Scalable Topic-Specific Influence Analysis on Microblogs

Bin Bi
UCLA
bbi@cs.ucla.edu

Yuanyuan Tian
IBM Almaden Research Center
ytian@us.ibm.com

Yannis Sismanis[◇]
Google
yannis@google.com

Andrey Balmin[*]
GraphSQL
andrey@graphsql.com

Junghoo Cho
UCLA
cho@cs.ucla.edu

ABSTRACT

Social influence analysis on microblogs, such as Twitter, has been playing a crucial role in online advertising and brand management. While most previous influence analysis schemes rely only on the links between users to find key influencers, they omit the important text content created by the users. As a result, there is no way to differentiate the social influence in different aspects of life (topics). Although a few prior works do support topic-specific influence analysis, they either separate the analysis of content from that of network structure, or assume that content is the only cause of links, which is clearly an inappropriate assumption for microblog networks.

To address the limitations of the previous approaches, we propose a novel Followship-LDA (FLDA) model, which integrates both content topic discovery and social influence analysis in the same generative process. This model properly captures the content-related and content-independent reasons why a user follows another in a microblog network. We demonstrate that FLDA produces results with significantly better precision than existing approaches. Furthermore, we propose a distributed Gibbs sampling algorithm for FLDA, and demonstrate that it provides excellent scalability on large clusters. Finally, we incorporate the FLDA model in a general search framework for topic-specific influencers. A user freely expresses his/her interest by typing a few keywords, the search framework will return a ranked list of key influencers that satisfy the user's interest.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

1. INTRODUCTION

Microblogging services, such as Twitter (`twitter.com`), have gained tremendous popularity in recent years. Using these services, a user can publish a short message, called a *tweet*, and *follow* other users to keep up with their latest updates. The "follow" relationship (or *followship*) is directed, with information only flowing from the *followee* to the *follower*. A large amount of microblog data has been accumulated over time. For example, according to a March 2012 report, Twitter had over 500 million registered users creating over 340 million tweets daily [27].

The rich text and social information in microblogs has become a popular resource for marketing campaigns to monitor the opinions of consumers on particular products and to launch viral advertising. Identifying key influencers in microblogs is required for such marketing activities. Although a lot of work has been done on social influence analysis, most of these studies [8, 16, 17, 18, 9] infer influence only from the network structure, while ignoring the valuable text content that the users created. As a result, the learned influence of each user is only *global*, with no way to assess the influence in a particular aspect of life (topic). For example, no one can deny that President Obama is a key influencer in general. But his impact is most prominent in politics. In other subjects, like choosing digital cameras, he is unlikely to be influential. Clearly, *topic-specific* influence analysis provides a more detailed influence portfolio for a user, which is critical for effective marketing.

A number of PageRank-based methods, such as Topic-Sensitive PageRank [15] and TwitterRank [28], are able to compute per-topic influence ranks, but they require the topics to be already created either manually or by a topic modeling preprocess. As content and links are related to each other in a microblog network, the separation between the analysis on content and the analysis on the network structure usually leads to inferior performance, compared to those methods, like Link-LDA [10], which can detect topics and infer influences at the same time. However, Link-LDA, as originally designed for citation networks, assumes that the generation of links is purely based on the content. This assumption clearly does not apply to microblogs, since it is prevalent for a user to follow celebrities simply because of their fame and stardom, with nothing to do with what he/she actually tweets about.

To correctly model topic-specific influence on microblogs, we propose a new Bernoulli-Multinomial mixture model, called Followship-LDA (FLDA). This model contains two levels of mixtures: an upper-level Bernoulli mixture with one of the components being a Multinomial mixture. FLDA jointly models text and followship in the same generative process. Furthermore, it is able to differentiate the different reasons why a user follows another. Sometimes, A follows B because they tweet in similar topics. This type of followship is content-based. In other times, A follows B purely because B is a pop star. In this case, the followship is content-independent. Using FLDA, we can not only learn the per-user preference of following by content or not, but also remove the stardom effect when computing the topic-specific influence. Our empirical study on two popular microblog datasets, Twitter and Tencent Weibo, shows that

[◇*] This work was done while the authors were at IBM Almaden Research Center.

WSDM'14, February 24–28, 2014, New York, New York, USA.

http://dx.doi.org/10.1145/2556195.2556229.

the FLDA model produces significantly higher quality results than the prior arts.

Gibbs sampling is a widely used approach to approximate target distributions for LDA-like Bayesian models. To meet the computational challenge posed by rapid growing microblog data, we propose a *distributed* Gibbs sampling algorithm which significantly speeds up the Gibbs sampling process. For example, a sequential job that would take 21 days on a high end server can finish in 1.5 days using the distributed algorithm on a cluster of 27 commodity machines! We chose to implement the distributed Gibbs sampling on top of the Spark cluster computing framework [29]. Several alternative platforms [7, 3, 5] have been proposed to address the problem of machine-learning at scale. Spark is such a parallel programming framework, which supports efficient iterative algorithms on datasets stored in the aggregate memory of a cluster. We pick Spark as the underlying framework, because of its extreme flexibility as far as cluster programming is concerned. In addition to machine-learning algorithms, which were the main motivation behind the design of Spark, various data-parallel applications can be expressed and executed efficiently using Spark; examples include MapReduce, Pregel[20], HaLoop[7] and many others(see [29]).

Finally, we propose a general search framework for topic-specific key influencers, which can flexibly plug in various topic-specific influence methods, including FLDA, Link-LDA, Topic-Sensitive PageRank and TwitterRank. A user just needs to enter a set of keywords to describe his/her interest, the search framework will infer a topic distribution from the keywords and return a ranked list of influencers in the corresponding topic combination.

In particular, this paper makes the following contributions:

- We propose a new Bayesian Bernoulli-Multinomial mixture model, FLDA, to jointly model both content and links in the same generative process, while separating the various reasons why a user follows another in a microblog network.
- We discuss and implement a distributed Gibbs-sampling technique for training FLDA over large clusters.
- We propose a general search framework for finding topic-specific key influencers with various models (including FLDA, Link-LDA and PageRank variants).
- Through extensive experimentation with two large real datasets, we demonstrate (a) the substantial better precision achieved by FLDA than previous work, (b) the excellent scalability of the distributed Gibbs-sampling technique over large clusters and (c) various interesting insights gained from the real datasets.

The rest of this paper is organized as follows. In Section 2, we describe the related work. The FLDA model is introduced in Section 3. Section 4 presents the distributed Gibbs sampling algorithm for FLDA, while Section 5 provides an overview of the general search framework for topic-specific key influencers. In Section 6, we present our experimental results. Finally we conclude the paper in Section 7.

2. RELATED WORK

Much work has been done on influence analysis in social networks. Kempe et. al. pioneered the Linear Threshold Model and Independent Cascade Model to explain the spread of influence in a social network and abstracted the key influencer problem into a maximization problem [17]. Along with subsequent works, such as [18] and [9], these methods are only after the identification of *global* influencers instead of influencers for specific topics. Although Barbieri et. al. extended the Linear Threshold Model and Independent Cascade Model to be topic-aware [2], the topics are still obtained based on the network structure, while totally ignoring the valuable content information.

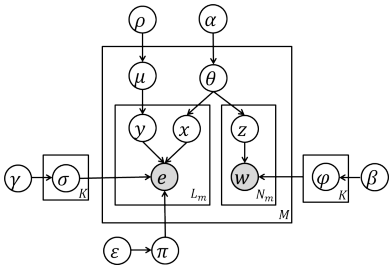


Figure 1: Followship-LDA

Given the popularity of PageRank [6], it is only natural to extend it for topical influence analysis. Topic-Sensitive PageRank (TSPR) [15] was such an extension for computing per-topic PageRank scores. TSPR biases the computation of PageRank by replacing the classic PageRank's uniform teleport vector with topic-specific ones. However, it requires a separate preprocess to create topics and provide per-topic teleport vectors. This preprocess can be done by either utilizing existing manually categorized topic hierarchies, such as suggested in [15], or applying well-known topic modeling methods like LDA [4] on the text content, as suggested in [28].

In [28], TwitterRank was proposed to find topic-level influencers on Twitter. A set of topics is first produced by LDA on the tweets. Then TwitterRank applies a method similar to TSPR to compute the per-topic influence rank. The transition probability between two users in TwitterRank is defined based on the number of tweets published by different followees and the topical similarity between the follower and the followee.

In the context of documents and citations (or hyperlinks), a mixed membership model was proposed in [10] to jointly model text and citations in the same generative process, which we will refer to as Link-LDA. In the generative process of Link-LDA, for a given document, a citation to another document is created in exactly the same way as a word is created, and they share the same per-document topic distribution. If we aggregate all the tweets for each user, and treat a user as a document and his/her followships to other users as citations, then Link-LDA can be applied to the microblog network, to learn the probability of each microblog user u being followed by someone given a specific topic t. This probability can be used to measure u's influence on the topic t. Link-PLSA-LDA [22] is an extension to Link-LDA, but also assumes that the cause of links is purely based on the text content.

