STATISTICAL OPTIMIZATION OF TRAINING DATA FOR SEMI-SUPERVISED TEXT DOCUMENT CLUSTERING

By

Cody Newbold

A thesis submitted in partial fulfillment

of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Middle Tennessee State University

August 2017

Committee: Dr. Joshua L. Phillips, Dr. Chrisila Pettey, Dr. Cen Li

ABSTRACT

Unsupervised machine learning algorithms suffer from uncertainty that results are accurate or useful. In particular, text document clustering algorithms such as Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) give no guarantee that documents are clustered in a manner similar to human readers. Using a semi-supervised approach on text document clustering, we show that the selection of training data can be statistically optimized using LDA and LSA. Using this method, a human reader categorizes a percentage of the data as an analysis step, then feeds the partially-labeled data into bootstrap training and testing steps. Using mutual information to discover which documents were better for training, the algorithm does a post-processing step using the optimized training set. The results show that mutual information values are higher when the statistically optimized training set is used and indicate that human-like performance is better achieved with optimized training data.

TABLE OF CONTENTS

LIST OF FIGURES iv

LIST OF SYMBOLS AND ABBREVIATIONS vi

Chapter

I. INTRODUCTION 1

II. BACKGROUND 4

Topic Modeling 4

The Afghan War Diary 5

Studies on the Afghan War Diary 6

Overview of Latent Dirichlet Allocation 9

Overview of Latent Semantic Analysis 10

Motivation to Compare LDA and LSA 10

III. METHODS 12

Reproducing the LDA and Model Tree Study 12

Curating the Dataset 14

Rewriting the LDA Code 15

Implementing Latent Semantic Analysis 27

Comparing the Two Algorithms 31

IV. RESULTS 33

V. DISCUSSION 40

BIBLIOGRAPHY 43

LIST OF FIGURES

Figure 1 – Overview of the topic modeling approach 4

Figure 2 – A sample report summary from the AWD 6

Figure 3 – A sample text summary from the AWD represented as a word cloud 8

Figure 4 – Sample LDA output, including the terms returned from a single run 10

Figure 5 – Model tree for the combined fatalities... 14

Figure 6 – Percentage breakdown of human-labeled topic assignments 15

Figure 7 – The cluster_sort() function in R 17

Figure 8 – LDA cluster assignment joint probability distribution function................. 18

Figure 9 – LDA cluster assignment joint probability distribution function................. 19

Figure 10 – LDA cluster assignment joint probability distribution function............... 19

Figure 11 – The alpha values of LDA on 100 documents of the AWD 21

Figure 12 – A diagram of the data structure used to contain the MI values 23

Figure 13 – A portion of the MI vector... 23

Figure 14 – LDA mutual information box plots for 50 testing documents 25

Figure 15 – Fold_in() function overview.. 28

Figure 16 – An excerpt of LSA output from one run of the algorithm......................... 29

Figure 17 – Program logic for calculating mutual information medians...................... 31

Figure 18 – Line plot of LDA and LSA median mutual information values................ 34

Figure 19 – Improvement between statistical analysis phase 36

Figure 20 – Line plot of LDA and LSA median mutual information values................ 37

Figure 21 – Improvement between statistical analysis phase 38

Figure 22 – LDA and LSA partial labeling box plots..39

LIST OF SYMBOLS AND ABBREVIATIONS

AWD – Afghan War Diary

JDM – Joint Distribution Matrix

LDA – Latent Dirichlet Allocation

LSA – Latent Semantic Analysis

MI – Mutual Information

PPM – Point Process Modeling

SVD – Singular Value Decomposition

CHAPTER I

INTRODUCTION

In the Information Age, it has become difficult for many fields of study to adequately process the large amounts of text data that exist [19, 5]. Specifically, journalists often have trouble when receiving huge datasets that contain unknown information to be able to report on the topics contained in them. Most people find information they need either through searching the internet using keywords or by clicking on relevant links [19]. Data is expanding in such a way that technology needs to enable everyone to find information in a smarter way. While there are some tools in existence that allow us to cluster and tag documents for analyses, the methods can be improved [20].

No clear solution has emerged that is best for categorizing a large, unsorted volume of text documents. Much research has been done on topic modeling, and while methods exist, they are not mainstream [12]. Topic modeling is difficult to accomplish and must be tailored for each dataset, since it is important that the results be comprehensible to humans [14]. There are also factors that must be adjusted, such as model parameters and training data size, to make it work properly [15]. Building on two methods that have been previously studied, we will focus on improving the ways to model text and extract data from large datasets.

The Afghan War Diary (AWD) serves as a prime example of a dataset chosen for research, due to journalists' inability to gain meaning out of the 77,000 text reports contained in it, in addition to its being leveraged by past research from others [6, 21]. The AWD is important because although it was processed by many professional data journalists, much information has yet to be extracted from it. Many of the analyses done on the dataset by journalists were descriptive in nature, whereas the methods utilized by computer scientists and statisticians obtained predictive results. Research has been done on the AWD dataset using Latent Dirichlet Allocation (LDA) and point process modeling (PPM), as well as other algorithms [6, 21].

