# SocialLDA:Scalable Topic Modeling in Social Networks

Ashish Bindra

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

University of Washington

2012

Committee:

Ankur Teredesai

Martine De Cock

Program Authorized to Offer Degree:
Computing & Software Systems

University of Washington

## Abstract

SocialLDA:Scalable Topic Modeling in Social Networks

Ashish Bindra

Chair of the Supervisory Committee:
Professor Dr. Ankur Teredesai
Institute of Technology, University of Washington Tacoma

Topical categorization of blogs, documents or other objects that can be tagged with text, improves the experience for end users. Latent Dirichlet allocation (LDA) is a well studied algorithm that discovers latent topics from a corpus of documents so that the documents can then be assigned automatically into appropriate topics. New documents can also be classified into topics based on these latent topics. However, when the set of documents is very large and varies significantly from user to user, the task of calculating a single global LDA topic model, or an individual topic model for each and every user can become very expensive in large scale internet settings. The problem is further compounded by the need to periodically update this model to keep up with the relatively dynamic nature of data in online social networks such as Facebook, Twitter, and FriendFeed. In this work we show that the computation cost of using LDA for a large number of users connected via a social network can be reduced without compromising the quality of the LDA model by taking into account the social connections among the users in the network. Instead of a single global model based on every document in the network we propose to use a model created from messages that are authored by and received by a fixed number of most influential users. We use PageRank as the influence measure and show that this Social LDA model provides an effective model to use as it reduces the number of documents

to process thereby reducing the cost of computing the LDA. Such a model can be used both for categorizing a users incoming document stream as well as finding user interest based on the user's authored documents. Further this also helps in the cold start problem where a model based on a users own messages is insufficient to create a good LDA model.

# TABLE OF CONTENTS

# LIST OF FIGURES

Figure Number Page

# LIST OF TABLES

Table Number Page

# ACKNOWLEDGMENTS

I am indebted to many of the faculty, staff and students at the Institute of Technology who have contributed immensely to my learning during these past few year culminating in this thesis. First and foremost, I would like to acknowledge the support, guidance and help I have received from Professor Ankur Teredesai. He has always encouraged me to expand my horizons and gone out of his way to make resources available to me. This thesis would not have been possible without his guidance and it has been an honor working with him. I would also like to thank Professor Martine De Cock for her very thorough reviews, insightful comments and patience. I am also grateful to Stephen Rondeau for his prompt help in making available computing resources.

Last but definitely not the least, I am deeply thankful to my wife Ritu. Graduate school would have remained a dream without her unconditional love, unwavering support and exceptional culinary skills.

# DEDICATION

To *Ritu*.

# Chapter 1

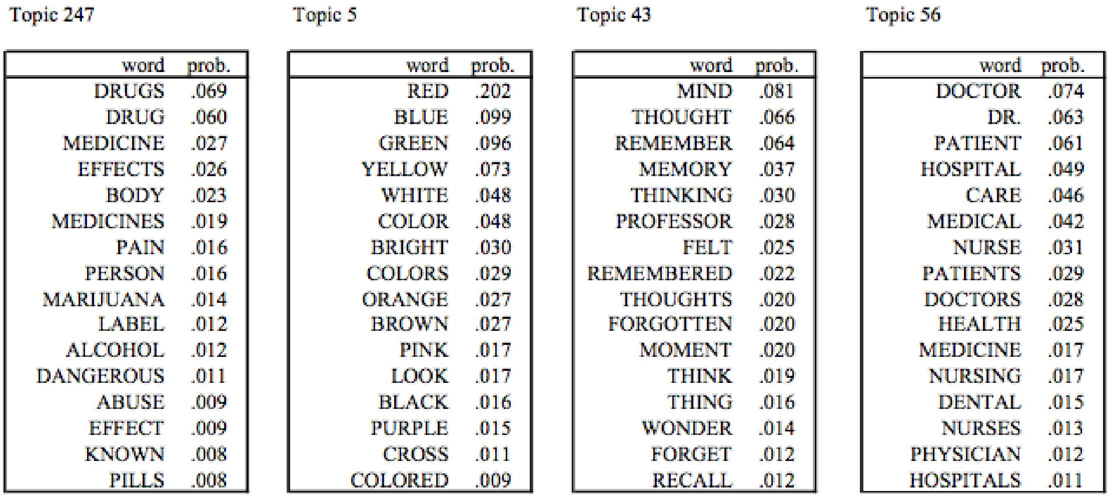
# INTRODUCTION

## *1.1 Motivation*

Real time social web [1] outputs a tremendous amount of information and this output is increasing everyday. The challenge in mining and managing this data is not only due to the large volume but also due to its temporal nature, which requires that data be processed with minimum delay. The problem of good, scalable categorization, filtering techniques that work on these large, real time social text streams is still considered unsolved.

Central to filtering out noisy data from social text streams is the notion of topic modeling, which automatically groups documents based on semantic similarity. These groups or topics can then be used to either improve tasks like search results or as a means to improve the user experience in exploring the underlying document dataset. Automatic categorization helps capture areas of interest for a user making it easier to filter out documents that are not relevant to the user's topic of interest.

The literature in Topic Modeling is extensive[2, 3, 4, 5, 6, 7, 8]. One of the earliest well known topic model is Latent Semantic Indexing (LSI), which is also known as Latent Semantic Analysis[9, 10]. This work originated as a model of human cognition developed to explain how very young children acquire new vocabulary at such a high rate without any external explicit direction. This model was adopted by and used extensively by the Information Retrieval research community as a model to improve search results with considerable success. However it not very widely used in the industry because it is unable to scale up to the extremely large internet scale data sets.



Topic 247

| word | prob. |
|---|---|
| DRUGS | .069 |
| DRUG | .060 |
| MEDICINE | .027 |
| EFFECTS | .026 |
| BODY | .023 |
| MEDICINES | .019 |
| PAIN | .016 |
| PERSON | .016 |
| MARIJUANA | .014 |
| LABEL | .012 |
| ALCOHOL | .012 |
| DANGEROUS | .011 |
| ABUSE | .009 |
| EFFECT | .009 |
| KNOWN | .008 |
| PILLS | .008 |

Topic 5

| word | prob. |
|---|---|
| RED | .202 |
| BLUE | .099 |
| GREEN | .096 |
| YELLOW | .073 |
| WHITE | .048 |
| COLOR | .048 |
| BRIGHT | .030 |
| COLORS | .029 |
| ORANGE | .027 |
| BROWN | .027 |
| PINK | .017 |
| LOOK | .017 |
| BLACK | .016 |
| PURPLE | .015 |
| CROSS | .011 |
| COLORED | .009 |

Topic 43

| word | prob. |
|---|---|
| MIND | .081 |
| THOUGHT | .066 |
| REMEMBER | .064 |
| MEMORY | .037 |
| THINKING | .030 |
| PROFESSOR | .028 |
| FELT | .025 |
| REMEMBERED | .022 |
| THOUGHTS | .020 |
| FORGOTTEN | .020 |
| MOMENT | .020 |
| THINK | .019 |
| THING | .016 |
| WONDER | .014 |
| FORGET | .012 |
| RECALL | .012 |

Topic 56

| word | prob. |
|---|---|
| DOCTOR | .074 |
| DR. | .063 |
| PATIENT | .061 |
| HOSPITAL | .049 |
| CARE | .046 |
| MEDICAL | .042 |
| NURSE | .031 |
| PATIENTS | .029 |
| DOCTORS | .028 |
| HEALTH | .025 |
| MEDICINE | .017 |
| NURSING | .017 |
| DENTAL | .015 |
| NURSES | .013 |
| PHYSICIAN | .012 |
| HOSPITALS | .011 |

Figure 1.1: An illustration of four (out of 300) LDA topics extracted from TASA corpus[4]

Another drawback of LSI is that it assigns each document to a single category, which is considered incorrect since a single document could be relevant to multiple topics. This led to the development of different Probabilistic Topic Models (PTM)[11, 4] that do soft clustering. In other words, they assign documents to multiple topics with a probability metric. One of the most well studied and well regarded PTM is the Latent Dirichlet Allocation [12, 4] or LDA model. We use LDA as our topic modeling technique because it is well studied and widely accepted in the field. We elaborate more about topic models in Chapter 2.3.

We would like to point out that in this work we are concerned mostly with unsupervised Topic Modeling, i.e. where there is no training data to learn from and instead we learn directly from the data set. LDA is originally an unsupervised Topic Modeling technique but there have been work on creating LDA like models using labelled data[13].

The problem with topic models like LDA is that they are computationally expensive. Calculating topic models like LDA or other PTM takes a lot of time on large corpuses. This problem is further compounded when the corpus itself is dynamic growing at a high rate, which is the case today with most data sources like the web taken as whole, or web sites like Twitter, Facebook, FriendFeed, etc. In these environments the topic models need to be trained frequently.

In a large social networking website like Tumblr, Friendfeed, Twitter there is an underlying social network. Each users sends and receives documents (blogs, posts, reviews, tweets, etc.). A user might receive documents from a large number of other users and wants to rank or prioritize or filter the incoming messages. LDA type topic models can be used in this scenario. Also, a topic model based on the authored documents can help shed light on the interests and expertise of that user, which can be used to recommend users, documents or target ads.

In both these there is the choice of either creating a single global topic model used across the entire network or creating a topic model for every user. A single global topic model might not perform well on every user in the network but individual topic models are more expensive and for a large number of users there are not enough documents to build good topic models. Further due to the dynamic nature the topic model(s) will need to be retrained periodically. In very large networks this is clearly computationally expensive and costly.

## *1.2 Problem*

In this study we investigate how to reduce the cost of calculating LDA in social networks like Twitter, Tumblr, Friendfeed that generate a lot of text documents shared across the network. This problem is motivated by work done at **davai.com**, a social media marketing startup that leverages social media. One of the problems was the high cost of calculating LDA topic models, which were then to be used in a proprietary algorithm to help improve user experience managing the large streams of documents that a user receives.

We are interested in two different things, namely

1. Calculating topic models on the the documents received by an author to help them filter and navigate through the information.

2. Calculating topic models on the documents authored by each author, which help discover their interests leading to better recommendations.

## *1.3 Contribution*

We show that the computation cost of of using LDA for a large number of users connected via a social network can be significantly reduced by taking into account the social connections among the users in the network. Instead of a single global model based on every document in the network we propose to use a model created from messages that are authored by and received by a fixed number of most influential users. We use PageRank as the influence measure and show that this Social LDA model provides an effective model to use as it reduces the number of documents to process thereby reducing the cost of computing the LDA. Such a model can be used both for categorizing a users incoming document stream as well as finding user interest based on the users authored documents. Further this also helps in the cold start problem where a model based on a users own messages is insufficient to create a good LDA model.

# Chapter 2
# BACKGROUND

## *2.1 Probabilistic Clustering*

Unsupervised topic modeling takes a set of documents and generates semantic clusters without needing labelled data. Clustering algorithms like K-means [14] create hard clusters, i.e. they partition data points into mutually exclusive clusters. On the other hand soft clustering maps each data point to each cluster over a probability distribution. This mitigates some of the shortcomings of hard clustering especially with overfitting.

If we assume that each cluster in the data is from a distinct distribution then the data points within a cluster will be described by the distribution for the cluster and the entire data set would be described by what is called a mixture model. When the number of clusters are finite then it is a finite mixture model[15]. So $k$-finite mixture is a set of $k$ probability distributions, representing $k$ clusters. The distributions in the mixture are present as per some probability distribution themselves so some may dominate or they could be uniformly distributed. For example figure 2.1 shows a mixture of two Gaussians.

