Distributed Diverging Topic Models
A Novel Algorithm for Large Scale Topic Modeling in Spark

James Andrew Marquardt

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

Master of Science

University of Washington

2014

Committee:

Martine De Cock, Chair

Jason Chuang

Sergio Davalos

Golnoosh Farnadi

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington

Abstract

Distributed Diverging Topic Models
A Novel Algorithm for Large Scale Topic Modeling in Spark

James Andrew Marquardt

Chair of the Supervisory Committee:
Associate Professor Martine De Cock
Institute of Technology

In their 2001 work *Latent Dirichlet Allocation*, Blei, Ng, and Jordan proposed the generative model of the same name that has since become the basis for most research in the field of topic modeling. The model overcame many of the shortcomings of previous probabilistic models such as allowing the inference of topics in documents not present in the learning phase, as well as allowing for topic mixtures. In the past decade the algorithm for inferring the probabilities associated with the model has been implemented in many different languages, been extended to allow topic relationships with other entities such as emotion and document label, and optimized in a variety of ways to allow faster learning. Latent Dirichlet Allocation (LDA) has found applications within a wide variety of disciplines; including digital humanities, computational social science, e-commerce, and government science policy. In short, the numerous advances and applications illustrate the significant influence of the original LDA algorithm.

However, in spite of the numerous publications and tools created as a result of LDA, the model suffers from one issue: it is extremely computationally intensive. This shortcoming is so great that its utility towards large datasets of the scale of those mined from the Internet is somewhat questionable. Additionally, the topic modeling algorithm often requires a degree of active learning, requiring feedback from a domain expert, which in certain circumstances would be ideally minimized.

In this work we present Distributed Diverging Latent Dirichlet Allocation (DD-LDA), a novel algorithm for the creation of topic models based on the original Latent Dirichlet Allocation model. The algorithm takes advantage of recent advances in distributed systems approaches to computation, and demonstrates its utility through decreased time requirements as well as increased model performance via the ability to intelligently determine appropriate model size.

TABLE OF CONTENTS

Page

LIST OF FIGURES

GLOSSARY

LDA: Latent Dirichlet Allocation, a generative model for representing document topic distributions

PLSI: Probabilistic Latent Semantic Indexing, a non-generative model for topic modeling; precursor to LDA

HADOOP: A popular cluster computing framework

SPARK: An in memory cluster computing framework

MAPREDUCE: A programming paradigm which allows distributed programs to be written as Map and Reduce phases

MPI: Message passing interface, a specification for distributed programming

HDFS: Hadoop distributed filesystem, a filesystem used in Hadoop clusters

ACKNOWLEDGMENTS

I would like to express my gratitude to everyone who supported me throughout the of course my Master's thesis. I am thankful for their inspiring guidance, invaluably constructive criticism, and friendy advice during my research. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to this work.

I express deepest thanks to Professor Martine De Cock. Her constant encouragement and feedback along with the support she has given me in my (excessively) varied academic pursuits as a student will never be forgotten.

I would also like to thank the members of my advisory committee for their dedication to helping make the most of this research.

Thank you,
James Andrew Marquardt

DEDICATION

I dedicate this work to my wonderful son, Edison, who is the light of my world. Always be the inquisitive mind you are, understand that you're never too old to learn something amazing, and know that Daddy always loves you.

Chapter 1

INTRODUCTION

Inundation with potentially excessive amounts of data is a fact of life. While this may have at one point been an inconvenience, the "big data" revolution has allowed enterprising organizations and individuals to discover a wealth of knowledge within the deluge. Many existing tools for doing this sort of profitable exploration are fairly limited to simple statistical analysis. While such tools are undoubtedly useful, there is much to be said for more abstract exploration. Often, the answer to the question "how many people are saying something" is far less interesting than the question "what are they saying."

Various approaches have been proposed for the addressing of such inquiries, topic modeling being a star among them. Topic modeling has become an indispensible tool in many investigations. It has been shown to be valuable in exploration of scientific literature and the discovery of what academics focus on similar fields [3] [11]. Topic models have been featured as an ideal way to summarize vast swaths of historical news articles in order to see how the attentions of journalists have changed over time [33]. They have helped examine early colonial perceptions of beauty [6] and model the political makeup of legislative districts [26]. In short, they have done a lot. Yet in spite of their many applications, are they feasible for use on more than just samples?

1.1 Motivations

There is a reason many of the studies which use topic modeling as their primary exploratory tool use a sample of the total data available: most available tools for topic modeling are quite slow. Research into pared down versions of the algorithms behind the inference of topic models has yielded some positive results [25], but there is definitely more that can be done.

In this work we will propose a novel approach to topic modeling which builds on years of

research on probabilistic models as well as modern advances in cluster computing. It is the goal of this work to move towards the feasibility of applying topic modeling algorithms deployed on big data frameworks to Internet scale data.

1.2 Thesis Layout

We will first cover an overview of the various big data technologies and topic modeling techniques we will use in the development of our novel algorithm in Chapter 2. We will then describe the algorithm that we are proposing, as well as provide justifications for using the proposed methods in Chapter 3. Next, we will describe the data sets we will use in order to test our proposed methods in Chapter 4. Descriptions of the various criteria we will use to evaluate our algorithms will be given in Chapter 5. Our experiment design and results will be described in Chapter 6. Finally, we will offer our observations and ideas for future work in Chapter 7.

Chapter 2

BACKGROUND

2.1 Distributed Computing

2.1.1 History

Distributed computing is a paradigm of computation that utilizes components located on networked computers. These components are able to function together through the passing of messages. Distributed computing is often differentiated from parallel computing in that a distributed system will typically not have shared memory among the networked components, as shown is Figure 2.1.

The primary motivation for the adoption of distributed computing paradigms deals with

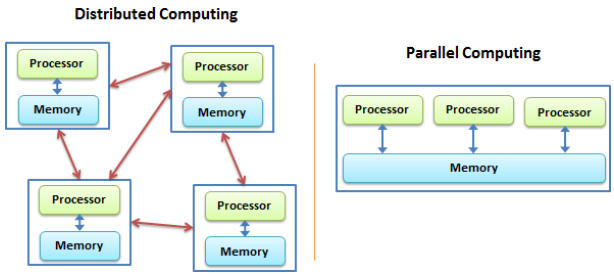

Figure 2.1: Distributed vs. parallel systems. Arrows indicate inter and intra node communication

scale. It has been re-stated many times that Moore's Law (the statement that every eighteen months either the number of transistors that can fit on a chip will double or the cost of the same number will be halved) may no longer hold true in the near future. As such, the cost of adding computational power to a single computational node is becoming such that it is becoming less cost effective. As an alternative to this type of vertical scaling, adding more nodes to the system (also known as horizontal scaling) is a way to add computational power. In theory, gains made by horizontal scaling are limited only by the number of nodes that

can be successfully incorporated to a computational cluster (although in truth, issues such as hardware and communication failure will cause the benefits to not be quite so great).
While the origins of distributed computing can be argued, one of the earliest works that laid out the formal engineering basis for performing parallel tasks on a cluster of computational nodes was put forward in [1]. In this work, the author describes a formulation for the expected speedup for using multiple computational nodes for an otherwise sequentially executed task. Many of the original computational clusters grew out of work founded on this paper, including early work on ARPANET's distributed computation of routing tables [19] and the commodity hardware Beowulf cluster described in [2].