Latent Semantic Analysis (LSA) is another algorithm that has popularity among those seeking to gain a better understanding of unorganized text. LSA was first discussed in a paper by Landauer et al. in 1998 and has been used in a multitude of studies surrounding text analysis [1, 3, 2, 14]. For example, LSA was used to provide emotional context of summaries from TV shows to generate user preferences [3]. Another study used LSA to provide a solution for identifying spam with unsupervised algorithms [2]. In a study done by Chang et al. that compared human performance to text modeling algorithms, LSA is mentioned as one of the first topic modeling methods that attempted to duplicate human selection when analyzing text [14].

While results from previous studies yielded a set of topics for the AWD dataset, the unsupervised nature of LDA and LSA have introduced significant uncertainty on whether the topic models produced were reflective of topic models humans might produce. In most cases, similarity to human-like topic modeling would be preferred, but efforts to better achieve human-like topic modeling of the AWD using the above approaches have yet to be implemented.

In this study, we aim to improve the performance of unsupervised algorithms on topic modeling of the AWD dataset by using a bootstrap statistical augmentation of the LDA and LSA methods. This statistical analysis is applied to the topic clustering results from the two methods using manually labeled data. Grooming the LDA and LSA methods using human-labeled data creates a process by which unsupervised algorithms can be statistically molded to "think" more like humans. This approach is not distinct to one method or dataset, but can be applied to any system that can be curated by hand. Executing this semi-supervised technique on uncategorized datasets has potential to shed much light on previously unexplored raw text.

CHAPTER II

BACKGROUND

Topic Modeling

Topic modeling is a way of distributing topics over documents to obtain a collection of documents grouped into topics. Figure 1 shows how topic modeling is done in several steps.

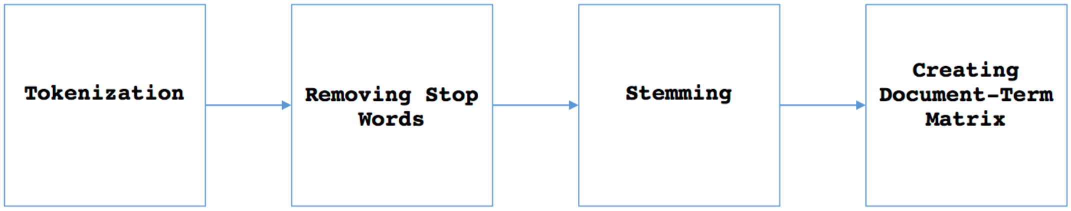


Figure 1. Overview of the topic modeling approach.

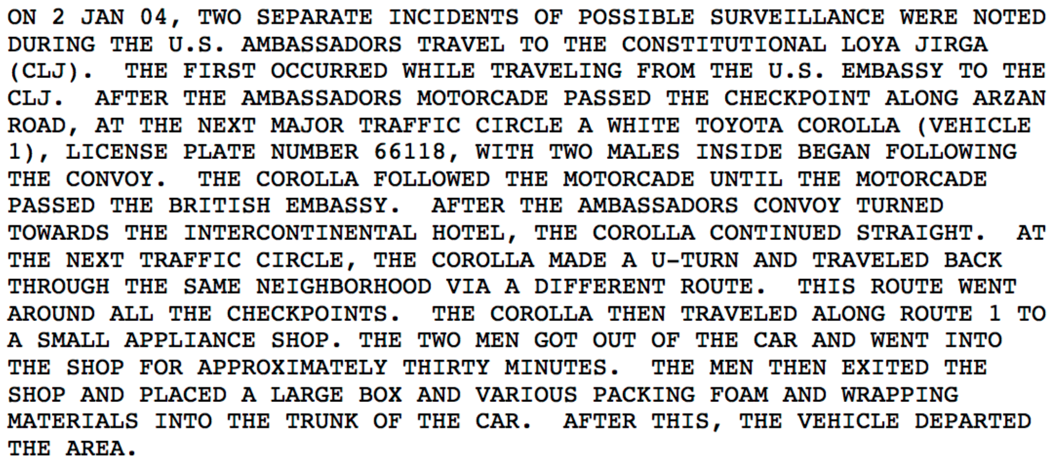
Tokenization segments text into a word array, typically breaking on punctuation or whitespace characters. Removing stop words is done next, which removes words like "for" and "the", which do not contribute to the overall meaning of the document. Stemming, which is the next step, is the process of reducing similar words to their roots. For example, "planning", "planner", and "planned" would all reduce to "plan". This allows multiple, semantically-related word forms to be treated equally regardless of how they are used in the language. Finally, a document-term matrix is created, which maps each word to how frequently it occurred in the text. Topic modeling algorithms usually differ in how they subsequently process the document-term matrix.

Output from topic modeling algorithms can vary. With LDA, a list of terms by topic is returned, along with a probability of topic membership for each document. LSA produces a matrix of documents and terms, where correlated terms are in rows and documents in columns. Each document is therefore quantified as a vector in a space of correlated terms, allowing for ease of clustering into topics using a variety of statistical clustering techniques. The output obtained in this study was a set of documents clustered into topics, much like the output of data clustering, but using raw text instead of quantitative data as input.