Consider the case where we have the data (instances, data points), the parameters for each cluster (e.g. mean $\mu$ and standard deviation $\sigma$ for normally distributed clusters) as well as information as to which cluster each data point belongs to. In such a case we have complete information. Now, consider being given the data, the fact that there are $k$ clusters and their distribution family but not the parameters. In this case one has to infer the parameters for each of the $k$ clusters.

In topic modeling each topic is from a separate distribution, which combine to form the total data. This is essentially an example of a problem that statistical learning aims to solve.

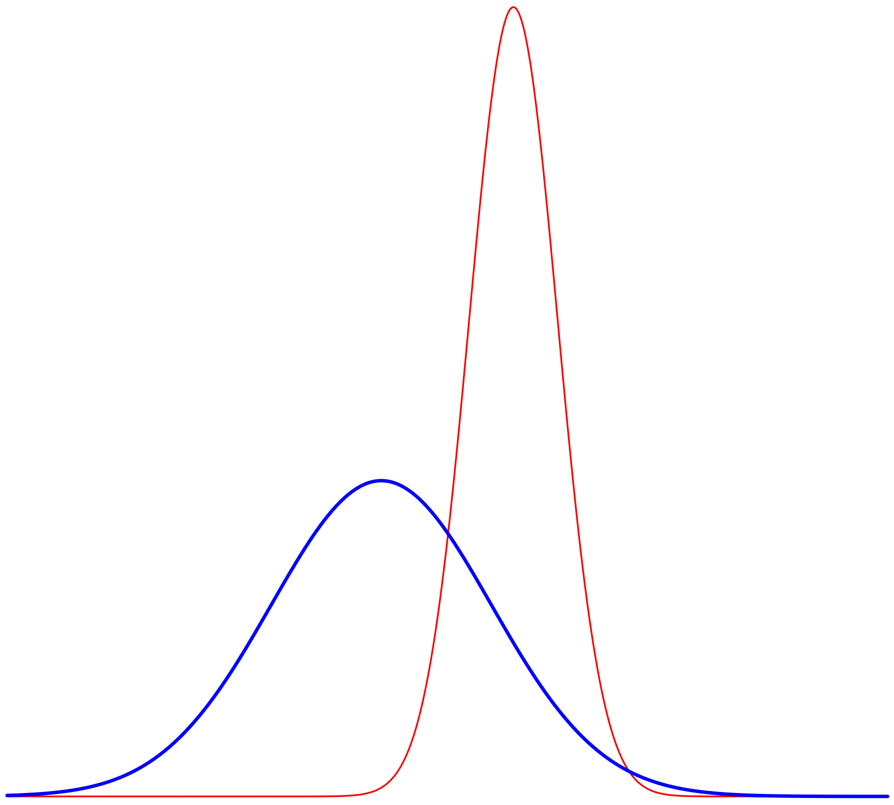


Figure 2.1: A mixture model over two separate Gaussians

## *2.2 Statistical Learning*

In statistical learning the primary idea is that we have an observed data set $D$ with $n$ observed points such that $D = \{d_1, d_2, \ldots, d_n\}$. This data is generated by some unknown distribution or mixture of distributions and we want to learn the parameters of these distributions so we can either predict future data points (i.e. $p(d_{n+j} \mid D)$, find the best clustering or utilize the learned parameters in some other task. For example, topic models can be used to classify incoming documents or cluster documents in the existing corpus.

One simplifying assumption is to consider the data points independent and identically distributed [16]. This maps to the documents being a bag of words.

Bayes' rule ( 2.1 ) relates these in a simple formula that is the foundation of Bayesian statistics. It provides a simple, effective way to learn and update knowledge from newly discovered evidence (data). Given a set of mutually exclusive hypothesis $H$ that compete to explain the model, newly discovered data $d$ and prior knowledge about the hypothesis $P(h_i)$ that was held before $d$ was observed, we can use 2.1 to update the probability of our hypothesis conditioned on the new data.

$$P(h_i \mid d) = \frac{P(d \mid h_i)P(h_i)}{P(d)} \tag{2.1}$$

$$where, P(d) = \sum_{h \in H} P(d \mid h)P(h) \tag{2.2}$$

In equation 2.1 the term $P(h_i \mid d)$ is known as the **posterior**, $P(d \mid h_i)$ is known as the **likelihood** and $P(h_i)$ is the **prior**. The denominator stays constant among competing hypothesis and normalizes the posterior probability. Thus,

$$posterior \propto likelihood * prior \tag{2.3}$$

In our case the hypothesis is the distribution parameter $\theta$ and the observed data set is $D$, so we will rewrite Bayes' rule as:

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)} \tag{2.4}$$

From a frequentist view point, Bayes' merely provides the conditional probability of one hypothesis even given the others. However, Bayes' rule is a powerful tool for statistical learning in subjective or belief oriented probability theory[17, 18]. It allows incorporation of beliefs and learning from newly observed data, which in some ways models human learning.

There are a number of ways to learn using this expression and we explore a few next.

### 2.2.1 Maximum Likelihood Estimation (MLE)

The goal of MLE is to find the parameters that maximize the likelihood ($L(\theta \mid D)$). It is considered a non-bayesian approach since it assumes a uniform prior [14, 16].

$$L(\theta \mid D) = P(D \mid \theta) = \prod_{d \in D} P(d \mid \theta) \tag{2.5}$$

In general we are interested in the ordering between parameters, so in practice log likelihood is used since it is a monotone and computationally more efficient.

$$\log L(\theta \mid D) = \log \prod_{d \in D} P(d \mid \theta) = \sum_{d \in D} \log P(d \mid \theta) \tag{2.6}$$

The MLE estimate then becomes an optimization problem as given in 2.7

$$\tilde{\theta}_{MLE} = \arg\max_{\theta} \sum_{d \in D} \log P(d \mid \theta) \tag{2.7}$$

The MLE estimate can be used to predict the distribution for newly arrived data. Say $\widehat{d}$ is a new observation. Then

$$P(\widehat{d} \mid D) \approx P(\widehat{d} \mid \tilde{\theta}_{MLE}) \tag{2.8}$$

### 2.2.2 Maximum á Posteriori (MAP)

MAP [16] estimation extends MLE by allowing the parameters to be weighed by a prior distribution. This prior distribution generally reflects a judgement that can prevent overfitting.

$$\tilde{\theta}_{MAP} = \arg\max_{\theta} P(\theta \mid D) \tag{2.9}$$

We can rewrite equation 2.9 using equation 2.4 as,

$$\begin{aligned}\tilde{\theta}_{MAP} = \arg\max_{\theta} P(\theta \mid D) = \arg\max_{\theta} \frac{P(D \mid \theta)P(\theta)}{P(D)} \\ = \arg\max_{\theta} P(D|\theta)P(\theta) \textit{[since P(D) is independent of } \theta \textit{]} \\ = \arg\max_{\theta} L(\theta \mid D)P(\theta) \\ = \arg\max_{\theta}\{\sum_{d \in D} logP(d \mid \theta) + logP(\theta)\}\end{aligned} \tag{2.10}$$

The distribution of a new data point, using MAP can be approximated by,

$$P(\widehat{d} \mid D) \approx P(\widehat{d} \mid \tilde{\theta}_{MAP}) = \int_{\theta \in \Theta} P(\widehat{d} \mid \tilde{\theta}_{MAP}) P(\theta \mid D) d\theta \tag{2.11}$$

### *2.2.3 Expectation Maximization (EM)*

The EM[15] algorithm is a procedure for obtaining a maximum-likelihood (or MAP ) estimate for parameter $\theta$ when we do not have complete information. EM can also be viewed as a generalization of the $K$-means clustering algorithm[19] and similar to $K$-means, the EM algorithm is composed of two steps. It starts out by "guessing" the parameter values then uses these values to estimate the cluster probabilities for each instance maximizing the likelihood. The first step is inferring the "expectation" (E-step) while the second is maximizing the likelihood (M-step), which is repeated till the change in likelihood is negligible. Note that this is different from $K$-means where iteration stops once instances do not change. EM is guaranteed to reach a local maximum, which means that the initialization step becomes important and it is a good idea to repeat the process multiple times with different initializations.

For example in figure 2.1 there are two clusters. Both clusters are gaussian but they have different parameters. If we are given just the data points and information that there are two clusters then we are dealing with incomplete information and the task at hand would be to infer the distribution parameters for both the clusters ($\mu$ and $\sigma$) and the likelihood of each cluster. EM solves this problem by making an initial estimate of these parameters, which are then used to infer the cluster probability for each data point. This constitutes the expectation step. This is followed by again estimating the parameters based on the newly inferred probabilities trying to maximize the likelihood. This constitutes the maximization step.

Let $C$ denote the set of $k$ clusters and $d_i$ is a data point, then the probability distribution over $d_i$ is:

$$P(d_i) = \sum_{c \in C} P(d_i \mid C = c) P(C = c) \tag{2.12}$$

Estimating the parameter(s) that define these distribution will allow us to infer the probability of each data point belonging to any of the clusters.

The likelihood for this mixture model with unknown variables $C$ is [17]:

$$P(d \mid \theta) = \sum_{c \in C} P(d, C \mid \theta) \tag{2.13}$$

The problem with expectation maximization is that it is not scalable as parameters increase and it provides point estimates of the parameter $\theta$. To infer a distribution over $\theta$ one needs Bayesian inference.

### *2.2.4 Bayesian Inference*

The MAP equation 2.10 is similar to MLE equation 2.7 except for the addition of a prior. The prior itself can be parameterized, i.e. $(P(\theta) = P(\theta|\alpha))$, which we will see in LDA. In this case $\alpha$ is known as a hyperparameter. This addition of a prior makes this a Bayesian inference problem [16] .

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)} \tag{2.14}$$

We are no longer limited to finding the maximum, which makes it necessary to calculate the marginal likelihood or normalization term $P(D)$, which can be calculated as:

$$P(D) = \int_{\theta \in \Theta} P(D \mid \theta)P(\theta)d\theta \tag{2.15}$$

To predict the distribution of a new incoming data point we can use:

$$\begin{aligned} P(d \mid D) &= \int_{\theta \in \Theta} P(d \mid \theta)P(\theta \mid D)d\theta \\ &= \int_{\theta \in \Theta} P(d \mid \theta)\frac{P(D \mid \theta)P(\theta)}{P(D)}d\theta \end{aligned} \tag{2.16}$$

The calculation of the marginal likelihood $P(D)$ is hard in practice as it becomes intractable. This has lead to the development of numerous approximation algorithms. One such method is knowns as Gibbs sampling, which we use for inference on the LDA topic model as well. We direct the interested user to [20, 16] for more details.

## 2.3 Topic Modeling

LDA is a probabilistic topic model that evolved from LSI and PLSI. We first introduce LSI and PLSI before presenting LDA in detail.

### 2.3.1 Latent Semantic Analysis/Indexing (LSI)

Latent Semantic Analysis [9, 10, 21] is an influential work that automatically finds higher order structures for text indexing to improve retrieval performance. It originally conceived as a model of human cognition, which was developed to explain how very young children acquire new vocabulary at such a high rate without any external explicit direction [22]. It captures the meaning or semantic information embedded in large text corpus without human supervision. In essence LSA takes the documents and words as input and then transforms them into a so called semantic space where documents and words that are closer in semantic are closer in this space. LSA has been used to automatically grade essays, to automate tutoring and has found utility in the information retrieval community, which is our primary concern here.