2.1.2 Message Passing Interface

Message Passing Interface (MPI) is a standardized message passing system to allow the coordination of parallel tasks over nodes in a computational cluster. This includes the assignment of tasks to a node, communication between nodes during computation, and how output from tasks is collected. The system allows for a high degree of control in the parallelization of tasks.
While the framework offers certain benefits, it suffers from multiple issues. Firstly, as a high degree of specificness is required, the resulting code can be fairly difficult to debug. In addition, computation requires the movement of data to a computational node, creating a severe bottleneck in terms of the communication overhead.

2.1.3 MapReduce

MapReduce is a programming model for distributed computation proposed for use by Google in [9]. The model specifies a programming paradigm that simplifies all operations for a map phase, wherein data on which operations are to be performed are mapped to a computational node, and a reduce phase, in which operations on the mapped data are performed on the node in which they exist. These map and reduce operations operate in much the same way as the map and reduce operations in many functional programming languages do. The popular

Hadoop distributed framework utilizes the MapReduce framework for writing programs. The paradigm benefits in several ways. Firstly, MapReduce makes use of data locality when performing computational tasks. Unlike MPI, which requires data to be moved to the node on which it will be operated, operations under the MapReduce scheme will occur on the computational node on which the data exists. Secondly, and also as a partial extension of the first benefit described, the MapReduce paradigm is extremely simple from a user perspective when compared to MPI. While in MPI a high degree of granularity is required when specifying the flow of tasks, MapReduce programs can be written as a specification of a single map and a single reduce phase. Finally, MapReduce benefits from being the programming paradigm of choice for many of the most common distributed frameworks, including Hadoop and Spark.

2.1.4 Distributed Filesystems

A Distributed Filesystem is a method of storing and accessing files based in a client/server architecture. In a distributed file system, one or more central servers store files that can be accessed, with proper authorization rights, by any number of remote clients in the network. Much like an operating system organizes files in a hierarchical file management system, a distributed system uses a uniform naming convention and a mapping scheme to keep track of where files are located. When the client device retrieves a file from the server, the file appears as a normal file on the client machine, and the user is able to work with the file in the same ways as if it were stored locally on the workstation. When the user finishes working with the file, it is returned over the network to the server, which stores the now-altered file for retrieval at a later time.

Distributed file systems can be advantageous because they make it easier to distribute documents to multiple clients and they provide a centralized storage system so that client machines are not using their resources to store files.

One of the most pervasive distributed filesystems is the Hadoop Distributed Filesystem (HDFS). HDFS is implemented in the Java programming language for Hadoop, but has found applications in other cluster frameworks. Files are distributed and maintained across

many “data” nodes, with a level of redundancy being maintained. Access to the files is controlled via a “head” node, which tracks the location of data within the cluster’s data nodes. Redundant head nodes are typically maintained as well.

2.1.5 Spark

Apache Spark is an open-source in-memory data analytics cluster computing framework. It is often seen as a successor to Hadoop, and has been shown to outperform Hadoop computation time for certain tasks by as much as 100 times. The major advantage Spark presents over traditional Hadoop disk-based MapReduce is its in memory computation paradigm. In Hadoop, MapReduce computation is performed by loading data from disk, performing computations on the data, and writing it back out to disk. With Spark, data is held in a cluster’s shared memory, allowing for repeated queries to the same data without waiting for slow disk reads. This has been shown to be very advantageous for highly iterative processes such as training a logistic regression model, where the same data will be referenced repeatedly. In [34], the time to convergence of training a logistic regression model on four node Spark and Hadoop models were compared using a 29 gigabyte dataset. Training using a large number of iterations was shown to complete as much as 100 times faster using a Spark cluster as opposed to a Hadoop cluster, as seen in Figure 2.2.

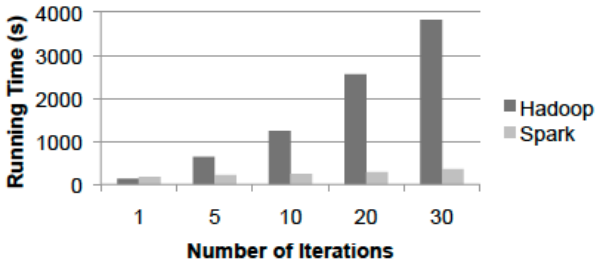

Figure 2.2: Runtime of Hadoop and Spark on logistic regression model training

2.2 Approximation Methods

The goal of all tasks described in this work is to determine a distribution of unobserved variables using various methods. In this section, we explain two of the most commonly used methods of approximation used in topic modeling.

2.2.1 Variational Inference

Variational inference is an increasingly popular method of approximating posterior probability distributions. The approach seeks to eliminate some of the "guess work" involved when attempting to approximate multiple distributions which co-vary to an unknown degree. Variational inference does this by assuming the relationships among distributions are known a priori. In some applications, such as topic modeling, the distributions are assumed to have little to no co-variance. By fixing the co-variance in this way, the distributions can then be sampled independently to arrive at an approximate posterior distribution [5].
While the potential independence assumptions made by variational inference lend themselves to easy computation using MapReduce, the volume of commonly used topic modeling toolkits [17] [18] which continue to use alternative approximation methods seems to indicate a reluctance to adopt variational inference. For this reason, in this work we choose to ignore this approximation method.

2.2.2 Gibbs Sampling

By far the most commonly used approach to posterior approximation in topic modeling is Gibbs sampling. Gibbs sampling or is a Markov Chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution. By continuously re-sampling from either a random or pre-specified distribution, the exact posterior can be approximated. Given an infinite number of sampling iterations, the approach will converge on the true posterior [7]. Conversely given a low number of sampling iterations, the posterior approximation will be poor.
Given its formulation as a MCMC method, it well suited to approximations within Bayesian Networks, such as those described in Section 2.3.2. As such, it is commonly used in many

topic modeling toolkits in order to perform approximation. Because of its pervasive adoption, we focus on Gibbs sampling for approximation for the remainder of this work.

2.3 Topic Models

One of the tasks of machine learning is to uncover abstract representations of documents. This task is relevant both for presenting summarizations of documents to be ingested by a human reader as well as for extracting features to be used in other machine learning tasks such as classification. Topic models are one such way of performing this extraction.

A topic model is a statistical model of the mixture of abstract "topics" that a document is comprised of. For example, given known topics for "banking" and "trade," one could represent the blog posts of a various amateur economists as some mixture of these two topics. For posts discovered to be mostly comprised of the "banking" topic, one would expect to see a large number of "banking" related words, and similarly for posts related to the "trade" topic. Posts with a roughly even proportion of the "banking" and "trade" topics may expect to see a similar distribution of both "banking" and "trade" related words. Probabilistic topic modeling captures this concept of topic proportions being affected by the presence of topical words in a mathematical framework.

It should be understood, however, that topic models are not limited to modeling textual documents. In fact, they have been applied in a variety of contexts including image processing [32] and finance [10]. However, in order to make this work more easily understandable, future references in this work to "document features" or "terms" will refer to a term vector document representation and the individual words contained in a document respectively.