The Afghan War Diary

The AWD is a set of nearly 77,000 documents detailing aspects of the Afghanistan War that were previously unknown to the public, and this dataset has been too large to be more than descriptively useful to journalists so far. The AWD was made public on July 25, 2010, but was sent to the New York Times, the Guardian, and Der Spiegel prior to it being released to the public. The dataset spans the years 2004 to 2009, and contains time, position, and a brief summary of each event, among other fields. The documents are one of the most complete looks at the Afghanistan War that is available to the public [17]. Since the documents are such a massive set of data, many journalists, including the three newspapers to which the dataset was originally sent, were not able to adequately sort and analyze them. Although several news sources did analyze the documents, much of the text data in specific was left uninvestigated, and it has been left to computer scientists to do much of the processing of the text [21].

Figure 2 shows a sample report summary included in the AWD. This report discusses a suspicious incident that occurred on a certain date, describing how a white car followed an ambassador's motorcade. The incident seemingly did not result in any action taken by the military. Someone wishing to get information on all the incidents of surveillance during this time period would need to rely on a keyword search of the AWD reports. Keyword searches, while valuable, require that a human know the exact words, or at the very least synonyms, of the words they would like to search, as well as an evaluation of the returned documents after the keyword search has taken place. Therefore, for this type of large dataset, artificial intelligence becomes crucial for adequate analysis of the text.



ON 2 JAN 04, TWO SEPARATE INCIDENTS OF POSSIBLE SURVEILLANCE WERE NOTED DURING THE U.S. AMBASSADORS TRAVEL TO THE CONSTITUTIONAL LOYA JIRGA (CLJ). THE FIRST OCCURRED WHILE TRAVELING FROM THE U.S. EMBASSY TO THE CLJ. AFTER THE AMBASSADORS MOTORCADE PASSED THE CHECKPOINT ALONG ARZAN ROAD, AT THE NEXT MAJOR TRAFFIC CIRCLE A WHITE TOYOTA COROLLA (VEHICLE 1), LICENSE PLATE NUMBER 66118, WITH TWO MALES INSIDE BEGAN FOLLOWING THE CONVOY. THE COROLLA FOLLOWED THE MOTORCADE UNTIL THE MOTORCADE PASSED THE BRITISH EMBASSY. AFTER THE AMBASSADORS CONVOY TURNED TOWARDS THE INTERCONTINENTAL HOTEL, THE COROLLA CONTINUED STRAIGHT. AT THE NEXT TRAFFIC CIRCLE, THE COROLLA MADE A U-TURN AND TRAVELED BACK THROUGH THE SAME NEIGHBORHOOD VIA A DIFFERENT ROUTE. THIS ROUTE WENT AROUND ALL THE CHECKPOINTS. THE COROLLA THEN TRAVELED ALONG ROUTE 1 TO A SMALL APPLIANCE SHOP. THE TWO MEN GOT OUT OF THE CAR AND WENT INTO THE SHOP FOR APPROXIMATELY THIRTY MINUTES. THE MEN THEN EXITED THE SHOP AND PLACED A LARGE BOX AND VARIOUS PACKING FOAM AND WRAPPING MATERIALS INTO THE TRUNK OF THE CAR. AFTER THIS, THE VEHICLE DEPARTED THE AREA.

Figure 2. A sample report summary from the AWD.

Studies on the Afghan War Diary

A few studies have attempted to use the data in the AWD to predict the outcome of incidents that occurred during the Afghanistan War, such as looking at fatalities and intensity of conflict [16, 21, 17]. These methods succeeded in being useful to not just describe what was in the documents, but to gain more knowledge than was included in the AWD metadata.

LDA and Model Trees

The LDA and model tree approach described by Rusch et al. was able to show that fatality rates per incident could be obtained from just the text portion of the documents [21]. Rusch et al. demonstrated that the AWD text can be modeled into topics using LDA and model trees [21]. LDA is a process that was presented by Blei et al. in 2012 [18]. Although LDA is not limited to text applications, it is very useful for them [18, 9, 5, 7]. LDA is a specific version of the broader study of topic modeling, of which there are many varieties [4, 7].

Many studies have been done on LDA, including topic modeling solutions that include searching for keywords after LDA preprocessing [11]. Some research used LDA for automatically tagging short texts [12, 6, 8]. There was even a study done on using LDA for the prevention of data leaks [10].

In the LDA and model tree approach, the results of the text modeling steps were then used to create a model tree that contained partitioned topics and the frequency of words used in them. Figure 3 shows a word cloud that very simply demonstrates how the sample report summary from Figure 2 might be split into more and less used words.

