8-2016

Analyzing Clustered Latent Dirichlet Allocation

Christopher Gropp
Clemson University, groppcw@gmail.com

Analyzing Clustered Latent Dirichlet Allocation

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science

by
Christopher Gropp
August 2016

Abstract

Dynamic Topic Models (DTM) are a way to extract time-variant information from a collection of documents. The only available implementation of this is slow, taking days to process a corpus of 533,588 documents. In order to see how topics - both their key words and their proportional size in all documents - change over time, we analyze Clustered Latent Dirichlet Allocation (CLDA) as an alternative to DTM. This algorithm is based on existing parallel components, using Latent Dirichlet Allocation (LDA) to extract topics at local times, and k-means clustering to combine topics from different time periods. This method is two orders of magnitude faster than DTM, and allows for more freedom of experiment design. Results show that most topics generated by this algorithm are similar to those generated by DTM at both the local and global level using the Jaccard index and Sørensen-Dice coefficient, and that this method's perplexity compares favorably to DTM. We also explore tradeoffs in CLDA method parameters.

Acknowledgments

Clemson University is acknowledged for a generous allotment of compute time on Palmetto Cluster.

This research is funded in part by NSF grants 1243436 and 1228312.

I would like to thank my parents and friends for their unwavering support and their confidence in my abilities even when I was skeptical.

I would like to thank Dr. Paul Wilson and Dr. Ilya Safro for their ideas which led to this thesis. Their clever suggestions form the core around which this project was built. I would also like to thank Dr. Alex Herzog for his work on the data cleaning, experimental pipeline, and figures, as well as generally being an excellent person to bounce ideas off of.

Last and most definitely not least, I would like to thank Dr. Amy Apon, whose contributions cannot be understated. From passing along useful papers to in depth style editing, she has been an invaluable resource whose guidance has made this thesis possible. Even more so than for her substantial help, though, I would like to thank her for putting up with me as I put this together.

Table of Contents

Title Page . . . . **i**

Abstract . . . . **ii**

Acknowledgments . . . . **iii**

List of Tables . . . . **vi**

List of Figures . . . . **vii**

1 Introduction **1**
- 1.1 Topic Modeling 1
- 1.2 Research Motivation 2
- 1.3 Approach 3
- 1.4 Solution Design 3
- 1.5 Results 4

2 Review of Literature **5**
- 2.1 Topic Models 5
- 2.2 Latent Dirichlet Allocation 7
- 2.3 Dynamic Topic Modeling 11
- 2.4 Clustering 12
- 2.5 Related Projects 16

3 Problem Motivation **20**
- 3.1 Impact of High-Performance Computing 20
- 3.2 Evaluating Research Output 21
- 3.3 Dynamic Topic Models 23

4 Clustered Latent Dirichlet Allocation **24**
- 4.1 Requirements 24
- 4.2 Design 26

5 Experimental Validation **34**

5.1 Data and Resources 34
5.2 Comparison to DTM 36
5.3 Parameter Optimization 45
5.4 Global and Local Topic Dynamics 45

6 Conclusion 50
6.1 Outcomes 50
6.2 Future Work 51

Appendices 52
A Sample Documents 53
B Additional Similarity Graphs 55

Bibliography 57

List of Tables

5.1 Dataset Details for Computer Science Journal Abstracts and NIPS Conference Proceedings 37
5.2 Runtime Results on Computer Science Abstracts 38
5.3 Perplexity results on computer science abstracts. 41
5.4 Probability of overlap of random topics among top 20 values. 44

List of Figures

4.1 Flowchart of the algorithm 28

5.1 Similarity of Dynamic Topic Means under Sørensen-Dice Coefficient and Jaccard Index. 43
5.2 Heatmap showing perplexity estimated from cross-validation for different combinations of topic counts on the NIPS data. 46
5.3 Heatmap showing perplexity estimated from cross-validation for different combinations of topic counts on the computer science abstracts data. 47
5.4 Evolution of three largest global topics for the NIPS data (left panel) and computer science abstracts data (right panel). 48
5.5 Evolution of global topic "Computer Networks". 48
5.6 Local topics for selected time segments corresponding to global topic "Computer Networks". 49

1 Similarity of Local Topic Means under Sørensen-Dice Coefficient . . . 55
2 Similarity of Local Topic Means under Jaccard Index 56

Chapter 1

Introduction

1.1 Topic Modeling

Topic modeling is the process of automating the task of identifying key themes in a collection of documents. The most common representation for this is based on Latent Dirichlet Allocation, wherein documents are assumed to be randomly generated from one or more topics, each of which is a word distribution. These topics are latent distributions inferred from the documents via a Dirichlet process. The algorithm repeatedly samples the documents and modifies the topics to better fit them until reaching a defined convergence. LDA has a number of strong assumptions, including that both words and documents are unordered. Various methods have been developed to relax these assumptions, generally at a substantial performance cost [8].

1.1.1 Dynamic Topic Modeling

Of particular interest is Dynamic Topic Modeling [7], which relaxes the assumption that all documents are generated simultaneously. It divides the corpus into a set of time segments, which each have their own topics. These topics represent the same theme in each time segment, but the specific language used is allowed to change through time. This lets a user not only see how the language of a topic changes over time, but also how well represented a topic is at any given point in time. However, DTM's update process is considerably slower than LDA's, and requires many more iterations for convergence. At the time of writing this document, the only available implementation for DTM is the one released by the algorithm's creator, David Blei [6], which is not parallelized for model generation. A parallel implementation of DTM was very recently developed by a different research group, but they have not made their code available [5].

1.2 Research Motivation

We are interested in examining how academic research changes over time, and especially how it reacts to new resources. Specifically, we have a corpus of over one million abstracts from journal articles published between 1996 and 2012. Of these abstracts, 396,959 are in chemistry, 533,588 are in computer science, and 101,318 are in economics. We want to see what changes occur in the subjects of research at various institutions when they gain access to new computing resources, and how this compares to the overall direction of research of top universities. DTM can give us the necessary output, but its performance is almost prohibitive on this volume of data, let alone any larger datasets. We need an algorithm that can both determine the shape of discussion across time and also allow for changes in those same topics; and be able to do this for very large datasets, preferably in parallel.

1.3 Approach

The original DTM code is not parallelized. However, LDA has several strong parallel implementations available. In this thesis, we utilize parallelized LDA on subsets of the data to construct a system roughly analogous to DTM out of parallel components. This system provides the same type of outputs as DTM. In order to succeed, our system needs to run considerably faster than DTM. Our system must also generate data that provides insight for our original scientific inquiry.

1.4 Solution Design

We solve this problem using Clustered Latent Dirichlet Allocation (CLDA). The basic structure of the system is as follows. First, the data are segmented as for DTM, breaking the corpus into discrete time segments. Each of these sub-corpora is the input to a separate run of PLDA+, our chosen implementation of parallelized LDA. The output for this step is set of topics for every timestep. The full list of these topics is passed to a parallelized implementation of k-means clustering, producing a set of topics representative of the full set. Each original topic corresponds to a particular centroid topic. Analysis can be performed on the overall mixture of topics at any given timestep, matching the original topic mixtures to their representative centroids to compare them across time. We can also study how the topics corresponding to a given centroid change over time.

1.4.1 Test Design

Evaluation is along multiple dimensions. First, we want to know if the results are obtained more quickly than when using the original implementation of DTM.

Secondly, are they of high quality? Third, we would also like to know if the results are similar to those obtained by DTM. Runtime is straightforward to test. The second question is more difficult. Overall quality of a model is often evaluated using perplexity measures, although there is evidence suggesting this measure does not necessarily correlate with human perception of topic quality [9]. Similarity can be tested using set comparison metrics like the Sørensen-Dice Coefficient or Jaccard Index. However, these are open questions in the topic modeling community, and there are multiple plausible metrics for similarity.

1.5 Results

We compare our implementation of CLDA to Blei's DTM along three dimensions; runtime, perplexity, and similarity. As our implementation is fundamentally constructed from a highly optimized parallel LDA, its runtime is dramatically faster than that of DTM. Its perplexity over the test set is also superior. The topics it generates are also broadly similar to those generated by DTM. We also explore the effect of CLDA's parameters on perplexity, and demonstrate CLDA's ability to provide detailed information about a given dynamic topic.

Overall, the system provides a practical alternative to DTM, although it does not match perfectly. Future work will focus on exploring the sensitivity of various input parameters, expanding the framework to use other clustering techniques, and developing improved metrics for topic evaluation.

Chapter 2

Review of Literature

2.1 Topic Models

Topic modeling is the process of automating the task of identifying key themes in a collection of documents. This information can be the end goal, allowing examination of the themes in a large corpus. This information can also be used to classify new documents.

2.1.1 Terminology

Topic modeling has formal definitions for a number of common terms.

- Word – The smallest meaningful element of data. Also called a "token".
- Vocabulary – A collection of all unique words. This is typically implemented as a vector.
- Document – A collection of words. A word may appear multiple times in a document. This is typically implemented by a vector where the ith element is the address of the ith word in the vocabulary.

- Corpus – A collection of documents.
- Topic – A means to describe a pattern of words; the specifics vary by method. In LDA, this is a word distribution represented by a vector of probabilities.

2.1.2 Brief History of Topic Modeling

Early attempts at reducing text corpora to their most useful components began with a method called term-frequency inverse-document-frequency (tf-idf). In this method, the frequency of a word in a document is compared to its overall occurrence in all documents, providing a rough estimate of its importance to the document in question. This provides information on word importance without giving undo weight to stopwords. However, it only minimally reduces the description of the corpus and provides no special insight into recurring trends. For example, tf-idf will note that a document contains an unusually high frequency of the words “car” and “door” but no information as to how common this combination is [8].

Latent Semantic Indexing (LSI) uses the importance matrix of tf-idf and applies an additional step. Singular Value Decomposition (SVD) is applied on this matrix to extract the key correlations. Continuing the previous example, LSI will detect that “car” and “door” often appear together.

Probabilistic LSI, despite the name similarity, is closer to LDA than LSI. PLSI takes the co-occurrence data of words and documents as observations of a generative model. To generate a word in a document, randomly choose a topic from that document’s topic distribution, and then randomly choose a word from that topic’s word distribution. Continuing the previous example, a document may have a high likelihood of choosing the “vehicle” topic, which in turn has a high likelihood of generating the words “car” and “door”, resulting in that document having a high occurence of those words. While the specific method used by PLSI leads to overfitting and prevents use as a classifier on held-out data, the model structure forms the core of LDA [11][19].

2.2 Latent Dirichlet Allocation

2.2.1 LDA Assumptions

Latent Dirichlet Allocation (LDA) is the most commonly used method for topic modeling. It has several key assumptions [8]:

1. Each document is equally important.
2. Each document is a “bag of words”: Words are assumed to be unordered.
3. A topic is described by a probability mass function of words.
4. Each document is a mixture of topics, represented by a probability mass function for topic selection.
5. Documents are generated by sampling a topic from their topic mixture, and then sampling a word from that topic. This is repeated to generate all words in the document.
6. Multiple topics can contribute to generating a document.