In [26], Tang et al. proposed a Topical Affinity Propagation (TAP) model for topic-level social influence. But, similar to TSPR and TwitterRank, TAP requires a separate topic modeling approach to be applied first to derive a set of topics on the content.

In [24], Pal et. al. proposed to identify topical authorities by clustering users using 15 carefully selected features and then rank users within each cluster. Cognos [11] heavily relies on the manually curated Twitter "Lists" to infer topics of expertise and rank experts for different topics. Liu et al. introduced a graphical model to learn influence in the context of general heterogeneous networks [19].

3. FOLLOWSHIP-LDA

The existing works on topic-specific influence analysis can be categorized into two camps. The first camp, represented by TSPR and TwitterRank, completely detach the topic detection process

Table 1: Notations used in FLDA

Notation	Description
θ	Per–user topic distribution
φ	Per-topic word distribution
σ	Per-topic followee distribution
π	Multinomial distribution over followees
μ	Per-user Bernoulli distribution over indicators
$\alpha, \beta, \gamma, \epsilon, \rho$	Parameters of the Dirichlet (Beta) priors on Multinomial (Bernoulli) distributions
w	Word identity
e	Followee identity
z	Identity of the topic of a word
x	Identity of the topic of a followee
y	Binary indicator of whether a followship is related to the content of tweets
M	Number of unique users
V	Number of words in the vocabulary
K	Number of unique topics
N_m	Number of words in the tweets of user m
L_m	Number of followees for user m

Choose $\pi \sim Dirichlet(\varepsilon)$
For each topic $k = 1, \dots, K$
  Choose $\varphi_k \sim Dirichlet(\beta)$
  Choose $\sigma_k \sim Dirichlet(\gamma)$
For each user $m = 1, \dots, M$
  Choose $\theta_m \sim Dirichlet(\alpha)$
  For the n^{th} word of the m^{th} user, where $n \in \{1, \dots, N_m\}$
    Choose a topic $z_{m,n} \sim Multinomial(\theta_m)$, where $z_{m,n} \in \{1, \dots, K\}$
    Choose a word $w_{m,n} \sim Multinomial(\varphi_{z_{m,n}})$, where $w_{m,n} \in \{1, \dots, V\}$
  Choose $\mu_m \sim Dirichlet(\rho)$
  For the l^{th} link of the m^{th} user, where $l \in \{1, \dots, L_m\}$
    Choose a topic $x_{m,l} \sim Multinomial(\theta_m)$, where $x_{m,l} \in \{1, \dots, K\}$
    Choose an indicator $y_{m,l} \sim Bernoulli(\mu_m)$, where $y_{m,l} \in \{0,1\}$
    If $y_{m,l} = 0$ then
      Choose a followee $e_{m,l} \sim Multinomial(\pi)$, where $e_{m,l} \in \{1, \dots, M\}$
    Else if $y_{m,l} = 1$ then
      Choose a followee $e_{m,l} \sim Multinomial(\sigma_{x_{m,l}})$, where $e_{m,l} \in \{1, \dots, M\}$

Figure 2: Generative process for Followship-LDA

from the influence analysis. As we will show later in this paper, these methods perform inferior to those approaches in the second camp that integrate text topic discovery and social influence analysis in the same model. Link-LDA represents the best prior work in the second camp. However, it was originally developed for citation and hyperlink networks. In Link-LDA, the topic assignments for words and for citations are drawn from the same topic distribution θ, assuming that the content of a document is topically related to that of its cited documents. This is a very reasonable assumption for citation/hyperlink networks, since an author most definitely chooses the topically related documents to cite. But this assumption no longer applies to microblog networks. There are many reasons for a microblog user to follow another. Some are content-related (they tweet in similar topics) and others are not. For example, President Obama has a massive number of followers in Twitter, but some of them have never tweeted about politics at all. It is very common to see users follow celebrities, not because they share any topic of interest, but just because they are famous and popular. Clearly, Link-LDA is not able to capture these non-content related factors in the influence analysis.

To correctly model the topics and social influence in microblog networks, we propose **Followship-LDA**, abbreviated as FLDA. The graphical model for FLDA is depicted in Figure 1, with the notations described in Table 1. The generative process of a user's content and links/followees is summarized in Figure 2.

For the generation of content, each user is viewed as a mixture of latent topics from which words are drawn, similar to LDA. To be more specific, for the m^{th} user, we first pick the per-user topic distribution θ_m from a Dirichlet prior with parameter α. Then, to generate the n^{th} word for the tweets of the user, a topic $z_{m,n}$ is first chosen from θ_m. Finally, the word $w_{m,n}$ is picked from the per-topic word distribution $\varphi_{z_{m,n}}$.

On the other hand, the links of the m^{th} user are generated by a much more complex three-stage stochastic process. First of all, every user has a unique preference of following others based on content or non-content reasons. The Bernoulli distribution μ_m characterizes this per-user preference. As a result, for the l^{th} link/followee of the m^{th} user, we first consult μ_m to decide on the value of the binary variable $y_{m,l}$. $y_{m,l} = 1$ indicates that the link creation is based on the user's content, whereas $y_{m,l} = 0$ means that content has nothing to do with the link. Now if $y_{m,l} = 1$, we use the same topic distribution θ_m to pick a topic $x_{m,n}$ of interest, just as in the content generation part of FLDA. Afterward, we choose a followee $e_{m,l}$ who well addresses the picked topic from the per-topic followee distribution $\sigma_{x_{m,l}}$. When $y_{m,l} = 0$, the user is following someone for non-content reasons. We use π to capture this probability distribution. In particular, a followee $e_{m,l}$ is chosen from the Multinomial distribution π.

Note that FLDA is a much more complex mixture model than LDA and Link-LDA. We call it a Bernoulli-Multinomial mixture model, because the model consists of two levels of mixtures: an upper-level Bernoulli mixture that includes a Multinomial mixture underneath. More specifically, each followee e of a user m is drawn from a Bernoulli mixture of two components. One of the mixture components is a Multinomial distribution with parameter π, corresponding to the global popularity. The other component, however, is itself a mixture of K Multinomial components, each corresponding to a topic. The distribution of followee e of user m is: $p(e|\mu, \pi, \theta, \sigma) = \mu_{m,0}\pi_e + \mu_{m,1} \sum_{k=1}^{K} \theta_{m,k}\sigma_{k,e}$, where μ are the outer mixing proportions, and θ are the inner mixing proportions.

The various probability distributions we can learn from the FLDA model characterize the different factors that affect the textual and social structures of a microblog network. For a user m, the probability $\theta_{z|m}$ represents the likelihood of m tweeting about topic z, and $\mu_{y|m}$ is the probability of the reason indicator y (content-related or not) why the user m follows others. For content of tweets, $\varphi_{w|z}$ gives the probability of word w belonging to topic z. In terms of links, $\sigma_{e|x}$ captures the likelihood of a user e being followed by someone for a given topic x. This value essentially quantifies the influence of user e on x and is exactly the topic-specific influence score we want to compute. Finally, π_e indicates the probability of a user e being followed for any non-content reason. In some sense, π_e is measuring the global popularity of e. We formally define:

Topic-Specific Influence: the influence of user e on topic x is measured by $\sigma_{e|x}$ which is the probability of e being followed for topic x in the FLDA model.

Content-Independent Popularity: the content-independent popularity of user e is measured by π_e which is the probability of e being followed for any content-independent reason in the FLDA model.