2.3.1 Probabilistic Latent Semantic Indexing

A precursor to LDA, Probabilistic Latent Semantic Indexing (pLSI) attempts to model the causal relationships between document content and the latent variables which it is comprised of [12]. In Figure 2.3, the words w which comprise a document d are shown to be generated by a latent variable c, which can be thought of as a "topic." These causal relationships are then said to apply to a corpus of M documents, each having N words. The probability of

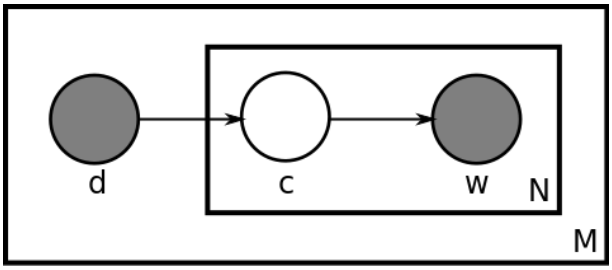

Figure 2.3: Plate notation for pLSI

co-occurence of a document and a word is modeled according to Equation 2.1.

$$P(w,d) = \sum_c P(c)P(d|c)P(w|c) = P(d)\sum_c P(c|d)P(w|c) \tag{2.1}$$

The model states, as shown in Equation 2.1, that the probability of word w and document d co-occurring is determined by the probability of occurrence of document d multiplied by the sum of the product of the probability of topic c occurring in document d and the probability of word w having topic c for all potential topics c.

The model has drawn a degree of ire as a result of its not being a generative model. It has also been shown to lead to overfitting during probability inference. Despite these issues, it is notable for being the basis for Latent Dirichlet Allocation.

2.3.2 Latent Dirichlet Allocation

One of the most common forms of topic models is Latent Dirichlet Allocation (LDA) [5]. LDA is a generative model that represents documents as a mixture of latent variables. These latent variables are the "topics" of the topic models, and are probability distributions over the possible document features that may be present in a document. The model assumes a Dirichlet prior over both the distribution of topics within a document and the distribution of features within a topic. A concise formulation of the model was presented in [5] as seen in Figure 2.4. In this model α and β are the paramaters for the Dirichlet priors for the per document topics and the per topic word distributions, respectively. θ is the topic distribution for a given document. Z is the topic which generates feature W. φ is the distribution of words for a given topic. M is the document count of the corpus being examined and N

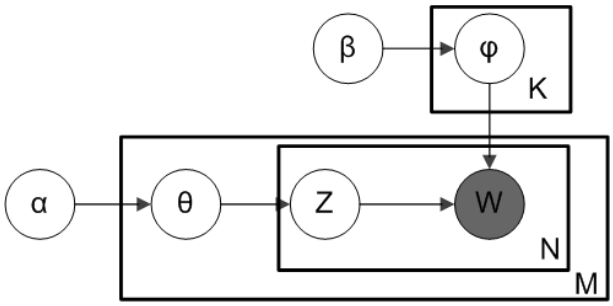

Figure 2.4: Plate diagram of LDA

is the number of unique terms within the corpus. By sampling from topics and features within topics, it is possible to generate all documents.

By envisioning documents as the product of the set of causal relationships described above, it is possible to uncover the probability of each word for each topic.

In order to estimate the probabilities associated with the aforementioned model, we use Gibbs Sampling. Other methods involving variational inference have been proposed, although those using Gibbs Sampling are by far the most common. The probability of a word w being generated given a topic z is estimated according to Equation 2.2,

$$P(w|z) = \hat{\varphi}_z^{(w)} = \frac{{n_z}^{(w)} + \beta}{{n_z}^{(\cdot)} + N\beta} \tag{2.2}$$

and the probability of a topic z being generated given a topic model τ is estimated according to Equation 2.3.

$$P(z|\tau) = \hat{\theta}_z^{(d)} = \frac{{n_z}^{(d)} + \alpha}{{n_\cdot}^{(d)} + K\alpha} \tag{2.3}$$

Parameters α and β intuitively specify how close Equations 2.2 and 2.3 are to a maximum likelihood estimation: if their value is zero, Equations 2.2 and 2.3 become a maximum likelihood estimation, while high values make them tend to a uniform distribution. Typical values for these parameters are $\alpha = 50/K$ and $\beta = 0.1$; values that are commonly used in many available topic modeling toolkits which implement LDA. The number of topics K depends on the data and therefore differs for each problem. A typical and straightforward solution is simply trying different values to see which one offers the best results for the

desired task. Other values used in the above equations are described in Table 2.1.

As mentioned, Gibbs Sampling is used to arrive at the above estimations. For LDA, the

Table 2.1: Values used in LDA with Gibbs sampling to find underlying topics

value	*description*
${n_z}^{(w)}$	Number of times term w is assumed to have been generated by topic z.
${n_z}^{(d)}$	Number of times a term instance of document d is assumed to have been generated by topic z.
${n_z}^{(\cdot)}$	Total number of times a term has supposedly been generated by topic z.
$n_{\cdot}^{(d)}$	Total number of term instances of document d generated by any topic.
${n'_z}^{(w)}$	Number of times term w is assumed to have been generated by topic z, but without counting the current assignment of w.
${n'_z}^{(d)}$	Number of times a term instance of document d is assumed to have been generated by topic z, but without counting the current assignment of w.
${n'_z}^{(\cdot)}$	Total number of times a term has supposedly been generated by topic z, but without counting the current assignment of w.
$n'^{(d)}_{\cdot}$	Total number of term instances of document d generated by any topic, but without counting the current assignment of w.

Gibbs Sampling algorithm requires the typical random initialization of all values described in Table 2.1. This is done by sampling topics from a uniform distribution for every term in every document. The term probabilities are then adjusted through an iterative process. At each step, a term is sampled and removed from the model. A topic is then sampled according to Equation 2.4.

$$P(z|w,\tau) \propto P(w|z) \times P(z|\tau) = \frac{{n'_z}^{(w)} + \beta}{{n'_z}^{(\cdot)} + N\beta} \cdot \frac{{n'_z}^{(d)} + \alpha}{n'^{(d)}_{\cdot} + K\alpha} \tag{2.4}$$

This topic is then associated with having generated the term that was previously removed, and the term is added back to the model. This process is repeated for all terms in all documents. Iterations typically continue until either the model satisfies some metric indicating convergence, or a predefined number of iterations have occured. Commonly perplexity, as described in Section 5.1, is used as a convergence metric [25] [5].

2.3.3 Extensions of Latent Dirichlet Allocation

Latent Dirichlet Allocation has been at the center of many extensions. The Hierarchical Dirichlet Process (HDP) model described in [30] adds an additional Dirichlet prior to the original model, H in Figure 2.5. The purpose of this additional causal relationship is to model corpora as a mixture of many topic models.

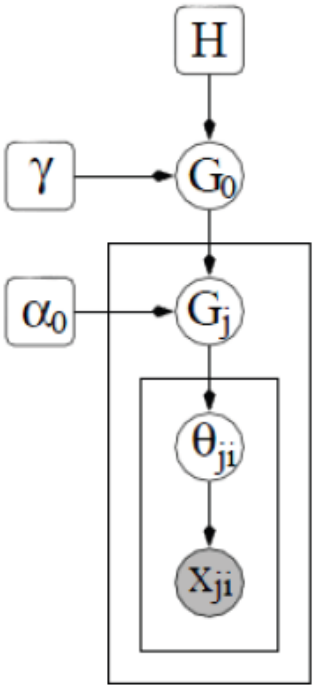

Figure 2.5: Hierarchical Dirichlet Process plate model

The HDP model has been the focus of works studying topic clustering as in [4], [21],

and [14]. Term-topic probabilities within this model can be inferred using an adaptation of the model described in Section 2.3.2.