Only the output of this model (documents) can be observed directly. The topics and topic mixtures are latent distributions that must be inferred. LDA assumes that the prior distribution of each topic is a Dirichlet distribution, which distinguishes it from more generalized methods. Topics are randomly seeded, and then iterated upon using Bayesian inference; during each iteration, the algorithm considers each document's relation with each topic and updates the topics for the next iteration. Iteration continues until some user-specified condition is met.

2.2.2 Bayesian Inference

Bayesian Inference is a critical component of LDA. It is the process of adapting hypotheses to better suit new data without disregarding prior knowledge. For example, suppose a coworker has recently arrived from another city. We have several hypotheses on how he or she traveled between cities;

- Coworker traveled by car.
- Coworker traveled by train.
- Coworker walked the whole way.

These hypotheses are not equally likely. Prior knowledge tells us that people do not generally travel between distant cities by foot. We can improve our estimates of the likelihood of each hypothesis by making observations of available data. If we notice a train ticket stub, this supports the hypothesis that our coworker took a train, and reduces the likelihood of the other hypotheses. If we notice mud on our coworker's shoes, this supports the hypothesis that he or she walked the whole way, but we will still be skeptical due to our prior low belief.

2.2.3 Dirichlet Distribution

To understand the Dirichlet distribution, we begin with the Beta distribution. This is a distribution over [0,1] with two parameters, α and β. These parameters control the shape of the distribution, and its probability density function is given by

$$P(x) = \frac{x^{\alpha-1} \cdot (1-x)^{\beta-1}}{B(\alpha, \beta)}. \tag{2.1}$$

Here, $B(\alpha, \beta)$ is the Beta function, which serves to normalize the distribution.

The Dirichlet distribution is the multivariate generalization of the beta distribution. It is defined by a vector of concentration parameters. Concentration values of 1 represent uniformity, while values approaching zero concentrate all probability mass into a single point. We are concerned with this distribution due to the Dirichlet process, wherein each sampling of the distribution's observed data alters the parameters for the next sample in a "rich get richer" fashion. LDA is essentially a modified Dirichlet process.

2.2.4 LDA Implementations

LDA has many implementations in a wide variety of languages. There are two standard formulations of it, depending on how the Dirichlet priors are updated for the next iteration. The implementation used in the original paper and uses variational Bayes, while many later works, including PLDA, rely on Gibbs sampling.

In both cases, directly computing the optimal values for the latent variables and other parameters is intractable. Variational Bayes divides these unknowns into smaller groups whose local optimums are tractable. Each subgroup is optimized analytically, holding the others constant. As the local optima rely on the value of the other parameters, this suggests an iterative process where each grouping is optimized in alternating sequence until convergence is reached. While this convergence is guaranteed, deriving the processes to perform the local optima represents a substantial development cost.

Gibbs sampling takes a different approach. Rather than a series of alternating optimizations, each observation is sampled repeatedly, updating the latent distributions using Bayesian inference. The posterior distribution after each observation is sampled will tend toward the values of the inferred parameters that are mostly likely to generate the observed data, but as a Monte Carlo method the rate of convergence is unbounded. In practice, there is a length of time where the estimated distribution does not represent the true distribution, called the burn-in period, followed by convergent behavior. While this behavior is less desirable than that of Variational Bayes, the implementation of Gibbs sampling is less complicated and comparatively straightforward to derive.

In either method, iteration is used to approximate the latent parameters. Convergence can be measured either by change in these estimates or in another objective metric of the model, such as the likelihood of producing the training set [8].

2.2.5 LDA Strengths and Weaknesses

One of the assumptions of LDA is that every document is equally important. When evaluating documents over a long span of time, this becomes troublesome. The classification of a document written in 2000 should be based more on how it compares with documents written in the 1990s than in the 1900s. This problem can be partially sidestepped by considering blocks of time as separate collections, and performing LDA on each of them independently. However, this not only greatly reduces the size of the corpus being used on any given task, but also neglects any information about topic evolution. This second element is of interest in many fields of study, and can only be crudely captured by baseline LDA [7].

2.3 Dynamic Topic Modeling

2.3.1 DTM Algorithm

Dynamic topic modeling is one approach to the time dependency problem; documents are sorted into discrete time segments, each containing a sizable corpus of its own. Each time segment has its own LDA model, and these models are linked together during parameter approximation. Each time segment contains the same number of topics, initially seeded by applying LDA to the entire corpus. As a result, topics with matching indices but from different time segments will be similar, having arisen from the same original topics. These topics are further tied together during each iteration of the DTM algorithm, where updates take into account not only the topics in the current time segment but also the one preceding it. We refer to the set of topics linked to each other over time as a *dynamic topic*, where the particular topic at a given timestep is a *local topic*. For example, if one models 20 topics on 10 time segments using DTM, there will be 20 dynamic topics each consisting of 10 local topics. During each iteration, topics are updated by repeated inference on documents in their own timestep, and also by consideration of the topic's form in the preceding timestep.

However, the multinomial model used by LDA and the Gaussian model for the time dynamics are non-conjugate, making posterior inference intractable, and an approximation must be used. Using approximations at each iteration has a negative effect on convergence, causing a process already slowed by splitting the timesteps to slow further [7].

2.3.2 DTM Strengths and Weaknesses

DTM is effective in capturing the transformation of a dynamic topic over time. It maintains the core strength of LDA while also allowing for variance across time periods to account for slowly changing language [7]. However, it possesses no mechanism for the birth or death of topics. Furthermore, the evolution model for the topics assumes the topics are recognizable from one year to the next. While a topic might gradually evolve to be unrecognizable from its original form, each individual jump must be smaller than the distance from that topic to the others in that time.

DTM also retains the weaknesses of LDA, primarily the necessity of specifying how many topics are present. DTM adds further complication to this, as the optimal number of topics may vary by time, which is not supported by the model.

2.4 Clustering

Topic models are by no means the only way to classify data. Clustering numerical data is an omnipresent problem in machine learning. The most relevant approaches are discussed here.

Clustering is a much more general problem than topic modeling, which is but one of many applications. Its terminology is correspondingly more general;

- Feature - Any axis of input used for classification. Height, weight, and color are examples of features. The set of possible combinations of values for each of these features is known as the *feature space*. Any given data point will have a number of features; most algorithms assume that all input data have the same features (though different values).

- Cluster - A set of data points that are "alike"; what this means varies considerably from method to method. Generally, points in the same cluster will be "closer" to each other than they are to points in other clusters, under some distance metric. Whether clusters are disjoint, can overlap, or are nested within one another varies as well.

- Classifier - A method to assign a data point to a cluster. A common way of verifying a classifier is to use it to classify points whose cluster is known (also called labeled data) to confirm they are assigned correctly.

One algorithm for clustering is k-means. This thesis uses k-means due to its combination of simplicity, familiarity, speed, and availability of a fast parallel implementation.

2.4.1 K-Means Algorithm

K-means is among the simplest clustering methods possible. Data points are divided into k clusters based on a simple Voronoi diagram on k points. A Voronoi Diagram is a partition of a space, based on a set of points. The area closer to a given point than to any other points is its Voronoi cell. Each of the k points is the mean of the training points in its cluster, and the surrounding Voronoi cell represents a classifier. K-means is a hard classifier, that is, points near the boundary between Voronoi cells are still treated as part of the cluster with the closer mean.

Within this structure, the quality of a set of clusters is determined via inter-cluster sum of squares, calculated as the sum of the distance between each point in a cluster and its center. Computing the optimal cluster for a given value of k is an NP-hard problem, and as such is generally impractical. However, heuristic algorithms can reach local maxima very quickly, and when run repeatedly on differing initial conditions can often find the optimal solution quickly.

The most well-known algorithm for generating k-means clusters is Lloyd's Algorithm, and is interchangeably known as the K-means algorithm. Beginning with some initial set of k points ("means"), the data are classified to the nearest such point using Euclidean distance. Once all the data have been classified, the means are updated to become the centroid of all points in their cluster. Repeat the process to classify the data, and update the means whose clusters changed, until the change in the centroids falls below a provided threshold, or stops changing entirely.

2.4.2 K-Means Strengths and Weaknesses

K-means' simplicity ensures it has excellent runtime, but it also carries with it a number of assumptions. As a Voronoi classifier, there is an intrinsic assumption that "correct" clusters should be roughly equally sized or that data in different clusters will be separated by considerable distance. It is thus very likely to not identify a small cluster near a large one, but will instead split the large cluster into two.

Even if the data lend themselves well to Voronoi classification, there is the matter of selecting k. While this is a well-studied problem, there is no general solution. Where it is possible to evaluate the value of the resulting classifier, such as with precision and recall, the algorithm can be run with varying values of k and the best result selected. Note that while technically the internal measure of inter-class sum of squares can be used for this purpose, it is optimized when $k = n$ and each data point has its own cluster for a total sum of 0; the choice of k is a prime example of a bias-variance trade-off.

The algorithm can be highly sensitive to its initial conditions, generating different clusters depending on the starting values. This is often handled by running the algorithm repeatedly with varying starting points. Combining this with the possibility of repeating the algorithm for varying k values as mentioned above, ensuring confident results can require many executions. Fortunately, each execution rarely requires more than a handful of iterations, so this whole process is still very fast compared to other clustering approaches.

2.4.3 K-Means Implementation

Most machine learning toolboxes implement K-means. However, some implementations re-calculate the distance between every point and every centroid at every iteration, even though the centroid may not have changed. This is an $O(k \cdot n \cdot d)$ operation, where n is the number of points and d their dimensionality. Even when a centroid has changed, it may not have moved nearly far enough to potentially reclassify a given point. There are a number of clever variants on K-means, such as Yinyang K-Means [13], that produce identical results without performing many of these unnecessary distance calculations. However, even the unmodified algorithm lends itself to parallel optimization. The classification of points at each iteration is a highly parallelizable operation, and the centroid adjustment can be performed with a broadcast-reduce pair.

The chosen implementation of K-means for this project was developed by Northwestern University. The code is readily available, and provides standalone executables for both serial operation and two varieties of parallel operation, using either shared or distributed memory. Its input and output formats are simple as well. It is a lightweight implementation that still fulfills the primary requirement of scalability.

2.5 Related Projects

2.5.1 PLDA+

While LDA alone is a powerful tool, applying it to large corpora, such as those possessed by Google, can surpass the practical capabilities of serial computing. PLDA is one answer to this problem, developed by collaboration between Google Beijing Research and CMU [28]. PLDA builds on a method called Approximate Distributed LDA (AD-LDA) [23]. Instead of a probability mass function, topics are represented by the count of each word assigned to them. For example, if the word "apple" is generated by a topic fifteen times and there are sixty words generated by that topic in total, AD-LDA will record fifteen whereas LDA would record 0.25. This method utilizes data parallelism by dividing up the set of documents across processes, and iterates over the corpus using Gibbs sampling. Each process has a copy of the word counts, and communicates any changes it makes to word assignment in its documents (and thus the resulting topic matrix) at the end of every iteration. During each iteration, processes do not communicate, and thus are working with stale results that are not globally accurate. As such, this can be considered an approximation to serial Gibbs sampling. Experiments show this approximation converges in practice.