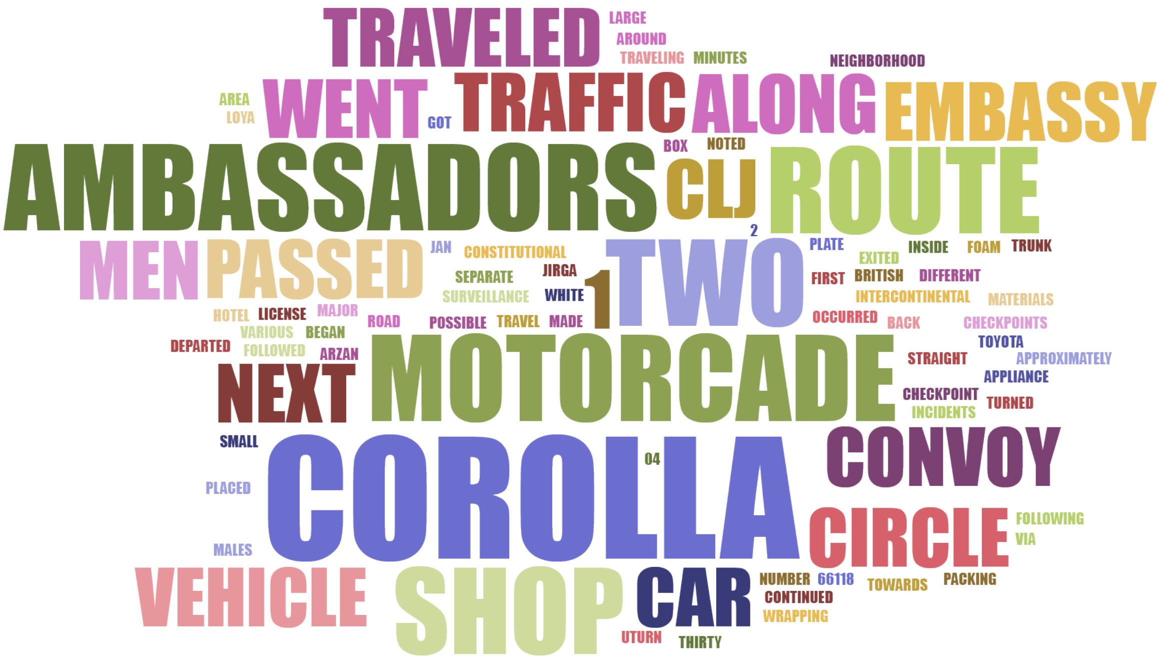


Figure 3. A sample text summary from the AWD represented as a word cloud, demonstrating the LDA method of finding most-used words.

Point Process Modeling

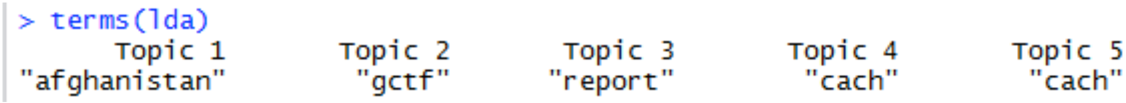
Point process modeling was used to show that the intensity of conflicts could be predicted using spatial and temporal data [17]. Zammit-Mangion et al. demonstrated that by looking at data from 2004 to 2009, they could predict conflicts and intensity of conflicts for a year after that [17]. Rusch et al. suggested combining the LDA and model tree approach with the point-process modeling approach using spatial and temporal data as the node model and partitioning the data based on the generated topics [21].

Continuing to study these documents may give further insight into the value of the tools used for text modeling and statistical analysis. For example, combining the two methods could potentially improve the accuracy of the PPM model by leveraging the quantitative summaries provided by LDA.

Overview of Latent Dirichlet Allocation

Latent Dirichlet Allocation, or LDA, is a probabilistic model. Blei describes a topic to be “a distribution over a fixed vocabulary” [19]. The method works in a few steps [18]. First, a number of topics must be supplied to the algorithm. Then, a set of topics is randomly spread over the document set using a Dirichlet distribution. After that, for each word in each document, a topic is chosen from the first step, and a word is chosen from the initial distribution [18]. This step is iterative, and the topic structure is gradually improved as the algorithm continues over multiple runs. In this way, the topics contained in the document set are discovered without any prior knowledge of what the document contained. “The central computational problem for topic modeling is to use the observed documents to infer the hidden topic structure,” describes the study by Blei et al. [19]. That is the main idea behind the LDA method, to gather a hidden topic structure. This can be considered opposite to a keyword search, where the topic structure is considered known and the documents need to be produced.

Figure 4 shows a sample run of the code used for the LDA approach described in this study, where LDA attempts to select topic labels for each document set. The LDA terms drawn from each topic are listed under the topic number. In this particular set of documents, there was a high volume of documents that discussed the military having discovered caches of weapons or ammunition (e.g. topics 4 and 5). In most of the runs of the LDA code on this document subset, there are multiple topics with the term “cach” as the main term, which is most likely due to that word appearing in so many of the report summaries.



```
> terms(lda)
     Topic 1       Topic 2       Topic 3       Topic 4       Topic 5
"afghanistan"       "gctf"      "report"        "cach"        "cach"
```

Figure 4. Sample LDA output, including the terms returned from a single run.

Overview of Latent Semantic Analysis

Latent Semantic Analysis (LSA) is another topic modeling approach. LSA combines a vector space model with singular value decomposition (SVD). In LSA, tokenization, stop word removal, and stemming are all done prior to creating the document-term matrix, where the text is converted into a text matrix, M, which is *m x n* in size. SVD is then applied to the text matrix, M. The SVD can be calculated by:

$$M = U\sum V^T$$

U is an *m x n* matrix, ∑ is an *m x n* matrix and V is an *n x n* matrix. U and V are unitary matrices and ∑ is a diagonal *m x n* matrix containing the singular values of M.

