Classification experiments

If a spamicity threshold is applied to decide spam and non-spam groups, supervised classification can also be applied. Using the thresholds of $\xi = 0.5$ and 0.7, we have the labeled data. We use SVM in SVM^{light} [Joachims, 1999] (with linear kernel) and Logistic Regression (LR) in WEKA (www.cs.waikato.ac.nz/ml/weka) as the learning algorithms. The commonly used measure AUC (Area Under the ROC Curve) is employed for classification evaluation.

Next we discuss the features that we consider in learning:

1. Group Spam Features (GSF) $f_1 \ldots f_8$: These are the proposed eight (8) group features presented in Section 4.3.1.
2. Individual Spammer Features (ISF): A set of features for detecting individual spammers was reported in [Lim et al., 2010]. Using these features, we represented each group with their average values of all the members of each group. We want to see whether such individual spammer features are also effective for groups. Note that these features cover those in Section 4.3.2.
3. Linguistic Features of reviews (LF): In [Ott et al., 2011], word and POS (part-of-speech) n-gram features were shown to be effective for detecting individual fake reviews. Here, we want to see whether such features are also effective for spam groups. For each group, we merged its reviews into one document and represented it with these linguistic features.

Table XIV (A) and (B) show the AUC values of the two classification algorithms for different feature settings using 10-fold cross validation for $\xi = 0.5$ and $\xi = 0.7$ respectively. It also includes the ranking algorithms in Section 4.7.2 as we can also compute their AUC values given the spam labels in the test data. Note that the relation-based model of GSRank could not use other features than GSF features and the features in Section 4.3.2. Here, again we observe that GSRank is significantly better than all other algorithms (with the 95% confidence level using paired *t*-test). RankBoost again performed the best among the existing methods. Individual spam features (ISF) performed poorly. This is understandable because they cannot represent group behaviors well. Linguistic features (LF) fared poorly too. We believe it is because content-based features are more useful if all reviews are about the same type of products. The language used in fake and genuine reviews can have some subtle differences. However, reviewers in a group can review different types of products. Even if there are some linguistic differences among spam and non-spam reviews, the features become quite sparse and less effective due to a large number of product types and not so many groups. We also see that combining all features (Table XIV, last row in each table) improves AUC slightly. RankBoost achieved AUC = 0.86 ($\xi = 0.5$) and 0.88 ($\xi = 0.7$), which are still significantly lower than AUC = 0.93 ($\xi = 0.5$) and 0.95 ($\xi = 0.7$) for GSRank

respectively. Finally, we observe that the results for $\xi = 0.7$ are slightly better than those for $\xi = 0.5$. This is because with the threshold $\xi = 0.7$, the spam and non-spam groups are well separated. In summary, we conclude that GSRank outperforms all baseline methods, including regression, learning to rank and classification. This is important considering that GSRank is an unsupervised method. This also shows that the relation-based model used in GSRank is indeed effective in detecting opinion spammer groups.

CHAPTER 5

LATENT VARIABLE MODELS FOR OPINION SPAM DETECTION

The previous Chapter dealt with the problem of collusion or group based spamming. In recent years, researchers have also studied other dimensions of the problem and proposed several techniques. However, the problem is still wide open. Unlike many other forms of spamming, the key difficulty for solving the opinion spam problem is that it is hard to find gold-standard data of fake and non-fake reviews for model building because it is very difficult, if not impossible, to manually recognize/label fake/non-fake reviews by mere reading [Jindal and Liu, 2008; Ott et al., 2011]. This leads to an important scenario in large-scale machine learning for anomaly detection where there is *absence of ground truth or positive labeled data*. This situation also arises in fields like *epidemiology* (accurate diagnosis of infected population), *credibility analysis* (positive samples of rumors), *cyber-security* (positive cases of phishing, sockpuppets, email fraud, etc.). A natural question is how do we build large scale machine learning models in such real-world settings? This Chapter proposes unsupervised Bayesian clustering models to address the problem. Before proceeding further, we review some related literature below and then propose the main intuitions.

Since it was first studied in [Jindal and Liu, 2008], various methods have been proposed to detect opinion spam. One of the main methods is supervised learning [Jindal and Liu, 2008; Ott et al., 2011; Li et al., 2011]. However, due to the lack of reliable ground truth label of fake/non-fake review data, existing works have relied mostly on ad-hoc or pseudo fake/non-fake labels for model building. In [Jindal and Liu, 2008], duplicate and near duplicate reviews were assumed to be fake reviews, which is restrictive and can be unreliable. In [Lim et al., 2010], a manually labeled dataset was used, which also has reliability issues because it has been shown that the accuracy of human labeling of fake reviews

is very poor [Ott et al., 2011]. In [Ott et al, 2011], Amazon Mechanical Turk (AMT) was employed to crowdsource fake hotel reviews by paying (US$1 per review) anonymous online workers (called *Turkers*) to write fake reviews for some hotels. Although these reviews are fake, they do not reflect the dynamics of fake reviews in a commercial website [Mukherjee et al., 2013c] as the Turkers do not have the same psychological state of mind when they write fake reviews as that of fake reviewers in a commercial website who have real business interests to promote or to demote. Also, Turkers may not have sufficient domain knowledge or experience to write convincing fake reviews. Due to the lack of labeled data, unsupervised methods have also been proposed for detecting individual [Lim et al., 2010; Wang et al., 2011] and group [Mukherjee et al., 2011; Mukherjee et al., 2012] spammers, time-series [Xie et al., 2012] and distributional [Mukherjee et al., 2013b; Feng et al., 2012] analysis, and mining reviewing patterns as unexpected association rules [Jindal et al., 2010] and reviewing burstiness [Fei et al., 2013].

The above existing works in opinion spam have made good progresses. However, they are largely based on heuristics and/or hinge on ad-hoc fake/non-fake labels for model building. No principled or theoretical models have been proposed so far.

This paper proposes a novel and principled technique to model and to detect opinion spamming in a Bayesian framework. It transcends the existing limitations discussed above and presents an unsupervised method for detecting opinion spam. We take a fully Bayesian approach and formulate opinion spam detection as a clustering problem. The Bayesian setting allows us to model *spamicity* of reviewers as latent with other observed behavioral features in our Author Spamicity Model (ASM). Spamicity here means the degree of being spamming. The key motivation hinges on the hypothesis that opinion spammers differ from others on behavioral dimensions [Mukherjee et al., 2012]. This creates a separation margin between population distributions of two naturally occurring clusters: spammers and non-spammers. Inference in ASM results in learning the distributions of two clusters (or classes)

based on a set of behavioral features. Various extensions of ASM are also proposed exploiting different priors.

The main contributions of this Chapter can be summarized as follows:

1. It proposes a novel and principled method to exploit observed behavioral footprints to detect spammers (fake reviewers) in an unsupervised Bayesian framework precluding the need of any manual labels for learning which is both hard [Jindal and Liu, 2008] and noisy [Ott et al., 2011]. A key advantage of employing Bayesian inference is that the model facilities characterization of various spamming activities using estimated latent variables and the posterior. It facilitates both detection and analysis in a single framework rendering a deep insight into the opinion spam problem. This cannot be done using existing methods. To our knowledge, this is the first principled model for solving this problem.
2. It proposes a novel technique to evaluate the results without using any labeled data. This method uses reviews of the top ranked and bottom ranked authors produced by the model as two classes of data to build a supervised classifier. The key idea is that the classification uses a complete different set of features than those used in modeling. Thus, if this classifier can classify accurately, it gives a good confidence that the unsupervised spamicity model is effective (details in Section 5.3.1).
3. It conducts a comprehensive set of experiments to evaluate the proposed model based on the classification evaluation method above and also human expert judgment. It also compares with a set of strong baseline techniques. The results show that the proposed models outperform the baselines significantly.

5.1 Model

This section details the proposed unsupervised model. We first discuss the basic intuition (Section 5.1.1) and the observed features (Section 5.1.2). In Section 5.1.3, we explain the generative process of our model and detail inference methods in Section 5.1.4 and Section 5.1.5.

5.1.1 Model overview

As discussed above, the proposed model formulates spam detection as an unsupervised clustering problem in the Bayesian setting. It belongs to the class of generative models for clustering [Duda et al., 2001] based on a set of observed features. It models spamicity, s_a (degree/tendency of spamming in the range [0, 1]) of an author, a; and spam/non-spam label, π_r of a review, r as latent variables. π_r is essentially the *class* variable reflecting the cluster memberships (we have two clusters, $K = 2$, spam and non-spam) for every review instance. Each author/reviewer (and respectively each review) has a set of observed features (behavioral clues) emitted according to the corresponding latent prior class distributions. Model inference learns the latent population distributions of the two clusters across various behavioral dimensions, and also the cluster assignments of reviews in the unsupervised setting based on the principle of probabilistic model-based clustering [Smyth, 1999].