Various other modifications to the original LDA model have yielded models optimized for modeling causal relationships between topics and various predefined class labels. In [15], a model is proposed that defines a causal relationship between sentiment expressed in a document and topic proportion. The model shown in Figure 2.6 presents the sentiment expressed

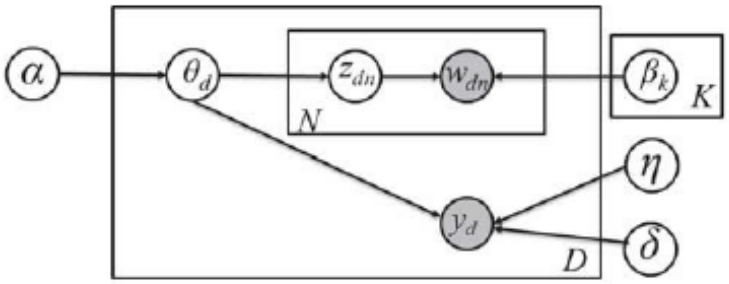

Figure 2.6: sLDA Plate model

as y, and attempts to define a causal relationship between it and the topic distribution of a document D.

2.4 Existing Distributed Algorithms for Topic Models

There exist several distributed implementations of the topic distribution inference algorithm described in Section 2.3.2. Many of these implementations attempt to perfectly replicate the sampling algorithm. In this section, we describe two implementations most relevant to this work.

2.4.1 Asynchronous - LDA (Async-LDA)

Async-LDA is an implementation of the Gibbs Sampling for LDA approach [28]. In this implementation, computational nodes within a MPI cluster perform the Gibbs Sampling algorithm over a subset of documents from a corpus. After a fixed number of sampling runs, the resulting topic distributions are merged, using an approach equivalent to what

is described in Section 3.2. In the work in which this method is proposed, Async-LDA is shown to achieve a near perfect speedup as computational nodes are added to the cluster for larger datasets (e.g. PUBMED with 737,869,083 terms in the corpus) while achieving less optimal speedups for smaller datasets (e.g. NYT with 99,542,125 terms in the corpus) as shown in Figure 2.7.

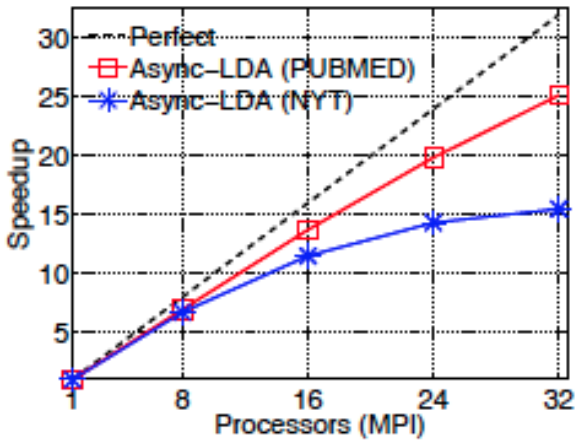

Figure 2.7: Speedup for Async-LDA on the PUBMED and NYT datasets

2.4.2 Approximate Distributed - LDA (AD-LDA)

In a parallel work to [28], AD-LDA was proposed to accomplish the same task of distributing the LDA Gibbs Sampling algorithm with the exception of using MapReduce as the distribution paradigm [22]. The procedure used is essentially equivalent to the one described in Section 2.4.1, with the map phase responsible for determining break up of documents and the reduce phase responsible for the aggregation of the resulting term-topic counts.

A particularly important contribution of the original AD-LDA work is the battery of scalability experiments conducted. The authors implement an equivalent version of AD-LDA using MPI, and note its performance with respect to the MapReduce implementation as the size of the cluster increases. The authors conclude that as the cluster size become large, the ability of the MPI implementation to scale decreases drastically. This serves as a motivation for adopting MapReduce as the programming paradigm for extremely large clusters

for performing topic model training.

Chapter 3

PROPOSED DISTRIBUTED ALGORITHM

3.1 Justification

It has been noted in the works discussed in Section 2.4 that an insufficient number of Gibbs Sampling iterations in distributed implementations of LDA will lead to topics which do not converge, leading to a suboptimal final solution. This is analogous to the problem of arriving at a poor approximation of the posterior distribution using Gibbs sampling for all applications mentions in Section 2.2.2. In this work, we propose to leverage diverging topics as an advantage to the overall topic model, rather than a hindrance.
Additionally, we propose to deploy this algorithm using the Spark cluster computing framework. We note that Gibbs sampling is a highly iterative task, and as discussed in Section 2.1.5, Spark performs well on highly iterative tasks.

3.2 Implementation

We propose two algorithms as an alternative to sequential methods for approximating topic distributions in LDA. In both, using MapReduce, subsets of documents from a corpus are mapped to different computational nodes. A topic model is trained for each of these subsets using Gibbs sampling. The resulting topic models are then returned from the reducer in the form of the counts described in Section 2.3.2, using the process described in the same section, acording to Listing 3.1. The only variation from the original process is that random initialization only occurs on the first iteration; successive iterations use counts forwarded from the post-reduce phase.

Listing 3.1: MapReduce code for topic model training

```
Function Map(document) {
```

```
    emit(hash(document), document);
}

Function Reduce(id, documents) {
    TopicModel = LDA(documents);
    emit(TopicModel);
}
```

After each iteration of algorithm, topic counts are merged in order to generate a single topic model. The resulting counts are then broadcast to all computational nodes and used for initialization for the next round of Gibbs sampling. This process is repeated until the perplexity of the models present at successive iterations improves by less than 0.1%. The entire procedure is presented in Figure 3.1.

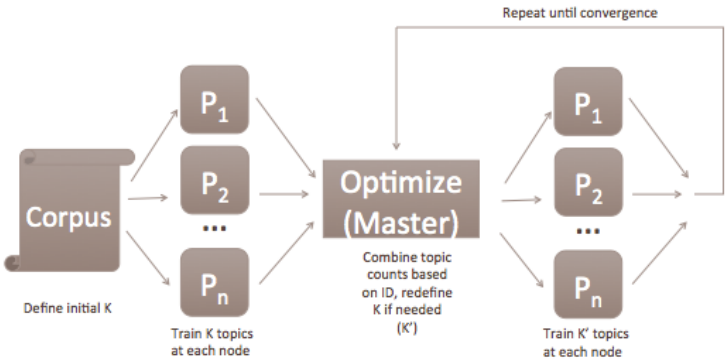

Figure 3.1: Flow of the proposed algorithms

In this workflow, an initial value K specifies the initial number of topics to assume for the model. Model training then occurs on a subset of documents within a corpus at n computational nodes (P). The topic counts are then merged in the Optimize phase, by summing the counts for matching topics from all nodes. The resulting counts are broadcast in order to use as the intial counts for the next round of Gibbs sampling. Training and Optimize then repeat until the perplexity convergence threshold is met.

The two algorithms proposed in the Sections 3.2.1 and 3.2.2 differ in how they determine

which topic models are to be merged at the Optimize step.