In the code used to run the LSA method in this study, k-means was used to cluster the documents from the *LSASpace*, the first k column vectors of the U matrix, into topics. This provided us with a list of document numbers corresponding to the topic in which they were placed, allowing us to quantitatively compare the LSA results with the LDA results.

Motivation to Compare LDA and LSA

Very few text analysis solutions have been used on the AWD in order to extract information from the report summaries alone. In this research, we compared LDA and

LSA using a bootstrap statistical method to improve them. Although the LDA and LSA methods have been used separately on other datasets, they have yet to be compared against each other while being tested on the AWD. Therefore, using these particular topic modeling algorithms on the AWD dataset is a useful exercise. LDA was selected due to its leverage in other studies on the AWD, in addition to its being a popular topic modeling algorithm [21]. LSA was selected due to its having been tested in past studies against human selection of topics [14].

Additionally, none of the previous studies on the AWD discussed ways in which LDA and LSA might be modified to gain larger insight into what lies in the 77,000 report summaries. Quantitatively contrasting LDA and LSA utilizing a statistical measurement should provide more depth and detail than what researchers uncovered in the past.

CHAPTER III

METHODS

The research presented here had five primary steps. The first was to reproduce the LDA and model tree approach on the AWD as described by Rusch et al. [21]. The second was to write LDA code that clustered 100 pre-labeled documents into five topics. Thirdly, an LSA approach was written that created an *LSASpace* and clustered the 100 documents into five topics using k-means. The fourth step in the process was to compare the two algorithms by statistically quantifying the topics returned from each one and performing post-processing with machine-selected training values. After those four goals were met, further analysis was done. The fifth step was to run both algorithms again using only 10% of the labeled data and discarding the other 90% of the labels. Using this partial labeling technique, it was hoped that the algorithms would still produce a training set for post-processing that would improve the performance of LDA and LSA.

Reproducing the LDA and Model Tree Study

We reproduced the LDA and model tree approach using source code obtained from the authors of the Rusch et al. study [21]. The code contained three phases and was written in the R programming language. This code needed several minor edits due to updates to the R language since the time it was written.

The three phases can be summarized as follows. Phase 1 completed the LDA and created a document-term matrix with the raw dataset. Then, the rows in the AWD that did not contain a summary were removed. Phase 2 removed the word “report” out of each report, since all the summaries contained this word. Then, 100 topics were derived from the dataset, and the 30 most frequent terms from each topic were stored. Phase 3 cleaned the data and stored it to a file. Then, the analysis of the data was done using a negative binomial S4 statistics model. Results were plotted into a tree based on the latent topics.

The source code to create the tree was also written by the authors of the Rusch et al. study [21]. The tree combined the topics discovered by LDA using a negative binomial model and helped to predict the number of fatalities in the AWD by obtaining the ten most frequent terms from the topics and the number of documents that were assigned to them. This approach provided results that predicted the fatalities that occurred during the span of the dataset.

Unfortunately, we were not able to produce the model tree from the raw AWD dataset, even though we obtained the source code from the authors. Final data had been saved at some point and allowed the tree to be created, but we were not able to produce it by running their full analysis. This was possibly due to the age of the code and the improvements made to the R language, but it appeared there were missing pieces of the data and incomplete code fragments in the materials and source code provided by the authors. Therefore, a full rewrite of the code was necessary to perform an LDA analysis on the AWD. The model tree from the Rusch et al. study is shown in Figure 5.

The deficiency of the Rusch et al. code led to a manual curation of a 100-document subset of the AWD. This was done to validate the performance of LDA and LSA on the data.