PLDA implements the AD-LDA algorithm in MPI, and extends it to provide error recovery, and demonstrate substantial speedup on large corpora. PLDA+ takes the MPI implementation of PLDA and goes further, optimizing the algorithm using the four strategies of data placement, pipeline processing, word bundling, and priority-based scheduling [21]. Data placement enables the pipeline to mask communication delays with further computation, working on one word bundle while communicating the results of another. These word bundles are chosen such that the computation time is long enough to mask communication, and arranged in a circular queue rather than statically assigned to processes. The queue and word bundles are managed by one set of processors while another set performs the Gibbs sampling, thus taking advantage of model parallelism.

PLDA+ succeeds in masking communication with computation, and as a result has superior scalability and performance to even PLDA, which is already fast. PLDA+ nears the theoretical maximum speedup for hundreds of processes and remains very high for all process counts tested.

2.5.2 Parallel DTM

Bhadury et al. [5] devised a method to address the normal complications with DTM's inference algorithm. Previous work relies on mean field approximations, which are costly to calculate. Their work instead utilizes developments in stochastic Markov Chain Monte Carlo methods, a category which also includes Gibbs sampling. This allows them to utilize the more easily parallelized Gibbs sampling framework to estimate posterior likelihood, but is also faster in serial operation. Their results show dramatic speedup over the original DTM implementation, but they have not made their code available as of this writing [5].

2.5.3 DCM-LDA

Dirichlet Compound Multinomial LDA is a method developed for organizing a library corpus, composed of many books. Its primary purpose is to facilitate corpus exploration, either by keyword or exploring related works, without the need for manual tagging. Its structure is quite similar to the system we devise in this thesis. Both systems divide their corpus into sub-corpora for LDA, then subsequently cluster the resulting topics. However, in DCM-LDA, each sub-corpus is a set of tightly interconnected documents. In their example, it is the pages of a single book, while in our data it might be articles within the same journal. These topics are then greedily clustered using a similarity matrix based on Jenson-Shannon divergence, then culled to roughly the most commonly occurring 10%. Despite the superficial similarities, this method is intended for local similarity and global diversity, while ours is designed around the reverse [22].

2.5.4 Other Related Works

A. Ahmed and E. P. Xing developed infinite dynamic topic models (iDTM), which operates similarly to DTM but allows for an unbounded number of topics. Each topic can be born or die out at any given timestep, as demonstrated on the NIPS conference proceedings [3]. Q. Diao et al. developed TimeUserLDA to discover "bursty topics" amongst microblogs such as twitter. Their method successfully identifies major events from noisy data by taking advantage of user history [12]. A. Dubey et al. propose non-parametric topics over time (npToT), which extends the topics over time algorithm to an unbounded number of topics. This treats document time as an observation rather than a classifier, and attaches topics to time distributions rather than single timesteps [14]. C. Chen et al. developed a dynamic topic model that utilizes normalized random measures instead of a Dirichlet process, yielding superior perplexity [10]. Q. He et al. integrate citation information into LDA, tying documents not to a fixed timestep but to the documents they cite [18]. Y. Tu et al. extend LDA into a Citation-Author Topic (CAT) model that identifies expert authors in each topic [25]. S. Xu et al. extend Topics over Time into the Author-Topic over Time (AToT) model, which infers not only topics but also the research interests of contributing authors [30]. K. W. Lim and W. L. Buntine develop a nonparametric model combining a Poisson mixed-topic link model with an author-topic model, using it to model authorship and content of research papers [20]. H. Yu et al. develop an improved Gibbs sampling routine using Fenwick trees, and apply this to speed up parallel LDA [31].

Chapter 3

Problem Motivation

3.1 Impact of High-Performance Computing

While high-performance computing (HPC) clusters have proliferated over the last few decades, they are still not omnipresent. A supercomputer requires substantial resources to obtain and maintain, far beyond discretionary budgeting. Acquiring the funding for such machines requires a convincing argument for the merit of the investment. Understandably, there is a considerable desire for evidence that such an expensive undertaking will have impressive results; however, while anecdotes are plentiful, quantitative descriptions of what happens when an institution gains high-performance resources are scarce.

While this problem plays out all over the globe in varying forms, the National Science Foundation (NSF) is particularly interested in the effectiveness of their investments. To this end, they have commissioned a grant to investigate how the acquisition of high-performance computing resources impacts research in academic institutions. Investigating this question requires examining the characteristics of institutions before and after they gain HPC resources; however, identifying the relevant timeframes and characteristics are both nontrivial problems.

3.2 Evaluating Research Output

The goal of HPC resources is to improve research, for some definition of "improve"; as such, evaluating the success of such investments logically means looking at the output of such research. There are many ways to do this, but a few possible quantitative analyses are of particular noteworthiness.

3.2.1 Graduate Output

One of the simplest ways to evaluate research output is to examine an institution's graduate program; every graduate student must produce new research in order to gain a degree, so simply counting the degrees awarded by a graduate school provides a very rough approximation of the volume of research being done. In theory, any advance that causes research to take less time would cause students to graduate more quickly, thus resulting in an increase in the number of degrees awarded.

Similar logic would suggest that more than just graduate students would finish projects more quickly; the overall rate of publishing would increase if the same work was being done, but more quickly. This, too, provides a rough metric for how much research an institution is performing. Publishing rates vary enormously by field; while this necessitates closely examining data for alternative explanations, changes in publishing rates may also indicate cross-field collaboration encouraged by HPC. Certainly, any anomaly is worthy of exploration to determine its cause [4].

3.2.2 Citation Count

In addition to evaluating the volume of research, one can evaluate its approximate quality. Citation count is a common metric for the importance of a paper, and can provide an easily calculated statistic to examine. However, citation count does not always indicate novelty; a paper discussing the state of the art may garner many citations as it forms a useful reference for other papers to draw on, despite not necessarily presenting anything new in its own right. Papers also unevenly accrue new citations as time passes, making it difficult to justify calibrations of relative importance for papers that were not published at the same time. Utilizing the citation count that papers possessed after a set period of time addresses that problem, but weakens the greatest advantage of using citations; the ease of calculation. While there is valuable information in citation counts, it is a tightly interconnected measure and driven more by external work than the work itself.

3.2.3 Type of Research

Another angle for examining research output is its subject; "nanotech" versus "material science" for example. If it is possible to identify the subject of research output, then changes in this makeup represent another feature to examine for impact. To do so, one must first find a quantitative definition of subject. Naïvely, one possibility of subject classification is the journal to which a paper was submitted. However, major journals often cover a wide range of topics within a field, so this alone is too coarse to observe the types of changes we expect to see. The articles themselves are usually tagged with a set of keywords, but use of these keywords is inconsistent. Instead, we examine the journal articles directly, using the text to define data-driven classifications. This is done through topic modeling, the demands of which lead directly to the subject of this thesis.

3.3 Dynamic Topic Models

Having chosen to examine the subjects of research output via topic models, the next question is what topic model to use. To evaluate a change in research output, we need several qualities in our topic model. Most importantly, we must be able to compare the proportions of topics from year to year. If years are not comparable to each other, we cannot measure differences. This can be done in a limited sense by running LDA on our entire corpus and adding up topic mixtures for each year once complete. However, we also seek insights on how the same overall subject changes over time. For example, we wish to notice if an institution continues to submit articles about material science from year to year, but their lexicon begins to include words relating to simulations once HPC resources are introduced. The combination of these requirements leads us to the need for a dynamic topic, whose overall subject matter remains consistent over time but can morph to incorporate new language. Dynamic topic models implement this requirement, with each timestep having a local topic as the form of a dynamic topic on that time's sub-corpus.

The available implementation of DTM is slow. Running the experiments for our research inquiry can take weeks, causing substantial setbacks if any modifications are necessary. In order to progress with this research, we require a system that generates dynamic topics, and does so more quickly than DTM.

Chapter 4

Clustered Latent Dirichlet Allocation

4.1 Requirements

The main goal of the algorithm is to provide a faster and more flexible alternative to Dynamic Topic Modeling. It must answer the same type of questions. It must accept the same type of input and produce the same types of output. It must be possible to replace DTM's presence in a workflow with this algorithm, both mechanically and conceptually.

The solution must address shortcomings of the original implementation of DTM. In particular, that the runtime of the existing C code is prohibitively long for large datasets. This algorithm must have superior performance to previous DTM implementations, and be scalable to large datasets. It is also desirable that the algorithm allow for the appearance and disappearance of topics over time.

4.1.1 Format

Specific requirements include:

1. Input must match previous DTM input. In this case, the chosen implementation was Blei and Gerrish's C code [6], so the input is the following:

    - A word ID file containing the entire vocabulary.
    - A sequence file containing the number of timesteps and their respective sizes.
    - A wordcount file containing the IDs and counts of each word that appears in a document. Each document occupies one line of this file.

2. Output must include the same information as previous implementations. In particular:

    - Every dynamic topic's form at each timestep
    - The topic mixture for each document

4.1.2 Performance

The algorithm must process much larger datasets in shorter timeframes. The system's runtime must be much shorter than the existing available DTM implementation, in terms of total run time.

4.1.3 Quality

In order to be useful, the topics produced by the algorithm must be either very similar to those produced by DTM, or superior to them. Measuring the quality of a topic model is an open question, but a standard approximation is the perplexity metric. This metric evaluates how likely the topic model is to generate a set of provided documents. A lower perplexity indicates a model more closely fits the documents. As perplexity is a function of probabilities rather than direct model parameters, it can be used to compare models over the same input. This metric is effective in evaluating a model's ability to predict output, but lower perplexity does not necessarily correlate with human perception of topic quality [9][26].

In addition to overall predictive quality, it is useful to evaluate the difference between the results generated by DTM and this system. The primary output of a topic model is a set of topics represented by vectors. While these vectors can be compared directly, their application in this numerical form is already measured by perplexity. Humans, however, do not generally examine these topics in their native numerical form; instead, it is easier to look at the most common words associated with a topic. This transforms the topic from a large vector to a small set, and necessitates a different type of comparison [9].

We can consider the output of two models to be similar if there is substantial overlap in the sets representing their topics, as a rough approximation to whether a human would consider the two outputs to be similar. This transforms the qualitative question of how similar topics look into a quantitative question that can be tested objectively [9]. If the output of this system and DTM are similar on this basis, then the system satisfies the need to generate output similar to that provided by DTM.

4.2 Design

The design for CLDA comes from evaluating the needs of the research questions.

The goal of DTM is to generate a consistent set of topics over a large corpus, and to modify them through an iterative process to better fit the documents of that corpus. There is a persistent subject, or *dynamic topic*, with many forms across time. DTM starts by discovering the overall topics with an LDA initialization step, and then iterates to discover how these dynamic topics take on local forms at each timestep. Our system does the inverse, searching for local topics at each timestep first. These local topics are then collected into dynamic topics in a later step.