As the generative process of ASM conditions review spam labels on author spamicites, inference also results in author spamicity estimates (probability of spamming) facilitating ranking of authors based on spamicity which is our main focus.

5.1.2 Observed features

Here we propose some characteristics of abnormal behaviors which are likely to be linked with spamming and thus can be exploited as observed features in our model for learning the spam and non-spam clusters. We first list the author features and then the review features. The notations are listed in Table XV.

Variable/Functions	Description
a; A; r; $r_a = (a, r)$	Author a; set of all authors; a review; review r by author a
$R_a, p(r_a)$	All reviews by a, $R_a = \{r_a\}$; associated product p for r_a
R_p; $R_{a,p}$	Reviews on product, p; Reviews on product p by author a
$\star\ (r_a, p(r_a))$	The $\star$ rating of r_a on product $p(r_a)$ on the 5-$\star$ rating scale
$MaxRev(a)$	Maximum # of reviews posted in a day by an author, a
$F(a)$; $L(a)$	First posting date of a; last posting date of a
$L(a, p)$; $A(p)$	Last review posting date by a on p; product p's launch date
$k \in \{\hat{s}, \hat{n}\}$	Class variable k for Spam/Non-spam class (label)
$s_a \sim Beta(\alpha_{\hat{s}}, \alpha_{\hat{n}})$	Spamicity of an author, a, , $s_a \in [0, 1]$
$\alpha^a_{k\in\{\hat{s},\hat{n}\}}$	Beta shape parameters (priors) for s_a for each author a
$\pi_{r_a} \sim Bern(s_a)$	Spam/Non-spam class label for review r_a, $\pi_{r_a} \in \{\hat{s}, \hat{n}\}$
$\theta^f_{k\in\{\hat{s},\hat{n}\}} \sim Beta(\gamma^f_{\hat{s}}, \gamma^f_{\hat{n}})$	Per class prob. of exhibiting the review behavior, $f_{5...9}$
$\gamma^f_{k\in\{\hat{s},\hat{n}\}}$	Beta shape parameters of θ^f for each review behavior, f
$\psi^f_{k\in\{\hat{s},\hat{n}\}} \sim Beta$	Per class probability of exhibiting the author behavior, $f_{1...4}$
$\psi^f_{k,1}$; $\psi^f_{k,2}$	Beta shape parameters of class k for behavior f
$x^f_{a,r} \sim Bern\left(\theta^f_{k\in\{\hat{s},\hat{n}\}}\right)$	Observed review feature, $f \in \{DUP, EXT, DEV, ETF, RA\}$
$y^f_{a,r} \sim \psi^f_{k\in\{\hat{s},\hat{n}\}}$	Observed author features $f \in \{CS, MNR, BST, RFR\}$
$n_{a,\hat{s}}$; $n_{a,\hat{n}}$	# of reviews of author a assigned to spam; non-spam class
$n^f_{k,P}$; $n^f_{k,A}$	# of reviews in class $k \in \{\hat{s}, \hat{n}\}$ which have review feature f (P)resent (f attains value 1); (A)bsent (f attains value 0)
n_a; $n_{k\in\{\hat{s},\hat{n}\}}$	# of reviews by author a; # of reviews in class k
K	Total number of clusters in the model

Table XV: List of notations

Author Features: Author features have values in [0, 1]. A value close to 1 (respectively 0) indicates spamming (non-spamming).

1. Content Similarity (CS): As crafting a new review every time is time consuming, spammers are likely to copy reviews across similar products. It is thus useful to capture the content similarity of

reviews (using cosine similarity) of the same author. We chose the maximum similarity to capture the worst spamming behavior.

$$f_{CS}(a) = f_1(a) = \max_{r_i, r_j \in R_a, i<j} cosine(r_i, r_j) \tag{5.1}$$

2. Maximum Number of Reviews (MNR): Posting many reviews in a single day also indicates an abnormal behavior. This feature computes the maximum number of reviews in a day for an author and normalizes it by the maximum value for our data.

$$f_{MNR}(a) = f_2(a) = \frac{MaxRev(a)}{\max_{a \in A}(MaxRev(a))} \tag{5.2}$$

3. Reviewing Burstiness (BST): The study in [Mukherjee et al., 2012] reports that opinion spammers are usually not longtime members of a site. Genuine reviewers, however, use their accounts from time to time to post reviews. It is thus useful to exploit the activity freshness of an account in detecting spamming. We define reviewing burstiness using the activity window (difference of first and last review posting dates). If reviews are posted over a reasonably long timeframe, it probably indicates normal activity. However, when all reviews are posted within a very short burst (τ = 28 days, estimated in Section 5.2.1), it is likely to be a spam infliction.

$$f_{BST}(a) = f_3(a) = \begin{cases} 0, L(a) - F(a) > \tau \\ 1 - \frac{L(a) - F(a)}{\tau}, otherwise \end{cases} \tag{5.3}$$

4. Ratio of First Reviews (RFR): Spamming early can impact the initial sales as people rely on the early reviews. Hence, spammers would try to be among the first reviewers for products as this enables them to control the sentiment [Lim et al., 2010]. We compute the ratio of *first reviews* to total reviews for each author. First reviews refer to reviews where the author is the first reviewer for the products.

$$f_{RFR}(a) = f_4(a) = \frac{|\{r \in R_a : r \; is \; a first \; review \}|}{|R_a|} \tag{5.4}$$

Review Features: We have 5 binary review features. Values of 1 indicate spamming while 0 non-spamming.

5. Duplicate/Near Duplicate Reviews (*DUP*): Spammers often post multiple reviews which are duplicate/near-duplicate versions of previous reviews on the same product to boost ratings [Lim et al., 2010]. To capture this phenomenon, we compute the review feature (f_{DUP}) *duplicate reviews* on the same product as follows:

$$f_{DUP}(r_a) = f_5(a,r) = \begin{cases} 1, \exists r \in R_{p=p(r_a)} \; cosine(r_a, r) > \beta_1 \\ 0, otherwise \end{cases} \tag{5.5}$$

f_{DUP} attains a value of 1 for a review r_a by an author a on product p, if it is similar (using cosine similarity based on some threshold, β_1= 0.7 (say)) to any other review on p. β_1 is estimated in Section 5.2.1.

6. Extreme Rating (*EXT*): On a 5-star ($\star$) rating scale, it reflects the intuition that to inflict spam, spammers are likely to give extreme ratings (1$\star$ or 5$\star$) in order to demote/promote products.

$$f_{EXT}(r_a) = f_6(a,r) = \begin{cases} 1, \star(r_a, p(r_a)) \in \{1,5\} \\ 0, \star(r_a, p(r_a)) \in \{2,3,4\} \end{cases} \tag{5.6}$$

7. Rating Deviation (*DEV*): Review spamming usually involves wrong projection either in the positive or negative light so as to alter the true sentiment on products. This hints that ratings of spammers often deviate from the average ratings given by other reviewers. This feature attains the value of 1 if the rating deviation of a review exceeds some threshold β_2. β_2 is estimated in Section 5.2.1. We normalize by the maximum deviation, 4 on a 5-star scale.

$$f_{DEV}(r_a) = f_7(a,r) = \begin{cases} 1, \frac{\left| \star(r_a, \; p(r_a)) - E\left[\star\left(r_{a' \neq a}, \; p(r_a)\right)\right] \right|}{4} > \beta_2 \\ 0, otherwise \end{cases} \tag{5.7}$$

The expectation is taken over all reviews on product $p = p(r_a)$ by other authors, $a' \neq a$, to get the average rating on p.

8. Early Time Frame (*ETF*): Lim et al., [2010] argues that spammers often review early to inflict spam as the early reviews can greatly impact people's sentiment on a product. To capture this spamming characteristic, we propose the following feature:

$$f_{ETF}(r_a) = f_8(a,r) = \begin{cases} 1, ETF(r_a, p(r_a)) > \beta_3 \\ 0, otherwise \end{cases}$$

$$ETF(r_a,p) = \begin{cases} 0, L(a,p) - A(p) > \delta \\ 1 - \frac{L(a,p)-A(p)}{\delta}, otherwise \end{cases} \quad (5.8)$$

$ETF(r_a, p)$ captures how early an author a reviewed the product p. δ = 7 months is a threshold for denoting earliness (estimated in Section 5.2.1). The definition says that if the latest review is beyond 7 months of product launch, it is no longer considered to be early. At the other extreme, if reviews are posted just after launch this feature attains a value of 1. β_3 is the corresponding threshold indicating spamming and is estimated in Section 5.2.1.