Generally searchers want to retrieve results on the basis of some meaning/semantic, which is not reliably captured by words in a document. Retrieving documents via simple lexical matching is a good starting point but not very accurate because words exhibit synonymy and polysemy. Synonymy reduces recall since documents that have a synonym of the query term are ignored in lexical matching. Polysemy on the other hand reduces precision since a document that is matched might be using the query term in a different context. Previous attempts have included standardizing vocabularies and domain specific thesaurus guided search but these are expensive operations that have not shown consistent improvement in search results.

LSI attempts to mitigate these shortcomings by automatically clustering text documents into semantic categories that improve the quality of information retrieval requests as well as the ability to navigate large text collections. The assumption is that there is some “latent” semantic structure in how words are used. Discovery of this latent structure allows defining the corpus and the queries in these higher order structures, which have reduced dimensionality and less noise improving both the accuracy as well as the efficiency of the retrieval process. In the paper the authors use singular-value decomposition (SVD) to perform their latent semantic indexing analysis. A large matrix relating terms and documents is factored into a set of 50-150 orthogonal factors from which the original matrix can be approximated. SVD was chosen since it is relatively scalable, allows throttling of power by controlling the number of dimensions and last, but not the least is that both terms and documents are projected onto the same vector space thereby allowing comparison.

The resulting reduced space allows clustering as per the corpus term usage reducing the effects of polysemy and synonymy allowing objects to be close to each other even if there is no common term. Cosine similarity is used to return a ranked list of the nearest objects. The first step is to create a document-term matrix, which is then factored using SVD to derive a particular latent semantic representation by approximating the original matrix using fewer orthogonal factors (derived dimensions). The underlying premise is that the reduction eliminates noise thereby improving the quality. This new model can approximate but not perfectly construct the original matrix. In the paper the authors use 50-100 derived dimensions.

Experiments done with one information science data set showed a 13% improvement over lexical matching while the results with another notoriously hard data set showed no difference. One experiment on a local data set without relevance judgements provided "motivating" results since the data set was known to the authors but there were no relevance judgements. The final experiment was in describing 480 groups of people characterized by the description on their projects.

One criticism of LSA, which also applies to PLSI, LDA is that it ignores word order and word syntax thereby not capturing the real meaning. While it is true that there is a lot of information that is not captured in not taking these into account, it must also be realized that there is a great deal of information that is carried by the words alone regardless of the order. For example, consider the words "mountain, boy, climbed, the". Even without knowing the order we can infer a lot. It is due to this that models like LSA, LDA while not perfect provide good results.

### *2.3.2 Probabilistic Latent Semantic Analysis/Indexing (PLSI)*

PLSI[11] adds a generative model and statistical foundation to LSI. The claim is that it addresses shortcomings of lexical matching and LSI by creating a better latent space based on solid statistical foundation amenable to statistical inference techniques. The authors claim that empirical results show that PLSI outperforms LSI.

The generative model in PLSI can be described as follows. We are given a set $D$ of $N$ documents, a set $W$ of $M$ words and a set $Z$ of $K$ topics, which is the latent variable. We first select a document $d$ with probability $P(d)$, then a topic $z$ with probability $P(z|d)$ and finally a word $w$ with probability $P(w|z)$. This process is repeated till the corpus is completely generated. The process does not restrict the distributions themselves, which as usual are selected for ease of inference. Note that $Z$ is not visible in the generated corpus and hence it is a latent factor. Unlike LSI the documents are assigned to each cluster with some weight/probability.

The actual model fitting is done via a modified version of EM known as Tempered EM[11], which reduces over-fitting when compared with EM. The main criticism of PLSI is that the number of parameters grows linearly with the number of documents, which causes the inference to grow exponentially. Further, the model is not amenable to the classification of new documents once the model has been inferred.

### *2.3.3 Latent Dirichlet Allocation*

Latent Dirichlet allocation (LDA)[12, 16, 4] is an unsupervised, probabilistic, text clustering algorithm. LDA can be viewed as another method that allows documents to be categorized into semantic topics similar to LSI and PLSI. It is however a soft clustering algorithm since it defines each documents as a distribution over these topics unlike LSI where each document is part of a single semantic cluster.

LDA defines a generative model that can be used to model how documents are generated given a set of topics and the words in the topics. A bayesian network allows for the definition of a probabilistic process that generates the observed data in the underlying network. Given the data one can use inference to find the parameter values of the underlying model. It is a flat clustering model, i.e. it does not infer relationships amongst the various topics. It is known as a soft clustering algorithm since it allocates every document to every topic with some probability. Figure 2.2

depicts a simple example.

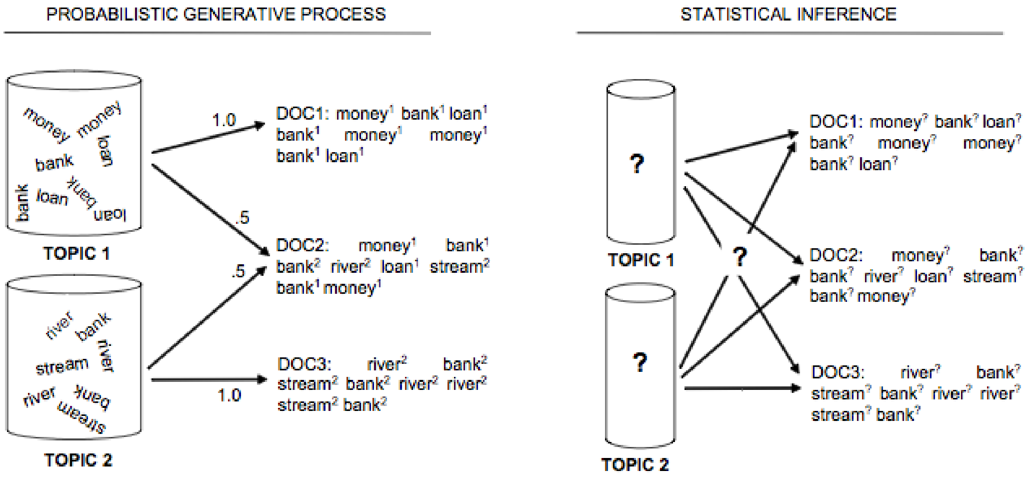

PROBABILISTIC GENERATIVE PROCESS
STATISTICAL INFERENCE
money money bank loan bank loan bank loan
TOPIC 1
river bank river stream river bank stream
TOPIC 2
1.0
.5
.5
1.0
DOC1: money$^1$ bank$^1$ loan$^1$ bank$^1$ money$^1$ money$^1$ bank$^1$ loan$^1$
DOC2: money$^1$ bank$^1$ bank$^2$ river$^2$ loan$^1$ stream$^2$ bank$^1$ money$^1$
DOC3: river$^2$ bank$^2$ stream$^2$ bank$^2$ river$^2$ river$^2$ stream$^2$ bank$^2$
?
TOPIC 1
?
TOPIC 2
?
DOC1: money$^?$ bank$^?$ loan$^?$ bank$^?$ money$^?$ money$^?$ bank$^?$ loan$^?$
DOC2: money$^?$ bank$^?$ bank$^?$ river$^?$ loan$^?$ stream$^?$ bank$^?$ money$^?$
DOC3: river$^?$ bank$^?$ stream$^?$ bank$^?$ river$^?$ river$^?$ stream$^?$ bank$^?$

Figure 2.2: Topic modeling as a statistical inference problem[4]. The left hand side show how 3 documents are generated using 2 topics LDA. The right hand side depicts the problem when all we have are the documents and we need to infer the parameters.

*Generative Model*

Figure 2.3 depicts a Bayesian network modeling document generation in the LDA model. Bayesian networks are a special case of Graphical Models, which are used to model probabilistic phenomenon by exploiting conditional independence simplifying inference in many cases. The diagram is represented in what is known as plate notation. The round circles denote random variables and observed variables are shaded with a double boundary while latent variables are not. Note that in this case the only observed r.v. are the words in documents. The edges denote conditional probability distributions with the the edge starting at the parent node and ends at the child node that is dependent on the parent node. The rectangles denote repetition with the number of the lower right signaling the number of repetitions. The notation used is described in Table 2.1 while Figure 2.4 explains the generative process in pseudo code format.

Table 2.1: LDA Notation

| Symbol | Description |
|---|---|
| $V$ | A vector denoting the fixed vocabulary. |
| $\omega$ | A unit vector that denotes a single word in $V$. |
| $w_{d,n}$ | The word assigned to the $n^{th}$ word in the document $d$. |
| $w$ | The set of all the words in the corpus. |
| $m$ | The number of documents. |
| $w_d$ or $d$ | A single document that is sequence of $n$ words denoted by $d = w_{d,1}, w_{d,2}, \ldots, w_{d,n}$. |
| $D$ | The set of all the documents. |
| $N_d$ | The number of words in document $d$. |
| $z_{d,n}$ | The topic representing $w_{d,n}$. |
| $k$ | The number of topics. |
| $\theta$ | A $m \times k$-dimensional vector denoting the topic distribution for all documents. |
| $\theta_d$ | A $k$-dimensional vector denoting the topic distribution for document $d$. |
| $\phi_i$ | A $k \times V$-dimensional vector denoting the word distribution for topic $i$. |
| $\phi_i$ | A $V$-dimensional vector denoting the word distribution for topic $i$. |
| $\alpha$ | Vector parameter for the Dirichlet for topic mixture proportion for a document. |
| $\beta$ | Vector parameter for the Dirichlet prior on per-topic word mixture. |

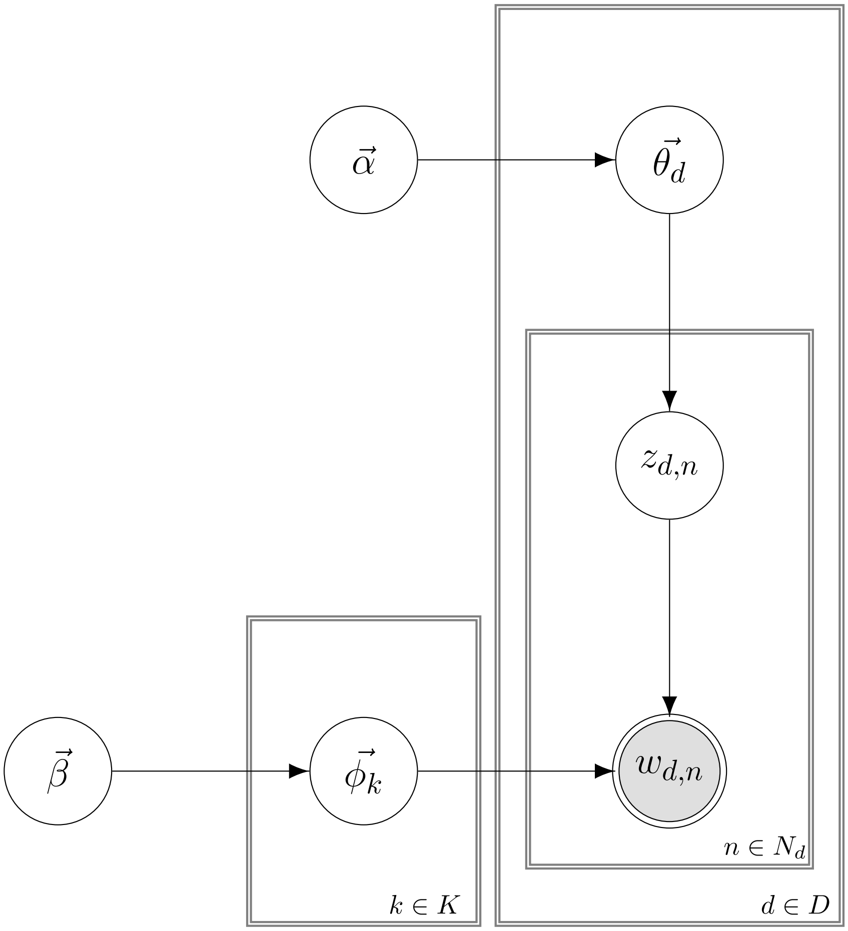


Figure 2.3: Generative model of LDA in plate notation

$\alpha$ and $\beta$ are the hyperparameters to the the document topic proportions $\theta$ and $\phi$ the topic word proportions. For each document $d$ we first generate the number of words $N_d$ then for each of these words we get the underlying topic $z_{d,n}$ from $theta_d$. $z_{d,n}$ conditioned on $phi_k$ decide the word instance $w_{d,n}$. This process is repeated till the entire corpus is generated.