3.1 Gibbs Sampling for FLDA

To learn the various distributions in the FLDA model, we use collapsed Gibbs sampling. However, the derivation of posterior distributions for Gibbs sampling in FLDA is complicated by the fact that followee distribution is a joint distribution of two-level mixtures. As a result, we need to compute the joint distribution of x and y in the Gibbs sampling process. The posterior distributions for Gibbs sampling in FLDA are given in the equations below. The detailed derivation of these equations is provided in the appendix.

$$p(z_{m,n}|z_{-(m,n)}, x, w, e, y, \alpha, \beta, \gamma, \varepsilon, \rho) \propto \frac{(c^{-(m,n)}_{z_{m,n},n,m,*} + d_{z_{m,n},n,m,*,*} + \alpha_{z_{m,n}})(c^{-(m,n)}_{z_{m,n},n,*,w_{m,n}} + \beta_{w_{m,n}})}{c^{-(m,n)}_{z_{m,n},n,*,*} + \sum_{i=1}^{W} \beta_i} \quad (1)$$

$$p(x_{m,l}, y_{m,l} = 0 | y_{-(m,l)}, x_{-(m,l)}, w, z, e, \alpha, \beta, \gamma, \varepsilon, \rho)$$
$$\propto (c_{x_{m,l},m,*} + d^{-(m,l)}_{x_{m,l},m,*,*} + \alpha_{x_{m,l}})(d^{-(m,l)}_{*,m,*,0} + \rho_0) \times \frac{d^{-(m,l)}_{*,*,e_{m,l},0} + \varepsilon_{e_{m,l}}}{d^{-(m,l)}_{*,*,*,0} + \sum_{i=1}^{M} \varepsilon_i} \quad (2)$$

$$p(x_{m,l}, y_{m,l} = 1 | y_{-(m,l)}, x_{-(m,l)}, w, z, e, \alpha, \beta, \gamma, \varepsilon, \rho)$$
$$\propto (c_{x_{m,l},m,*} + d^{-(m,l)}_{x_{m,l},m,*,*} + \alpha_{x_{m,l}})(d^{-(m,l)}_{*,m,*,1} + \rho_1) \times \frac{d^{-(m,l)}_{x_{m,l},*,e_{m,l},1} + \gamma_{e_{m,l}}}{d^{-(m,l)}_{x_{m,l},*,*,1} + \sum_{i=1}^{M} \gamma_i} \quad (3)$$

In the above equations, $z_{m,n}$ denotes the topic of the n^{th} word for the m^{th} user, and $y_{m,l}$ is the reason indicator (content or non-content) of the l^{th} link for the m^{th} user. $w_{m,n}$, $x_{m,l}$ and $e_{m,l}$ follow similar definitions. Let $z_{-(m,n)}$ denote the topics for all words except $z_{m,n}$, and $y_{-(m,l)}$ and $x_{-(m,l)}$ follow an analogous definition. We define $c_{z,m,w}$ as the number of times word w is assigned to topic z for the m^{th} user, and $d_{x,m,e,y}$ as the number of times link e is assigned to topic x for the m^{th} user with indicator y. If any of the dimensions in above notations is not limited to a specific value, we use $*$ to denote. Essentially, $*$ represents an aggregation on the corresponding dimension. For example, $c_{z,*,w}$ is the total number of times word w is assigned to topic z in the entire document collection. Finally, let $c^{-(m,n)}_{z,m,w}$ be the same meaning of $c_{z,m,w}$ only with the n^{th} word for the m^{th} user excluded. Similarly, $d^{-(m,l)}_{x,m,e,y}$ is defined in the same way as $d_{x,m,e,y}$ only without the count for the l^{th} link for the m^{th} user.

After the sampling algorithm has run for an appropriate number of iterations (until the chain has converged to a stationary distribution), the estimates for the parameters of θ, φ, μ, σ and π can be obtained via the following equations:

$$\theta_{x|m} = \frac{c_{x,m,*} + d_{x,m,*,*} + \alpha_x}{c_{*,m,*} + d_{*,m,*,*} + \sum_{i=1}^{K} \alpha_i} \quad (4)$$

$$\varphi_{w|z} = \frac{c_{z,*,w} + \beta_w}{c_{z,*,*} + \sum_{i=1}^{W} \beta_i} \quad (5)$$

$$\mu_{y|m} = \frac{d_{*,m,*,y} + \rho_y}{d_{*,m,*,*} + \rho_0 + \rho_1} \quad (6)$$

$$\sigma_{e|x} = \frac{d_{x,*,e,1} + \gamma_e}{d_{x,*,*,1} + \sum_{i=1}^{M} \gamma_i} \quad (7)$$

$$\pi_e = \frac{d_{*,*,e,0} + \varepsilon_e}{d_{*,*,*,0} + \sum_{i=1}^{M} \varepsilon_i} \quad (8)$$

4. SCALABLE GIBBS SAMPLING FOR FLDA

The rapid growth of microblog data poses a significant challenge for influence analysis in terms of both computation time and memory requirements. Scalable solutions that can take advantage of the computation power and memory capacity of multiple computers are becoming more crucial. However, the Gibbs sampling updates of FLDA shown in equations (1)-(3) are inherently sequential, which makes it very difficult to parallelize the computation. However, given the abundance of words and the large number of links in a microblog dataset, the dependency between different topic assignments or indicator assignments in equations (1)-(3) is relatively weak. As a result, we can relax the sequential requirement of the Gibbs sampling updates and distribute the computation to a number of processes running in parallel. In fact, similar observations were used to develop approximate parallel/distributed Gibbs sampling algorithm for LDA in [23], [25] and [1]. We implemented our distributed FLDA Gibbs sampling algorithm on a distributed cluster computing framework called Spark [29]. Before the details of our distributed algorithm, we first provide a brief overview of Spark.

```
1  val baseRDD = sc.textFile("hdfs://master/baseData.log")
2  val lowerRDD = lines.map(String.toLowerCase _)
3  val regexB = sc.broadcast(REGEX)
4  val nMatches = sc.accumulator(0)
5  lowerRDD.foreach (s =>
6      if ( s.matches(regexB.value) )
7       nMatches += 1
8  )
9   println ("#Matches is:\%d".format(nMatches.value))
```

Listing 1: Sample Spark code.

4.1 Spark Overview

Spark is a large-scale distributed processing framework specifically targeted at machine-learning iterative workloads. It uses a functional programming paradigm, and applies it on large clusters by providing a fault-tolerant implementation of distributed data sets called Resilient Distributed Data (RDD). RDDs can either reside in the aggregate main-memory of the cluster, or in efficiently serialized disk blocks. Especially for iterative processing, the opportunity to store the data in main-memory can significantly speed up processing. An RDD contains immutable data; i.e. it cannot be modified, however, a new RDD can be constructed by transforming an existing RDD.

The Spark runtime consists of a single coordinator node and multiple worker nodes. The coordinator keeps track of how to re-construct any partition of the RDD when any of the workers fails.

Computation in Spark is expressed using functional transformations over RDDs. For instance, assume that we have a log file, and that we want to transform each string to lower case. Consider the first two lines of actual Spark code in Listing 1: The first line of code defines an RDD of strings, called `baseRDD`, over a file "baseData.log" stored in a Hadoop Distributed FileSystem; each text line of the log file, corresponds to a string of the RDD. The second line of code, uses the `map` function to transform each string in `baseRDD` through the function `String.toLowerCase`. The transformation happens in parallel on all the workers, and defines a new RDD, called `lowerRDD` that contains the lower-case string of each string in `baseRDD`.

Spark's programming model provides additionally two useful abstractions: broadcast variables and accumulators. Broadcast variables are initialized at the coordinator node, and made available to all worker nodes, through efficient network broadcast algorithms. Spark chooses a topology-aware network-efficient algorithm to disseminate the data. Line 3 in Listing 1 initializes a broadcast variable called `regexB` to a regular expression (called `REGEX`). In Line 6, this value is used inside the `foreach` loop to check if any of the lines in the RDD called `lowerRDD` matches that regular expression. Note that broadcast variables are immutable, read-only, objects and cannot be modified by the workers.

Similar to a broadcast variable, an accumulator is also a variable that is initialized on the coordinator node, and sent to all the worker nodes. However, unlike a broadcast variable, an accumulator is mutable and can be used to aggregate results of computations at worker nodes. Worker nodes may update the state of the accumulator (usually just by incrementing it, or by using computations such as count and sum). At the end of the RDD transformation, each worker node sends its locally-updated accumulator back to the coordinator node, where all the accumulators are combined (using either a default or user-supplied combine function) into a final result. In

our example listing, `nMatches` is an accumulator that is locally incremented by all workers (line 7) before it is globally aggregated (through an implicit addition over all the partial results in line 9).

4.2 Distributed FLDA using Spark

We now describe how we use the Spark framework to implement the distributed Gibbs sampling algorithm for FLDA. In particular we discuss technical issues that distributed approaches encounter, and propose solutions and justify various implementation choices.

First, we define the notion of a *user object*. Each user object corresponds to a single user m, and holds information about the content (i.e. the actual words used by m) and the link structure (i.e. other users that m is following). For each word w and link e, the user object holds the last topic assignment, i.e. the corresponding latent variables z and x. For each link additionally it holds the last binary state (i.e. content-related or content-independent) for the y latent variable. Finally, each user object holds the *user-local* counters $d_{x,m,e,y}$, $c_{x,m,w}$, as well as all aggregates of these (like $d_{*,m,*,*}$) that show up in equations (1)-(3). Note that the corresponding local aggregates always have a m index in the subscript. Such aggregates are entirely local to a user and need not be shared.

Note that in addition to the user-local counters and aggregates, equations (1)-(3) require *global* aggregates (like $d_{x,*,*,1}$) over all the users. Such global aggregates always have a $*$ instead of m in the corresponding subscript index.