Searching for topics in a given timestep is an easily solved problem. Each timestep can be treated as a corpus of its own, and be input into LDA. Similarly, collecting output into like subjects is another solved problem. Topics are vectors, and can be processed using clustering algorithms such as K-means. Both LDA and K-means have readily available parallel implementations, leaving only the data manipulation to be developed.

Figure 4.1 shows the overall process of CLDA.

4.2.1 Step 1: Decompose the Corpus

In order to process each timestep with LDA, the data first need to be separated by timestep. The DTM input files contain all the necessary information, so carving up the full corpus is a simple task for Python or similar languages. The bulk of the data manipulation was handled in Python for this reason; the manipulation operations represent a fairly small portion of the overall runtime, so the language was chosen for ease of implementation. Once the timesteps are separated, there is another data manipulation step to convert them from the DTM input format to the input format of the chosen LDA implementation.

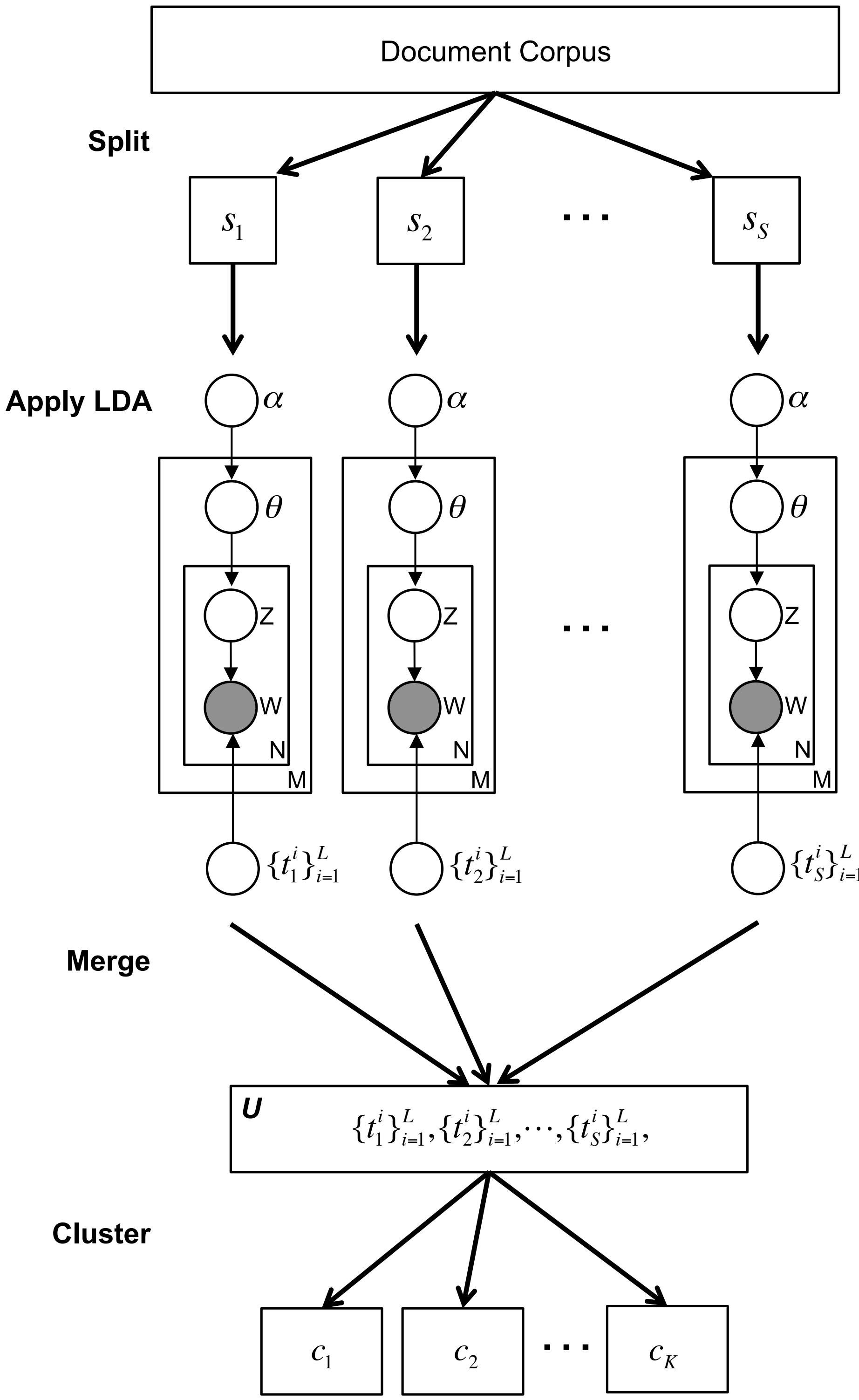


Figure 4.1: Flowchart of the algorithm

The division of the overall corpus into individual timesteps is a serial task, but the remaining data manipulation before each LDA run can act independently on the resulting chunks as long as that process can access the vocabulary list. Most of the manipulation required for the *Merge* step can similarly be done independently, and as such this entire sequence can be trivially parallelized. However, only the LDA step takes substantial time to complete, so parallelizing the manipulation as well is not necessary.

4.2.2 Step 2: Generate Local Topics

Once the corpus has been decomposed into individual timesteps, it is time to process them with LDA. This is a straightforward operation, with a number of LDA runs equal to the number of timesteps; LDA run 1 processes timestep 1, and so on. As these are independent, they can run simultaneously on separate processors - or groups of processors, if using parallel implementations of LDA - for embarrassing parallelism. When this is complete, there will be one set of outputs for each timestep, to be merged in the next step.

To test the algorithm, PLDA+ was chosen as the LDA implementation [21]. The foremost reason for this was its ready availability, but it boasts many other advantages already detailed in a previous section.

4.2.3 Step 3: Merge Local Topics

Once the LDA runs are complete, their output is prepared for input to k-means. At the conceptual level this requires concatenating the emitted topics into a single list, but in practice this step is considerably more involved. The individual outputs have indexing entries that must be removed before they can be concatenated, and then re-indexed to match the specific demands of k-means. More complex than such reformatting is ensuring that the topics generated are comparable; LDA acts on a vocabulary consisting of everything that appears in its source documents, and produces topics with a value for each element in the vocabulary. If a word appears in one document collection but not another, these topics are not directly comparable. As such, if any of the timesteps did not contain the full vocabulary, it is necessary at this stage to pad their topics with the missing entries as zero values.

Proper data cleaning minimizes the importance of padding topics with missing entries; in many of the experiments performed on real data it was not necessary. However, there is no guarantee that this is true on any given dataset, and missing entries are especially likely to occur if data is divided into small subcorpora or spans a wide range of subjects.

In addition to ensuring the vectors are comparable in dimension, they must be comparable in scale. PLDA+ provides varying magnitudes for vectors based on their occurance in the data, but the intention of this algorithm is to cluster based on the meaning of topics, not their occurrence. As such, we normalize the topics before clustering them, using Manhattan distance. This operation is not complicated and has no dependence on other topics, let alone other timesteps, and can thus be done independently before the merge, or all at once afterwards. Our implementation performs this normalization after the merge, but there is no difference in results either way.

4.2.4 Step 4: Clustering Local Topics

After the topic collections have been merged into one file, they must be clustered. While executing this step requires only a one-line script to call an implementation of k-means, there are some important caveats worthy of discussion. Most notably, the choice of k and the initial clusters both represent user specifications with substantial consequences.

The choice of what value to use for k is an open question, and has no general solution. There are a finite range of acceptable values of k for any given input. In the extreme cases, $k = 1$ defines a single cluster containing the entire dataset, and $k = n$ defines a cluster for each and every data point individually. It can be optimized for any given metric of quality, even if just by brute force, but the notion of "quality" has no intrinsic definition. Evaluating the quality of a given set of topics is the subject of future work[1]; for now, the choice of k is left to the judgment of the users.

The k-means algorithm is also sensitive to its initial clusters. It is a heuristic algorithm that settles into local minima, since computing the optimal clustering under inter-class sum of squares is NP-hard. One set of initial values may result in different topics than another, and as before, there is no obvious notion of quality. The internal metric of inter-class sum of squares, while unhelpful for comparing varying values of k because it is minimized by $k = n$ in all input sets, does allow comparison of clusters using the same k; from a set of clusterings, it can be used to choose the one closest to optimal. Even running k-means repeatedly and selecting the best results has its problems, though; generating random initial clusters that are different enough from each other to be useful is a challenge whose solution depends on the data, and every method is likely to skew results toward a particular shape. Fortunately, in this particular case, there is a useful initial guess to utilize; the results of LDA itself. The initial values can either come from random topics sampled from the merged set, or one can run LDA on the entire corpus with k topics. In the latter case, this can be done simultaneously with the other LDA runs, avoiding an unnecessary delay in

[1]See the Conclusions chapter for preliminary thoughts on this subject.

overall runtime. This implementation uses LDA results as its initial clusters, although experiments with other initialization strategies will be the subject of future work.

The implementation used for these experiments performs clustering serially, since the size of the topic matrix is small enough that it can be clustered faster than a job can be scheduled on parallel resources (generally taking less than a second). However, for much larger experiments, the selected k-means code provides parallel implementations for both OpenMP and MPI.

Once clustering is complete, there are two important outputs. The first is the centroids, each usable as a topic in its own right, and the second is the assignment of the original topics to their corresponding centroids.

4.2.5 Step 5: Backtracking for Supplemental Information

The clustering outputs are useful on their own, but they provide entirely different information than the outputs of DTM. DTM does not provide a general vision of a given dynamic topic, only its local topics at each timestep. Clustering provides both a time-agnostic version of a topic and a varying number of local topics at each timestep; including potentially none at all, indicating that topic was not meaningfully present at that time. The likelihood of this depends on the ratio of local to dynamic topics, becoming more unlikely as the number of local topics is increased.

DTM's major outputs are a matrix of local topics for each timestep, and topic mixture values for each document in the corpus. Generating the first one is fairly straightforward; for each dynamic topic generated by clustering, collect the local topics assigned to it at each timestep. As there may be many such local topics, it is necessary to combine them into something directly comparable to DTM; an arithmetic mean can combine the vectors into a topic representative of the subject at that timestep, but this is an area worthy of further exploration. Other possibilities include other statistics, such as the median, or selecting one topic as a paragon based on intra-class similarity.

Generating topic mixture data requires more steps. PLDA+, unlike many LDA implementations, does not provide as output the mixtures used during its iterations. Instead, it comes bundled with a program that estimates topic mixtures given a PLDA+ topic model and a set of documents, which need not be the documents on which the model was trained. To gather the mixture data for the corpus, we run this program using the local topic models and the local data. While this step is presented here for clarity, it is performed immediately after PLDA+ runs, as it is fully independent of other timesteps and can thus be done in parallel.

Combining the local mixture data into the global dynamic topics follows a process similar to combining the local topics. For each document, the topic mixture value for a dynamic topic is the sum of the topic mixture values for each local topic assigned to that dynamic topic.