3.2.1 Optimized Distributed - LDA (OD-LDA)

This algorithm attempts to perfectly reproduce the sequentially trained models from LDA in a distributed system. This approach a very similar one to that taken in [22], with one notable exception. In [22], the authors note that it is unnecessary to attempt to find a perfect matching of topics from each computational node. Rather, the authors note that if topics with the same topic ID from each node are merged across all nodes the model will tend to converge upon additional iterations.

In our approach, we search the space of all possible topics from all computational nodes to find a matching that lowers the divergence between topics, as determined by their Jensen-Shannon Divergence, defined in Equation 3.1.

$$JSD_{\pi_1,...,\pi_n}(P_1, P_2, ..., P_n) = H(\sum_{i=1}^{n} \pi_i P_i) - \sum_{i=1}^{n} \pi_i H(P_i) \tag{3.1}$$

In this equation, π_i refers to the weight assigned to probability distribution P_i. For our purposes, each distribution is weighted equally. $H(P_i)$ refers to the Shannon entropy of probability distribution P_i, as defined in Equation 3.2.

$$H(P) = \sum_{j=1}^{n} p_j \log(\frac{1}{p_j}) \tag{3.2}$$

In this equation, P is a discrete probability distribution, and p_i is the jth term of the distribution. The equation for $H(P)$ will in effect capture the amount of information received for every event in P

We choose this divergence metric both for it being a symmetrical measurement as well as its ability to measure divergence between more than two distributions (as opposed to other metrics such as KullbackLeibler divergence).

In order to arrive at the combinations of topics to be merged, we examine all possible groupings (equivalent to the number of topics) of topic models such that no two models involved can originate from the same computational node. For example, assume an example where we seek to create a topic model of K=3 topics on a cluster with three computational

nodes named A, B and C. In this example the topic model with an ID of 1 from node A is denoted as A_1. The possible groupings include [{A_1,B_1,C_1},{A_2,B_2,C_2},{A_3,B_3,C_3}], [{A_1,B_2,C_3},{A_2,B_3,C_1},{A_3,B_1,C_2}], and so on. Each of these possible groupings will have its JSD, and the grouping whose mean divergence is lowest will be merged. This differs from the approach taken in [22] in that the authors of that work only consider merging topics with the same topic ID from each node.

3.2.2 Distributed Diverging - LDA (DD-LDA)

This implementation makes use of an observation made on the work in [22]. In it, the authors state that if the number of Gibbs sampling iterations performed during the inter-node topic model training is low enough, topics of the same topic ID will tend to be less similar. We interpret this observation to mean that the resulting distributions may in fact belong to completely disparate topics, and should be treated as such by not automatically merging them.

In this proposed model, upon receiving the counts from the MapReduce phase, we examine topics of the same ID from each computational node. Using Jensen-Shannon Divergence, we determine if the term-topic distribution they produce is below a pre-selected threshold. If it is, we determine that the topics should be merged by summing counts. If they are not, we examine all possible groupings of these same topics of same ID and same computational node to determine the configuration such that the resulting groups will each have a divergence less than the selected threshold and the mean divergence of the two groups is minimized. These groupings of models will be merged and the topic count will be increased by one. If no such grouping exists, we determine the topics to be unique, and perform no merging. In this case, we increase the topic count by the number of computational nodes minus one. This workflow is shown in Figure 3.2 and Figure 3.3, which is the internal operation of DD-LDA for the Optimize phase of Figure 3.1.

In Figure 3.2, the JSD of all topics with the same topic ID are checked to see if they satisfy the divergence criteria. If they do not, an additional check is performed in Figure 3.3 to find

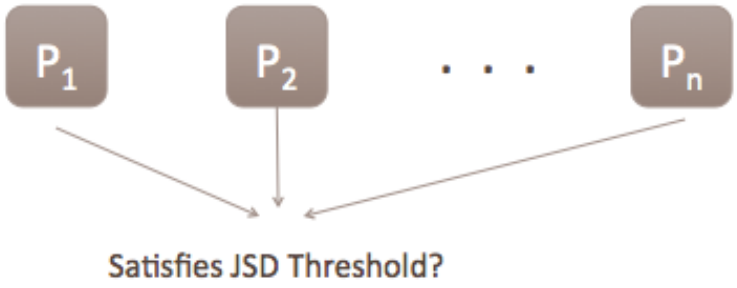

Figure 3.2: Initial check of topic models from reduce phase in Optimize

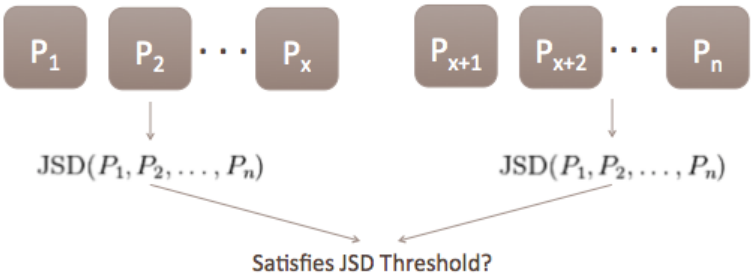

Figure 3.3: Check of split of topic models from reduce phase in Optimize

a splitting of the topic distributions for one where the mean JSD of the two arrangements would be below the threshold.

A unique facet of this implementation is that the pre-specified topic number K serves only as a lower limit to the number of topics that the model will eventually be comprised of.

Chapter 4

DATA SETS

As we aim to test the scalability (in addition to other evaluation metrics) of the algorithms proposed in Section 3, we require all experiments to run using reasonably large data sets. The choice to use large data sets is additionally necessary as a large amount of the original data is removed during the various pre-processing steps detailed in Chapter 4.4.
To satisfy this size requirement, we evaluate our experiments on the myPersonality and Stack Overflow data sets. We additionally use the Wikipedia abstracts data set for evaluation metrics requiring an external corpus.

4.1 myPersonality Project

The myPersonality data set is available through the myPersonality Project [13]. myPersonality is a popular Facebook application that allows users to take real psychometric tests, and is used to record their psychological and Facebook profile. Data collection began in June 2007. The data set contains more than 6,000,000 survey results together with more than 4,000,000 individual Facebook profiles.
Aside from its sheer size, the myPersonality data set is attractive for this research due to its being representative of a diverse population, ensuring that quality metrics for these experiments are not unrealistic due to homogeneous data. While gender is not necessarily balanced, both males and females are well represented, as shown in Table 4.1.

Gender	Proportion
Male	0.632
Female	0.368

Table 4.1: Distribution of gender in the myPersonality data set

In the data set, the vast number of entries contain only one status update per user. Higher posting counts decrease in frequency, with the overall distribution of post counts having a long right tail as shown in Figure 4.1.