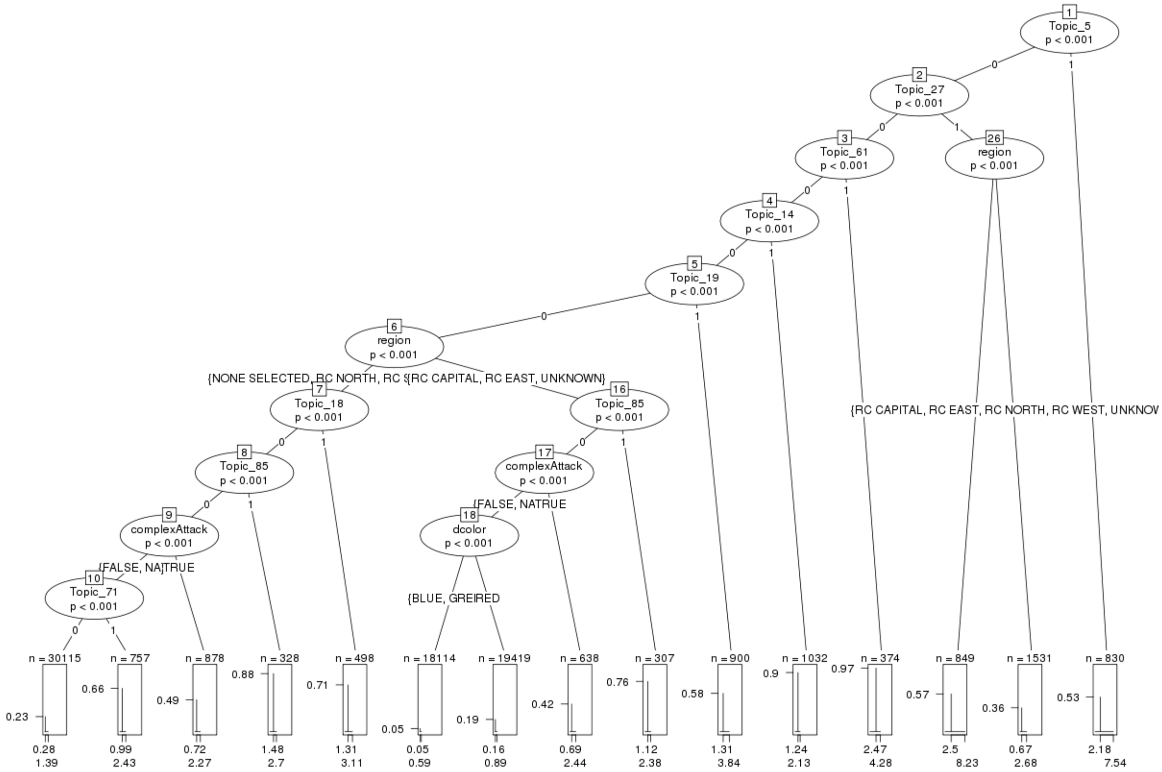


Figure 5. Model tree for the combined fatalities, created using a negative binomial model. This figure was recreated for this study with pre-processed data and code provided by Rusch et al. [21].

Curating the Dataset

To analyze how well the LDA and LSA methods were sorting the dataset, a random 100-document subset of the Afghan War Diary was first extracted and sorted into topics by hand for comparison. Figure 6 shows a breakdown of the percentages of human labeled topics in a pie chart. We see that 54% of documents fell into one topic and 32% into another.

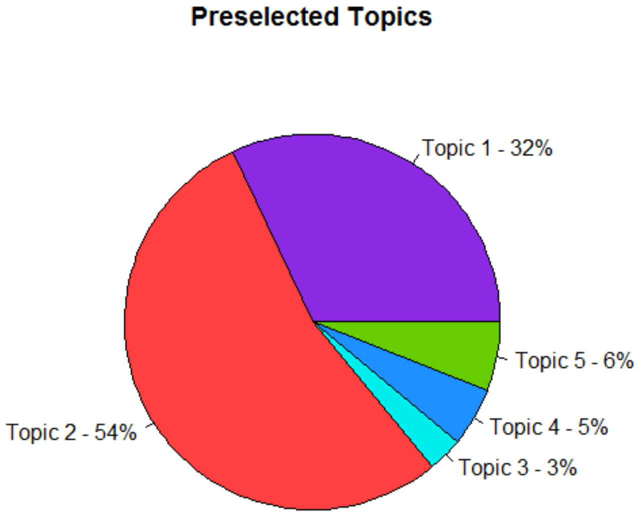


Figure 6. Percentage breakdown of human-labeled topic assignments.

The human labeled topics can be summarized as follows: Topic 1 – Direct/Indirect Fire, Topic 2 – Cache/Mine/IED Found, Topic 3 – Explosion/Hostile Action, Topic 4 – Surveillance/Reconnaissance, and Topic 5 – Propaganda. This curated subset was used as the trusted data against which the remainder of the analysis was tested. LDA and LSA were compared against the topics chosen by hand to review how accurately each algorithm was categorizing the report summaries.

Rewriting the LDA Code

Beginning with a simple LDA example found online and code fragments from the Rusch et al. study, the following LDA method was used for the initial test run of the algorithm [21, 22]. Like the original source code from the Rusch et al. paper, the code for this research was written in R. The code for all analyses included in this paper is publicly available online at https://github.com/varali/lda-lsa/.

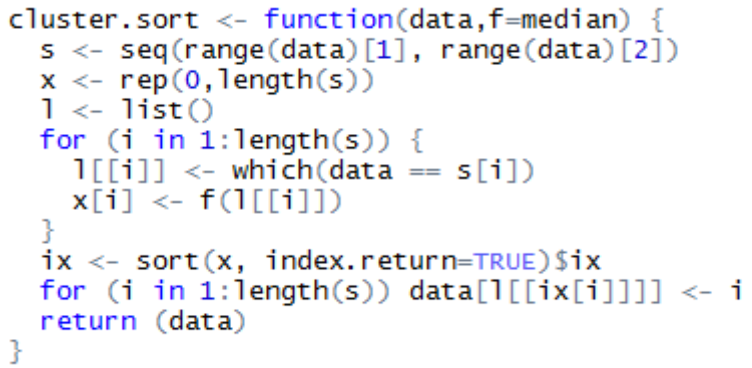
The documents were read in from a CSV file, which was drawn from the original AWD dataset. The hand-curated labels were contained in a separate file of two columns: a document index that corresponded to each of the 100 documents and a topic number from one to five. Then, the documents were mapped into topics using LDA.