Based on previous work ([25]), the global aggregates are not stored in the user object, but are computed periodically and distributed to all workers through an accumulator. The idea is that such aggregates should change slowly and thus any inaccuracies (because of the periodic synchronization) shouldn't affect the quality of the final result. Although this assumption has been shown to work well in practice for basic LDA, it is not evidently clear whether it works for FLDA. A big difference is that the global aggregates, that distributed LDA periodically needs to synchronize, are only per document terms and thus their count is limited (especially after typical pre-processing, like stop-word removal or stemming). However, distributing FLDA requires the periodic synchronization of orders of magnitude more aggregates;not only per document terms but also per user terms (and typically there are many more users). In the experiments, we show that for real datasets with millions of users, distributing FLDA still works very well.

Second, we define a mapping function, `GibbsSampleMap`, which takes as input one such user object, runs Gibbs sampling once and returns a new user object. In particular, this function goes over all the words and links in the object and (a) "undoes" the effects of the last assignment to the latent variables x, y and z (by properly decreasing the corresponding counts $d_{x,m,e,y}$, $c_{x,m,w}$ as well as all the corresponding local and global aggregates), (b) computes the new probabilities for the latent variables x, y and z according to the equations (1)-(3), and finally (c) assigns new latent variables according to these probabilities, and increases the corresponding counts and all user-local and global aggregates.

Putting all these together, first we initialize an RDD of user objects by (a) properly parsing and co-grouping the content and the link structure for each user, (b) randomly initializing the latent variable assignments and (c) computing the corresponding user-local counters and aggregates based on these initial assignments. Then we run a number of iterations over the RDD, where each iteration maps all user objects (in parallel) to new user objects using the `GibbsSampleMap` function we defined above. At the beginning of each iteration we accumulate and broadcast the global aggregates. We note, that each worker has its each own copy of the global aggregates, that the mapping function modifies. Thus although each worker starts with the same global aggregates, as user objects are transformed through the mapping functions, the workers' copies of the global aggregates get "out-of-sync", until the start of the next iteration when new global aggregates are computed and broadcasted. Finally, when all the iterations are done, we use equations (4)-(8) to estimate the parameters θ, φ, μ, σ and π of the FLDA model.

4.3 Discussion

The distributed Gibbs sampling algorithm is quite generic and could be used to train other Bayesian models as well. We emphasize that the final result depends on the assumptions made in [23] and [25]. In particular, the global-aggregates change slowly and thus are not updated continuously, but only once every iteration (or even less often and asynchronously in the case of [25]). This choice does not seem to affect the final result of topic models like LDA. The literature shows that the quality of the result (in terms of perplexity or log-likelihood) is equivalent to that from a purely sequential implementation (where the global aggregates are always updated). We empirically show that the same holds for FLDA. We emphasize that it is not immediately clear whether this approach works for FLDA, since it requires the synchronization of order of magnitude more global aggregates. In Section 6 we compare the resulting user rankings produced by serial and distributed version of the algorithm and show virtually no difference. Finally we note that in the experiments with various real datasets, we observed that computing and synchronizing the global-aggregates just once every ten iterations doesn't seem to affect the quality of the results. It is an open problem exactly how infrequent such global updates can be.

Although Spark provides a lineage based fault-recovery mechanism, we chose to complement it using manual checkpoints for every ten iterations. The reason, is that replaying all the iterations from the beginning for every failed worker (although infrequent) takes quite some time. With the checkpoints, we guarantee that at most ten iterations will have to be replayed in case of failures. Our choice was also based on the fact that the cost of a checkpoint was negligible (a small fraction of the time required to do an iteration).

Finally, we took extra care for the correctness of the distributed Monte-Carlo simulation. Since multiple workers are spawned at roughly the same time, typical random-number generators are seeded with similar (or even exactly the same) seeds. This introduces correlations between the pseudo random numbers generated across the workers. In extreme cases, two workers could "see" exactly the same stream of pseudo-random numbers. Such correlation jeopardizes the quality of the returned results. To guarantee correctness of the distributed simulation, we use the technique discussed in [14] for generating multiple streams of uniform numbers that are provably independent. In particular, we assign a unique stream for each RDD block and iteration pair (i.e. if we have 100 RDD blocks and 500 iterations, we have 100×500 independent streams of random numbers). This approach guarantees not only the correctness of the simulation, but also repeatability; every time we run the simulation with the same initial seed we get exactly the same results, regardless of the number of workers or possible worker failures.

5. QUERYING TOPICAL INFLUENCERS

Finally, we propose a general search framework for topic-specific key influencers, called **SKIT**. Inspired by the popular search engine framework, SKIT allows a user to freely express his/her interests by typing a set of keywords. Then, SKIT returns an ordered list of key influencers by their influence scores that satisfy the user's intent.

SKIT flexibly allows plugging in different topical influence methods. All that it needs from the underlying influence analysis are (a) the derivation of interested topics from the query keywords, and

Table 2: Statistics of Experimental Datasets.

Dataset	# users	# dist. words	# total words	# links
Twitter	1.76 M	159 K	2363 M	183 M
Weibo	2.33 M	714 K	492 M	51 M

(b) the per-topic influence scores for every microblog user. More specifically, given a set of key words as a query q, SKIT first derives a weight $W(t, q)$ for each topic t in the set of all topics T, indicating the likelihood of topic t being represented by query q. Then, utilizing the learned per-topic influence score for each user $\text{INFL}(t, u)$, the final influence score $\text{INFL}(q, u)$ for a user u given a query q is computed as

$$\text{INFL}(q, u) = \sum_{t \in T} W(t, q) \cdot \text{INFL}(t, u). \quad (9)$$

Finally, the users are returned in decreasing order of their influence scores $\text{INFL}(q, u)$.

When our FLDA model is used as the underlying topic-specific influence analysis method, the probability distributions $\theta_{z|m}$ and $\sigma_{e|x}$ are produced as part of the results. Here, $\theta_{z|m}$ represents the probability of topic z given user m, and $\sigma_{e|x}$ is the probability of user e being followed by someone given topic x. If we treat a query q as a new user, we can use the folding-in [13] or the variational inference [25] technique on FLDA to quickly learn $\theta_{z=t|m=q}$, the probability of topic t given the query q, and use this value as $W(t, q)$ in Equation (9). On the other hand, the per-topic influence score $\text{INFL}(t, u)$ for each user can be quantified by $\sigma_{e=u|x=t}$.

Besides FLDA, our flexible SKIT search framework can also easily plug in Link-LDA, TSPR and TwitterRank. The folding-in and the variational inference techniques equally apply to Link-LDA and LDA, if LDA is used in the topic modeling preprocess for TSPR and TwitterRank, to compute $W(t, q)$. The definition of $\text{INFL}(t, u)$ for Link-LDA is the same as in FLDA. For both TSPR and TwitterRank, $\text{INFL}(t, u)$ is simply the PageRank score for user u and topic t.

6. EXPERIMENTS

In this section, we start with evaluating the effectiveness of our FLDA model on two microblog datasets, Twitter and Tencent Weibo. On the Twitter dataset, we give examples of topics and influencers found by the FLDA model, then we use the Tencent Weibo dataset to systematically compare FLDA with a number of existing approaches including TSPR, TwitterRank and Link-LDA. Finally, we demonstrate the scalability of the distributed Gibbs sampling algorithm, and show that it produces results with indistinguishable quality as the sequential algorithm.

Experiment Setup. The sequential FLDA Gibbs sampling algorithm was run on an 4-core Intel Xeon (X5672) 64-bit 3.2GHz server with 192GB RAM. For distributed FLDA Gibbs sampling, we used a cluster of 27 IBM System x iDataPlex dx340 servers. Each server consisted of two quad-core Intel Xeon (E5540) 64-bit 2.5GHz processors, 32GB RAM, and interconnected using 1GB Ethernet. We reserved one server as the Spark coordinator, and use the remaining ones for workers. Each machine was configured to run up to 8 concurrent workers. By default, we used 200 workers for distributed FLDA.

6.1 Effectiveness on Twitter Dataset

We first evaluate our FLDA model on a Twitter dataset[1], crawled between October 2009 and January 2010. The raw dataset consists of roughly half a terabyte of text and link information. The basic statistics of this dataset are given in Table 2. We used the tokenizer from the TweetNLP project [12] in order to improve the accuracy of the recognized terms in the noisy text. We tried to further reduce the inherent noise of tweets, by removing terms that appear in less than 50 tweets. We set the number of topics to 100 and run the distributed FLDA Gibbs sampling for 500 iterations. All the priors were set to 0.1 except ρ which was set to 1. These settings are fairly typical for LDA-based approaches and their tuning is beyond the scope of this paper.

Table 3 shows some of the resulting topics with their top keywords and influencers. We named these topics to simplify the presentation. Intuitively, it is clear that the influencers are very relevant to the corresponding topics. For example, one would expect O'Reily publishers, Gartner research, and popular software bloggers to be influential for an IT-related topic. Just as one would expect school age kids to be influenced by pop stars. Some of the findings are insightful. For example, Australians seem to be particularly influenced by comedians. While the first person in the list is Prime Minister of Australia (at the time of the crawl), the following three are comics.