The probability density of the $k-$dimensional Dirichlet random variable $\theta$ is given in equation 2.17 and similarly the prob[1]ability density for $phi$ is given in equation 2.18,

```
proc GenerateDocuments($\alpha, \beta$)
    k = | $\alpha$ |
    for $i = 1$ To $k$ do
        Choose $\phi_k \sim Dir(\beta)$
    end for

    for $d \in D$ do
        Choose $N_d \sim$ relevant distribution (e.g. Poisson)
        Choose $\theta_d \sim Dir(\alpha)$
        for $n = 1$ To $N_d$ do
            Choose $z_{d,n} \sim \theta_d$
            Choose $w_{d,n} \sim \phi_{z_{d,n}}$
        end for
    end for
end proc
```

Figure 2.4: Generating documents in LDA

where $\Gamma(x) = (x-1)!$ known as the Gamma function.

$$P(\theta \mid \alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_k^{\alpha_k - 1} \tag{2.17}$$

$$P(\phi \mid \beta) = \frac{\Gamma(\sum_{i=1}^{k} \beta_i)}{\prod_{i=1}^{k} \Gamma(\beta_i)} \prod_{i=1}^{k} \phi_k^{\beta_k - 1} \tag{2.18}$$

The probability of an individual word in this model is:

$$P(\omega) = \sum_{k} p(\omega \mid z) p(z) \tag{2.19}$$

The joint probability distribution for a single document $d$ of all the random variables given $\alpha$ & $\beta$ is:

$$P(d, z_d, \theta_d, \phi \mid \alpha, \beta) = \prod_{n=1}^{N} p(w_{d,n} \mid \phi_{z_{d,n}}) p(z_{d,n} \mid \theta_d) p(\theta_d \mid \alpha) p(\phi \mid \beta) \tag{2.20}$$

By integrating over the continuous distribution $\theta$ and summing over topic assignment $z_d$ The probability of an individual word $w_{d,n}$ in a document $d$ can be found by marginalizing over $z_{d,n}$:

$$P(w_{d,n} = \omega \mid \theta_d, \phi) = \sum_k P(w_{d,n} = \omega \mid \phi_k) P(z_{d,n} = k \mid \theta_d) \tag{2.21}$$

Equation 2.20 can be generalized to obtain the joint distribution for the entire corpus $D$:

$$P(D \mid \theta, \phi) = \prod_{d \in D} p(d \mid \theta_d, \phi) = \prod_{d \in D} \prod_{n=1}^{N_d} P(w_{d,n} \mid \theta_d, \phi) \tag{2.22}$$

The Dirichlet hyperparameter $\alpha$ and $\beta$ influence the document topic distribution and the the topic word distribution respectively. Lower values increase sparsity, which results in more decisive assignments. In general a symmetric hyperparameter is used. Also the hyperparameters themselves can be estimated[12, 16].

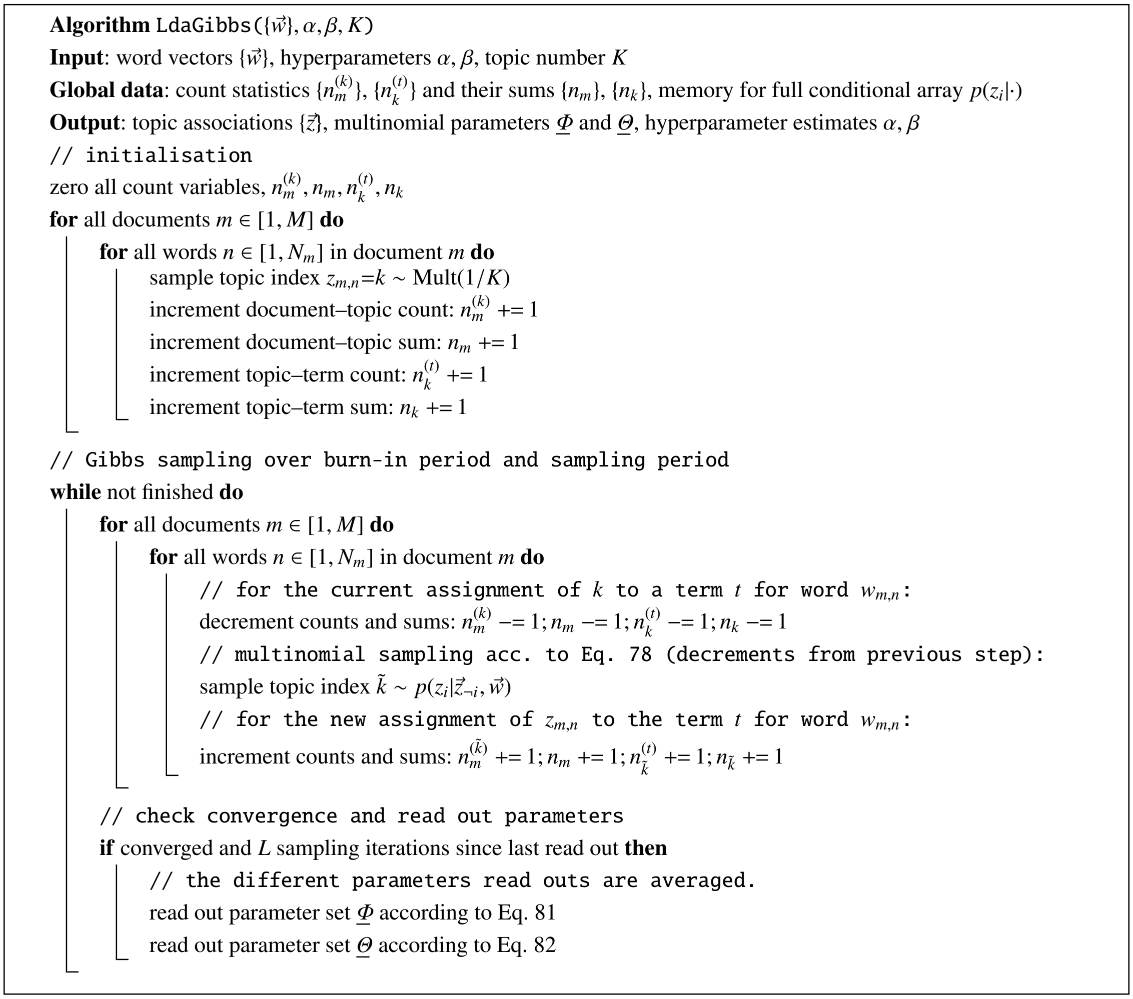
*Inference*

Given a set of documents $D$ and the LDA model we can use inference to find out $\theta$, $\phi$ and the topic assignment for each word $z$, which can be views as:

$$P(z, \theta, \phi \mid w, \alpha, \beta) = \frac{P(z, \theta, \phi, w \mid \alpha, \beta)}{P(w \mid \alpha, \beta)} = \frac{P(z, \theta, \phi, w \mid \alpha, \beta)}{\int_{\phi_{i:k}} \int_{\theta_{1:m}} \sum_{z_{1:m}} P(z, \theta, \phi, w \mid \alpha, \beta)} \tag{2.23}$$

However exact inference of LDA is generally intractable [12, 16] due to the denominator in equation 2.23 and approximate inference algorithms are used. The original paper [12] used a mean-field variational expectation maximization algorithm though Gibbs sampling [4, 16], a specialized Markov Chain Monte Carlo (MCMC) algorithm, is more frequently used.

Gibbs sampling provides us with posterior estimates of the topic assignments $z$, which can then be used to estimate $\theta$ and $\phi$. Figure 2.5 from [16] shows the gibbs inference algorithm for LDA. See [4, 16] for a detailed explanation and derivation of the collapsed gibbs sampler for LDA.



```
Algorithm LdaGibbs($\{\vec{w}\}, \alpha, \beta, K$)
Input: word vectors $\{\vec{w}\}$, hyperparameters $\alpha, \beta$, topic number $K$
Global data: count statistics $\{n_m^{(k)}\}$, $\{n_k^{(t)}\}$ and their sums $\{n_m\}$, $\{n_k\}$, memory for full conditional array $p(z_i|\cdot)$
Output: topic associations $\{\vec{z}\}$, multinomial parameters $\underline{\Phi}$ and $\underline{\Theta}$, hyperparameter estimates $\alpha, \beta$
// initialisation
zero all count variables, $n_m^{(k)}, n_m, n_k^{(t)}, n_k$
for all documents $m \in [1, M]$ do
    for all words $n \in [1, N_m]$ in document $m$ do
        sample topic index $z_{m,n}=k \sim \mathrm{Mult}(1/K)$
        increment document–topic count: $n_m^{(k)}$ += 1
        increment document–topic sum: $n_m$ += 1
        increment topic–term count: $n_k^{(t)}$ += 1
        increment topic–term sum: $n_k$ += 1
// Gibbs sampling over burn-in period and sampling period
while not finished do
    for all documents $m \in [1, M]$ do
        for all words $n \in [1, N_m]$ in document $m$ do
            // for the current assignment of $k$ to a term $t$ for word $w_{m,n}$:
            decrement counts and sums: $n_m^{(k)}$ −= 1; $n_m$ −= 1; $n_k^{(t)}$ −= 1; $n_k$ −= 1
            // multinomial sampling acc. to Eq. 78 (decrements from previous step):
            sample topic index $\tilde{k} \sim p(z_i|\vec{z}_{\neg i}, \vec{w})$
            // for the new assignment of $z_{m,n}$ to the term $t$ for word $w_{m,n}$:
            increment counts and sums: $n_m^{(\tilde{k})}$ += 1; $n_m$ += 1; $n_{\tilde{k}}^{(t)}$ += 1; $n_{\tilde{k}}$ += 1
    // check convergence and read out parameters
    if converged and $L$ sampling iterations since last read out then
        // the different parameters read outs are averaged.
        read out parameter set $\underline{\Phi}$ according to Eq. 81
        read out parameter set $\underline{\Theta}$ according to Eq. 82
```

Figure 2.5: Gibbs sampling algorithm for LDA [16].

*Evaluation*

There are in general two broad categories of evaluation metrics for probabilistic topic models like LDA. The metrics in the first category are motivated by how the trained topic model will be put to use. The metrics are then derived from the improvement in the performance of the task at hand using the topic model. For example if the topic model is to be used for information retrieval then the improvement in precision and recall can be used. The problem with this approach is that it is hard to measure the quality of topic models in general.

The second category of metrics is concerned with measuring the quality of the trained topic model without taking into consideration the task at hand. The most accepted evaluation metric for probabilistic topic models like LDA is to train the model using a fixed proportion of the data and then evaluate the likelihood of the held out documents using the trained LDA model. Yet another approach trains using a part of each document and then gauges the quality of the trained model by its ability to predict the held out words in each document. There are also metrics from clustering algorithms that are used to measure the quality of the inferred topics but these require a set of objective evaluation for comparison.