9. Rating Abuse (RA): This feature captures the abuse caused by multiple ratings on the same product. Multiple ratings/reviews on the same product are unusual. Although this feature is similar to DUP, it focuses on the rating dimension rather than content. Rating abuse, $RA(a,p)$ is defined by the similarity of ratings of an author, a towards a product, p across multiple reviews by the author weighted by the total number of reviews on the product.

$$f_{RA}(r_a) = f_9(a,r) = \begin{cases} 1, RA(a, p(r_a)) > \beta_4 \\ 0, otherwise \end{cases};$$

$$RA(a,p) = |R_{a,p}| \left(1 - \frac{1}{4} \left(\max_{r \in R_{a,p}} (\star(r,p)) - \min_{r \in R_{a,p}} (\star(r,p)) \right) \right) \quad (5.9)$$

The similarity of multiple star rating is computed using the difference between maximum and minimum star rating on a 5-star scale to capture consistent high/low ratings. The normalization constant is 4 as it is the maximum possible rating difference. For multiple ratings in genuine cases where ratings change (e.g., after correct use), the feature attains lower values. β_4 is the rating abuse threshold indicating spamming and is estimated in Section 5.2.1.

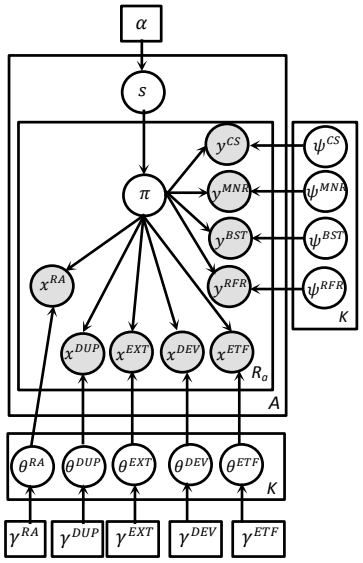


Figure 9: Plate notation of ASM.

5.1.3 Generative process

In ASM, spam detection is influenced by review and author features. Normalized continuous author features in [0, 1] are modeled as following a Beta distribution ($y^f_{a,r} \sim \psi^f_{k\in\{\hat{s},\hat{n}\}}$) (Table XV). This enables ASM to capture more fine grained dependencies of author's behaviors with spamming. However, review features being more objective, we found that they are better captured when modeled as binary variables being emitted from a Bernoulli distribution ($x^f_{a,r} \sim Bern\left(\theta^f_{k\in\{\hat{s},\hat{n}\}}\right)$ (Table XV). $\theta^f_{k\in\{\hat{s},\hat{n}\}}$ for each review feature $f \in \{DUP, EXT, DEV, ETF, RA\}$ and $\psi^f_{k\in\{\hat{s},\hat{n}\}}$ for each author feature $f \in \{CS, MNR, BST, RFR\}$ denote the per class/cluster (spam vs. non-spam) probability of emitting feature f.

Latent variables s_a and π_r denote the spamicity of an author, a and the (spam/non-spam) class of each review, r. The objective of ASM is to learn the latent behavior distributions for spam and non-spam clusters (K = 2) along with spamicites of authors from the observed features. We now detail its generative process.

1. For each class/cluster, $k \in \{\hat{s}, \hat{n}\}$:

   Draw $\theta_k^{f \in \{DUP,\ldots,RA\}} \sim Beta(\ \gamma^f)$

2. For each (author), $a \in \{1 \ldots A\}$:

   i. Draw spamicity, $s_a \sim Beta(\alpha^a)$;

   ii. For each review, $r_a \in \{1 \ldots R_a\}$:

   a. Draw its class, $\pi_{r_a} \sim Bern(s_a)$

   b. Emit review features $f \in \{DUP, \ldots, RA\}$:

   $x_{r_a}^f \sim Bern\left(\theta_{\pi_{r_a}}^f\right)$;

   c. Emit author features $f \in \{CS, \ldots, RFR\}$:

   $y_{r_a}^f \sim \psi_{\pi_{r_a}}^f$;

We note that the observed author features are placed in the review plate (Figure 9). This is because each author behavior can be thought of as percolating through reviews of that author and emitted across each review to some extent. Doing this renders two key advantages: i) It permits us to exploit a larger co-occurrence domain. ii) It paves the way for a simpler sampling distribution providing for faster inference.

5.1.4 Inference

We employ approximate posterior inference with Monte Carlo Gibbs sampling, and use Rao-Blackwellization [Bishop, 2006] to reduce sampling variance by collapsing on the latent variables s and θ^f. As observed author features obtaining values in [0, 1] are modeled as continuous Beta distributions, sparsity is considerably reduced as far as parameter estimation of $\psi^{f \in \{CS,\ldots RFR\}}$ is concerned. Hence, to ensure speed, we estimate ψ_k^f using the method of moments, once per sweep of Gibbs sampling. The Gibbs sampler is given by:

Algorithm 2: Inference using MCMC Gibbs Sampling

1. Initialization:
   Randomly assign review clusters, $\pi_{r_a} = \begin{cases} \hat{n}, z < 0.5 \\ \hat{s}, z \geq 0.5 \end{cases}; z \sim U(0,1)$
2. Iterate $n = 1$ to N_{max}: // $N_{max} = 3000$
   For author, $a = 1$ to A:
   For review $r_a = 1$ to R_a:
   i. Flush cluster assignment, π_{r_a};
   ii. Sample $\pi_{r_a} \sim p(\pi = k| \dots)$ using (5.10);
   iii. Update $n_{k,[\,]}^{f=DUP}$, $n_{k,[\,]}^{f=EXT}$, $n_{k,[\,]}^{f=DEV}$, $n_{k,[\,]}^{f=ETF}$, $n_{k,[\,]}^{f=RA}$ for $k \in \{\hat{s}, \hat{n}\}$
   End for
   End for
   If $n > N_{Burn\_In}$: // $N_{BurnIn} = 250$
   For author, $a = 1$ to A:
   For review $r_a = 1$ to R_a:
   Update $\psi_k^{f=CS}$, $\psi_k^{f=MNR}$, $\psi_k^{f=BST}$, $\psi_k^{f=RFR}$; $k \in \{\hat{s}, \hat{n}\}$ using (5.11)
   End for
   End for
   End if

Algorithm 3: Single-sample Monte Carlo EM

1. Initialization:
   Start with uninformed priors: $\alpha^a \leftarrow (1,1)$; $\gamma^f \leftarrow (1,1)$
2. Repeat:
   i. Run Gibbs sampling to steady state (Algorithm 1) using current values of α^a, γ^f.
   ii.Optimize α^a using (5.12) and γ^f using (5.14)
   Until convergence of α^a, γ^f

$$p(\pi_i = k|\boldsymbol{\pi}_{\neg \boldsymbol{i}} \dots) \propto \frac{n_{a,k_{\neg i}} + \alpha_k^a}{(n_a + \alpha_{\hat{s}}^a + \alpha_{\hat{n}}^a)_{\neg i}} \times \prod_{f \in \{DUP,EXT,DEV,ETF,RA\}} \left(g(f,k,x_{a,r}^f)\right) \times$$

$$\prod_{f \in \{CS,MNR,BST,RFR\}} \left(p\left(y_{a,r}^f | \psi_{\pi_i}^f\right)\right) \tag{5.10}$$

where functions g and $p\left(y_{a,r}^f | \psi_{\pi_i}^f\right)$ are defined below:

$$g(f,k,x_{a,r}^f) = \begin{cases} \dfrac{\left(n_{k,P}^f + \gamma_k^f\right)_{\neg i}}{\left(n_k + \gamma_{\hat{s}}^f + \gamma_{\hat{n}}^f\right)_{\neg i}}, & if \quad x_{a,r}^f = 1 \\ \dfrac{\left(n_{k,A}^f + \gamma_{\neg k}^f\right)_{\neg i}}{\left(n_k + \gamma_{\hat{s}}^f + \gamma_{\hat{n}}^f\right)_{\neg i}}, & if \quad x_{a,r}^f = 0 \end{cases}$$

$$p\left(y_{a,r}^f | \psi_{\pi_i}^f\right) \propto \left(y_{a,r}^f\right)^{\psi_{i_1}^f - 1} \left(1 - y_{a,r}^f\right)^{\psi_{i_2}^f - 1}$$

The subscript $\neg i$ denotes counts excluding review $i = r_a = (a, r)$. The Beta shape parameter updates for ψ_k^f using method of moments are given as follows:

$$\psi_k^f = (\psi_{k,1}^f, \psi_{k,2}^f) = \left(\mu_k^f \left(\frac{\mu_k^f (1-\mu_k^f)}{\sigma_k^f} - 1 \right), (1-\mu_k^f) \left(\frac{\mu_k^f (1-\mu_k^f)}{\sigma_k^f} - 1 \right) \right) \tag{5.11}$$

where μ_k^f and σ_k^f denote the mean and biased sample variance for feature f corresponding to class k. Algorithm 1 details the full inference procedure for learning ASM using the above Gibbs conditional. Omission of a latter index denoted by [ ] corresponds to the row vector spanning over the latter index.