Chapter 5

Experimental Validation

There are several questions to be answered in this analysis:

- How does the runtime of CLDA compare to DTM?
- How does the output of CLDA compare to DTM? Specifically:
  - Is the output of CLDA similar to the output of DTM?
  - How does the quality of the output of CLDA compare to the output of DTM?
- How do the user specified parameters of CLDA influence output quality?
- What insights can CLDA provide that DTM cannot?

5.1 Data and Resources

We chose two datasets to use as case studies for our experiments. These datasets are both collections of research text in computer science, allowing us to intuit meaning from the topics more easily than we would for data from other fields.

Other types of text data may have other properties, and exploring them will be the subject of future work. Coincidentally, both datasets also span 17 years. Future work will also investigate the effects of data spanning a much larger time scale.

5.1.1 Journal Abstracts

Clemson's Data-Intensive Computing Ecosystems lab has a collection of journal abstracts acquired from Elsevier through partnership with LexisNexis. These abstracts span 1996-2012 and 3 subjects: chemistry, computer science, and economics. The focus of this analysis will be on the computer science abstracts, as it is the most familiar and thus simplest to verify intuitively.

Cleaning the data requires several steps. The raw data also contain substantial metadata stored in XML, which must be removed to extract the raw text. Samples of this raw text can be found in Appendix A. From there, we remove special characters and punctuation; these are replaced with spaces, as we discovered that simply removing punctuation often leads to words being combined. We then convert all letters to lower-case. It is important to do these steps before removing words, since most string comparison functions will not recognize that "Apple:" and "apple" are the same word. The first words removed are stopwords. Then, we remove every word that does not appear in most documents. We initially removed words that did not appear in at least 1% of the documents, but this reduces the vocabulary down to 1,253 unique words. As an alternative, we prepared a second version of the dataset that only removed words that did not appear in at least 0.01% of the documents, which leaves 22,410 unique words in the vocabulary. The 1% version of the dataset is used in most of the experiments directly comparing CLDA and DTM, while the 0.01% version is used in the experiments about CLDA's parameters. Each experiment will note which version of the data is used. Details on the properties of both versions of this dataset are found in Table 5.1.

5.1.2 NIPS Conference Proceedings

The Neural Information Processing Systems (NIPS) Conference provides the full text of every paper accepted there, and is commonly used as a point of reference for a variety of machine learning techniques. We use a version of the data containing papers scanned using Optical Character Recognition (OCR) for the years 1987-2003. This version has stopwords already removed and minor manual data cleaning [15]. We left this dataset as-is to maintain comparability with other results on these data and did not perform any additional pre-processing beyond format changes. Details of the data are found in Table 5.1.

5.1.3 Computing Resources

All experiments were run on Clemson University's Palmetto Cluster. Palmetto is a highly ranked academic research cluster, consisting of 1,978 compute nodes with 20,728 cores and 598 NVIDIA Tesla GPU accelerators, with 36,608 GB of memory [2][1]. Since Palmetto is a diverse resource, care was taken to ensure different experiments landed on the same types of nodes.

5.2 Comparison to DTM

To compare our system to DTM, we used Blei and Gerrish's implementation of DTM [6], used in the original paper developing the method [7]. Since the start of this project, another implementation has been published by Bhadbury et al. [5].

Table 5.1: Dataset Details for Computer Science Journal Abstracts and NIPS Conference Proceedings

Dataset	Vocabulary Size	Total Documents	Total Tokens
Abstracts (1%)	1,253	533,560	25,201,799
Abstracts (0.01%)	22,410	533,560	32,551,540
NIPS	14,036	2,484	3,280,697

Abstracts		NIPS	
Year	Documents	Year	Documents
1996	17191	1987	90
1997	19277	1988	95
1998	18778	1989	101
1999	15663	1990	143
2000	16293	1991	144
2001	16005	1992	127
2002	17434	1993	144
2003	22315	1994	140
2004	21151	1995	152
2005	29054	1996	152
2006	34499	1997	151
2007	42115	1998	151
2008	48218	1999	150
2009	52821	2000	151
2010	54103	2001	192
2011	54946	2002	203
2012	53639	2003	197

Table 5.2: Runtime Results on Computer Science Abstracts

	# of Processors	# of Iterations	Walltime (minutes)	Walltime (hours)
DTM (Blei)	1	100	3497	58.3
CLDA	12	1,000	12	0.2
CLDA	24	1,000	6	0.1
CLDA	48	1,000	2	0.03
CLDA	48	10,000	18	0.3

Their implementation is parallelized and reports excellent performance, but its code is not yet publicly available; as such, our comparisons are restricted to the original implementation.

5.2.1 Runtime

To compare runtimes, we used the same input files for both our system and DTM. However, for our system, there are additional parameters for parallel operation, as well as the number of local topics. For this test, we used the computer science abstracts using the 1% word appearance threshold. All tests used 20 global topics, with our system using 50 local topics.

The results shown in Table 5.2 demonstrate that the algorithm is orders of magnitude faster than the original implementation of DTM. This is unsurprising; the primary operation of consequence is the LDA phase of the algorithm, which utilizes the highly optimized PLDA+. The other operations largely consist of data manipulation to normalize or rotate files, and the clustering step. However, since only the topics themselves are used for clustering, k-means can process these data in seconds serially.

As a consequence of this speed, we can run LDA for many more iterations than is practical with DTM, and on many more local topics. PLDA+ is also highly scalable, which is encouraging for applications to larger corpora.

5.2.2 Perplexity

Perplexity is a standard measure of the overall quality of a topic model. It captures how well the model matches the provided data. It is the exponent of entropy, which is calculated as the negative inverse of the log-probability. Calculating perplexity requires calculating the probability of an input set being generated by the model, which is the product of the probability of generating each document in the input set. The probability of generating a document i is the product of the probability of generating each of its words $w_{i,j}$, of which there are n_i. As multiplying these probabilities quickly results in underflow in floating point arithmetic, these products are expressed as sums of logarithms instead. Here, this simply pushes the existing logarithm through the product operations to where it no longer causes problems, as shown in Equation 5.1 [27]. The formula used for perplexity is thus

$$Perplexity = e^{\frac{-\log\left(\prod_{1..d}^{i}\left(\prod_{1..n_i}^{j} P(w_{i,j})\right)\right)}{\sum_{1..d}^{i} n_i}} = e^{\frac{-\sum_{1..d}^{i}\left(\sum_{1..n_i}^{j} \log P(w_{i,j})\right)}{\sum_{1..d}^{i} n_i}}. \tag{5.1}$$

Calculating $P(w)$ depends on the model used. For most topic models, the probability of a word appearing in a document is based on the topic mixture of that document. Each token has a chance of being drawn from any of the document's constituent topics, each of which has its own chance of generating any particular word. The probability of a token appearing in a document is

$$P(w) = \sum_{1..T}^{t} P(w|t) * P(t). \tag{5.2}$$

$P(w|t)$ is an entry in the topic matrix, and estimating it consists of a lookup. $P(t)$ is a value from the topic mixtures, for which estimates are not always available. The DTM code used for these experiments estimates the topic mixtures for each document in its training corpus, but the LDA code used in our system does not. See the "Backtracking" section for how this is handled. Further, testing on a set of held-out documents, rather than the training set, requires calculating new topic mixtures regardless of implementation. PLDA+ comes equipped with a program to perform this task for its particular form of topic output, but DTM does not. Algorithms exist for approximating these values, but there is no intrinsic method. Wallach et al. describes several of these algorithms, and evaluates their strengths and weaknesses [26]. For the purposes of our experiment, we rely only on the methods provided with the code.

We applied DTM to the computer science abstract data while specifying 20 global topics, using the topic mixtures output by the DTM code to calculate perplexity. We applied CLDA with 20 global topics and 50 local topics to this same data, calculating topic mixtures using the code provided with PLDA+. We also applied CLDA to a randomly selected subset of this data, consisting of 80 percent of the documents in each time step, and then calculated perplexity using the 20 percent of documents not used to generate the model. The results of this experiment can be found in Table 5.3.

Our system has a substantially lower perplexity than the DTM implementa-

Table 5.3: Perplexity results on computer science abstracts.

	Vocab. Size	Total Log-Likelihood	# of Tokens	Perplexity
DTM (All Data)	1253	-196,815,247	25,201,799	2,464.07
CLDA (All Data)	1253	-149,717,941	25,201,799	380.22
CLDA (Heldout)	1253	-31,723,462	5,027,567	549.99

tion, meaning that the model much more closely fits the training data. Future work will explore other ways to evaluate the quality of a topic model.

5.2.3 Similarity

The previous results indicate that our system is both very fast and has low perplexity. We wish to know how similar the generated topics are to those generated by DTM.

Topics are probability mass functions represented by vectors, but this is not how humans interpret them [9]. Rather than look holistically at the entire vector, a human will typically examine the most heavily weighted words in a topic; for example, the top five. These words will provide insight as to the conceptual meaning of a topic. In order to compare the insights gleaned from a set of topics, we thus need to compare what a human compares; the words most strongly tied to a topic [9].

Set theory provides several metrics for comparing finite sets. Of particular interest are the Sørensen-Dice coefficient

$$S(A,B) = \frac{2 * |A \cap B|}{|A| + |B|} \tag{5.3}$$

and the Jaccard index

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{5.4}$$

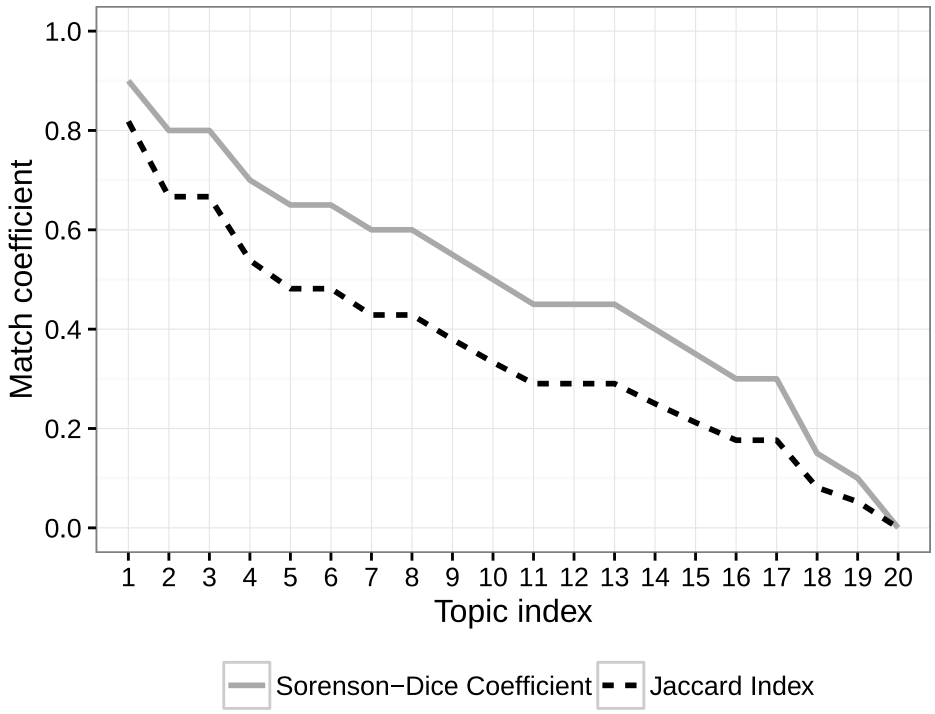
both of which examine the relative size of the intersection. Sets that share most of their elements with each other will generate values closer to 1, while sets that share few elements will be closer to 0.