When a priori categorization is not available the likelihood of the held out data set is standard metric to use [12, 16]. A better model will have a higher likelihood on the held-out test set. However since likelihood numbers are generally very large negative numbers and there is some variability in the length of documents in the corpus an alternative measure known as ***perplexity*** is used, which is defined as:

$$perplexity(D_{test}) = exp\{-\frac{\sum_{d \in D} \log p(w_d)}{\sum_{d \in D} N_d}\} \tag{2.24}$$

A model with lower perplexity on the test data set signal a better fit. To calculate perplexity we need to compute the likelihood of every single document in the test data using the LDA model created from the training data.

$$p(D_{test} \mid D_{train}) = \int P(D_{test} \mid \theta, \phi) P(\theta, \phi \mid D_{train}) d\theta d\phi \tag{2.25}$$

This computation is intractable and various approximation algorithms have been derived. The interested reader is directed to [23, 24] for details on the state of the art on estimating likelihood .The left-to-right algorithm [23] is well accepted in the research community for language modeling.

### *2.4 Social Network Analysis*

A social network is a graph where the nodes (or actors) are entities like humans, organizations, etc. and the edges connecting them represent social relationships. These edges could be explicit like in the case of Twitter where users follow other users or implicit, e.g. connecting coauthors in a citation network.

One important area of research in social networks is how to measure the importance or influence or authority of actors in a social network. PageRank [25] is the most well known ranking algorithm for networks and a lot of other algorithms have been developed based on PageRank. One such metric is TwitterRank [8], which extends PageRank with topical similarity to find influential users in Twitter like networks. The are other more general algorithms like the "hubs & authorities"[26] model that predates and influences PageRank.

More generally the concept of centrality measures how important a node is within a graph. The most simple centrality metric is to look at only the degree, i.e. the number edges of a node. Social networks borrow heavily from network theory and a large number of measures have been developed to quantify various characteristics. We direct the interested reader to [27] for a detailed exposition.

We are interested in social networks where the actors produce and consume textual information. In a social network users follow other users for a wide variety of reasons but we argue that in general the influence of a user should be highly correlated with the importance of their content. In other words users that produce content that is important and relevant to other users will gain more influence, which also implies that over time a user with better content will have more followers. We use PageRank to rank actors based on influence in our study because it is well accepted in the community and is relatively easy to implement.

# Chapter 3

# SocialLDA

In this chapter we present a detailed description of the SocialLDA topic modeling approach.

## *3.1 Exploiting social links*

The LDA topic modeling process in a large social network is depicted in Figure 3.1. This process has been improved such that when newer documents come in one does not have to recalculate the entire LDA but save time by partial calculations. In [28] the authors present three different approaches. They then improve upon by actually presenting a maximum-entropy based classification method for assigning the topic distribution to newer documents. The authors point out that in streaming environments the topic model might drift requiring the original LDA model to be regenerated and also suggest a simple heuristic as to when the model needs to be retrained. The general idea of this approach is captured in Figure 3.2. Even though this process is supposed to improve the process it still requires recalculation of the LDA model so our approach will further improve this streaming process as well.

In a social network like Twitter, Tumblr, etc. users follow other users. There are numerous reasons why people follow other users but one important factor is the quality of content created by users. Users who produce high quality, original content tend to have more influence in social networks. We claim that an LDA model based on the messages, authored by and received by, a fixed number of users ($Top-K$) that are highly ranked on an influence metric like PageRank provides a good approximate model to use for classifying documents of other users in the network. This approach

```
proc CreateTopicModels($G, Z, P$)
▷ G is the social graph, Z is the number of topics and P is the repeating interval.
    for every time interval $P$ do
        for every user $u_i$ in $G$ do
            Calculate LDA with Z topics
        end for
    end for
end proc
```

Figure 3.1: Computing LDA Topic Models in Social Networks

```
$First$ :
for every user $u_i$ in $SocialGraph$ do
    Calculate LDA $L_i$ with Z number of topics on existing corpus
    Train $Max - Ent$ classifier $ME_i$ for $u_i$ using $L_i$
end for

$Then$ :
for  each new document $D$ arriving for user $u_i$ do
    Use $ME_i$ to assign topic distribution to $D$
end for

$Periodically\ checking$ :
if Topic drift significant for user $u_i$ then
    ▷ [For example if less than 70% words common between test & training data]
    Recalculate LDA $L_i$ with Z number of topics on current corpus
    Retrain $Max - Ent$ classifier $ME_i$ for $u_i$ using $L_i$
end if
```

Figure 3.2: Streaming LDA Process

reduces the cost of computing the LDA model since a good LDA topic model can be estimated while also reducing the computational cost since the total number of messages is significantly reduced. Further this helps in the cold start problem where a model based on a user's own messages is insufficient to create a good LDA model. The performance and quality of of the approach in Figure 3.1 and **??** can be improved by taking into account the social connections to reduce the amount of recalculation as well as improving the results for most users. To this end we ran some experiments and summarize their results in Section 4.4.

This approach has an additional step of calculating PageRank on the graph. However, the cost of this step is much less when compared to the cost of calculating topic models for the rest of the users other than the $Top - K$. Also most social networks already rank users based on their authority and thus in practice this is not an issue as the information is already available.

```
proc SocialLDA($G, Z, P$)
▷ G is the social graph, Z is the number of topics and P is the repeating interval.
    for every time interval $P$ do
        Get $Top - K$ users (where $K <<| G |$)
        for every user $u_i$ in $Top - K$ do
            Calculate LDA with Z topics
        end for
    end for
end proc
```

Figure 3.3: SocialLDA Algorithm

# Chapter 4
# EVALUATION

In this chapter we present the details of our dataset, experiments and results.

## *4.1 Experimental Environment*

We use MALLET (MAchine Learning for LanguagE Toolkit)[29] for computing LDA topic models and evaluation. It is an open source toolkit that has been cited in various related papers and has working implementations of LDA inference and evaluation. We also used Java Jung[30] for computing PageRank and Gephi[31] for computing PageRank, visualizations and some other graph metrics, which we eventually ended up not using.

## *4.2 Data*

We needed a representative social graph along with timestamped messages over a significant amount of time authored by users and had initially planned to use Twitter data but they no longer allow new whitelisted clients (`http://support.twitter.com/entries/160385-how-do-i-get-whitelisted`), which limited the amount of data that we can crawl so we decided to use friendfeed data set instead.

The data set [32] is sourced from the social networking website FriendFeed (`http://friendfeed.com/`). FriendFeed is very similar to other social networking sites like Twitter, Facebook, Tumblr, etc. FriendFeed allows user to aggregate information from numerous other social networking sites and allows commenting on other entries. The dataset used was extracted by the Special Interest Group on Social Network Analysis (`http://larica.uniurb.it/sigsna/about/`) in 2009-2010 by sampling posts from publicly available messages on FriendFeed. We use the 2010-a dataset while [32] describes an earlier sampling. The characteristics of the dataset are similar. Figure 4.1 and figure 4.2 confirm that the number of followers as well as the number of posts both generally follow a power law distribution.

The dataset has been imported into postgresql RDBMS and we do not use all the fields available in the original dataset. We ignore the data on comments, likes and only use the original entries.

Table 4.1: Entries table

| Field | Description |
|---|---|
| PostID | Identifier of the entry, assigned by Friendfeed. |
| PostedBy | Identifier of the author of the post. |
| Text | Body of the message. |

Table 4.2: Users table

| Field | Description |
|---|---|
| ID | Identifier of the user, assigned by Friendfeed. |
| Name | display name. |

### *4.2.1 Feature selection*

From this dataset we remove words using a standard stop word list. We also remove infrequent words that appear in less than 2% of the documents We form our vocabulary set from the remaining words and keep that fixed for all other processing. The

Table 4.3: Following table

| Field | Description |
|---|---|
| FollowerID | ID of the user following FollowedID. |
| FollowedID | ID of the user followed by FollowerID. |

Table 4.4: Posts and relationship statistics

| | Posts | Followed by | Following |
|---|---|---|---|
| Min | 0 | 0 | 0 |
| Max | 166965 | 113923 | 112145 |
| Total | 11048231 | 27811816 | 27811816 |
| Average | 16.91 | 42.55 | 42.55 |
| SD | 264.33 | 311.60 | 652.77 |

Table 4.5: FriendFeed Dataset Characteristics

| | |
|---|---|
| Total number of users | 671,840 |
| Total number of entries | 12,450,658 |
| Total number of edges | 27,811,816 |
| Total authors who have at least a single post | 212054 |
| Total authors who have at least 25 posts | 62614 |
| Correlation coefficient between pagerank and total posts | 0.0108353974403349 |
| Correlation coefficient between pagerank and number of followers | 0.756587208245108 |
| Correlation coefficient between pagerank and number of followed | 0.0943826170571718 |

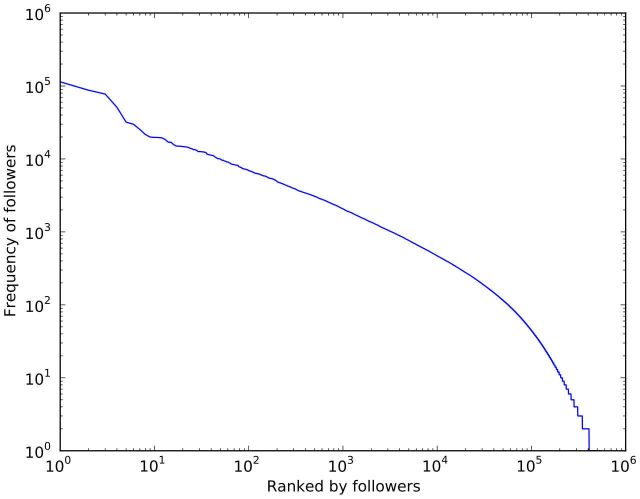


Figure 4.1: LogLog plot of Rank vs Followers

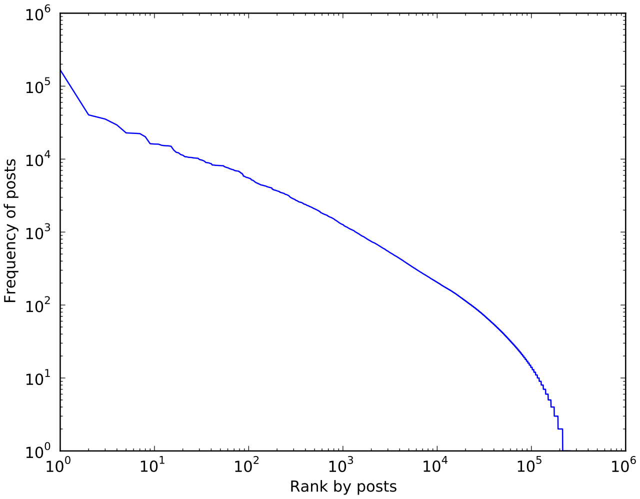


Figure 4.2: LogLog plot of Rank vs Posts

decision to prune words that appear less than 2% of the time is based on [33] where the authors empirically study and evaluate the performance of feature selection on accuracy of text classification. Unlike clustering, the text categories are known a priori. Feature selection is important because high dimensionality of the feature space is a major challenge and while what constitutes high dimensionality has changed over time - the problem itself lingers. One goal of automatic feature selection is to remove non-informative terms according to corpus statistics and the task at hand. Another use case would be to reduce dimensionality so that the data can be processed with minimal loss of information/utility.