5.1.5 Hyperparameter-EM

Algorithm 2 performs inference using uninformed priors (i.e., hyperparameters α and γ are set to $(1, 1)$). Posterior estimates of spamicity can be improved if hyperparameters α and γ are estimated from the data. This is because the priors for author spamicity and latent review behaviors (α and γ) directly affect spam/non-spam cluster assignment to reviews. Algorithm 2 details hyperparameter estimation using the single sample Monte Carlo EM.

Algorithm 2 learns hyperparameters α and γ which maximize the model's complete log-likelihood, $\mathcal{L}$. We employ an L-BFGS optimizer for maximization. L-BFGS is a quasi-Newton method which does not require the Hessian matrix of second order derivatives. It approximates the Hessian using rank-one updates of first order gradient. A careful observation of the model's complete log-likelihood reveals that it is a separable function in $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ allowing the hyperparameters to be maximized independently. Due to space limits, we only provide the final update equations:

$$\alpha_k^a = \underset{\alpha_k^a}{\operatorname{argmax}} \begin{pmatrix} \log\Gamma(\alpha_{\hat{s}}^a + \alpha_{\hat{n}}^a) + \log\Gamma(\alpha_{\hat{s}}^a + n_{a,\hat{s}}) + \log\Gamma(\alpha_{\hat{n}}^a + n_{a,\hat{n}}) \\ -\log\Gamma(\alpha_{\hat{s}}^a) - \log\Gamma(\alpha_{\hat{n}}^a) - \log\Gamma(n_a + \alpha_{\hat{s}}^a + \alpha_{\hat{n}}^a) \end{pmatrix} \tag{5.12}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_k^a} = \Psi(\alpha_{\hat{s}}^a + \alpha_{\hat{n}}^a) + \Psi(\alpha_k^a + n_{a,k}) - \Psi(\alpha_k^a) - \Psi(n_a + \alpha_{\hat{s}}^a + \alpha_{\hat{n}}^a) \tag{5.13}$$

$$\gamma_k^f = \underset{\gamma_k^f}{\operatorname{argmax}} \begin{pmatrix} \log\Gamma(\gamma_{\hat{s}}^f + \gamma_{\hat{n}}^f) + \log\Gamma(\gamma_{\hat{s}}^f + n_{k,P}^f) + \log\Gamma(\gamma_{\hat{n}}^f + n_{k,A}^f) \\ -\log\Gamma(\gamma_{\hat{s}}^f) - \log\Gamma(\gamma_{\hat{n}}^f) - \log\Gamma(n_k + \gamma_{\hat{s}}^f + \gamma_{\hat{n}}^f) \end{pmatrix} \tag{5.14}$$

$$\frac{\partial \mathcal{L}}{\partial \gamma_{\hat{s}}^{f}} = \Psi\left(\gamma_{\hat{s}}^{f} + \gamma_{\hat{n}}^{f}\right) + \Psi\left(\gamma_{\hat{s}}^{f} + n_{\hat{s},P}^{f}\right) - \Psi\left(\gamma_{\hat{s}}^{f}\right) - \Psi\left(n_{k} + \gamma_{\hat{s}}^{f} + \gamma_{\hat{n}}^{f}\right) \tag{5.15}$$

$$\frac{\partial \mathcal{L}}{\partial \gamma_{\hat{n}}^{f}} = \Psi\left(\gamma_{\hat{s}}^{f} + \gamma_{\hat{n}}^{f}\right) + \Psi\left(\gamma_{\hat{n}}^{f} + n_{\hat{n},A}^{f}\right) - \Psi\left(\gamma_{\hat{n}}^{f}\right) - \Psi\left(n_{k} + \gamma_{\hat{s}}^{f} + \gamma_{\hat{n}}^{f}\right) \tag{5.16}$$

where, $\Psi(\cdot)$ denotes the digamma function.

5.2 Experiment settings

We now evaluate the proposed ASM model. We use the reviews of manufactured products from Amazon.com. We consider only authors/reviewers with at least 3 reviews as authors with fewer reviews have few behavior characteristics. For reviewers with fewer reviews, the method in [Xie et al., 2012] can be applied. Our final data comprises of 50,704 reviewers, 985,765 reviews, and 112,055 products. Below we describe parameter estimation, baseline systems, and evaluation results.

5.2.1 Learning feature thresholds

As noted in Section 5.1.2, the proposed feature constructions contain thresholds. The thresholds can either be set heuristically or learned using some weak supervision. In this work, we use weak supervision to learn the thresholds from the Amazon group spam dataset in [Mukherjee et al., 2012], which provides a small set of labeled spammer groups and their reviews. Using this small set of available data is not a limitation of our method because the actual spamicty modeling of ASM still remains unsupervised as it does not use any spamicty labels for authors/reviews in model building. In fact, ASM does not have any response variable where supervision can be fed using labels. To keep evaluation fair, the labeled data in [Mukherjee et al., 2012] is not used for our evaluation in Section 5.2.3.

Thresholds of burstiness, τ = 28 days and earliness, δ = 7 months (in Section 5.1.2) were estimated using greedy hill-climbing search maximizing the likelihood of the data in [Mukherjee et al., 2012]. It is also important to note that above thresholds apply to feature constructions and not specific to ASM.

As we will see in Section 5.2.2, the same features are used by other baseline methods, so improvements of ASM are attributed to its process and not the choice of feature thresholds.

Thresholds for binary discretization of continuous review features $\beta_{1...4}$ (in Section 5.1.2) are learned using Recursive Minimal Entropy Partitioning (RMEP) [Fayyad and Irani, 1993]. The estimated values are as follows: β_1 = 0.72, β_2 = 0.63, β_3 = 0.69, β_4 = 2.01.

5.2.2 Systems in our experiments

Although ASM clusters reviews and estimates author spamicities, in this work, our focus is to evaluate the ranking of authors based on estimated author spamicities.

5.2.2.1 Generative models: ASM Variants

ASM with Uninformed Priors (ASM-UP): This is the fully unsupervised version of ASM. Uninformed priors are used for Beta distributed variables, s^a, θ^f, $f \in \{DUP, .., RA\}$, i.e., $\forall a \in A$, $\alpha^a \leftarrow (1,1); \gamma^f \leftarrow (1,1)$. This setting is called uninformed because any value in [0, 1] is equally likely to be assigned to the Beta distributed variables and the model doesn't benefit from any domain knowledge based priors. Posterior estimates are drawn after 3000 iterations with an initial burn-in of 250 iterations.

ASM with Informed Priors (ASM-IP): Here we employ guidance using some domain knowledge heuristics. Amazon tags each review with AVP (Amazon Verified Purchase) if the reviewer actually bought the product. Keeping other settings the same as ASM-UP, for each author, a, if none of his reviews have AVP tags, then we set $s_a \sim Beta(5,1)$ else we set $s_a \sim Beta(1,5)$ [1]. The rationale is that authors who have not bought a single product on Amazon are likely to be less reliable (hence probably

[1] A potential non-spammer (NS), $s^{NS} \sim Beta(1,5)$ has an expected value, $E[s^{NS}] = \frac{1}{6}$ which is much smaller than a potential spammer (S), $s^{NS} \sim Beta(5,1)$, with expected value, $E[s^S] = \frac{5}{6}$ in the range [0, 1].

spamming) than those who also review products that they have purchased on Amazon (i.e., receiving AVP tags).

ASM with Hyperparameter Estimation (ASM-HE): This setting estimates the hyperparameters α^a, $a \in A$ and γ^f, $f \in \{DUP, EXT, DEV, ETF, RA\}$ using Algorithm 2 keeping all other settings fixed as ASM-UP.