FLDA separated the "globally" popular users from the content-specific influencers, and elected that 15% of all links were content-independent. In other words, 15% of the time, these popular users were followed regardless of what people tweet about. By comparison, the largest topic was associated with less than 2.5% of all links. The top five globally popular users detected by FLDA were: Pete Wentz (singer), Ashton Kutcher (actor), Greg Grunberg (actor, author of Yowza mobile app), Britney Spears (singer), and Ellen DeGeneres (comedian, TV host). In comparison, the top five most-followed Twitter accounts were: Barack Obama, Ashton Kutcher, Britney Spears, Ellen DeGeneres, Shaquille OŠNeal (basketball player). Although President Obama was most followed, we found his impact was most prominent in politics, which was topic-related. Similarly, the basketball star Shaquille O'Neal was mostly followed, due to his impact in basketball. FLDA can correctly identify topic-specific influence from the content-independent popularity.

There were a number of similar topics produced by both FLDA and Link-LDA. Table 4 compares the top five influencers from FLDA and Link-LDA for a few example topics. As shown from the table, FLDA produced dramatically better results than Link-LDA. For example, the "Jobs" topic produced by FLDA and Link-LDA had virtually the same top-10 keywords. The top five influencers identified by FLDA were all popular job-search websites, whereas the influencers found by Link-LDA were mostly tech bloggers. Upon inspection of their tweets it seems clear that the Link-LDA list was much less relevant to the job search topic. As another example, on the "Music" topic, FLDA successfully identified popular music media as influencers, whereas Link-LDA misidentified Club Ubuntu and a consultant as influencers in the music topic.

Naturally, such anecdotal evidence is very hard to generalize and quantify. Luckily, 2012 KDD Cup provided us with the data needed to objectively measure the quality of FLDA and other approaches, as we describe next.

6.2 Effectiveness on Tencent Weibo Dataset

In this section, we systematically evaluate the effectiveness of our FLDA model on a sample dataset from the popular Chinese microblog site – Tencent Weibo (`t.qq.com`).

This Tencent Weibo dataset is released by KDD Cup 2012[2]. The basic statistics of this dataset [3] are given in Table 2. A very nice feature of the Weibo dataset is the set of provided VIP users (also

[1]This dataset was crawled in a BSF manner with the top 1000 users in `twitterholic.com` as the seeds.

[2]`www.kddcup2012.org/c/kddcup2012-track1/data`

[3]In the Tencent Weibo dataset, for each user, the appearance of each word is

Table 3: A sample of FLDA topics and their influencers.

Topic	Top-10 keywords	Top-5 influencers
"Information Technology"	data, web, cloud, software, open, windows, microsoft, server, security, code	Tim O'Reilly, Gartner Inc., Scott Hanselman (software blogger), Jeff Atwood (software blogger, co-founder of stackoverflow.com), Elijah Manor (software blogger)
"Food and drink"	food, chocolate, coffee, eat, chicken, lunch dinner, cheese, recipe, tea	Whole Foods (organic grocery chain), Foodimentary.com (food blog), WineTwits.com (wine community), Barack Obama, L.A. Times Food
"Cycling and running"	bike, ride, race, training, running, miles, team, workout, marathon, fitness	Lance Armstrong, Levi Leipheimer, George Hincapie (all 3, US Postal pro cycling team members), Johan Bruyneel (US Postal team director), RSLT (radioshackleopardtrek.com – pro cycling team)
"Advertiser's dream"	class, sleep, hate, bed, tired, movie, homework, finally, bored, ugh	Taylor Swift, Pete Wentz, Katy Perry (all 3 singers), Perez Hilton (celeb blogger), Lady Gaga
"Ppl can't spell"	ppl, dnt, nite, tht, jus, goin, lov, wat, abt, plz	Kim Kardashian, Kourtney Kardashian, Khloe Kardashian, Tila Tequila (all 4 reality TV stars), Ciara (singer)
"Down under"	travel, Australia, latest, Sydney, Melbourne fishing, Australian, trip, hotel, island	Kevin Rudd (Australian PM in 2010), Rove McManus, Dave Hughes, Wil Anderson (all three are Aussie comedians and TV hosts), Ruby Rose (Australian model and TV presenter)

Table 4: A sample of the topics of FLDA and Link-LDA together with their influencers

Topic	Model	Top-10 keywords	Top-5 influencers
"Job"	FLDA	business, job, jobs, management, manager, sales, services, company, service, hiring	job-hunt.org, jobsguy.com, integritystaffing.com/blog (Job Search Ninja), JobConcierge.com, careerealism.com
	Link-LDA	job, jobs, manager, business, sales, management, service, company, services, hiring	Paul Terry Walhus (web host developer and blogger), Wayne Sutton (startup advisor), Adam Glickman (tech professional), bloggersblog.com (blogging news), Mary Hodder (tech blogger)
"Music"	FLDA	listening, album, rock, band, song, john, black, top, tour, artists	pitchfork.com (music website), Trent Reznor (singer), Paste Magazine (music magazine), New Musical Express, Sub Pop Records
	Link-LDA	listening, rock, album, song, band, black, john, top, beatles, bob	Club Ubuntu, Nithin Jawali (tech enthusiast), Paul Shaffer (musician) Debra Zimmer (consultant), iheartquotes.com (a collection of quotes)
"Justice"	FLDA	police, court, case, law, report, death, story, arrested, woman, state	Barack Obama, CNN Breaking News, The New York Times, CanadaCool.com, BBC Breaking News
	Link-LDA	police, ap, law, court, report, press, case, death, reuters, state	Health Brand, 2humor.com (funny stuff), healthsmartme.tumblr.com Daniel Vega (lawyer), Multiplaza Shopping (shopping deals)
"Weather"	FLDA	snow, weather, rain, winter, high, nc, denver, storm, wind, county	CNN Breaking News, The Denver Post, NPR News, The Weather Channel, CBS Denver
	Link-LDA	snow, weather, atlanta, rain, high, nc, county, south, fire, north	DiningPerks.com, Georgia Aquarium, Atlanta Journal-Constitution (Atlanta newspaper), HelloNorthGeorgia.com, Q100 Atlanta (radio)

called items in Weibo), which enables us to systematically evaluate the precision of various key influencer methods. These VIP users are manually labeled by Weibo administrators, and organized in hierarchical categories. An example hierarchical category is *science_and_technology.internet.mobile*, where categories in different levels are separated by a dot ".". In this dataset, categories are anonymized as integers, such as 1.4.2.3. There are 377 categories and on average each category contains 16.2 VIPs. According to Weibo, the VIP users are typically famous people and organizations. In other words, they are "key influencers" in their corresponding categories. As a result, the VIP users can be used as the "ground truth" for our empirical evaluation. While we don't expect VIP categories to have 100% precision or recall, they give us enough information to facilitate relative comparisons across different schemes.

Based on this information, we have set up the following experiment. For a given category, we use one VIP (i.e. all the words of this user) as the query, and observe how many of the fellow VIP users in the same category are identified as top influencers by the different schemes. In the following, we compare our FLDA model with TSPR, TwitterRank and Link-LDA. We maintain the number of topics at 100 for all the methods and run 500 iterations for LDA (used in TSPR and TwitterRank), Link-LDA and FLDA. The priors used are 1.0 for α, 0.01 for β, γ and ϵ, and 0.1 for ρ.

Figure 3a and Figure 3b compare our FLDA model with TSPR, TwitterRank and Link-LDA on two of the largest categories in the Weibo dataset. For each category, we use every VIP user as a key influencer query and check how many of the top K returned users are the fellow VIP users. We report the average number of VIPs among the top K returned results across all the queries. As shown in both figures, our FLDA model consistently produces better precision results than the others by a significant margin. TSPR is usually the worst among all methods, followed by TwitterRank. Link-LDA performs slightly better than the two PageRank-based approaches.

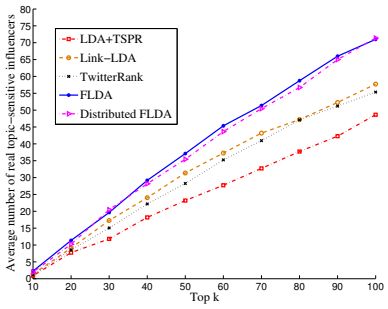
To analyze the results across all categories we employ a standard Mean Average Precision (MAP) [21] metric. MAP for a set of queries is defined as the mean of the average precision scores ($AvgP$) for each query. $AvgP$ of a list of top-k query results is defined as the average of precision values for all k prefixes.