In order to transform our data into a form where it can be evaluated with these metrics, we use the top 20 words in a topic as its representative set. Chang et al. [9] used the top 5 words as the core of a topic for their intruder experiment, but they were using humans to detect outliers instead of searching for broad similarity. We chose this value as it is low enough to be human-readable, but high enough to dampen the impact of minor value differences on ordering. However, this value is still arbitrary. Future work will explore other means of transforming topics into sets.

We compared the systems on two levels, local and global, using both measures. Each comparison requires having a single set of topics to compare to a single set of topics. At the local level, DTM has exactly one local topic per dynamic topic, but CLDA does not. We computed a local topic centroid for each of CLDA's dynamic topics using the local topics assigned to it, averaging these local topics together to form a new topic which could be compared to a DTM topic. Specifically, each word probability estimate for this local centroid is the mean of the word probability estimates of its component local topics. For the global comparison, CLDA provides a centroid topic for each dynamic topic but DTM does not. We estimated a global centroid topic for each of DTM's dynamic topics using the same process, averaging together the local topic from each time segment. These means provide us with an equal length set of topics for each system both globally and at each timestep.

Both the Sørensen-Dice coefficient and the Jaccard index only compare single sets to each other. Comparing the output requires assigning a bipartite matching between each collection of 20 topics. This is a "Stable Marriage Problem" and is well studied in matching. While the general problem is NP-hard, knowledge of our specific

Figure 5.1: Similarity of Dynamic Topic Means under Sørensen-Dice Coefficient and Jaccard Index.



problem allows us to avoid the need for a general solution. If each collection contains a topic describing a concept; for example, "neural nets"; these topics should ideally match each other more closely than they match any other topics in the opposite collection. If they do not, then our results are not particularly similar, and a low value is ensured regardless of the optimality of matching. Our experiment utilizes this assumption by greedily matching the pair of unassigned topics that are closest to each other under the Jaccard index out of all possible pairings, repeating the process until all topics are assigned. The Jaccard index and Sørensen-Dice coefficient are calculated for each match. The values of the global matches are shown in Figure 5.1, sorted from best to worst. The local matches for each timestep can be found in Appendix B.

Table 5.4: Probability of overlap of random topics among top 20 values.

Overlap of at least:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Probability:	0.275	0.040	0.0038	0.00026	1.3E-05	5.3E-07	1.7E-08	4.5E-10	9.6E-12	1.7E-13	2.5E-15	3.1E-17	3.0E-19	2.5E-21	1.6E-23	8.1E-26	3.1E-28	8.3E-31	1.4E-33	1.2E-36

Even low values under these metrics represent an overlap in topics unlikely to occur by coincidence. If a topic was created completely at random by choosing 20 words from this vocabulary, the chance at least one of those words overlaps with a selected topic is only 28%. We calculate this by noting the chance of a single random word lands in the selected topic is 20 (the number of words in the topic set) divided by the number of words in the vocabulary (here, 1253), which is approximately 1.6%. Subsequent probabilities can be modeled using the binomial distribution, where each event represents choosing a word that matches the topic in question. This approximation is imperfect; it assumes that all words are equally likely to be in the random topic's top 20, and that a word could potentially appear more than once. However, the predictions approach zero likelihood rapidly enough that we consider this a useful estimate; see Table 5.4. Using these probability estimates, we can estimate the expected value of the Sørensen-Dice coefficient and Jaccard index for random topics, by adding together the products of each overlap's probability estimate and the value of the metric for that overlap. Using this process, the expected value of the Sørensen-Dice coefficient for this experiment is estimated at 0.016, and the expected value of the Jaccard index is estimated at 0.014. With this in mind, the coefficients presented earlier represent a considerable similarity that is extremely unlikely to have occurred by coincidence.

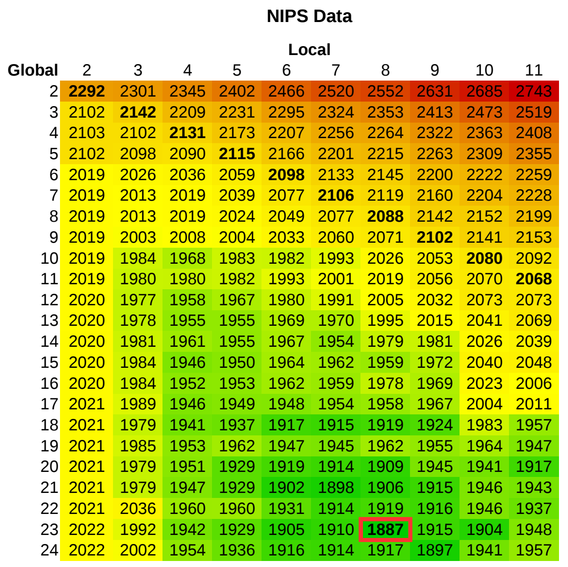
5.3 Parameter Optimization

We wish to know if there is an optimal ratio between the number of local topics L and the number of clusters K. Perplexity can be used to compare varying parameters of the same model as well as different models. We show perplexity results for a range of values for L local topics and K global topics. In Figure 5.2 we show the results of CLDA models varying both L and K from 2 to 30 in steps of 4 on the NIPS data. In Figure 5.3 we show the results of CLDA models varying K from 14 to 90 and L from 34 to 70 on the computer science abstract data, using the 0.01% word appearance threshold.

Our estimated perplexity on these data is comparable to that obtained by Bhadury et al. [5]. These results show several interesting trends. The overall direction of improving perplexity is increasing in both K and L at a similar rate, but increasing one while fixing the other does not appear to cause improvement. We hypothesize that a given value of K or L implies an optimal value of the opposite parameter, beyond which the model degrades. While the abstract data have optimal perplexity values with close values of K and L, the NIPS data shows superior perplexity when K is substantially larger than L. Further investigation is needed to determine the source of this difference.

5.4 Global and Local Topic Dynamics

LDA can be used to capture changes in topic proportions over time by training topics over a whole corpus and then evaluating segments of it. This does not capture any change in topic language over time, forcing each segment to use the same topics. DTM relaxes this constraint by allowing the topics to vary over time. DTM produces



NIPS Data

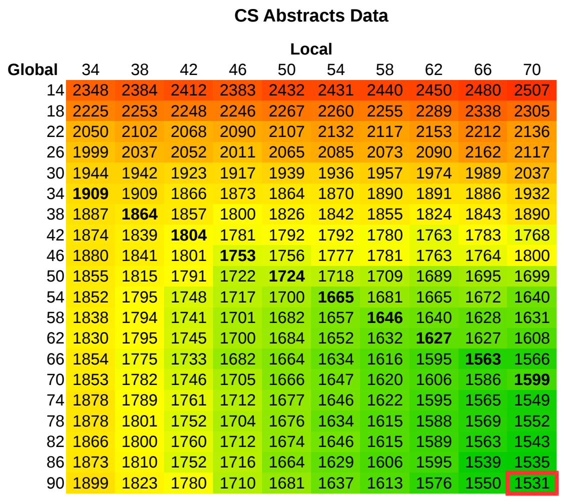
	Local									
Global	2	3	4	5	6	7	8	9	10	11
2	**2292**	2301	2345	2402	2466	2520	2552	2631	2685	2743
3	2102	**2142**	2209	2231	2295	2324	2353	2413	2473	2519
4	2103	2102	**2131**	2173	2207	2256	2264	2322	2363	2408
5	2102	2098	2090	**2115**	2166	2201	2215	2263	2309	2355
6	2019	2026	2036	2059	**2098**	2133	2145	2200	2222	2259
7	2019	2013	2019	2039	2077	**2106**	2119	2160	2204	2228
8	2019	2013	2019	2024	2049	2077	**2088**	2142	2152	2199
9	2019	2003	2008	2004	2033	2060	2071	**2102**	2141	2153
10	2019	1984	1968	1983	1982	1993	2026	2053	**2080**	2092
11	2019	1980	1980	1982	1993	2001	2019	2056	2070	**2068**
12	2020	1977	1958	1967	1980	1991	2005	2032	2073	2073
13	2020	1978	1955	1955	1969	1970	1995	2015	2041	2069
14	2020	1981	1961	1955	1967	1954	1979	1981	2026	2039
15	2020	1984	1946	1950	1964	1962	1959	1972	2040	2048
16	2020	1984	1952	1953	1962	1959	1978	1969	2023	2006
17	2021	1989	1946	1949	1948	1954	1958	1967	2004	2011
18	2021	1979	1941	1937	1917	1915	1919	1924	1983	1957
19	2021	1985	1953	1962	1947	1945	1962	1955	1964	1947
20	2021	1979	1951	1929	1919	1914	1909	1945	1941	1917
21	2021	1979	1947	1929	1902	1898	1906	1915	1946	1943
22	2021	2036	1960	1960	1931	1914	1919	1916	1946	1937
23	2022	1992	1942	1929	1905	1910	**1887**	1915	1904	1948
24	2022	2002	1954	1936	1916	1914	1917	1897	1941	1957

Figure 5.2: Heatmap showing perplexity estimated from cross-validation for different combinations of topic counts on the NIPS data. Best perplexity is highlighted in red.

both a version of each topic at each segment, as well as the relative proportion of each topic at each segment, demonstrating how both language and representation change over time [7]. However, DTM fixes the number of topics across time, with each overall topic having one representative per segment. CLDA relaxes this further, allowing a global topic to have any number of local representatives at each segment. In addition to allowing for topics to branch out, better fitting their local data, this also allows for global topics to appear and disappear entirely.

The strength of DTM is the variation of topics over time, taking on forms better suited to their local data while remaining tied together by a common theme. Blei et al. [7] demonstrate this by examining the changing form of a topic at several time steps, as well as their changing proportions over time. CLDA produces output to provide this same type of insight into a corpus.