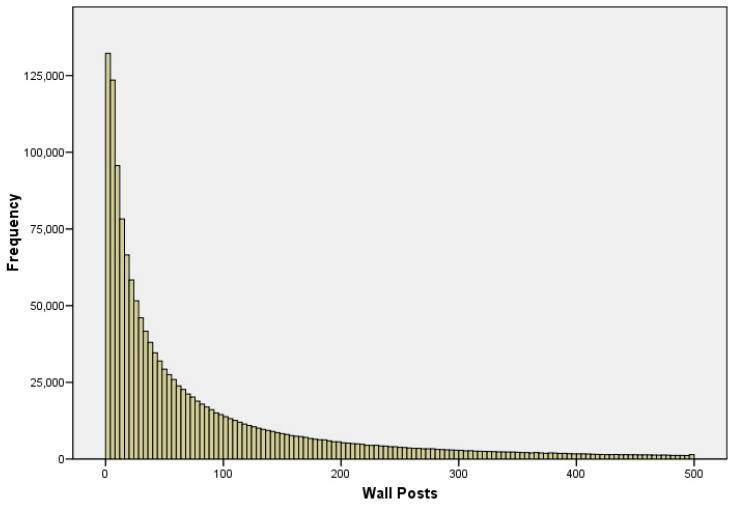

Figure 4.1: Distribution of wall posts per user within the myPersonality dataset

In terms of age, the data set is well representative of online social network users, as shown in Figure 4.2.

Reported relationship statuses for users in the data set are varied as well, as shown in Figure 4.3.

As this research is focused on the application of probabilisitc models to textual data, we use only those users within the data set that have consented to have their Facebook status updates made available. At the the time the data set was acquired for the research presented in this work, the status updates of 442,815 users had been made available. For these users, there were 25,407,612 total status updates, for an average of 57.4 status updates per user. Statistics for this section of the total data set are presented in Table 4.2.

4.2 Stack Overflow

Stack Overflow is the flagship site of the Stack Exchange Network, a network of question and answer websites. Within the network, each site focuses on addressing questions for a

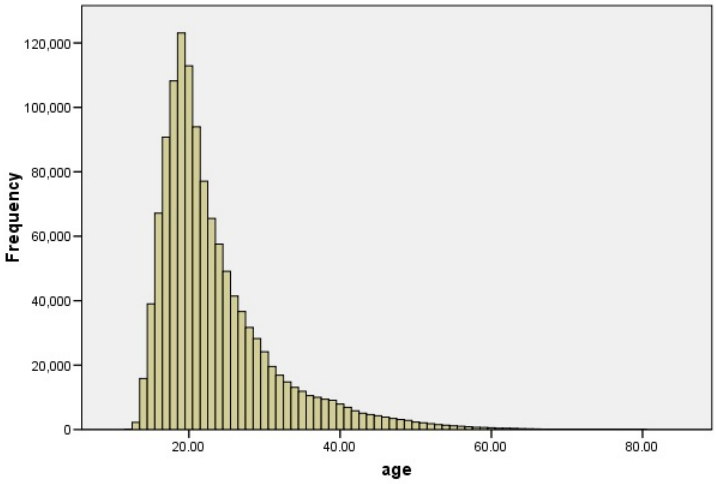

Figure 4.2: Distribution of user age within the myPersonality dataset

Mean	50.6
Median	1
Minimum	1
Maximum	54,083
Standard Deviation	143.8

Table 4.2: Per user status update count statistics in myPersonality data set

particular area of interest, e.g. statistics, server management, and Unix. The Stack Overflow site focuses on questions involving computer programming. At the time of this writing, Stack Overflow maintains a population of 2,700,000 registered users having asked more than 7,100,000 questions.

The data available on Stack Overflow is interesting for this research as it can be seen as being more homogeneous than the data present in the myPersonality data set. While the specific technologies mentioned are varied, the Stack Overflow community ensures that all posts are focused on some aspect of computer programming. This aspect of the data is important for this research as probabilistic approaches modeling documents may perform

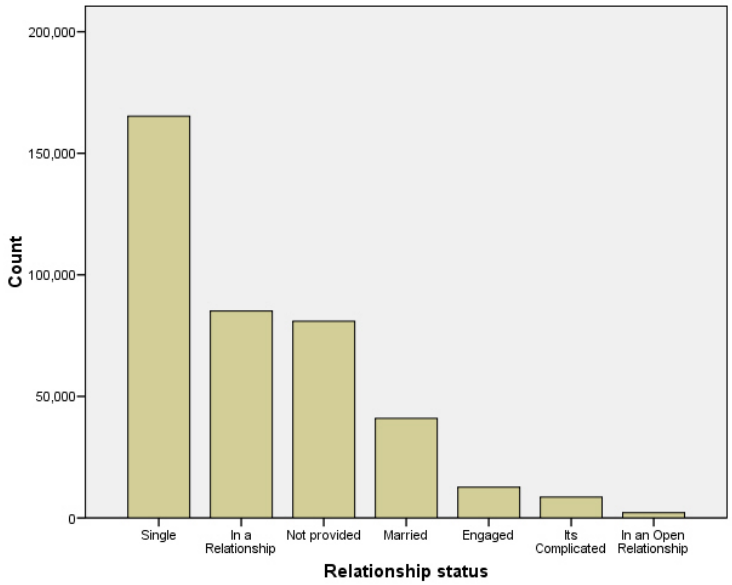

Figure 4.3: Reported relationship status frequency for users in the myPersonality data set

with differing degrees of quality on heterogenous and homogenous data.

Stack Overflow profiles are made publicly available through periodic data dumps that are accessible without pre-authorization through the organization's data website. For this work we use the data dump of the Stack Overflow user posts accessed on August 28, 2014. The word count per post distribution is again very skewed, as shown is Table 4.3.

Mean	89.5
Median	57.0
Minimum	3
Maximum	65,849
Standard Deviation	108.6

Table 4.3: Post word count statistics in Stack Overflow data set

4.3 Wikipedia

Some evaluation metrics require the use of a large external corpus. For such metrics, we choose to use Wikipedia abstracts data set. Wikipedia provides access to downloads of extended abstracts for articles within their collection.

The use of this data set as an external corpus for validation is important due to its extreme subject variety. Metrics that rely on calculating the probability of a word occurence in an external data set may unrealistically favor models trained on texts with similar subject matter.

This data dump is composed of 4,636,227 documents with 414,954,300 total words. The distribution of word counts is extremely skewed, as shown in Table 4.4.

Mean	89.5
Median	57.0
Minimum	3
Maximum	65,849
Standard Deviation	108.6

Table 4.4: Abstract word count statistics in Wikipedia data set

4.4 Pre-processing Steps

Prior to all experiments, data sets undergo several pre-processing steps. These pre-processing steps are designed to address the issues mentioned in [29], which address several issues surrounding nonsensical topics in probabilistic topic models. It has been noted that often many topic models will contain topics which assign high probability to words which do not convey much meaning to a human reader (e.g. “a”, “at”, “the”, “like”). We attempt to minimize the occurrence of such topics in our experiments by filtering out words with low information content. To do this, we apply a $tf - idf$ filter to the corpus [16]. Here, tf is simply the raw

frequency of a word within a corpus, and idf is defined in Equation 4.1.

$$idf(t, D) = \log \frac{N}{|\{d \in D\} : \{t \in d\}} \tag{4.1}$$

In this evaluation t is the term to evaluate and N is the number of documents in the corpus D in which the term is in. From this formulation, we define $tf - idf$ as simply the product of a term's frequency and its idf.

Once this value has been calculated, we filter out the lowest decile of terms based on their $tf - idf$ content.

In addition to this filter, we additionally apply a simple stop word check to all terms in the corpus. These words have been determined by experts using some external metric of utility in natural language processing tasks. For this work, we use the stop word lists provided as part of the Stanford NLP Toolkit [17].