The study looks at four different feature selection mechanisms that are each based on "term-goodness" criterion. The first one was

1. Document Frequency Thresholding (DF) Document frequency of a term signals the number of documents in which a term occurs. The idea is to remove terms in the corpus where DF is below some predetermined threshold on the assumption that rare terms do not contribute much. This is in contrast to the assumptions in web search where rare terms are given more weight.

2. Information Gain (IG) Information gain is a widely used metric quantify term-goodness in text classification problems [34]. It measures the additional amount of information that a term provides us about the class label. It can also be viewed as the change in entropy of the distribution after a term. $IG(w) = H(C) - H(C|w)$
$$= -\sum_{c\in C} P(c)logP(c) + \sum_{w\in[0,1]} P(w) \sum_{c\in C} P(c|w)logP(w|c)$$
$$= -\sum_{c\in C} P(c)logP(c)+P(w)\sum_{c\in C} P(c|w)logP(w|c)+P(\neg w)\sum_{c\in C} P(c|\neg w)logP(\neg w|c)$$

3. Mutual Information (MI) The mutual information

4. $\chi^2 - test$ (CHI)

5. Term Strength (TF)

The two classification methods used to test the effects of the above feature selection were:

1. k-nearest-neighbor classifier (kNN)
2. Linear Least Squares Fit mapping (LLSF)

The two corpora for this study:

1. Reuters-22173 collection
2. OHSUMED collection

The authors conclude that IG and CHI were the most effective in that a large number of features could be removed while improving the results of the classifier. However DF had a very similar effect and was also the lowest cost solution so it is a good choice.

The outcome provides firm footing, at least for text classification, for pruning features at high thresholds thereby reducing the dimensionality of the problem. In terms of application to LDA or topic models it gives us some foundation that using DF might be worth trying. Other metrics do not seem to be good candidates for calculation of topic models since they rely on existing labels. I do not use stemming since there is general consensus that there is not much utility in doing so. Lemmatization, spell checking, and non-standard abbreviation expansion might be useful and should be considered in a real product to remove noise.

### *4.3 Experiments*

The general idea of our algorithm is to reduce the computation cost of LDA by only calculating the model for a few selected users and then sharing the model among other users in the network. Figure 4.3 depicts this idea. We use PageRank as the influence metric and create a LDA topic model by combing all the documents authored by and passing through the Top K users. We need to verify the suitability of substituting this model in place of models created by individual users as well as a global LDA model.

We first partition our data set into a training and a test set in a 1:1 ratio in time order, i.e. every message in training has a creation time before that in test, which simulates processing in the real world. We train a global lda based on the training set as well as $Top - K$ LDA models with 25, 50, 100, 200 highest ranked users. We take a random sample of users and for each one of them we train a topic model based on the messages they authored in training and another on messages they received in training. We consider the Left-To-Right score of the user's LDA model on the training set itself to be the best score. We also evaluate the user's LDA model on the test data set. We then compare every other model against this. The experiment is explained in 4.3 and we summarize the results in the next section.

```
for Number of topics in 10, 20, 30, 40, 50 do
    $First$ :
    Partition dataset into 50% training $\tau$ and 50% test $\phi$
    Identify Top K actors in $Social - Graph$ using PageRank
    Calculate Top 25 LDA topic model $\omega_{25}$ based on documents in $\tau$.
    Calculate Top 50 DA topic model $\omega_{50}$ based on documents in $\tau$.
    Calculate Top 100 LDA topic model $\omega_{100}$ based on documents in $\tau$.
    Calculate Top 200 LDA topic model $\omega_{200}$ based on documents in $\tau$.
authored by or received by top k actors in $\tau$.
    Calculate Global LDA topic model $\gamma$ from documents in $\tau$.

    Add $n$ users randomly from the network to $\lambda$
    for every user $u_i$ in $\lambda$ do
        OutTrainingSet = documents authored by $u_i$ in $\tau$
        OutTestSet = documents authored by $u_i$ in $\phi$
        InTrainingSet = documents received by $u_i$ in $\tau$
        InTestSet = documents received by $u_i$ in $\phi$
        Create LDA topic model $\sigma$ based on InTrainingSet
        Create LDA topic model $\sigma$ based on OutTrainingSet
        Evaluate performance of $\sigma$ using (In/Out)TrainingSet itself, baseline
        Evaluate performance of $\sigma$ using TestSet
        Evaluate performance of $\gamma$ using TestSet
        Evaluate performance of all $\omega$ using TestSet
        Compare performance
    end for
end for
```

Figure 4.3: Experiment Design

## *4.4 Results*

Table 4.6 describes the relationship between $Top - K$ users, posts and links. Using one of the $Top - K$ models results in a large reduction in the amount of documents, which results in a significant drop in the running time of the topic modeling process. Figure 4.4 shows the time taken by different models using a varying number of topics.

Table 4.6: $Top - K$ Statistics

| | |
|---|---:|
| Documents in training set | 5,665,193 |
| Documents in test set | 5,629,295 |
| Messages authored by Top 25 | 2986 |
| Messages authored by Top 50 | 3854 |
| Messages authored by Top 100 | 6854 |
| Messages authored by Top 200 | 11163 |
| Messages authored & received by Top 25 | 550,600 (09.7%) |
| Messages authored & received by by Top 50 | 566,159 (10.0%) |
| Messages authored & received by by Top 100 | 615,748 (10.8%) |
| Messages authored & received by by Top 200 | 1,662,497 (30.0%) |
| Correlation coefficient between pagerank and total posts | 0.0108353974403349 |
| Correlation coefficient between pagerank and number of followers | 0.756587208245108 |
| Correlation coefficient between pagerank and number of followed | 0.0943826170571718 |

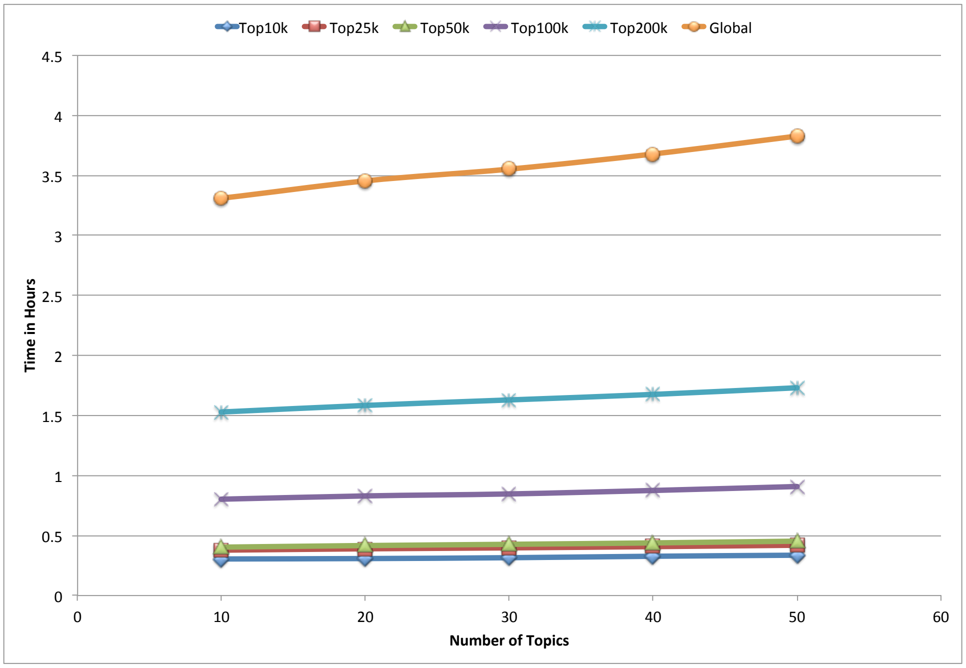
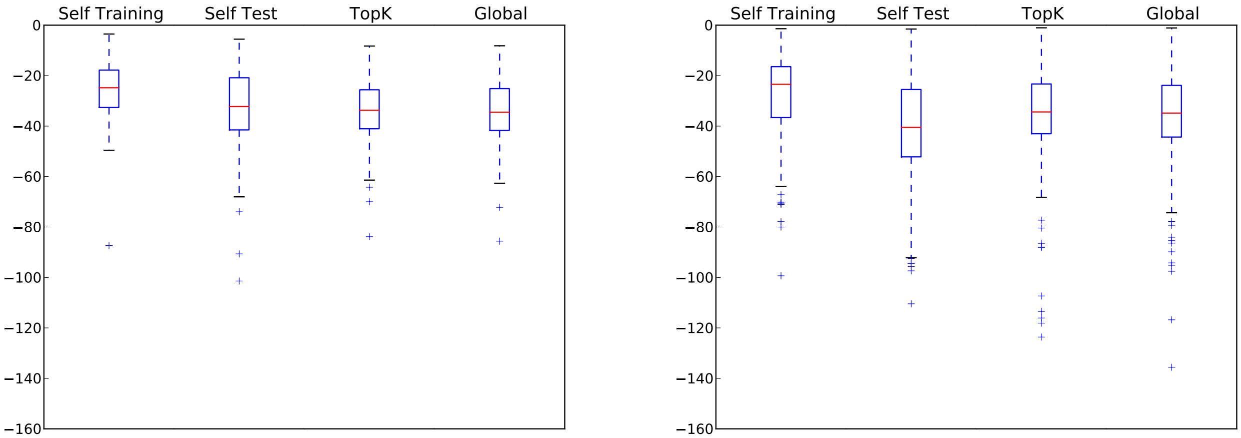


Figure 4.4: Comparing time complexity across the different approaches

Figures 4.5 to 4.13 present box plots that reflect the perplexity of the documents in the test set, with the likelihood being calculated using the Left-To-Right evaluation method as mentioned in section 2.3.3. A box plot, also known as a box and whisker plot, is generally used to show the distribution of a data set. It divides data into quartiles and the box starts at the first quartile ending at the third quartile. The line in the center denotes the second quartile, which is the median. This helps in depicting skew of the data set. The two external horizontal lines smallest and largest outliers. Points outside these whiskers denote outliers.

Each of our figure has two box plots. The one on the left shows the evaluation over incoming messages for a user and the one on the right side shows the evaluation over outgoing messages. For example, in figure 4.5 a 10 topic LDA model was evaluated. The evaluation over incoming messages clearly depicts that the TopK model is closer to the global and personal topic models with less skew. The evaluation over outgoing messages shows that it is close to the global and might be somewhat better than even the personal though with higher outliers. Similar trends are observed through the rest of the figures, i.e. the $Top - K$ models seem to be close to the global model for incoming stream and for outgoing they are closer to the user's performance. This indicates that using data from highly ranked users might produce reasonable results at a significantly lower cost.