We show the changing topic proportions for selected topics in both the NIPS data and computer science abstract data in Figure 5.4. Like DTM, CLDA provides



CS Abstracts Data

	Local									
Global	34	38	42	46	50	54	58	62	66	70
14	2348	2384	2412	2383	2432	2431	2440	2450	2480	2507
18	2225	2253	2248	2246	2267	2260	2255	2289	2338	2305
22	2050	2102	2068	2090	2107	2132	2117	2153	2212	2136
26	1999	2037	2052	2011	2065	2085	2073	2090	2162	2117
30	1944	1942	1923	1917	1939	1936	1957	1974	1989	2037
34	**1909**	1909	1866	1873	1864	1870	1890	1891	1886	1932
38	1887	**1864**	1857	1800	1826	1842	1855	1824	1843	1890
42	1874	1839	**1804**	1781	1792	1792	1780	1763	1783	1768
46	1880	1841	1801	**1753**	1756	1777	1781	1763	1764	1800
50	1855	1815	1791	1722	**1724**	1718	1709	1689	1695	1699
54	1852	1795	1748	1717	1700	**1665**	1681	1665	1672	1640
58	1838	1794	1741	1701	1682	1657	**1646**	1640	1628	1631
62	1830	1795	1745	1700	1684	1652	1632	**1627**	1627	1608
66	1854	1775	1733	1682	1664	1634	1616	1595	**1563**	1566
70	1853	1782	1746	1705	1666	1647	1620	1606	1586	**1599**
74	1878	1789	1761	1712	1677	1646	1622	1595	1565	1549
78	1878	1801	1752	1704	1676	1634	1615	1588	1569	1552
82	1866	1800	1760	1712	1674	1646	1615	1589	1563	1543
86	1873	1810	1752	1716	1664	1629	1606	1595	1539	1535
90	1899	1823	1780	1710	1681	1637	1613	1576	1550	1531

Figure 5.3: Heatmap showing perplexity estimated from cross-validation for different combinations of topic counts on the computer science abstracts data. Best perplexity is highlighted in red.

insight into the rising and falling predominance of various topics in a corpus. Unlike DTM, CLDA global topics need not be composed of exactly one topic at each segment. Figure 5.5 shows how a changing number of local topics represent a global topic we identify as "Computer Networks", along with their relative proportions.

We show the top words for these local topics at selected segments in Figure 5.6. While these topics are all clustered together, they represent distinct ideas within the overall concept of "Computer Networks". One may focus on software defined networking, while another may focus on the communication between remote sensors. While this distinction is useful to examine, treating these as fully separate topics does not produce an accurate picture of how prevalent computer networks research is in the corpus as a whole. Clustering these topics together provides both the global insight of overall representation and local insight into a research area's subdomains.

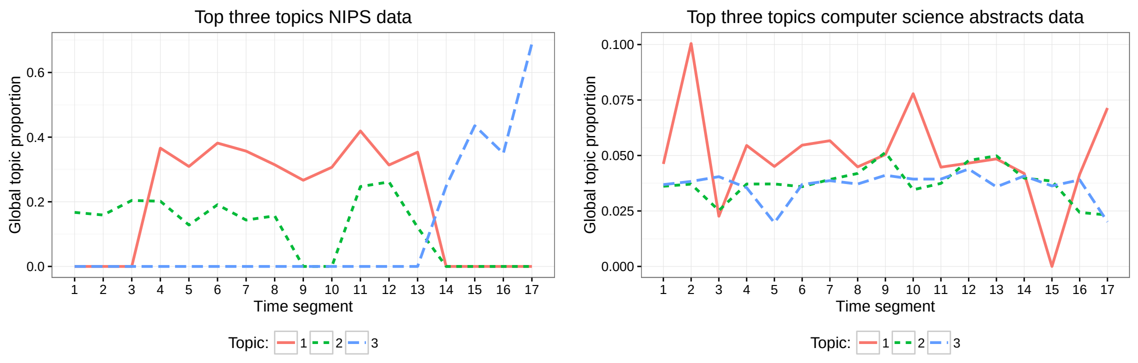


Figure 5.4: Evolution of three largest global topics for the NIPS data (left panel) and computer science abstracts data (right panel).

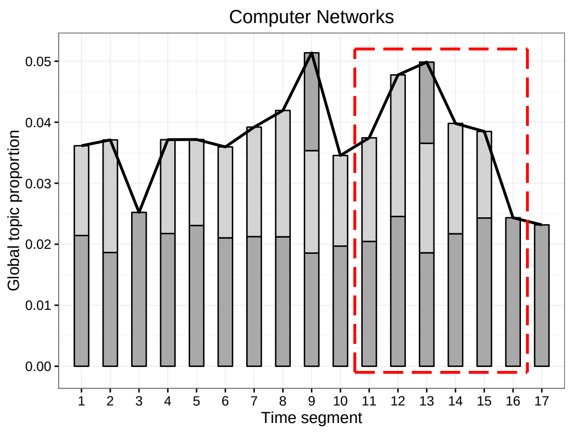


Figure 5.5: Evolution of global topic “Computer Networks”. Each bar represents a local topic, with bar height corresponding to the proportion a local topic contributes to the global topic at a time segment. See Figure 5.6 for a more detailed description of the local topics inside the red rectangle.

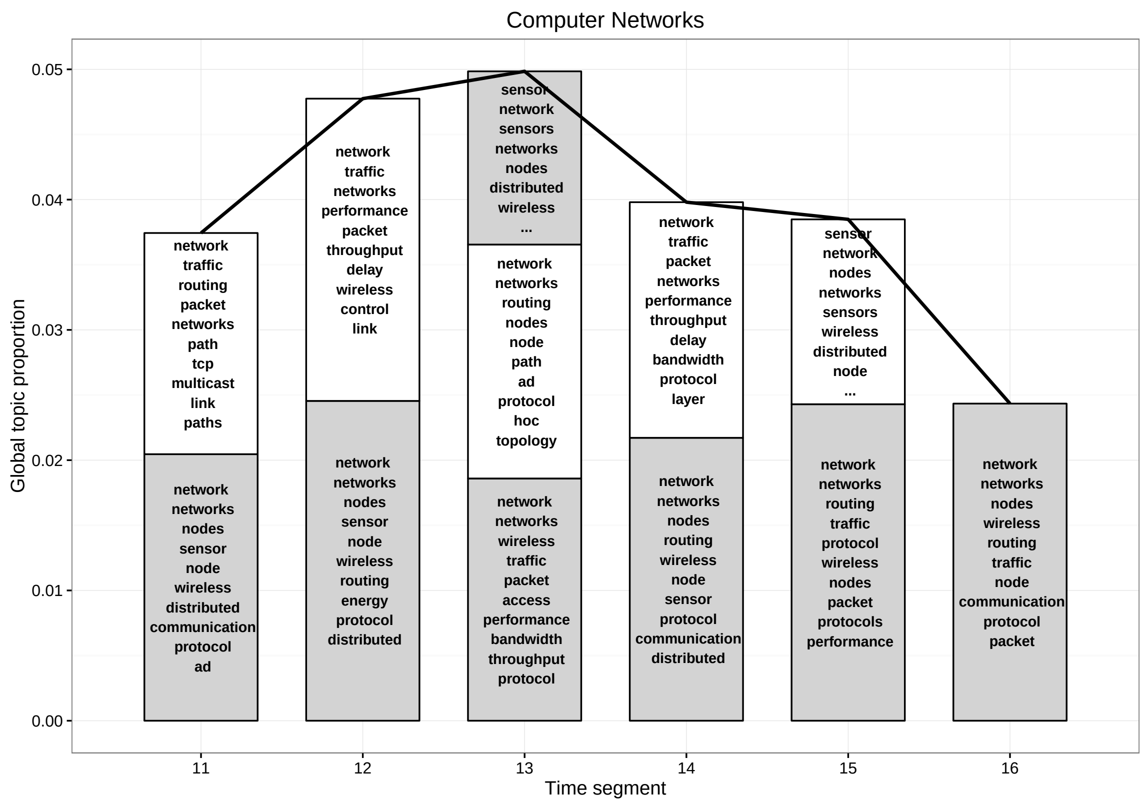


Figure 5.6: Local topics for selected time segments corresponding to global topic "Computer Networks" from the computer science abstracts data using 62 global topics and 50 local topics in each segment. Each bar lists the top words in each local topic. The height of each bar corresponds to the proportion a local topic contributes to the global topic.

Chapter 6

Conclusion

6.1 Outcomes

We have analyzed Clustered Latent Dirichlet Allocation, an alternative to Dynamic Topic Modeling. CLDA uses existing parallel components to vastly increase speed and facilitate the use of large corpora. It begins by dividing up data by timestep, and performing Latent Dirichlet Allocation on each timestep individually. The resulting local topics are merged together using k-means clustering, producing a smaller number of dynamic topics; each dynamic topic is composed of a number of local topics at each timestep, and provides a vision of what a cohesive idea looks like at different times. Our system was built using PLDA+, with R and Python code performing the data manipulations.

We find that our system performs faster than the original implementation of DTM by two orders of magnitude. CLDA also has a lower perplexity than DTM. The topics generated by CLDA are similar to those generated by DTM. CLDA shows a more detailed composition of local topics than is possible with DTM, and enables global topics to emerge and disappear over the time span. Taken together, these results show that CLDA is a promising approach for modeling dynamics in topics estimated from textual data.

6.2 Future Work

The system has already demonstrated usefulness in its intended purpose. However, many questions remain unanswered, and there are many ways to explore the full potential of this design. Some of them are listed below;

- How does performance compare across a wider variety of corpora? Small? Large? Diverse?
- Are there better metrics for evaluating the quality of a set of topics? If so, can these metrics be used to guide a data-driven approach to the choice of how many topics to use?
- How does the system perform using different clustering techniques? Does it function better with hard classifiers (like k-means) or fuzzy classifiers?

Appendices

Appendix A Sample Documents

The following are a selection of documents from the Journal Abstracts corpus used in our experiments. This is the raw text, before any pre-processing is applied.

- 2-s2.0-0030181502 1996 y eng <abstract original="y" xml:lang="eng"><ce:para>Several loop applications of wireless technology are aimed at reducing the cost of deploying communications services ranging from telephone to wideband video. In these applications, wireless links replace a portion of a wireline loop from a central location (a central office or cable headend) to a subscriber. The replacement of labor-intensive wireline technology by complex mass-produced integrated electronics in wireless transceivers is projected to reduce the overall cost of the resulting loop. These wireless loop applications attempt to provide existing communications services or small modifications to existing communications services. A different interpretation of a wireless loop makes use of low-power digital radio technology to provide the last thousand feet or so of a loop. Low-power low-complexity wireless loop technology in small base units can be integrated with network intelligence to provide the fixed-infrastructure network needed to support economical personal communications services (PCS) to small, lightweight, low-power personal voice and/or data communicators. Low-complexity communicators can provide many hours of "talk time" or data transmission time and perhaps several days of standby time from small batteries ($\leq$ 1.5 oz). Because this application of wireless loop technology can reduce the inherent costs in several parts of a wireline loop, it has the potential to provide convenient widespread PCS at less costs than providing telephone services over conventional wireline loops. This low-power wireless loop application does not fit into any existing communications system paradigm. Wireless technology with tetherless access and wide-ranging mobility, e.g., the personal access communications system (PACS), does not fit the accumulated wisdom of the wireline telephony paradigm. It also does not fit the paradigm of existing cellular radio that has sparsely distributed expensive cell sites, and it is not targeted at fixed video services as is wireless cable. Because a significant change in thinking is required in addressing this new low-power low-complexity widespread wireless loop paradigm, its large economic advantages and service benefits have not yet been embraced by many of the existing communications providers, who appear to be more comfortable pursuing the better-known paradigms of video using wireless cable, or of cellular radio in the guise of high-tier PCS, or in the guise of rapid economical deployment of telephone services in developing nations. This paper discusses the inherent economic advantages and service benefits of low-power low-complexity wireless loop technology integrated with network intelligence aimed at providing economical low-tier PCS to everyone. Âľ 1996 Plenum Publishing Corporation.</ce:para></abstract>

- 2-s2.0-0029753866 1996 y eng <abstract original="y" xml:lang="eng"> <ce:para>This paper presents 35, 95, and 225 GHz polarimetric radar backscatter data from snow-cover. It compares measured backscatter data with detailed in situ measurements

of the snowcover including niicrostructural anisotropies within the snowpack. Observations of backscatter were made during melt-freeze cycles, and measurable differences in the normalized radar cross section between older metamorphic snow and fresh low-density snow were observed. In addition, these data show that the average phase difference between the copolarized terms of the scattering matrix, Svv and S/,? is nonzero for certain snow types. This phase difference was found to be related to snowpack features including anisotropy, wetness, density, and particle size. A simple backscatter model based on measured particle size and anisotropy is found to predict the Mueller matrix for dry snowcover with reasonable accuracy. ÂĬ1996 IEEE.</ce:para> </abstract>

Appendix B Additional Similarity Graphs

These graphs show the matchings between the topics generated by CLDA and DTM for each local time segment. They are sorted from best match to worst match.