Figure 4 shows MAP of all the queries across all categories in the Weibo dataset. Again, FLDA produces significantly better results than the competing methods, more than 2 times better than TSPR and TwitterRank, and around 1.6 times better than Link-LDA. Interestingly, FLDA elected only 50% of Weibo links to be content-related. This explains the significant advantage of FLDA over Link-LDA, which assumes that all links are topic-specific.

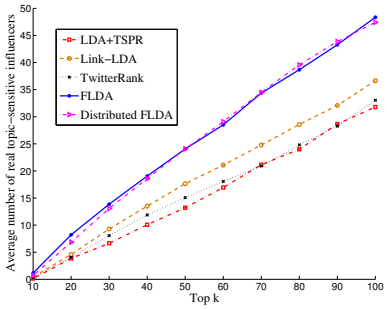
As shown in Figure 3a, 3b and Figure 4, the distributed FLDA consistently produces result with quality almost identical to that of the sequential FLDA. This confirms the relaxed dependency assumption on which our distributed FLDA Gibbs sampling is based.

Throughout the experiments we measured the time taken by the on-line component of our SKIT search framework. On average, each query takes 1.7 sec to get the results, and this time does not depend on the off-line modeling scheme we use.

associated with a weight (usually ≤ 1.0). We multiply this weight by 100 to approximate the underlying word frequency.



(a) category 1.6.2.1 (#VIPs in this category: 277)



(b) category 1.6.2.2 (#VIPs in this category: 123)

Figure 3: Average number of returned VIPs

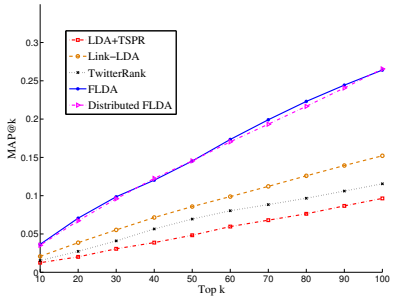


Figure 4: Mean Average Precision

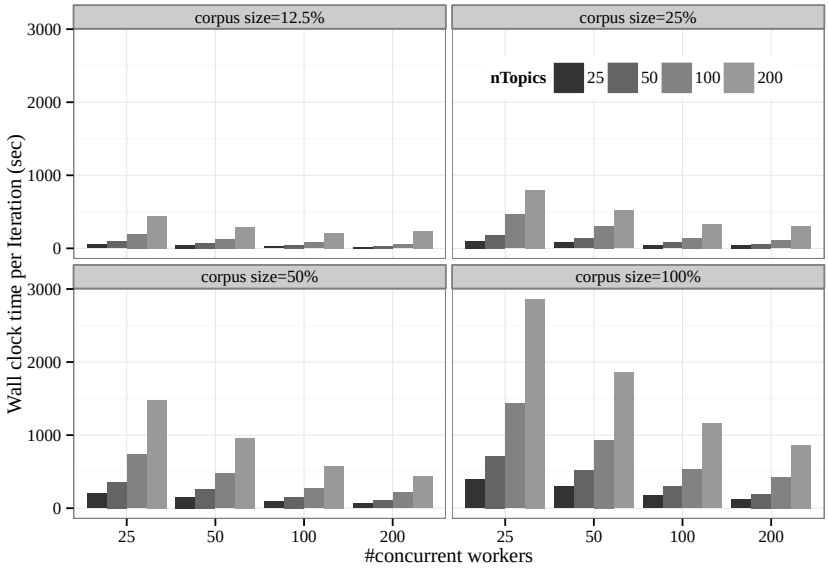


Figure 5: Speed-Up of Distributed FLDA on Twitter dataset.

6.3 Scalability

Before evaluating the scalability of our distributed FLDA Gibbs sampling algorithm, we first report the execution times for the sequential algorithm. For the KDD Weibo dataset, the sequential Gibbs sampling on a high-end server (192GB RAM, 3.2GHz processor) takes around 13 minutes per iteration, and for the Twitter dataset, it takes more than one hour per iteration. On Twitter dataset FLDA runs longer because there are many more words and links to sample. Running sequential Gibbs sampling for 500 iterations takes around 4.6 days for the KDD Weibo dataset, and would take 21 days for the Twitter dataset! This clearly motivates the need for a scalable solution.

Our distributed algorithm completes 500 iterations on Twitter data in about 36 hours, using 200 workers on 27 machines. Overall, the distributed FLDA in this instance is about 14 times faster than a sequential implementation running on single, large-memory server.

We tested the scalability of the distributed algorithm along three dimensions: data size, number of topics, and the number of concurrent workers. To obtain the scaled down dataset we performed uniform random sampling of users, for example to generate a 4 times smaller dataset we use a sampling rate of 25%. The results are summarized in Figure 5 where the scaled-down dataset is denoted as *corpus size* and is measured by the sampling rate used. We explore a wide range of sizes (from 12.5% all the way up to 100%), number of topics (from 25 to 200) and number of workers (from 25 to 200). The figure shows that the distributed FLDA scales well along all dimensions, given the limitations of our cluster. Our (older) CPU's were significantly oversubscribed with 8 workers per node, which was the case with 200 workers.

7. CONCLUSION

This paper addresses the problem of identifying topic-specific key influencers in microblog networks. To model the per-topic influence of each user, we introduce a novel Bernoulli-Multinomial mixture model called FLDA. FLDA incorporates the content of tweets and the network structure of microblogs into one unified model. Different from the previous work, such as Link-LDA, our FLDA model is specifically designed for microblogs in that it captures the fact that in reality a user sometimes follows another due to content-independent reasons. Moreover, in order to apply FLDA to a web-scale microblog network, we design a distributed Gibbs sampling algorithm for FLDA on the Spark distributed computing framework. Finally, the FLDA model is incorporated in a proposed general search framework for topic-specific key influencers, which provides a keyword search interface for users to freely query key influencers in different topic combinations.

Through experiments on two real-world microblog datasets, we demonstrate that FLDA significantly outperforms the state-of-the-

art methods in terms of precision. Furthermore, the distributed Gibbs sampling algorithm for FLDA provides excellent speed-up to hundreds of workers.

8. REFERENCES

[1] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. J. Smola. Scalable inference in latent variable models. In *WSDM'12*, pages 123–132, 2012.

[2] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, pages 81–90, 2012.

[3] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/pacts: a programming model and execution framework for web-scale analytical processing. In *SoCC*, 2010.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[5] V. R. Borkar, M. J. Carey, R. Grover, N. Onose, and R. Vernica. Hyracks: A flexible and extensible foundation for data-intensive computing. In *ICDE*, 2011.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW'98*, pages 107–117, 1998.

[7] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst. Haloop: efficient iterative data processing on large clusters. *PVLDB*, 2010.

[8] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM COMPUTING SURVEYS*, 38(1):2, 2006.

[9] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD '09*, pages 199–208, 2009.

[10] E. Erosheva, S. Fienberg, and J. Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, 101:5220–5227, 2004.

[11] S. Ghosh, N. Sharma, F. Benevenuto, N. Ganguly, and K. Gummadi. Cognos: crowdsourcing search for topic experts in microblogs. In *SIGIR '12*, pages 575–590, 2012.

[12] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *ACL*, pages 42–47, 2011.

[13] M. Girolami and A. Kabán. On an equivalence between plsi and lda. In *SIGIR '03*, pages 433–434, 2003.

[14] H. Haramoto, M. Matsumoto, T. Nishimura, F. Panneton, and P. L'Ecuyer. Efficient Jump Ahead for 2-Linear Random Number Generators. *INFORMS Journal on Computing*, 20(3):385–390, 2008.

[15] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW '02*, pages 517–526, 2002.

[16] A. Java, P. Kolari, T. Finin, and T. Oates. Modeling the spread of influence on the blogosphere. In *WWW 2006 Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, 2006.

[17] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03*, pages 137–146, 2003.

[18] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD '07*, pages 420–429, 2007.

[19] L. Liu, J. Tang, J. Han, and S. Yang. Learning influence from heterogeneous social networks. *Data Mining and Knowledge Discovery*, 25:511–544, 2012.

[20] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD*, 2010.

[21] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[22] R. Nallapati and W. W. Cohen. Link-plsa-lda: A new unsupervised model for topics and influence of blogs. In *Proceedings of the Second International Conference on Weblogs and Social Media*, 2008.

[23] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed algorithms for topic models. *J. Mach. Learn. Res.*, 10:1801–1828, Dec. 2009.

[24] A. Pal and S. Counts. Identifying topical authorities in microblogs. In *WSDM '11*, pages 45–54, 2011.

[25] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *PVLDB*, 3(1-2):703–710, Sept. 2010.

[26] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD '09*, pages 807–816, 2009.

[27] Twitter.com. Twitter turns six, 2012.

[28] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM '10*, pages 261–270, 2010.

[29] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In *NSDI'12*, 2012.