Joint Distribution Matrices and Mutual Information

A heat map plot was created for each run of the program in the initial LDA testing phase, which showed the confusion of the algorithm between different topics. The plot was generated from a joint probability distribution, which was a matrix of probabilities that correlated the machine-selected topic with the human-selected one. The y-axis of this plot corresponded to the expected topic (human-selected) and the x-axis corresponded to the topic in which the algorithm placed the document. The grid was initialized to all zeroes, and was incremented for each testing value in the set, at the x-value for the machine-selected topic number and the y-value for the human-selected topic number. For instance, if the human-selected topic was two and the machine placed the document in topic four, the grid was incremented at the location (4, 2). When all values in the set had been incremented in their proper locations, the entire matrix was divided by the total number of testing values, creating a joint probability matrix.

The topics were compared with the original human-labeled topics by passing them through a function called *cluster_sort()* [23]. This function used the median value of the indices of the topics to make sure that a difference in the arbitrary topic labels (i.e.

1, 2, 3, 4, 5) per run would not constitute a difference in the logical topic in which the document was placed. The cluster sort function is shown in Figure 7.



```
cluster.sort <- function(data,f=median) {
  s <- seq(range(data)[1], range(data)[2])
  x <- rep(0,length(s))
  l <- list()
  for (i in 1:length(s)) {
    l[[i]] <- which(data == s[i])
    x[i] <- f(l[[i]])
  }
  ix <- sort(x, index.return=TRUE)$ix
  for (i in 1:length(s)) data[l[[ix[i]]]] <- i
  return (data)
}
```

Figure 7. The *cluster_sort()* function in R. From Phillips, Colvin, and Newsam, 2011 [23].

The plots were constructed in heat map form, showing by color where exactly LDA differed from the human-labeled choices from topic to topic. Three plots are shown in Figures 8-10, one containing a low mutual information (MI) value, one a mid-range value, and one a high value. The colors in the heat maps demonstrate how high the probability was for all documents being placed in that topic, and the grid allows the machine-selected topic to be correlated with the expected topic. The legend on the right side of the graph shows how the probability values map to the colors. See Figures 8-10 for an example of this.

As for the MI values given at the top of each plot, those were calculated by summing the rows and columns of the joint probability distribution matrix, then computing the outer product of those vectors. The MI value is achieved by multiplying the probabilities from the joint distribution matrix times the $\log_2$ of the joint distribution matrix divided by the outer product of the marginal probability vector. The MI value measures the mutual dependence between the human-labeled documents and the machine-labeled ones. The higher the MI value, the better the machine-selected topics matched the human-selected topics. Mutual information can be calculated by the following equation, where *p(x, y)* is the joint probability distribution and *p(x)* and *p(y)* are the marginal probabilities.

$$I(X;Y) = \sum_{y \in Y} \sum_{x \varepsilon X} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

In Figures 8-10, a perfect match between the machine- and human-labeled choices would be that all high-value color blocks fall on the upward diagonal of the chart, where topic one meets one, two meets two, etc. This scenario would also produce the highest MI.

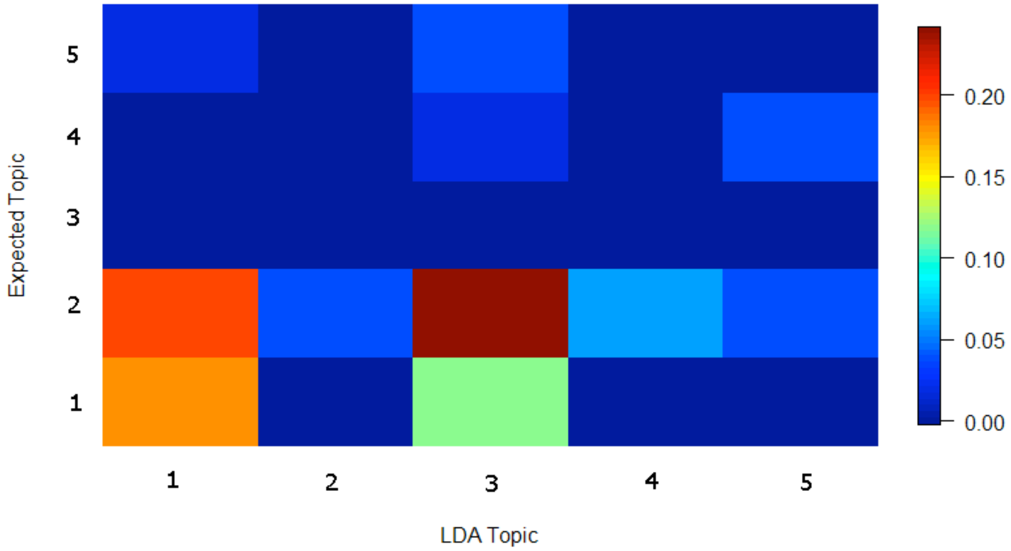


Figure 8. LDA cluster assignment joint probability distribution function for a mutual information value of 0.255979.