Figure 1: Similarity of Local Topic Means under Sørensen-Dice Coefficient

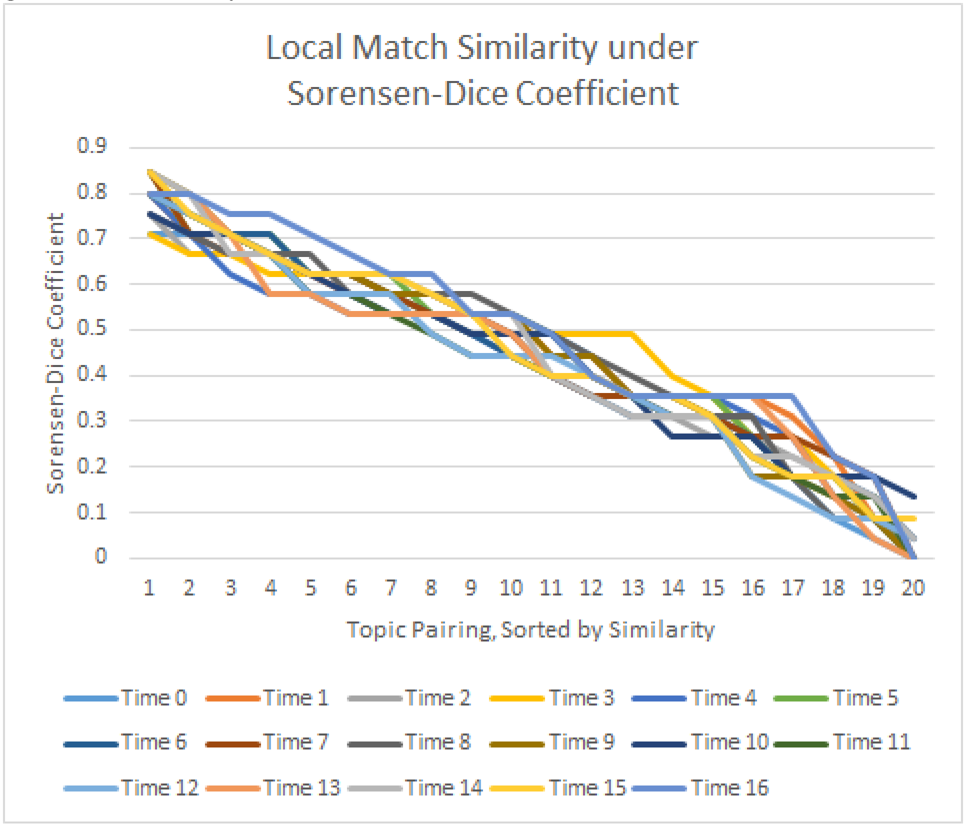
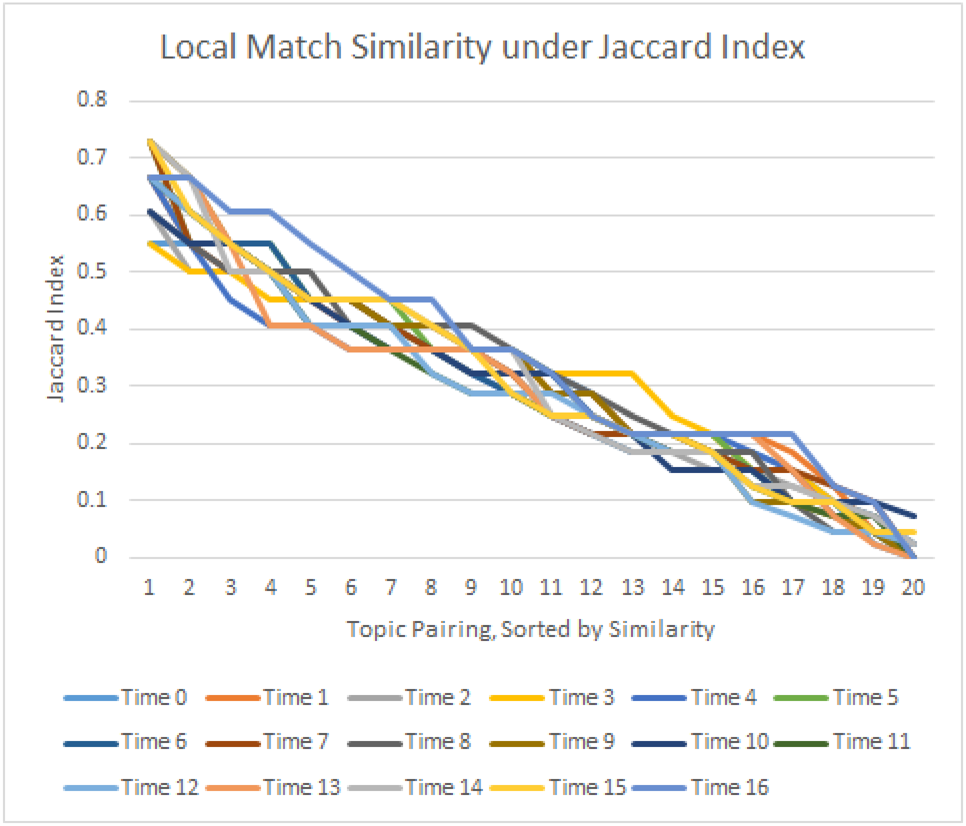


Figure 2: Similarity of Local Topic Means under Jaccard Index



Bibliography

[1] Palmetto user's guide, 2016. Online; accessed April 7, 2016.

[2] Palmetto2 — top500 supercomputer sites, 2016. Online; accessed March 24, 2016.

[3] Amr Ahmed and Eric P Xing. Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream. *arXiv preprint arXiv:1203.3463*, 2012.

[4] Amy W Apon, Linh B Ngo, Michael E Payne, and Paul W Wilson. Assessing the effect of high performance computing capabilities on academic research output. *Empirical Economics*, 48(1):283–312, 2015.

[5] Arnab Bhadury, Jianfei Chen, Jun Zhu, and Shixia Liu. Scaling up dynamic topic models. *arXiv preprint arXiv:1602.06049*, 2016.

[6] David M. Blei. Topic modeling, 2016. Online; accessed April 12, 2016.

[7] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

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

[9] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296, 2009.

[10] Changyou Chen, Nan Ding, and Wray Buntine. Dependent hierarchical normalized random measures for dynamic topic modeling. *arXiv preprint arXiv:1206.4671*, 2012.

[11] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6):391–407, 1990.

[12] Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. Finding bursty topics from microblogs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 536–544. Association for Computational Linguistics, 2012.

[13] Yufei Ding, Yue Zhao, Xipeng Shen, Madanlal Musuvathi, and Todd Mytkowicz. Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 579–587, 2015.

[14] Avinava Dubey, Ahmed Hefny, Sinead Williamson, and Eric P Xing. A nonparametric mixture model for topic modeling over time. In *SDM*, pages 530–538. SIAM, 2013.

[15] A. Globerson, G. Chechik, F. Pereira, and N. Tishby. Euclidean Embedding of Co-occurrence Data. *The Journal of Machine Learning Research*, 8:2265–2295, 2007.

[16] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

[17] William Gropp, Ewing Lusk, and Rajeev Thakur. *Using MPI-2: Advanced features of the message-passing interface*. MIT press, 1999.

[18] Qi He, Bi Chen, Jian Pei, Baojun Qiu, Prasenjit Mitra, and Lee Giles. Detecting topic evolution in scientific literature: how can citations help? In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 957–966. ACM, 2009.

[19] 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.

[20] Kar Wai Lim and Wray L Buntine. Bibliographic analysis with the citation network topic model. In *ACML*, 2014.

[21] Zhiyuan Liu, Yuzhou Zhang, Edward Y Chang, and Maosong Sun. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):26, 2011.

[22] David Mimno and Andrew McCallum. Organizing the oca: learning faceted subjects from a library of digital books. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 376–385. ACM, 2007.

[23] David Newman, Padhraic Smyth, Max Welling, and Arthur U Asuncion. Distributed inference for latent dirichlet allocation. In *Advances in neural information processing systems*, pages 1081–1088, 2007.

[24] Peter S Pacheco. *Parallel programming with MPI.* Morgan Kaufmann, 1997.

[25] Yuancheng Tu, Nikhil Johri, Dan Roth, and Julia Hockenmaier. Citation author topic model in expert search. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1265–1273. Association for Computational Linguistics, 2010.

[26] 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.

[27] Chong Wang, David Blei, and David Heckerman. Continuous time dynamic topic models. *arXiv preprint arXiv:1206.3298*, 2012.

[28] Yi Wang, Hongjie Bai, Matt Stanton, Wen-Yen Chen, and Edward Y Chang. Plda: Parallel latent dirichlet allocation for large-scale applications. In *Algorithmic Aspects in Information and Management*, pages 301–314. Springer, 2009.

[29] Eric P Xing, Qirong Ho, Pengtao Xie, and Wei Dai. Strategies and principles of distributed machine learning on big data. *arXiv preprint arXiv:1512.09295*, 2015.

[30] Shuo Xu, Qingwei Shi, Xiaodong Qiao, Lijun Zhu, Hanmin Jung, Seungwoo Lee, and Sung-Pil Choi. Author-topic over time (atot): a dynamic users' interest model. In *Mobile, Ubiquitous, and Intelligent Computing*, pages 239–245. Springer, 2014.

[31] Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, SVN Vishwanathan, and Inderjit S Dhillon. A scalable asynchronous distributed algorithm for topic modeling. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1340–1350. ACM, 2015.