APPENDIX

The Gibbs sampling equation for latent variable z can be derived in a similar way to Link-LDA. We omit its derivation due to space limitation. Let us derive the posterior probability of latent variables x and y:

$$
\begin{aligned}
&p(x_{m,l}, y_{m,l} | y_{-(m,l)}, x_{-(m,l)}, w, z, e, \alpha, \beta, \gamma, \varepsilon, \rho) \\
&\propto p(x, y, w, z, e | \alpha, \beta, \gamma, \varepsilon, \rho) \\
&= \int\int\int\int\int p(x, y, w, z, e, \theta, \varphi, \sigma, \pi, \mu | \alpha, \beta, \gamma, \varepsilon, \rho) d\theta d\varphi d\sigma d\pi d\mu \\
&= \int\int\int\int\int p(x|\theta) p(y|\mu) p(w|z, \varphi) p(z|\theta) p(e|x, y, \sigma, \pi) \\
&\quad \times p(\theta|\alpha) p(\varphi|\beta) p(\sigma|\gamma) p(\pi|\varepsilon) p(\mu|\rho) d\theta d\varphi d\sigma d\pi d\mu \qquad (10)
\end{aligned}
$$

As $p(e|x, y, \sigma, \pi) = p(e|x, \sigma)^y p(e|\pi)^{1-y}$, we get,

$$
\begin{aligned}
&= \int p(\theta|\alpha) p(z|\theta) p(x|\theta) d\theta \int p(\sigma|\gamma) p(e|x, \sigma)^y d\sigma \\
&\quad \times \int p(\varphi|\beta) p(w|z, \varphi) d\varphi \int p(\pi|\varepsilon) p(e|\pi)^{1-y} d\pi \int p(\mu|\rho) p(y|\mu) d\mu \qquad (11)
\end{aligned}
$$

Let us derive the first two integrals in Equation (11).

$$
\begin{aligned}
&\int p(\theta|\alpha) p(z|\theta) p(x|\theta) d\theta \int p(\sigma|\gamma) p(e|x, \sigma)^y d\sigma \\
&= \int \prod_{j=1}^{M} p(\theta_j|\alpha) \prod_{j=1}^{M} \prod_{u=1}^{N_j} p(z_{j,u}|\theta_j) \prod_{j=1}^{M} \prod_{v=1}^{L_j} p(x_{j,v}|\theta_j) d\theta \\
&\quad \times \int \prod_{k=1}^{K} p(\sigma_k|\gamma) \prod_{j=1}^{M} \prod_{v=1}^{L_j} p(e_{j,v}|\sigma_{x_{j,v}})^{y_{j,v}} d\sigma \qquad (12)
\end{aligned}
$$

Expand each probability formula based on its density,

$$
\begin{aligned}
&= \int \prod_{j=1}^{M} \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{j,k}^{\alpha_k - 1} \prod_{j=1}^{M} \prod_{u=1}^{N_j} \theta_{z_{j,u}} \prod_{j=1}^{M} \prod_{v=1}^{L_j} \theta_{x_{j,v}} d\theta \\
&\quad \times \prod_{k=1}^{K} \int \frac{\Gamma(\sum_{i=1}^{M} \gamma_i)}{\prod_{i=1}^{M} \Gamma(\gamma_i)} \prod_{i=1}^{M} \sigma_{k,i}^{\gamma_i - 1} \prod_{j=1}^{M} \prod_{v=1}^{L_j} \sigma_{x_{j,v}, e_{j,v}}^{y_{j,v}} d\sigma_k \qquad (13)
\end{aligned}
$$

Replace the innermost products over words in a document N_m by exponentiating to the sum of the counts, and do the same replacement for the products over users,

$$
\begin{aligned}
&= \prod_{j=1}^{M} \int \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{j,k}^{\alpha_k - 1} \prod_{k=1}^{K} \theta_{j,k}^{c_{k,j,*}} \prod_{k=1}^{K} \theta_{j,k}^{d_{k,j,*,*}} d\theta_j \\
&\quad \times \prod_{k=1}^{K} \int \frac{\Gamma(\sum_{i=1}^{M} \gamma_i)}{\prod_{i=1}^{M} \Gamma(\gamma_i)} \prod_{i=1}^{M} \sigma_{k,i}^{\gamma_i - 1} \prod_{i=1}^{M} \sigma_{k,i}^{d_{k,*,i,1}} d\sigma_k \\
&= \prod_{j=1}^{M} \int \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{j,k}^{\alpha_k + c_{k,j,*} + d_{k,j,*,*} - 1} d\theta_j \\
&\quad \times \prod_{k=1}^{K} \int \frac{\Gamma(\sum_{i=1}^{M} \gamma_i)}{\prod_{i=1}^{M} \Gamma(\gamma_i)} \prod_{i=1}^{M} \sigma_{k,i}^{\gamma_i + d_{k,*,i,1} - 1} d\sigma_k \qquad (14)
\end{aligned}
$$

Multiply each term by a constant equal to one (consisting of two inverse fractions), and distribute the integral over the original constant Γ-function fraction for the priors,

$$
\begin{aligned}
&= \prod_{j=1}^{M} \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + c_{k,j,*} + d_{k,j,*,*})}{\Gamma(\sum_{k=1}^{K} \alpha_k + c_{k,j,*} + d_{k,j,*,*})} \\
&\times \int \frac{\Gamma(\sum_{k=1}^{K} \alpha_k + c_{k,j,*} + d_{k,j,*,*})}{\prod_{k=1}^{K} \Gamma(\alpha_k + c_{k,j,*} + d_{k,j,*,*})} \prod_{k=1}^{K} \theta_{j,k}^{\alpha_k + c_{k,j,*} + d_{k,j,*,*} - 1} d\theta_j \\
&\times \prod_{k=1}^{K} \frac{\Gamma(\sum_{i=1}^{M} \gamma_i)}{\prod_{i=1}^{M} \Gamma(\gamma_i)} \frac{\prod_{i=1}^{M} \Gamma(\gamma_i + d_{k,*,i,1})}{\Gamma(\sum_{i=1}^{M} \gamma_i + d_{k,*,i,1})} \\
&\times \int \frac{\Gamma(\sum_{i=1}^{M} \gamma_i + d_{k,*,i,1})}{\prod_{i=1}^{M} \Gamma(\gamma_i + d_{k,*,i,1})} \prod_{i=1}^{M} \sigma_{k,i}^{\gamma_i + d_{k,*,i,1} - 1} d\sigma_k
\end{aligned} \tag{15}
$$

Note that both integrals are over the entire support of Dirichlet densities, so they both evaluate to 1, and hence drop out of the products,

$$
\begin{aligned}
&= \prod_{j=1}^{M} \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + c_{k,j,*} + d_{k,j,*,*})}{\Gamma(\sum_{k=1}^{K} \alpha_k + c_{k,j,*} + d_{k,j,*,*})} \\
&\times \prod_{k=1}^{K} \frac{\Gamma(\sum_{i=1}^{M} \gamma_i)}{\prod_{i=1}^{M} \Gamma(\gamma_i)} \frac{\prod_{i=1}^{M} \Gamma(\gamma_i + d_{k,*,i,1})}{\Gamma(\sum_{i=1}^{M} \gamma_i + d_{k,*,i,1})}
\end{aligned} \tag{16}
$$

Eliminate constant terms that do not depend on the position (m, l),

$$
\propto \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + c_{k,m,*} + d_{k,m,*,*})}{\Gamma(\sum_{k=1}^{K} \alpha_k + c_{k,m,*} + d_{k,m,*,*})} \times \prod_{k=1}^{K} \frac{\Gamma(\gamma_{e_{m,l}} + d_{k,*,e_{m,l},1})}{\Gamma(\sum_{i=1}^{M} \gamma_i + d_{k,*,i,1})} \tag{17}
$$

Define $c^{-(m,l)}$ the same way as c, only without the counts for position (m, l). Then, for counts that do not include position (m, l), $c^{-(m,l)} = c$. For ones that do include (m, l), $c^{-(m,l)} = c + 1$. $d^{-(m,l)}$ is defined in the same way. Also, using the fact that $\Gamma(x + 1) = x \times \Gamma(x)$, expand out the incremented terms depending on (m, l),

$$
\begin{aligned}
&= \frac{\prod_{k \neq x_{m,l}} \Gamma(\alpha_k + c_{k,m,*} + d_{k,m,*,*}^{-(m,l)})}{\Gamma(1 + \sum_{k=1}^{K} \alpha_k + c_{k,m,*} + d_{k,m,*,*}^{-(m,l)})} \\
&\times \Gamma(\alpha_{x_{m,l}} + c_{x_{m,l},m,*} + d_{x_{m,l},m,*,*}^{-(m,l)}) \\
&\times (\alpha_{x_{m,l}} + c_{x_{m,l},m,*} + d_{x_{m,l},m,*,*}^{-(m,l)}) \\
&\times \prod_{k \neq x_{m,l}} \frac{\Gamma(\gamma_{e_{m,l}} + d_{k,*,e_{m,l},1}^{-(m,l)})}{\Gamma(\sum_{i=1}^{M} \gamma_i + d_{k,*,i,1})} \times \frac{\Gamma(\gamma_{e_{m,l}} + d_{x_{m,l},*,e_{m,l},1}^{-(m,l)})}{\Gamma(\sum_{i=1}^{M} \gamma_i + d_{x_{m,l},*,i,1}^{-(m,l)})} \\
&\times \frac{\gamma_{e_{m,l}} + d_{x_{m,l},*,e_{m,l},1}^{-(m,l)}}{\sum_{i=1}^{M} (\gamma_i + d_{x_{m,l},*,i,1}^{-(m,l)})}
\end{aligned} \tag{18}
$$