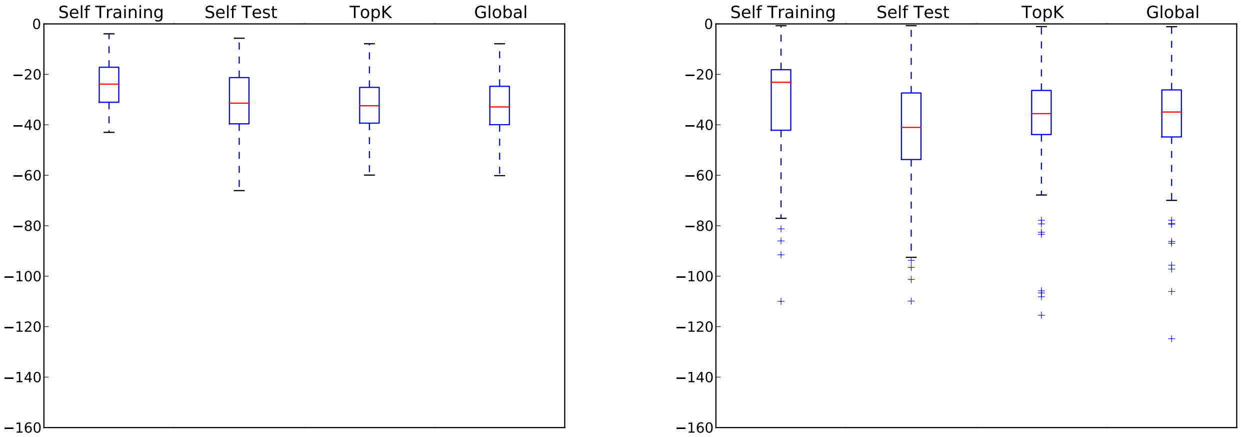


(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.5: Comparison across 10 Topics and 100 user $Top - K$ LDA.

Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top - K$ model's distribution is close to the global and the self test models.

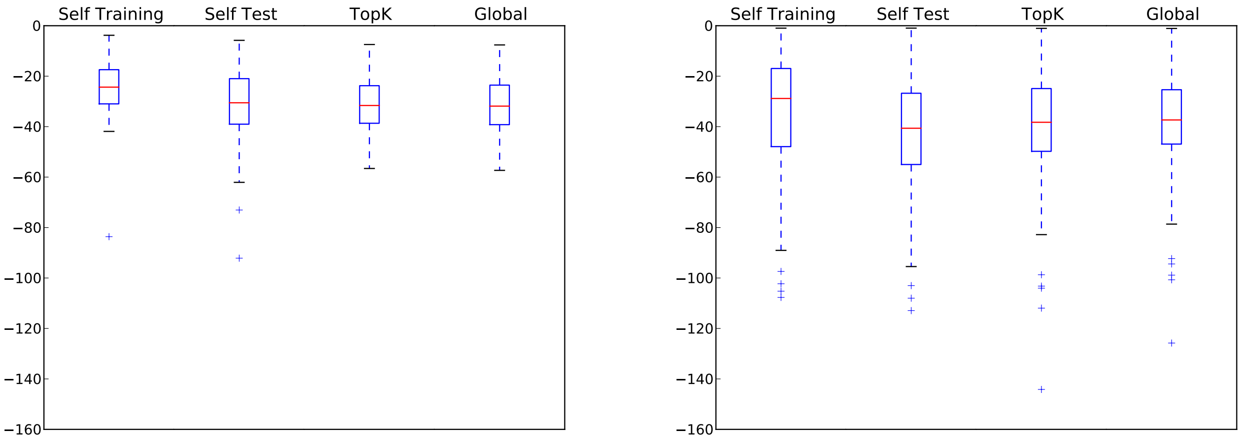


(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.6: Comparison across 20 Topics and 100 user $Top - K$ LDA.

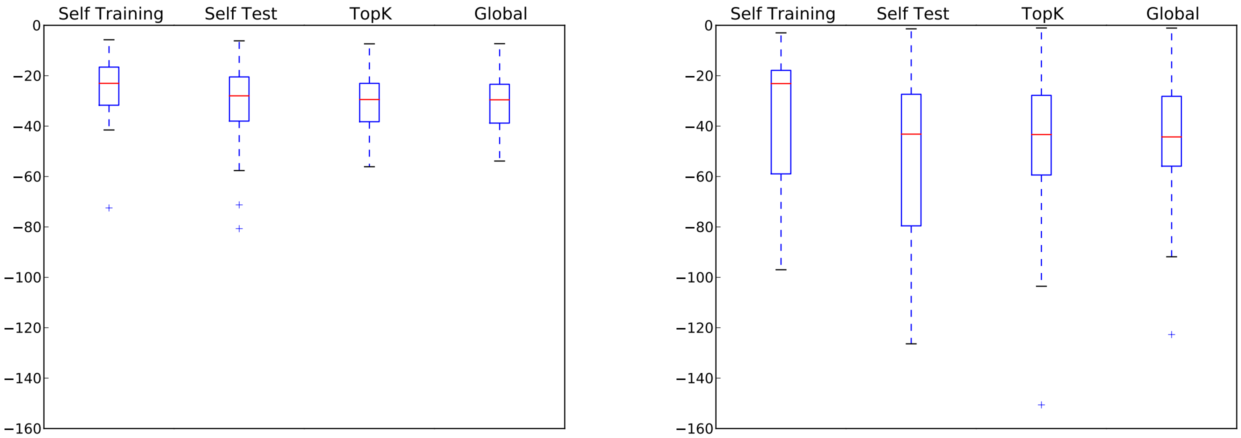
Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top - K$ model's distribution is close to the global and the self test models.



(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.7: 30Comparison across 30 Topics and 100 user $Top - K$ LDA.

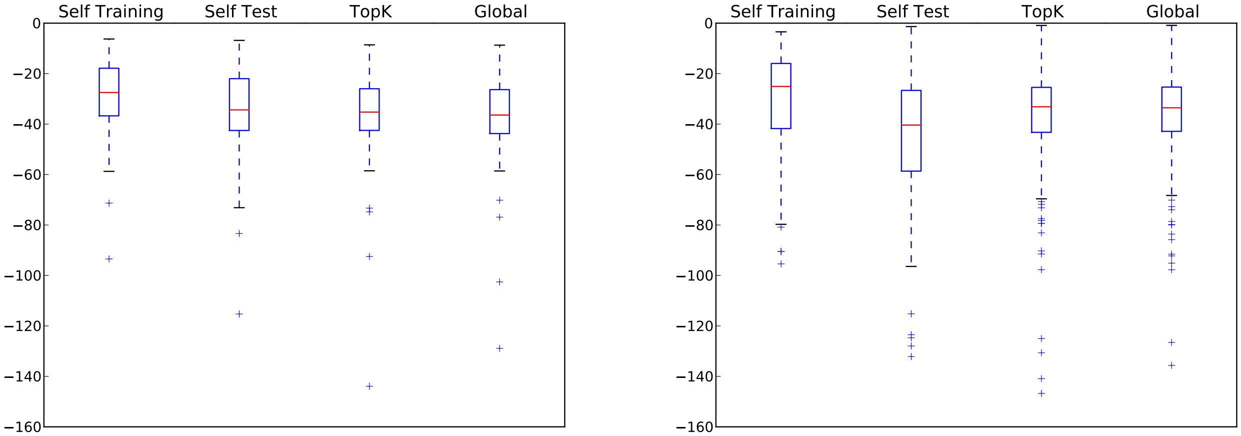


(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.8: Comparison across 40 Topics and 100 user $Top - K$ LDA.

Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top - K$ model's distribution is close to the global and the self test models.

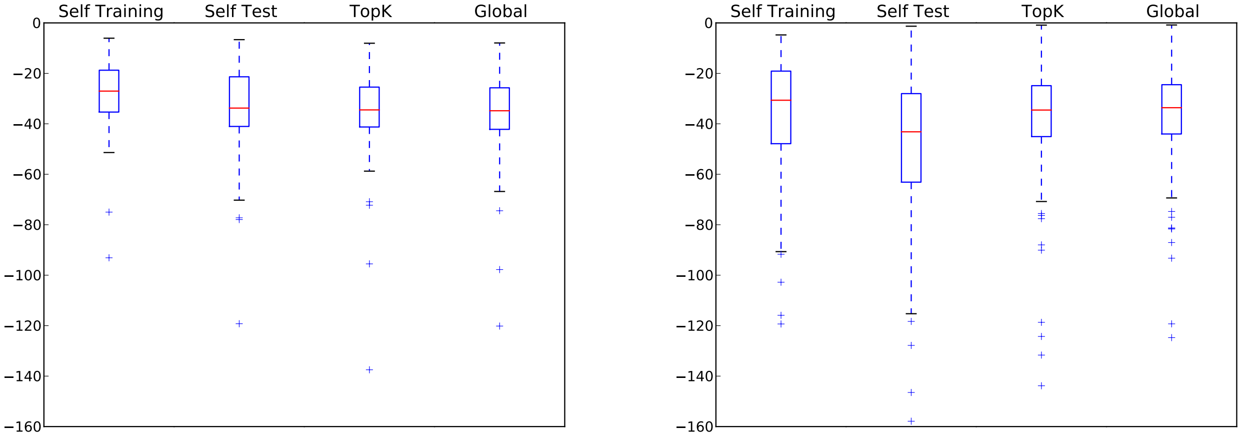


(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.9: Comparison across 10 Topics and 50 user $Top - K$ LDA.

Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top - K$ model's distribution is close to the global and the self test models.

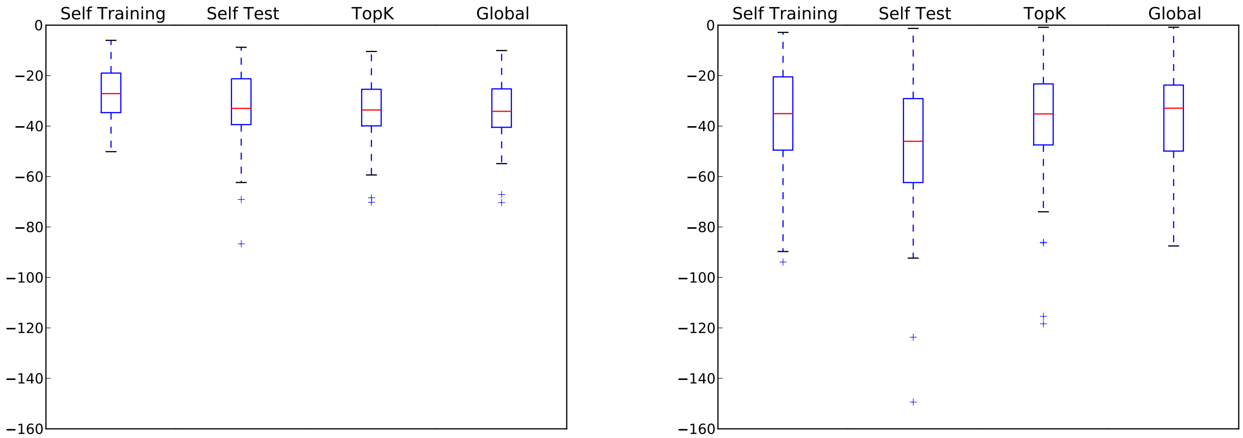


(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.10: Comparison across 20 Topics and 50 user $Top - K$ LDA.

Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top - K$ model's distribution is close to the global and the self test models.