5.2.2.2 Unsupervised rank aggregation

ASM estimates reviewer spamicities as scores in [0, 1] (the posterior on $s \sim Beta$). We can also regard the observed behaviors as ranking functions in [0, 1] with extreme values 0 (respectively 1) indicating non-spamming (spamming) on various behavior dimensions. Then, estimating the final spamicites of authors becomes unsupervised rank learning using aggregation [Klementieve et al., 2007]. The problem setting is described as follows.

Let $x \in X$ denote an item (e.g., author) in the instance space X (e.g., set of authors/reviewers) that need to be ranked relative to each other according to some criterion (e.g., spamicity). Let $q \in Q$ denote a query and $r : Q \times X \rightarrow \mathbb{R}$ denote a ranking function whose output governs the rank position of an item, i.e., $r(q, x) > r(q, x')$ specifies that x is ranked higher than x' on query q using ranking function r. The notation $r_u \succ_\epsilon r_v$ signifies that the ranking function r_u is better than r_v with respect to a certain evaluation criterion (e.g., NDCG, MAP, L_2 loss, etc.). Given a set of ranking functions, $\{r_i\}_{i=1}^N$, rank aggregation learns the optimal ranking function, r_{Opt} (using a weighted combination of $\{r_i\}_{i=1}^N$) such that $\forall i, r_{Opt} \succ_\epsilon r_i$. This problem setting is also called *unsupervised rank fusion* or *ensemble ranking*. In the supervised setting, one usually employs regression to learn the weights [Vogt et al., 1999]. In the unsupervised setting, the approach is two-fold: i) Derive an *incidental*/*surrogate* supervision signal. ii) Employ a learning algorithm to learn the parameters of r_{Opt}. In [Klementieve et al., 2007], the *surrogate* supervision signal was computed using some ranker agreement heuristics and iterative gradient descent was used as the learning algorithm.

In our case of ranking reviewers according to spamicites, we have 9 ranking functions $\{r_i\}_{i=1}^{N=9}$ corresponding to the 9 behavior feature dimensions (in Section 5.1.2). For each author feature, $f_{1...4} \in \{CS, MNR, BST, RFR\}$, we directly use the value of the feature as the ranking function while for each review feature, $f_{5...9} \in \{DUP, EXT, DEV, ETF, RA\}$, we take the expected value of the feature across all reviews of an author to compute the corresponding author feature. We then directly use the value of each author feature as a ranking function to produce 9 training ranks. Given no other knowledge, this is a reasonable approach since $f_{1...9}$ being anomalous reviewing behaviors are likely to be correlated with spammers. Thus, the training ranking produced by each feature function is based on a certain spamicity dimension. However, none of the training rankings may be optimal. We employ learning to rank [Liu, 2009] to learn an optimal ranking function by aggregating 9 ranking functions $\{f_i\}_{i=1}^{N=9}$.

Learning to rank [Liu, 2009] is a supervised technique that takes a set of rankings for training and produces a single optimal aggregated ranking function. Each of our training rankings is produced by sorting instances (authors) based on values of a feature in $\{f_i\}_{i=1}^{N=9}$. Each instance/author in a training ranking is represented as a vector of $f_{1...9}$ spam features. We experimented with two popular learning to rank algorithms: SVMRank [Joachims, 2002] and RankBoost [Freund et al., 2003]. For SVMRank, we used the system in [Joachims, 2002]. RankBoost was from RankLib[2]. We use the pair-wise L_2 loss metric as the optimization criterion for RankBoost and SVMRank. We also experimented with RankNet, AdaRank, Coordinate Ascent in RankLib, but their results were poorer and hence not included.

5.2.2.3 Baselines

For a more comprehensive comparison, we also experiment with the following baseline approaches:

[2] http://www.cs.umass.edu/~vdang/ranklib.html

Feature Sum (FSum): As each feature $f_{1...9}$ measures spam behavior on a specific dimension, an obvious approach is to rank the authors in descending order of the sum of all feature values.

Helpfulness Score (HS): In many review sites (e.g., Amazon), readers can provide helpfulness feedback to each review. It is reasonable to assume that spam reviews should get less helpfulness feedback. HS uses the mean helpfulness score (percentage of people who found a review helpful) of reviews of each reviewer to rank reviewers in ascending order of the scores.

Finally, we note that although in ASM spam detection is modeled as clustering, our key task is to estimate author spamcities (the posterior on the latent variable, $s_a \sim Beta \in [0, 1]$) for ranking authors according to their spamicities.. Standard clustering algorithms (e.g., *k*-means) are not suitable baselines because they only produce clusters, but do not generate a ranking of authors.

Table XVI (A)

k (%)	ASM-UP				ASM-IP				ASM-HE				SVMRank			
	P	R	F1	A	P	R	F1	A	P	R	F1	A	P	R	F1	A
5	77.7	74.0	75.8	75.5	77.9	74.8	76.3	75.7	79.6	75.1	77.3	77.4	72.1	74.7	73.4	73.1
10	68.5	62.9	65.6	63.5	72.1	69.5	70.8	72.8	76.8	70.3	73.4	73.4	67.9	70.3	69.1	70.4
15	62.9	59.9	61.4	60.2	66.8	64.5	65.6	66.1	68.9	67.4	68.1	66.7	57.2	60.9	58.9	59.2

Table XVI (B)

k (%)	RankBoost				FSum				HS			
	P	R	F1	A	P	R	F1	A	P	R	F1	A
5	74.6	75.1	74.8	74.6	76.1	73.6	74.8	75.2	57.8	61.7	59.7	59.8
10	68.1	71.6	69.8	71.2	67.3	60.2	63.6	61.4	58.9	60.8	59.8	60.6
15	58.3	57.8	58.0	59.8	60.2	55.3	57.6	57.2	61.7	58.0	59.8	58.2

Table XVI (A, B): 5-fold SVM CV for review classification using top *k* (%) authors' reviews as the spam (+) class and bottom *k* % authors' reviews as the non-spam (-) class. P: Precision, R:Recall, F1:F1-Score, A:Accuray.

	ASM-UP			ASM-IP			ASM-HE			SVMRank			RankBoost			FSum			HS		
	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3	B_1	B_2	B_3
J_1	31	15	3	36	11	1	43	5	0	36	19	1	37	13	1	34	13	0	6	14	17
J_2	28	14	3	31	6	1	36	6	0	32	16	4	34	8	2	32	11	0	5	12	14
J_3	29	13	2	33	8	0	39	3	0	33	11	2	34	11	0	31	8	0	8	9	10
Avg.	29.3	14.0	2.67	33.3	8.33	0.67	39.3	4.67	0	33.7	15.3	2.33	35.0	10.7	1	32.3	10.7	0	6.33	11.7	13.7
κ_{Fleiss}	0.73			0.68			0.74			0.71			0.72			0.76			0.73		

Table XVII: Number of spammers detected in each bucket (B_1, B_2, B_3) by each judge (J_1, J_2, J_3) across each method. Last row reports the agreement of judges using Fleiss' multi-rater kappa (κ_{Fleiss}) for each method.

5.3 Results

As noted in Section 5, we are not aware of any gold-standard ground truth labeled data for opinion spammers. Hence, this work focuses on Bayesian inference in the unsupervised setting. To evaluate the author spamicities computed by different systems, we use two methods: *review classification* and *human evaluation*.

Running the systems in Section 5.2.2 on our data generates a ranking of 50,704 reviewers. However, human evaluation on all authors is clearly impossible. Even for review classification, the number of reviews is huge for such a large number of reviewers. We thus sample the rank positions 1, 10, 20,…,

50000 (with a sampling lag/interval of 10 ranks) to construct the evaluation set, E of 5000 rank positions. Using a fixed sampling lag ensures that performance on E is a good approximation over the entire range. 5000 is reasonable for review classification, but for human evaluation we need to use a subset (see below).

5.3.1 Review classification

This is a new evaluation method in the unsupervised setting. The idea is that if ASM is effective, it must rank the highly likely spammers at the top and highly likely non-spammers at the bottom. We use a supervised classification of likely spammers and likely non-spammers to evaluate this ranking. Instead of classifying reviewers, we classify their reviews.