As a final pre-processing step, we filter out all formatting and markup artifacts within each corpus. While this may not necessarily be a required task in many topic modeling applications, the presence of formatting and markup may have detrimental effects on the conceived interpretability of a topic model. As one of the evaluation metrics we describe attempts to address the understandability of topic models, we exclude these artifacts from the data.

Chapter 5

EVALUATION METRICS

5.1 Perplexity

The most common way to evaluate a probabilistic model as seen in a wide review of existing literature is to measure the log-likelihood of a held-out test set. Perplexity for topic models is a measurement of how well a model can generate a set of unseen documents, with lower perplexity corresponding to greater generative power. In this evaluation, the test set is a collection of documents w of size d not used in the inference of the topic distributions. The model is represented as the matrix of topic-term probabilities φ and the hyperparameter β, representative of the distribution of topics over documents. The LDA topic distribution parameter θ is not taken into consideration as it represents the topic-distributions for the documents of the training set, and can therefore be ignored to compute the likelihood of unseen documents. These measurements are evaluated using Equations 5.1 and 5.2.

$$\mathcal{L}(w) = \log p(w|\varphi, \beta) = \sum_{d} \log p(w_d|\phi, \beta) \tag{5.1}$$

$$\text{perplexity}(test\ set\ w) = e^{(-\frac{\mathcal{L}(w)}{count\ of\ tokens})} \tag{5.2}$$

5.2 Coherence

While perplexity gives a good evaluation of the representative power of a topic model, it has been shown to be negatively correlated with human understanding of topics in certain cases [8]. For this reason, topic coherence is often used as an additional evaluation metric for topic models. The intuition behind this metric, which uses pointwise mutual information (PMI) as described in Equation 5.3 using external data, comes from the observation that occasionally a topic has some odd-words-out within the top ten most probable words for that topic. This leads to the idea of a scoring model based on word association between

pairs of words, for all word pairs in a topic. But instead of using the collection itself to measure word association (which could reinforce noise or unusual word statistics), a large external text data source is used to provide regularization.

For this work, we measure co-occurrence of word pairs from a single huge external text dataset: all articles from English Wikipedia. For the dataset, we counted a co-occurrence as words w_i and w_j co-occurring in any 10-word window in any article. These co-occurrences are counted over a corpus of over a billion words, so they will produce reasonably reliable statistics.

$$PMI(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \tag{5.3}$$

In this equation, we calculate the log of the division of the probability of terms w_i and w_j occurring in a document by the product of the probability of occurrence for both w_i and w_j.

Chapter 6

EXPERIMENTAL RESULTS

6.1 Experiment Design

In order to evaluate the utility of the algorithms proposed in Section 3, we compare both DD-LDA and OD-LDA to the implementation of LDA in Mallet [18], a popular topic modeling toolkit, as well as to a version of AD-LDA adapted to Spark from [22]. All experiments are run using virtual machines running on the Microsoft Azure cloud computing platform. The virtual machines used all run the Ubuntu 12.04 operating system, have 14 gigabytes of memory, and two core processors clocked at approximately 2.2 gigahertz.

For DD-LDA/OD-LDA cluster, we utilize a group of up to four of the above mentioned virtual machines. Each virtual machine has an installation of Spark 1.0.0. Spark is deployed in standalone mode (without extra resource management software) using HDFS as the distributed file-system.

All experiments are repeated ten times using both the myPersonality and Stack Overflow datasets.

6.2 Scalability Analysis

In order to determine the speed improvements achieved by DD-LDA, OD-LDA, and AD-LDA over LDA, we run the algorithm on three cluster configurations: two nodes, three nodes, and four nodes. We do not run the algorithms on one node as in this configuration the algorithm is essentially utilizing the same sequential algorithm as LDA.

In both the OD-LDA and DD-LDA implementations, the time required to train a model is less than that required for the Mallet implementation of LDA, as seen in Table 6.1 and Table 6.2. It should be noted, however, that in OD-LDA, the performance gains are very slight considering the number of nodes being added to the computation, with a less than

	Algorithm			
Nodes	LDA	AP-LDA	OD-LDA	DD-LDA
1	216.75	-	-	-
2	-	125.52	183.30	130.76
3	-	92.56	147.04	100.28
4	-	74.26	126.83	82.19

Table 6.1: Runtime performance of LDA and DD-LDA/OD-LDA in minutes on myPersonality dataset

	Algorithm			
Nodes	LDA	AD-LDA	OD-LDA	DD-LDA
1	628.31	-	-	-
2	-	352.69	536.67	375.21
3	-	269.46	429.04	288.88
4	-	214.02	381.59	238.64

Table 6.2: Runtime performance of LDA and DD-LDA/OD-LDA in minutes on Stack Overflow dataset

50% improvement seen for the four node cluster. This is due to the amount of computations required to find an "ideal" matching of topics from all nodes as described in Section 3.2. For the DD-LDA implementation, however, the improvement seen is closer to being linear as computational nodes are introduced. The speedup is not quite linear, as is expected from the operations required to determine a proper "merging" of the topics from each node as described in Section 3.2. Speedups for the Stack Overflow and myPersonality datasets are visualized in Figure 6.1 and Figure 6.2. Speedup in DD-LDA is comparable to that achieved by AD-LDA, although slightly slower.

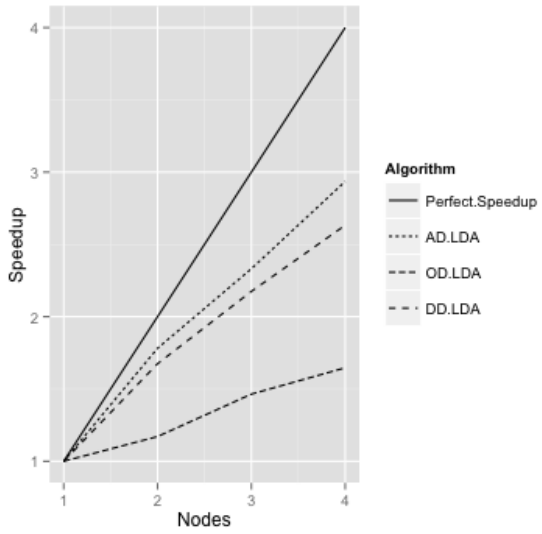

Figure 6.1: Speedups for DD-LDA, OD-LDA, and AD-LDA based on cluster size for Stack Overflow Data

6.3 Model Quality Analysis

While improved computation time is the primary goal of this work, in order to be viable our algorithm must also create topic models that are at least as good of quality as those created using LDA in a non-distributed fashion. To evaluate topic model quality, we examine both the perplexity and coherence of the topic models created by DD-LDA, OD-LDA, AD-LDA and Mallet LDA. All metrics for the distributed algorithms are collected using a four node cluster. We use the Wikipedia abstracts dataset in order to evaluate both perplexity and coherence metrics using external data.

As shown in Figure 6.3 and Figure 6.4, perplexity results for DD-LDA are demonstrably improved over those seen using all other methods used, and are statistically significant with $p < 0.05$.

Coherence results are, unlike those for perplexity, worse at first look for DD-LDA compared to other approaches. However, using two one sided tests, the coherence of the DD-LDA models are both equivalent to those using Mallet LDA with $p < 0.05$.