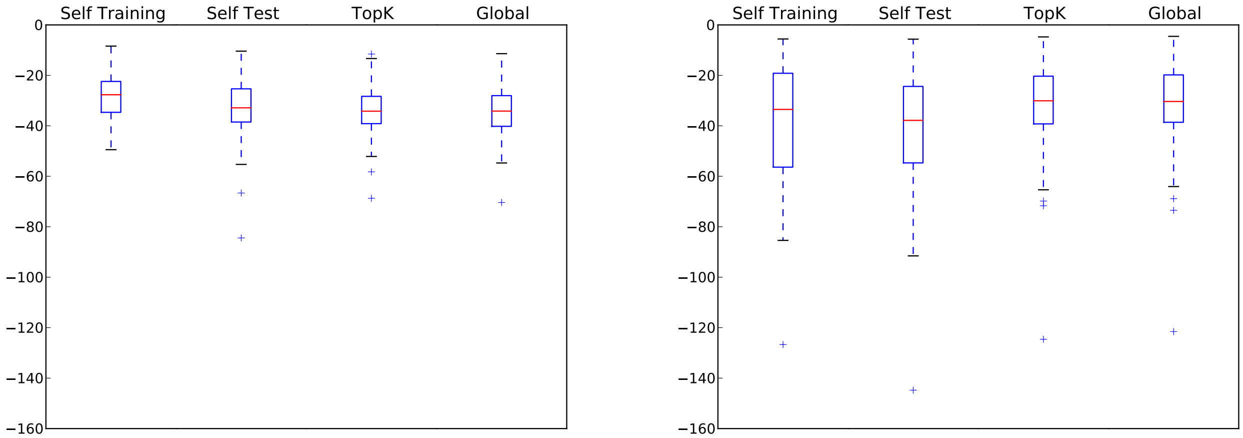


(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.11: Comparison across 30 Topics and 50 user $Top-K$ LDA.

Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top-K$ model's distribution is close to the global and the self test models.

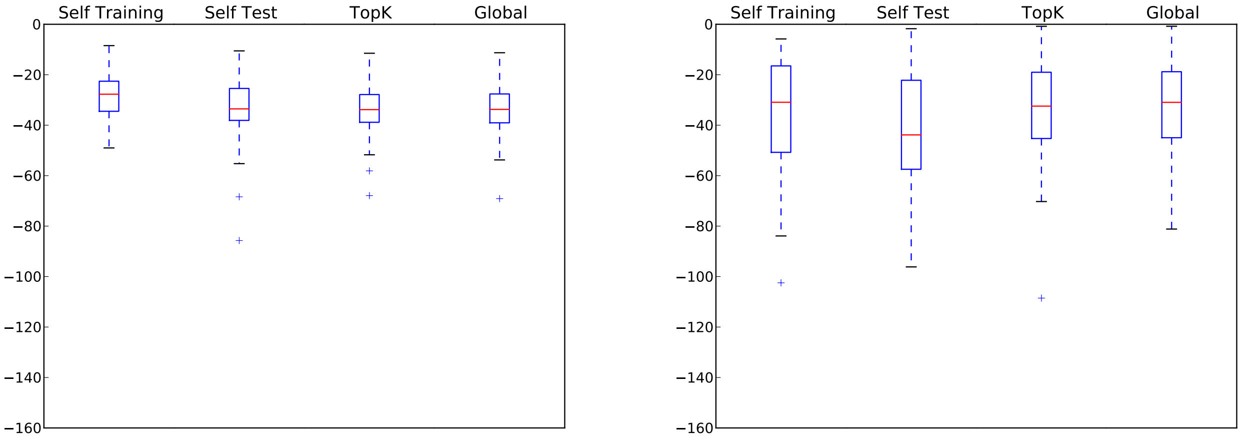


(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.12: Comparison across 40 Topics and 50 user $Top-K$ LDA.

Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top-K$ model's distribution is close to the global and the self test models.



(a) Evaluating only incoming messages

(b) Evaluating only outgoing messages

Figure 4.13: Comparison across 50 Topics and 50 user $Top - K$ LDA.

Each box depicts the distribution of the test set over different models. For both outgoing and incoming messages the $Top - K$ model's distribution is close to the global and the self test models.

# Chapter 5
# CONCLUSION & FUTURE WORK

The explosion in generation of text documents in the last decade has bought increasing attention to automatic text categorization algorithms. LDA and other probabilistic topic modeling techniques, which allow unsupervised probabilistic clustering of text documents into semantic categories have gained wide acceptance. However computing LDA over a large corpus is an expensive process. In social networks LDA based topic models can be used to decipher user interests for targeted recommendations or used by users to explore and manage their received documents. In such a scenario the computation of LDA based topic models is even more expensive because of the need to recompute periodically or compute topic models for individual users.

In this work we show that the computation cost of using LDA for a large number of users connected via a social network can be reduced without compromising the quality of the LDA model by taking into account the social connections among the users in the network. Instead of a single global model based on every document in the network we propose to use a model created from messages that are authored by and received by a fixed number of most influential users. We use PageRank as the influence measure and show that this Social LDA model provides an effective model to use as it reduces the number of documents to process thereby reducing the cost of computing the LDA. Such a model can be used both for categorizing a users incoming document stream as well as finding user interest based on the users authored documents. Further this also helps in the cold start problem where a model based on a users own messages is insufficient to create a good LDA model. The results might be explained in terms of a filtering effect. Messages that pass through highly ranked users are more likely to be more useful. An influential user is less likely to spread spam messages. Using a $Top-K$ model allows filtering of the lower quality messages. Also, the influence of the content generated by a user should be highly correlated with their influence in the network.

We consider this as the first attempt to relate topic models with social links and it is merely a small step. In terms of future research it might be interesting to compare the results of influence metrics other than PageRank. It is also quite possible that the process can be improved by partitioning the graph into segments that have their own $Top - K$ model. Yet another interesting idea would be bias the model with messages flowing through highly ranked users. Currently each message is used only once regardless of how many $Top - K$ users have seen it. If a message $m_1$ is received by a large number of $Top - K$ ranked users and another message $m_2$ is seen by a much smaller number of $Top - K$ ranked users then if we give more weight to $m_1$ in our model we could end up with a better model. Even more intriguing is to actually create a new generative model that takes into account the social links as part of the model itself.

# GLOSSARY

STATISTICAL PARAMETER: is a numerical characteristic of a population or a model. For example, a normal distribution has two parameters $\mu$ (mean) and $\sigma$ (variance). The beta distribution has two parameters and the dirichlet, which generalizes beta has k parameters.

LIKELIHOOD: The likelihood of a set of parameter values, given some observed outcomes is the probability of those observed outcomes given the parameter value.

$L(\theta \mid X) \approx P(X \mid \theta)$, where $X$ is the observed data and $\theta$ is the parameter value.

PRIOR: The estimated probability without observing the (new) data.

DIRICHLET: A dirichlet is a continuous probability distribution that is defined by a vector parameter $\alpha$ of positive real numbers.

BETA: The beta distribution is a dirichlet over two dimensions.

MODEL BASED CLUSTERING: assumes that a model generated the data and we can learn the model from the data.

PERPLEXITY: Perplexity is a measure of the quality of a proposed probability model $q$. We assume that $q$ was created using a training data set $\{t_1, t_2, \ldots, t_m\}$ and then use $q$ to predict new data points in the set $X$ of points $\{x_1, x_2, \ldots, x_n\}$ generated from the same process that generated the training data set.

$$2^{-\sum_{i=1}^{N} \frac{1}{N} \log_2 q(x_i)} \tag{5.1}$$

MIXTURE MODEL:

BAYESIAN NETWORK: A data structure that represents the dependencies among random variables so as to define a complete joint probability distribution. Also known as belief network, causal network, knowledge map. A bayesian network is itself a specific type of graphical model.

GENERATIVE MODEL: A bayesian network allows for the definition of a probabilistic process that generates the observed data in the underlying network. Given the data one can use inference to find the parameter values of the underlying model.

MONOTONE: Order preserving function.

PARAMETER: A numeric quantity usually unknown that describes a certain population characteristic.

STATISTIC: A quantity calculated from a sample of data used to estimate parameter.

# BIBLIOGRAPHY

[1] B. J. Jansen, G. Campbell, and M. Gregg, "Real time search user behavior," in *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, pp. 3961–3966, 2010.

[2] A. Asuncion, P. Smyth, and M. Welling, "Asynchronous distributed learning of topic models," *Advances in Neural Information Processing Systems*, vol. 21, 2008.

[3] L. Hong and B. D. Davison, "Empirical study of topic modeling in twitter," 2010.

[4] M. Steyvers and T. Griffiths, "Probabilistic topic models," *Handbook of Latent Semantic Analysis*, vol. 427, 2007.

[5] C. Clifton, R. Cooley, and J. Rennie, "TopCat: data mining for topic identification in a text corpus," *IEEE transactions on knowledge and data engineering*, pp. 949–964, 2004.

[6] D. M. Blei and J. D. Lafferty, "Topic models," in *Text Mining: Theory and Applications.* (A. Srivastava and M. Sahami, eds.), pp. 71–94, CRC Press, Taylor and Francis Group, June 2009.

[7] A. Banerjee and S. Basu, "Topic models over text streams: A study of batch and online unsupervised learning," in *SIAM Data Mining*, 2007.

[8] J. Weng, E. P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential twitterers," in *Proceedings of the third ACM international conference on Web search and data mining*, pp. 261–270, 2010.

[9] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[10] S. Dumais, G. Furnas, T. Landauer, S. Deerwester, and R. Harshman, "Using latent semantic analysis to improve access to textual information," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 281–285, ACM, 1988.

[11] T. Hofmann, “Probabilistic latent semantic analysis,” in *Proc. of Uncertainty in Artificial Intelligence, UAI99*, pp. 289–296, Citeseer, 1999.

[12] D. Blei, A. Ng, and M. Jordan, “Latent dirichlet allocation,” *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[13] D. Ramage, D. Hall, R. Nallapati, and C. Manning, “Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pp. 248–256, Association for Computational Linguistics, 2009.

[14] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 ed., 2003.

[15] I. Witten, E. Frank, and M. Hall, *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2011.

[16] G. Heinrich, “Parameter estimation for text analysis,” *Web: http://www. arbylon. net/publications/text-est. pdf*, 2005.

[17] T. Griffiths and A. Yuille, “A primer on probabilistic inference,” *The probabilistic mind: Prospects for Bayesian cognitive science*, pp. 33–57, 2008.

[18] I. Hacking, *An introduction to probability and inductive logic.* Cambridge Univ Pr, 2001.

[19] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* Cambridge University Press, 1 ed., 2008.

[20] P. Resnik and E. Hardisty, “Gibbs sampling for the uninitiated,” *Online manuscript: http://www. umiacs. umd. edu/resnik/pubs/gibbs. pdf*, 2009.

[21] D. Martin and M. Berry, “Mathematical foundations behind latent semantic analysis,” *Handbook of latent semantic analysis*, pp. 35–56, 2007.

[22] W. Kintsch, D. McNamara, S. Dennis, and T. Landauer, “Lsa and meaning: In theory and application,”

[23] H. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, “Evaluation methods for topic models,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1105–1112, ACM, 2009.

[24] W. Buntine, "Estimating likelihoods for topic models," *Advances in Machine Learning*, pp. 51–64, 2009.

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

[26] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[27] S. Wasserman and J. Galaskiewicz, *Advances in social network analysis: Research in the social and behavioral sciences.* Sage Publications, Inc, 1994.

[28] L. Yao, D. Mimno, and A. McCallum, "Efficient methods for topic model inference on streaming document collections," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, (Paris, France), pp. 937–946, ACM, 2009.

[29] A. K. McCallum, "Mallet: A machine learning for language toolkit." http://mallet.cs.umass.edu, 2002.

[30] J. OMadadhain, D. Fisher, S. White, and Y. Boey, "The jung (java universal network/graph) framework," *University of California, Irvine, California*, 2003.

[31] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *International AAAI Conference on Weblogs and Social Media*, pp. 361–362, 2009.

[32] F. Celli, F. Di Lascio, M. Magnani, B. Pacelli, and L. Rossi, "Social network data and practices: the case of friendfeed," *Advances in Social Computing*, pp. 346–353, 2010.

[33] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization," in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pp. 412–420, Citeseer, 1997.

[34] W. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice.* Addison-Wesley Reading, MA, 2010.