Prior works in [Mukherjee et al., 2013c] have shown that faking or lying usually involves more use of personal pronouns and associated verb actions to justify fake statements eventually resulting in more use of positive sentiments and emotion words. The hypothesis has been attested in [Ott et al., 2011] where text classification using *n*-gram features have been shown quite effective in detecting spam and non-spam reviews. Thus, if our classification of reviews based on text features is good, it implies that the ASM ranking of reviewers according to spamicites is effective because text classification concurs with the abnormal behavior spam detection of ASM. The key characteristic of this classification is that the text features have not been used in ASM. Clearly, using the same set of features will not be meaningful.

For this set of experiments, we consider the reviews from the top k% ranked authors to be in the spam (+) class while reviews from the bottom k% of the ranked authors to be in the non-spam (–) class. Although the actual percentage of spammers is unknown, deception prevalence studies [Wang, 2010] have reported 8-15% spam rate in online review sites. We thus report results for k = 5%, 10%, and 15%.

We induce a linear kernel SVM[3] and report 5-fold cross validation results in Table XVI. The features are 1-3-grams. We note the following observations:

1. As *k* increases, we find a monotonic degradation in review classification performance (except HS) which is expected as the spamicities of top and bottom authors get closer which makes the corresponding review classification harder.
2. For *k* = 5%, ASM models performs best on F1 and Accuracy metrics. Next in order are FSum, RankBoost, and SVMRank. It is interesting that the simple baseline FSum performs quite well for *k* = 5%. The reason is attributed to the fact that the top positions are mostly populated by heavy spammers while the bottom positions are populated by genuine reviewers and hence a naïve un-weighted FSum could capture this phenomenon.
3. For *k* = 10, 15%, FSum does not perform so well (SVMRank and RankBoost outperform Fsum). This is because for *k* = 10, 15%, the ranked positions involve more difficult cases of authors/reviewers and a mere sum is not able to balance the feature weights as not all features are equally discriminating.
4. For *k* = 10%, SVMRank and RankBoost outperform ASM-UP and perform close to ASM-IP. ASM-HE still outperforms SVMRank and RankBoost by 4% in F1 and 2-3 % in accuracy.
5. For *k* = 15%, ASM variants outperform other methods and increase F1 by a margin of 2-10% and accuracy by 3-7%.
6. Performance of HS remains much poorer and similar for each *k* showing that it is not able to rank spammers well, indicating that helpfulness is not a good metric for spam detection. In fact, helpfulness votes are subject to abuse.

We note that no significance test is applied here to compare performance of different methods because each method uses *different* reviews in classification as the output rankings from the systems are

[3] Other kernels, e.g., polynomial, rbf, sigmoid didn't perform so well.

different. However, we experimented with multiple and different randomized binning in 5-fold CV which showed the same trend as in Table XVI.

To further analyze the nature of spam and non-spam reviews by spammers and non-spammers (based on the rankings of ASM), we do an additional experiment. We consider the top 15% authors ranked by ASM-HE and randomly split them into two sets of authors and construct two classes of reviews coming from each set of authors. Classification on this set of reviews (coming from two sets of authors belonging to spammers) yielded 57% accuracy. Similarly, classification of reviews by two sets of authors belonging to non-spammers (bottom 15% ranked authors of ASM-HE) yielded 58% accuracy. We tried different random selection of authors which also yielded similar accuracies. We note that the accuracy is greater than 50% but less than the classification accuracy using ASM-HE (Table XVI). This shows that the above experiment is likely to separate authors and not spam vs. non-spam reviews. As different authors have differences in writing, the accuracy is greater than 50% (random guessing). But it is lower than spam and non-spam review classification because reviews of both classes come from spammers (or non-spammers) and hence noisy.

5.3.2 Human evaluation

Our second evaluation is based on human expert judgment, which is commonly used in research on Web spam. Human evaluation has also been used for opinion spam in prior works [Xie et al., 2011; Mukherjee et al., 2011]. It is however important to note that just by reading a single review without any context, it is very hard to determine whether a review is fake (spam) or not [Ott et al., 2011; Jindal and Liu, 2008]. However, it has been shown in [Mukherjee et al., 2012] that when sufficient context is provided e.g., reviewing patterns, ratings, type/brand of products reviewed, posting activity trails, etc., human expert evaluation becomes easier.

For this work, we used 3 domain expert judges for evaluating our ranked reviewers based on spamicites. The judges were briefed with many opinion spam signals: i) Having zero caveats, and full

of empty adjectives. ii) Purely glowing praises with no downsides. iii) Suspicious brand affinity/aversion, posting trails, etc., from prior findings and consumer sites. These signals are sensible as they have been compiled by consumer domain experts with extensive know-how on fake reviews. Our judges were also familiar with Amazon reviews and given access to additional metadata, e.g., reviewing profile, demographic information, etc. Although the judges were not provided the proposed features, they were encouraged to use their own signals along with the above existing signals and reviewer metadata. It is important here to note that providing various signals compiled from prior works and domain experts in consumer sites to the judges does not introduce a bias but enhances judgment. Without any signals, as mentioned above, it is very difficult to judge by merely reading reviews. It is also hard for anyone to know a large number of signals without extensive experience in opinion spam detection. Given a reviewer and his reviews, the judges were asked to independently examine his entire profile and to provide a label as spammer or non-spammer.

Due to the large number (5000) of reviewers in the evaluation set, E, it would have taken too much time for human judges to assess all the reviewers in E. We selected the following three ranked buckets (B_1, B_2, B_3) for evaluation by our judges:

B_1 (Top 50): Reviewers ranked from 1 to 50 by each system in E.

B_2 (Middle 50): Reviewers ranked from 2501 to 2550.

B_3 (Bottom 50): Reviewers ranked at bottom 50 ranks in E.

This is reasonable because these rank-positions in E reflect the performance trend on the entire range. Table XVII reports the results of each judge (as the count of reviewers labeled as spammers) for each bucket across each method. Additionally, we report the agreement of judges using Fleiss multi-rater kappa [Fleiss, 1971] for each method in the last row of Table XVII. We note the following observations:

1. All 3 judges perform similarly with slight variation (e.g., J_1 seems to be stricter and identifies more spammers in each bucket than J_3). This shows that the judges have consensus in spam

judgments. Further, the kappa values being in the range of substantial agreement according to scale[4] in [Landis and Koch, 1977] bolsters confidence in the judgments.

2. For all methods except HS, we find the average number of spammers detected by judges decreasing monotonically for buckets B_1, B_2, and B_3 which is expected from the ranking produced by the algorithms. HS performs poorly (placing more spammers in B2, B3 which is undesireable) as also observed in Section 5.3.1. Thus, helpfulness votes aren't useful for spam detection.
3. For B_1, ASM-HE performs best. On average, it is able to place 39 spammers in the top 50 rank positions. ASM-IP, RankBoost, and SVMRank perform poorer than ASM-HE. ASM-UP does not perform well as uninformed priors are weaker.
4. For B_2, the performance order is ASM-HE → ASM-IP → RankBoost → FSum → ASM-UP → SVMRank. At this bucket, a good method should place fewer spammers as B_2 is in the middle. This is indeed the case in Table XVII. Note that an ideal ranking should not place any spammers in the middle range as it is very unlikely that 50% of the reviewers are spammers.
5. For the last bucket, B_3, we find ASM-HE and FSum performing best by not placing any spammers in the bottom ranks. ASM-IP does quite well too. Next in performance order are RankBoost → SVMRank → ASM-UP. FSum performed very well in B_3 because its ranking (based on descending order of feature sum value) placed authors who ranked very low in all features. As the features are all abnormal and indicate suspicious behaviors, reviewers attaining very low FSum values are likely to be genuine reviewers which explain the good results of FSum.

In summary, we can conclude that proposed ASM-HE is effective and outperforms other methods and baselines. ASM-IP and ASM-UP are slightly inferior which is reasonable as they use weaker priors. Learning to rank methods SVMRank and RankBoost using proposed behaviors are strong competitors showing that the proposed behavior features are effective.

[4] No agreement ($\kappa < 0$), slight agreement ($0 < \kappa \leq 0.2$), fair agreement ($0.2 < \kappa \leq 0.4$), moderate agreement ($0.4 < \kappa \leq 0.6$), substantial agreement ($0.6 < \kappa \leq 0.8$), and almost perfect agreement for $0.8 < \kappa \leq 1.0$.