Refold the residual Γ-function terms back into their general products,

$$
\begin{aligned}
&= \frac{\prod_{k=1}^{K} \Gamma(\alpha_k + c_{k,m,*} + d_{k,m,*,*}^{-(m,l)})}{\Gamma(1 + \sum_{k=1}^{K} \alpha_k + c_{k,m,*} + d_{k,m,*,*}^{-(m,l)})} \\
&\times (\alpha_{x_{m,l}} + c_{x_{m,l},a,*} + d_{x_{m,l},m,*,*}^{-(m,l)}) \\
&\times \prod_{k=1}^{K} \frac{\Gamma(\gamma_{e_{m,l}} + d_{k,*,e_{m,l},1}^{-(m,l)})}{\Gamma(\sum_{i=1}^{M} \gamma_i + d_{k,*,i,1})} \times \frac{\gamma_{e_{m,l}} + d_{x_{m,l},*,e_{m,l},1}^{-(m,l)}}{\sum_{i=1}^{M} (\gamma_i + d_{x_{m,l},*,i,1}^{-(m,l)})}
\end{aligned} \tag{19}
$$

Remove all the terms that do not depend on $x_{m,l}$ or $y_{m,l}$.

If $y_{m,l} = 0$, the last term $\frac{\gamma_{e_{m,l}} + d_{x_{m,l},*,e_{m,l},1}^{-(m,l)}}{\sum_{i=1}^{M} (\gamma_i + d_{x_{m,l},*,i,1}^{-(m,l)})}$ in Equation (19) does not exist,

$$
\propto \alpha_{x_{m,l}} + c_{x_{m,l},m,*} + d_{x_{m,l},m,*,*}^{-(m,l)} \tag{20}
$$

If $y_{m,l} = 1$,

$$
\propto \frac{(\alpha_{x_{m,l}} + c_{x_{m,l},m,*} + d_{x_{m,l},m,*,*}^{-(m,l)})(\gamma_{e_{m,l}} + d_{x_{m,l},*,e_{m,l},1}^{-(m,l)})}{\sum_{i=1}^{M} (\gamma_i + d_{x_{m,l},*,i,1}^{-(m,l)})} \tag{21}
$$

The third integral $\int p(\varphi|\beta) p(w|z, \varphi) d\varphi$ in Equation (11) is independent of both x and y, so it can be safely canceled out. Let us turn to the forth integral in Equation (11).

$$
\begin{aligned}
&\int p(\pi|\varepsilon) p(e|\pi)^{1-y} d\pi \\
&= \int \frac{\Gamma(\sum_{i=1}^{M} \varepsilon_i)}{\prod_{i=1}^{M} \Gamma(\varepsilon_i)} \prod_{i=1}^{M} \pi_i^{\varepsilon_i - 1} \prod_{j=1}^{M} \prod_{v=1}^{L_j} \pi_{e_{j,v}}^{1 - y_{j,v}} d\pi \\
&= \int \frac{\Gamma(\sum_{i=1}^{M} \varepsilon_i)}{\prod_{i=1}^{M} \Gamma(\varepsilon_i)} \prod_{i=1}^{M} \pi_i^{d_{*,*,i,0} + \varepsilon_i - 1} d\pi \\
&\propto \frac{\Gamma(\sum_{i=1}^{M} \varepsilon_i)}{\prod_{i=1}^{M} \Gamma(\varepsilon_i)} \times \frac{\prod_{i=1}^{M} \Gamma(d_{*,*,i,0} + \varepsilon_i)}{\Gamma(\sum_{i=1}^{M} d_{*,*,i,0} + \varepsilon_i)} \\
&\propto \frac{\prod_{i \neq e_{m,l}} \Gamma(d_{*,*,i,0} + \varepsilon_i) \times \Gamma(d_{*,*,e_{m,l},0} + \varepsilon_{e_{m,l}})}{\Gamma(\sum_{i=1}^{M} d_{*,*,i,0} + \varepsilon_i)}
\end{aligned} \tag{22}
$$

If $y_{m,l} = 0$, Equation (22) can be written as:

$$
\begin{aligned}
&= \frac{\prod_{i \neq e_{m,l}} \Gamma(d_{*,*,i,0}^{-(m,l)} + \varepsilon_i) \times \Gamma(d_{*,*,e_{m,l},0}^{-(m,l)} + \varepsilon_{e_{m,l}} + 1)}{\Gamma(1 + \sum_{i=1}^{M} d_{*,*,i,0}^{-(m,l)} + \varepsilon_i)} \\
&= \frac{\prod_i \Gamma(d_{*,*,i,0}^{-(m,l)} + \varepsilon_i)}{\Gamma(\sum_{i=1}^{M} d_{*,*,i,0}^{-(m,l)} + \varepsilon_i)} \times \frac{d_{*,*,e_{m,l},0}^{-(m,l)} + \varepsilon_{e_{m,l}}}{\sum_{i=1}^{M} d_{*,*,i,0}^{-(m,l)} + \varepsilon_i} \\
&\propto \frac{d_{*,*,e_{m,l},0}^{-(m,l)} + \varepsilon_{e_{m,l}}}{\sum_{i=1}^{M} d_{*,*,i,0}^{-(m,l)} + \varepsilon_i}
\end{aligned} \tag{23}
$$

If $y_{m,l} = 1$, Equation (22) can be written as:

$$
= \frac{\prod_i \Gamma(d_{*,*,i,0}^{-(m,l)} + \varepsilon_i)}{\Gamma(\sum_{i=1}^{M} d_{*,*,i,0}^{-(m,l)} + \varepsilon_i)} \propto 1 \tag{24}
$$

Finally, we derive the last integral in Equation (11).

$$
\begin{aligned}
&\int p(\mu|\rho) p(y|\mu) d\mu \\
&= \prod_{j=1}^{M} \int \frac{\Gamma(\sum_s \rho_s)}{\prod_s \Gamma(\rho_s)} \prod_s \mu_{j,s}^{\rho_s - 1} \prod_{j=1}^{M} \prod_{v=1}^{L_j} \mu_{j,y_{j,v}} d\mu_j \\
&= \prod_{j=1}^{M} \int \frac{\Gamma(\sum_s \rho_s)}{\prod_s \Gamma(\rho_s)} \prod_s \mu_{j,s}^{d_{*,j,*,s} + \rho_s - 1} d\mu_j \\
&\propto \prod_{j=1}^{M} \frac{\prod_s \Gamma(d_{*,j,*,s} + \rho_s)}{\Gamma(\sum_s d_{*,j,*,s} + \rho_s)} \\
&= \prod_{j \neq m} \frac{\prod_s \Gamma(d_{*,j,*,s} + \rho_s)}{\Gamma(\sum_s d_{*,j,*,s} + \rho_s)} \times \frac{\prod_s \Gamma(d_{*,m,*,s} + \rho_s)}{\Gamma(\sum_s d_{*,m,*,s} + \rho_s)} \\
&\propto \frac{\prod_{s \neq y_{m,l}} \Gamma(d_{*,m,*,s}^{-(m,l)} + \rho_s) \times \Gamma(d_{*,m,*,y_{m,l}}^{-(m,l)} + \rho_{y_{m,l}} + 1)}{\Gamma(1 + \sum_s d_{*,m,*,s}^{-(m,l)} + \rho_s)} \\
&= \frac{\prod_s \Gamma(d_{*,m,*,s}^{-(m,l)} + \rho_s)}{\Gamma(\sum_s d_{*,m,*,s}^{-(m,l)} + \rho_s)} \times \frac{d_{*,m,*,y_{m,l}}^{-(m,l)} + \rho_{y_{m,l}}}{d_{*,m,*,*}^{-(m,l)} + \sum_s \rho_s} \\
&\propto d_{*,m,*,y_{m,l}}^{-(m,l)} + \rho_{y_{m,l}}
\end{aligned} \tag{25}
$$

Finally, substituting Equations (20), (23) and (25) into Equation (11) gives Equation (2). Similary, substituting Equations (21), (24) and (25) into Equation (11) gives Equation (3).