A notable difference between the approaches is the number of topics contained in the final

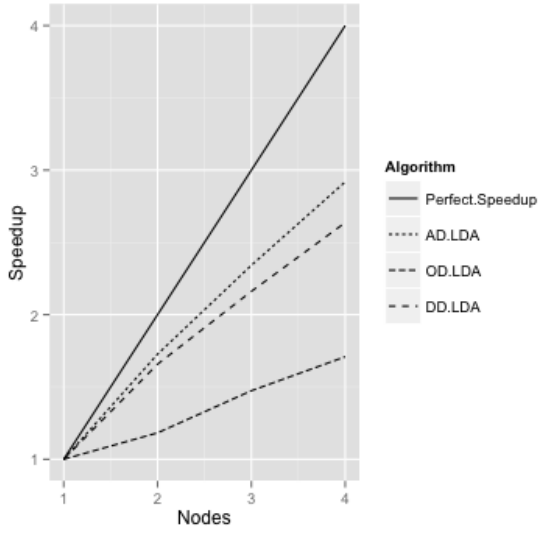

Figure 6.2: Speedups for DD-LD, OD-LDA, and AD-LDA based on cluster size for myPersonality Data

models. For all approaches, the topic count stays constant at 100, with the exception of DD-LDA. This is due to the ability given to DD-LDA to increase the resulting model's topic count.

Algorithm	Perplexity	Coherence	Topic Count
Mallet LDA	1329.62	2.19	100
AD-LDA	1335.30	2.14	100
OD-LDA	1349.73	2.21	100
DD-LDA	1284.26	1.97	457

Table 6.3: Perplexity and Coherence of LDA and DD-LDA models on myPersonality dataset

Algorithm	Perplexity	Coherence	Topic Count
Mallet LDA	1517.14	3.78	100
AD-LDA	1537.42	3.69	100
OD-LDA	1531.73	3.72	100
DD-LDA	1484.12	3.49	457

Table 6.4: Perplexity and Coherence of LDA and DD-LDA models on Stack Overflow dataset

Chapter 7

CONCLUSION AND FUTURE WORK

In this work we have reviewed the various applications of topic models with a specific focus placed on the generative LDA model. The LDA model allows for documents to be represented as a mixture of latent topics; a useful trait when performing tasks such as document summarization and information extraction. However, as noted in this work, the Gibbs sampling algorithm typically associated with topic distribution inference in LDA is extremely computationally demanding. In order to make the application of topic models feasible in the era of big data, we have noted several proposed methods for distributing topic inference for LDA models.

Leading up to the proposal of a new algorithm for topic model training, we reviewed existing frameworks and paradigms used for big data processing such as MPI, Hadoop, and Spark. We explained how the iterative nature of the Gibbs sampling algorithm for LDA makes it a perfect candidate for the Spark cluster computing framework. We justified this by noting in Section 2.1.5 that Spark has been shown to perform well on such iterative tasks.

We then went on to propose two novel algorithms for topic model inference: OD-LDA and DD-LDA. The two extend ideas presented in existing works on distributed topic model training, with extensions to the manner in which results from different computational nodes were resolved.

We compared the time, perplexity, and coherence performance of the two proposed algorithms to existing inference algorithms; one from the popular topic modeling toolkit Mallet and one from an existing work on distributed topic model inference. Models for experiments were completed on two large datasets, with a third external dataset used for metric evaluation.

Experimental results showed that OD-LDA did not yield satisfying speedups over sequentially performed LDA Gibbs sampling. However, we found that by modifying the topic

merging optimization problem, significant performance could be gained. For this variation, which we presented as DD-LDA, we have seen speedups in computation time that are much improved over sequential LDA, and comparable to the existing AD-LDA algorithm.

Additionally, we have shown that by allowing topics to diverge as in DD-LDA, improvements in model perplexity can be seen compared to all reference algorithms.

Finally, we have shown that, while topic coherence for our DD-LDA algorithm were slightly lower than those seen in other algorithms, they are statistically equivalent and thus not concerning.

Interesting directions for future work are additional scaling experiments using larger computational clusters for DD-LDA and the application of DD-LDA to various case studies.

In order to better understand the limits of DD-LDA, future research may involve applying the algorithm to at least one truly massive dataset (e.g. the Wikipedia dump of all abstracts in all languages) on a very large Spark cluster.

As this was a novel approach to inferring topic models, various case studies must be pursued in order to determine the usability of models trained using DD-LDA. Case studies may include tested applications of LDA based topic models such as document similarity in academic journals, as well as high stakes applications such as modeling healthcare data sets as a means of data exploration.

BIBLIOGRAPHY

[1] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[2] Donald J Becker, Thomas Sterling, Daniel Savarese, John E Dorband, Udaya A Ranawak, and Charles V Packer. Beowulf: A parallel workstation for scientific computation. In *Proceedings, International Conference on Parallel Processing*, volume 95, 1995.

[3] David M Blei and John D Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

[4] David M Blei and John D Lafferty. Topic models. *Text mining: classification, clustering, and applications*, 10:71, 2009.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[6] Sharon Block. Doing more with digitization. *Common-place*, 6(2), 2006.

[7] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[8] Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems*, 2009.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters osdo 2004, 2004.

[10] Gabriel Doyle and Charles Elkan. Financial topic models. In *NIPS 2009 Workshop on Applications of Topic Models: Text and Beyond*, 2009.

[11] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[12] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[13] Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.

[14] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM, 2006.

[15] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM, 2009.

[16] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[17] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[18] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[19] John M McQuillan, Ira Richer, and Eric Rosen. The new routing algorithm for the arpanet. *Communications, IEEE Transactions on*, 28(5):711–719, 1980.

[20] David Mimno. Computational historiography: Data mining in a century of classics journals. *Journal on Computing and Cultural Heritage (JOCCH)*, 5(1):3, 2012.

[21] David Mimno, Wei Li, and Andrew McCallum. Mixtures of hierarchical topics with pachinko allocation. In *Proceedings of the 24th international conference on Machine learning*, pages 633–640. ACM, 2007.

[22] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.

[23] David Newman, Sarvnaz Karimi, Lawrence Cavedon, Judy Kay, Paul Thomas, and Andrew Trotman. External evaluation of topic models. In *Australasian Document Computing Symposium (ADCS)*, pages 1–8. School of Information Technologies, University of Sydney, 2009.

[24] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics, 2010.

[25] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM, 2008.

[26] Joshua D Potter. Demographic diversity and district-level party systems. *Comparative Political Studies*, page 0010414013516918, 2014.

[27] H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one*, 8(9):e73791, 2013.

[28] Padhraic Smyth, Max Welling, and Arthur U Asuncion. Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems*, pages 81–88, 2009.

[29] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[30] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476), 2006.

[31] Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM, 2009.

[32] Chong Wang, David Blei, and Fei-Fei Li. Simultaneous image classification and annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1903–1910. IEEE, 2009.

[33] Tze-I Yang, Andrew J Torget, and Rada Mihalcea. Topic modeling on historical newspapers. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 96–104. Association for Computational Linguistics, 2011.

[34] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pages 10–10, 2010.

VITA

James Marquardt was born in Houston, Texas on July 25, 1985. After serving in the United States Army from 2004 to 2010, James began attending the University of Washington at Tacoma. He received a Bachelor of Science in Computer Science and Systems in 2013. He is currently a second year graduate student at the University of Washington at Tacoma.