CHAPTER 6

CONCLUSION AND FUTURE DIRECTIONS

This thesis proposes novel solution methodologies for fine-grained opinion mining. Particularly, probabilistic aspect and sentiment models have been proposed for modeling social conversation in debates, discussions and consumer review comments. These models are capable of capturing various fine-grained sentiment dimensions (e.g., agreement, disagreement, question, answer acknowledgement, thumbs up, thumbs down) and discover the sentiment expressions in each dimension. Further, the posterior estimates of the models allow various downstream NLP tasks such as discovering point of contention, questioned aspects in reviews, classifying debate posts as agreeing/disagreeing, classifying authors pairs based on their arguing natures, etc. The base JTE model jointly models topics and linguistic expressions. It also encodes social behaviors like topical interactions (using reply-to relations) and author interactions (through pair structures) which yielded the JTE-R and JTE-P models. Experimental results showed that the proposed models outperformed baselines for our tasks: i) discovering topics and CA-expressions; and ii) for each contentious post, discovering the contention points or topics. Experiments using perplexity and KL-Divergence metrics were also conducted. They showed that the proposed models fit the data better and discover more distinctive topics and CA-expressions. In all experiments, the interaction models JTE-R and JTE-P consistently gave better results. The models can be further extended to capture other socio-linguistic aspects of language in social media such as tolerance, partisan attachment, popularity of a debate topic, thread volume, predicting responses by authors, etc. We defer these to our future work.

Chapter 3 studies the core problem of aspect extraction in sentiment analysis. Specifically, semi-supervised techniques have been proposed using user defined seeds to discover more coherent aspects

and address the issue of semantically incoherent clustering in fully unsupervised aspect extraction models. To our knowledge, prior works do not deal with this problem. Yet, it is important because in practice the user often has something in mind to find. The results obtained in a completely unsupervised manner may not suit the user's need. To solve this problem, we proposed two models SAS and ME-SAS which take seeds reflecting the user needs to discover specific aspects. ME-SAS also does not need any additional help from the user in its Max-Ent training. Our results showed that both models outperformed two state-of-the-art existing models ME-LDA and DF-LDA by large margins. The models have further room for improvement. For instance, there is no notion of *word sense* encoded in the SAS and ME-SAS models. It also cannot deal with wrong knowledge or seed sets. It is also desirable to have the model standalone and leverage large review bases or knowledge bases (e.g., Wikipedia, WordNet) to generate vast amounts of knowledge automatically. A plausible approach is to run frequent pattern mining on topics (discovered from LDA) as transactions. The frequent patterns are likely to be semantically coherent pieces of knowledge which may be used as seed sets. We defer these approaches to future work.

Chapter 4 explores the notion of group opinion spam. Filtering opinion spam is an important precondition for reliable opinion mining. There have been previous studies which have addressed the notion of individual opinion spammers and fake reviews. However, group spam is more damaging as it can take total control of the sentiment on a product. Chapter 4 proposes a method to detect group spammers in product reviews. The proposed method first used frequent itemset mining to find a set of candidate groups, from which a labeled set of spammer groups was produced. We then proposed several behavior features derived from collusion among fake reviewers. A novel relation-based model, called GSRank, was proposed which can consider relationships among groups, individual reviewers, and products they reviewed to detect spammer groups in a mutual reinforcement framework. This model is very different from the traditional supervised learning approach to spam detection. Experimental results

showed that GSRank significantly outperformed the state-of-the-art supervised classification, regression, and learning to rank algorithms.

Next, Chapter 5, throws light on unsupervised anomaly detection, i.e., detecting opinion spam in the absence of labeled ground truth data. Specifically, a generative Bayesian clustering method is proposed which exploits the observed reviewing behaviors to detect opinion spammers (fake reviewers). To our knowledge, this is the first such attempt. Existing methods are mostly based on heuristics and/or ad-hoc labels for opinion spam detection. The proposed model has its basis in the theoretical foundation of probabilistic model based clustering. The Bayesian framework facilitates characterization of many behavioral phenomena of opinion spammers using the estimated latent population distributions. It also enables detection and posterior density analysis in a single framework. This cannot be done by any of the existing methods. Additionally, a novel technique to evaluate the results of unsupervised opinion spam models using supervised classification was proposed without the need of any manually labeled data. Finally, a comprehensive set of experiments based on the proposed automated classification evaluation and human expert evaluation have been conducted to evaluate the proposed model. The results across both evaluation metrics show that the proposed model is effective and outperforms strong competitors. There are yet more research questions that remain unexplored in the content of opinion spam. For instance, how users behave temporally and how much spam is accumulated over time across businesses/entities? Is there any competition among entities and if yes how do those affect the opinion spamming dynamics? Further how much does opinion spam actually impact sales. We defer these research questions to our future work.

CITED LITERATURE

Agarwal, R. Rajagopalan, S. Srikant, R. Xu. Y. 2003. Mining newsgroups using networks arising from social behavior. Proceedings ACM conference on World Wide Web.

Agrawal, R., & Srikant, R. 1994. Fast algorithms for mining association rules. Proceedings of the 20th international conference on very large data bases, VLDB (Vol. 1215, pp. 487-499).

Andrzejewski, D., Zhu, X. and Craven, M. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. Proceedings of International Conference on Machine Learning.

Bishop, C.M. 2006. Pattern Recognition and Machine Learning. Springer.

Blei, D., Ng, A., Jordan, M. 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research.

Blei, D. and Lafferty J. 2009. Visualizing topics with multi-word expressions. Tech. Report. 2009. arXiv:0907.1013v1.

Blei, D. and McAuliffe, J. 2007. Supervised topic models. Proceedings of Neural Information Processing Systems.

Brody, S. and Elhadad, S. 2010. An Unsupervised Aspect-Sentiment Model for Online Reviews. Proceedings of the 2010 Annual Conference of the North American Chapter of the ACL.

Burfoot, C., Bird, S., Baldwin, T. Collective Classification of Congressional Floor-Debate Transcripts. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.

Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., and Vigna, S. 2006. A reference collection for web spam. SIGIR Forum 40, 2, 11–24, S.

Chang, J., Boyd-Graber, J., Wang, C. Gerrish, S. and Blei, D. 2009. Reading tea leaves: How humans interpret topic models. Proceedings of Neural Information Processing Systems.

Chen, Z., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., and Ghosh, R. 2013a. Exploiting Domain Knowledge in Aspect Extraction. Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Chen, Z., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., and Ghosh, R. 2013b. Leveraging Multi-Domain Prior Knowledge into Topic Models. Proceedings of the 23rd International Joint Conference on Artificial Intelligence.

Chen, Z., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., and Ghosh, R. 2013c. Discovering Coherent Topics using General Knowledge. Proceedings of the ACM Conference of Knowledge and Information Extraction. doi:10.1145/2505515.2505519.

Chirita, P.A., Diederich, J., and Nejdl, W. 2005. MailRank: using ranking for spam detection. Proceedings of the ACM Conference of Knowledge and Information Extraction.

Duda, R. O., Hart, P. E., and Stork, D.J. 2001. Pattern Recognition. Wiley.

Easley D. and Kleinberg. J. 2010. Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge.

Fayyad, U. M. and Irani, K. B. 1993. Multi-interval discretization of continuous-valued attributes for classification learning. Proceedings of International Joint Conference on Artificial Intelligence.

Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., Ghosh, R.. 2013. Exploiting Burstiness in Reviews for Review Spammer Detection. Proceedings of the 7th International AAAI Conference on Weblogs and Social Media.

Fleiss, J. L. 1971. Measuring nominal scale agreement among many raters. Psychological Bulletin, 76(5), pp. 378–382, 1971.

Freund, Y., Iyer, R., Schapire, R. and Singer, Y. 2003. An efficient boosting algorithm for combining preference. Journal of Machine Learning Research.

Griffiths, T. and Steyvers, M. 2004. Finding scientific topics. Proceedings of the National Academy of Sciences.

Heinrich, G. 2009. A Generic Approach to Topic Models. Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.

Hofmann, T. 1999. Probabilistic Latent Semantic Analysis. Proceedings of the conference on Uncertainty in Artificial Intelligence.

Hu, M. and Liu, B. 2004. Mining and summarizing customer reviews. Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Jindal, N. and Liu, B. 2008. Opinion spam and analysis. Proceedings of ACM conference of Web Search and Data Mining.

Jindal, N., Liu, B. and Lim, E. P. 2010. Finding Unusual Review Patterns Using Unexpected Rules. Proceedings of the ACM Conference of Knowledge and Information Extraction.

Jo, Y. and Oh, A. 2011. Aspect and sentiment unification model for online review analysis. Proceedings of the ACM Conference in Web Search and Data Mining.

Joachims, T. 1999. Making large-scale support vector machine learning practical. Advances in Kernel Methods, MIT Press.

Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Kawamae, N. 2010. Latent interest-topic model. Proceedings of Conference in Information and Knowledge Management.

Kim, S., P. Pantel, T. Chklovski, and M. Pennacchiotti. 2006. Automatically assessing review helpfulness. Proceedings of Empirical Methods in Natural Language Processing.

Klementiev, A., Roth, D. and Small, K. 2007. An Unsupervised Learning Algorithm for Rank Aggregation. Proceedings of European Conference on Machine Learning.

Landis, J. R. and Koch, G. G. 1977. The measurement of observer agreement for categorical data. Biometrics, 33, 159-174.

Li, F., Huang, M., Yang, Y. and Zhu, X. 2011. Learning to identify review Spam. Proceedings of International Joint Conference on Artificial Intelligence.

Lim, E. Nguyen, V. A., Jindal, N., Liu, B., and Lauw, H. 2010. Detecting Product Review Spammers Using Rating Behavior. Proceedings of ACM International Conference on Information and Knowledge Management.

Lin, C. and He, Y. 2009. Joint sentiment/topic model for sentiment analysis. Proceedings of ACM International Conference on Information and Knowledge Management.

Liu, B. 2012. Sentiment analysis and opinion mining. Morgan & Claypool Publishers.

Liu, T-Y. 2009. Learning to Rank for Information Retrieval. Foundations and Trends in Information Retrieval, 3(3): 225-331. 2009.

Mukherjee, A. and Liu, B. 2013. Discovering User Interactions in Ideological Discussions. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A., Venkataraman, A., Liu, B., Meraz, S. 2013a. Public Dialogue: Analysis of Tolerance in Online Discussions. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M. Castellanos, M., and Ghosh, R. 2013b. Spotting Opinion Spammers using Behavioral Footprints. Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. doi:10.1145/2487575.2487580.

Mukherjee, A., Venkataraman, V., Liu, B., Glance, N. 2013c. What Yelp Fake Review Filter might be Doing? Proceedings of the 7th International AAAI Conference on Weblogs and Social Media.

Mukherjee, A., Liu, B., and Glance, N. 2012. Spotting Fake Reviewer Groups in Consumer Reviews. Proceedings of the ACM International World Wide Web Conference. doi: 10.1145/2187836.2187863.

Mukherjee, A. and Liu, B. 2012a. Mining Contentions from Discussions and Debates. Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. doi:10.1145/2339530.2339664.

Mukherjee, A., Liu, B. 2012b. Aspect Extraction through Semi-Supervised Modeling. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A. and Liu, B. 2012c. Modeling Review Comments. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A., Liu, B., Wang, J., Glance, N., Jindal, N. 2011. Detecting Group Review Spam. Proceedings of the International World Wide Web Conference. doi:10.1145/1963192.1963240.

Mukherjee A. and Liu, B. 2010. Improving Gender Classification of Blog Authors. Proceedings of the conference on Empirical Methods in Natural Language Processing.

Murakami A., and R. Raymond, 2010. Support or Oppose? Classifying Positions in Online Debates from Reply Activities and Opinion Expressions. Proceedings of International Conference on Computational Linguistics.

Ott, M., Choi, Y., Cardie, C. Hancock, J. 2011. Finding Deceptive Opinion Spam by Any Stretch of the Imagination. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.

Pang, B. and Lee, L. 2008. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval.

Rosen-Zvi, M., T. Griffiths, M. Steyvers, and P. Smith. 2004. The author-topic model for authors and documents. Uncertainty in Artificial Intelligence.

Si, J., Mukherjee, A., Liu, B., Li, Q., and Li, H. 2013. Exploiting Topic based Twitter Sentiment for Stock Prediction. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Smyth, P. 1999. Probabilistic Model-Based Clustering of Multivariate and Sequential Data. Proceedings of Conference on Artificial Intelligence and Statistics.

Somasundaran, S., J. Wiebe. 2009. Recognizing stances in online debates. Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP.

Teh, Y., M. Jordan, M. Beal and D. Blei. 2006. Hierarchical Dirichlet Processes. Journal of the American Statistical Association.

Thomas, M., B. Pang and L. Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. Proceedings of Empirical Methods in Natural Language Processing.

Titov, I. and R. McDonald. 2008. Modeling online reviews with multi-grain topic models. Proceedings of International Conference on World Wide Web.

Vogt, C.C., Cottrell, G.W. 1999. Fusion via a linear combination of scores. Information Retrieval. (1999), 151–173.

Wallach, H. 2006. Topic modeling: Beyond bag of words. Proceedings of International Conference in Machine Learning.

Wang, G., Xie, S., Liu, B., and Yu, P. S. 2011. Review Graph based Online Store Review Spammer Detection. Proceedings of the International Conference on Data Mining.

Wang, Z. 2010. Anonymity, Social Image, and the Competition for Volunteers: A Case Study of the Online Market for Reviews. The B.E. Journal of Economic Analysis & Policy. 10, 1 (Jan. 2010), 1–34.

Xie, S., Wang, G., Lin, S. and Yu, P.S. 2012. Review spam detection via temporal pattern discovery. Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Zhai, Z., Liu, B. Xu, H. and Jia, P. 2010. Grouping Product Features Using Semi-Supervised Learning with Soft-Constraints. Proceedings of International Conference on Computational Linguistics.

Zhang, Z. and B. Varadarajan. 2006. Utility scoring of product reviews. Proceedings of ACM International Conference on Information and Knowledge Management.

Zhao, X., J. Jiang, H. Yan, and X. Li. 2010. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. Proceedings of Empirical Methods in Natural Language Processing.

VITA

NAME: Arjun Mukherjee

EDUCATION: Ph.D., Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2014.

B.Tech., Computer Science and Engineering, Sikkim Manipal Institute of Technology, India, 2009.

HONORS: Dean's Scholar Fellowship, 2013.

Chancellor's Graduate Research Fellowship, 2013, 2012.

NSF SoCS Doctoral Symposium Scholarship, 2013.

Facebook Ph.D. Fellowship (Finalist), 2013.

EMNLP-CoNLL Best Reviewer Award, 2012.

Provost's & Deiss Fellowship for Graduate Research, 2012.

KDD NSF Student Travel Award, 2012.

PUBLICATIONS: Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M., Castellanos, M., and Ghosh, R. 2013. Spotting Opinion Spammers using Behavioral Footprints. Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. doi:10.1145/2487575.2487580.

Chen, Z., Mukherjee, A., Liu, B., Hsu, M., Castellanos, C., and Ghosh, R. 2013. Exploiting Domain Knowledge in Aspect Extraction. Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Chen, Z., Arjun Mukherjee, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Discovering Coherent Topics using General Knowledge. Proceedings of the ACM Conference of Knowledge and Information Extraction. doi:10.1145/2505515.2505519.

Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H. 2013. Exploiting Topic based Twitter Sentiment for Stock Prediction. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A., Venkataraman, V., Liu B., Meraz, S. 2013. Public Dialogue: Analysis of Tolerance in Online Discussions. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A., Liu, B. 2013. Discovering User Interactions in Ideological Discussions. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A., Venkataraman, V., Liu, B., Glance, N. 2013. What Yelp Fake Review Filter might be Doing? Proceedings of the 7th International AAAI Conference on Weblogs and Social Media.

Chen, Z., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., Ghosh, R. 2013. Leveraging Multi-Domain Prior Knowledge into Topic Models. Proceedings of the 23rd International Joint Conference on Artificial Intelligence.

Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., Ghosh, R. 2013. Exploiting Burstiness in Reviews for Review Spammer Detection. Proceedings of the 7th International AAAI Conference on Weblogs and Social Media.

Mukherjee, A. and Liu, B. 2012. Mining Contentions from Discussions and Debates. Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. doi:10.1145/2339530.2339664.

Mukherjee, A., Liu, B., and Glance, N. 2012. Spotting Fake Reviewer Groups in Consumer Reviews. Proceedings of the ACM International World Wide Web Conference. doi: 10.1145/2187836.2187863.

Mukherjee, A. and Liu, B. 2012. Aspect Extraction through Semi-Supervised Modeling. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A. and Liu, B. 2012. Modeling Review Comments. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.

Mukherjee, A., Liu, B., Wang, J., Glance, N., Jindal, N. 2011. Detecting Group Review Spam. Proceedings of the International World Wide Web Conference. doi:10.1145/1963192.1963240.

Mukherjee, A. and Liu, B. 2010. Improving Gender Classification of Blog Authors. Proceedings of the conference on Empirical Methods in Natural Language Processing.