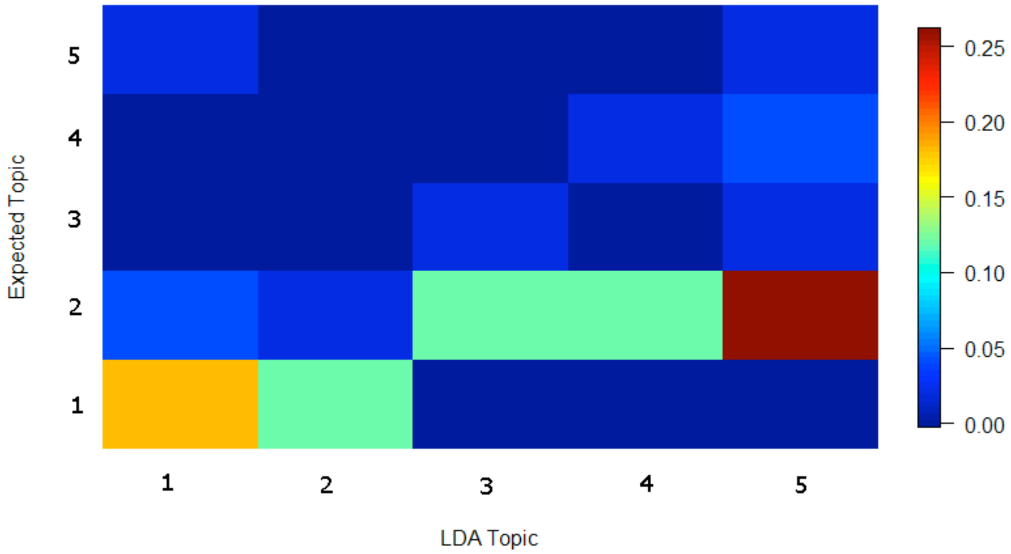


Figure 9. LDA cluster assignment joint probability distribution function for a mutual information value of 0.718642.

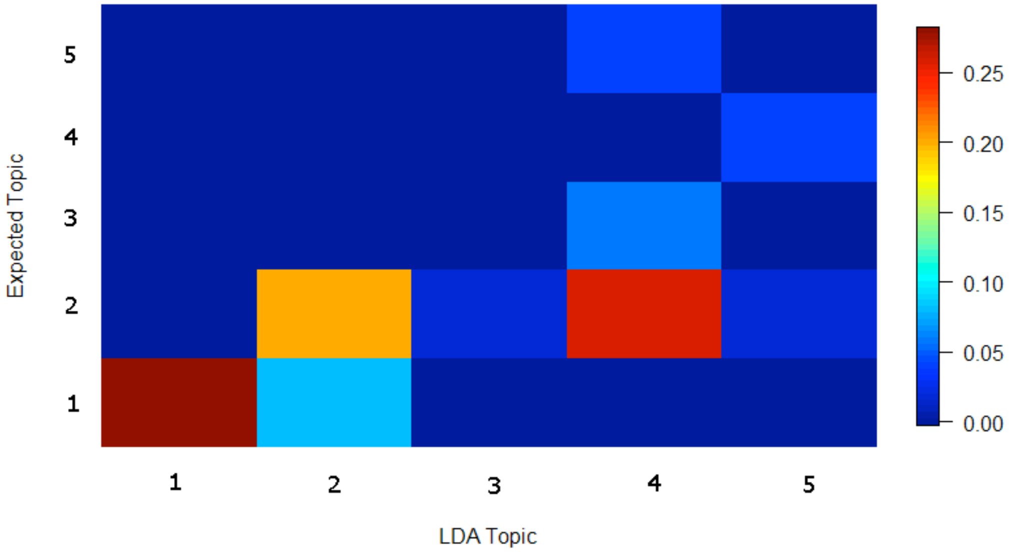


Figure 10. LDA cluster assignment joint probability distribution function for a mutual information value of 0.944925.

For the plot in Figure 8 with the mutual information value of 0.26, the LDA algorithm aligned with the expected topic decently well for topic one. For the other topics, however, LDA misclassified nearly all documents, and did not classify anything correctly for topics three, four, and five. For topic three in particular, LDA put all of them in topic two. Looking back at the pie chart of manually labeled data (Figure 6), we see that topics three, four and five comprise only 3%, 5%, and 6% respectively, so it could be that these topics are data starved, which is why the algorithm did not do well in classifying them.

For the plot in Figure 9 with the mutual information of 0.72, LDA did better, as evidenced by the upward diagonal of blue boxes. In this run, it put some of the right documents into each topic, but it also did not accurately classify most documents that should have been in topic two.

For the plot in Figure 10 with the highest mutual information value in this set, the value of 0.94, the algorithm was quite accurate in classifying the documents in topic one and some of the documents in topic two, but did not do very well in the other topics. LDA continued to do badly when it came to putting documents into other topics when they should have been classified in topic two.

As evidenced above, the confusion matrix provided a lot more information about how LDA was classifying the documents. With better mutual information values, the image plots tended to look cleaner and have simpler explanations on what went wrong when the algorithm was run. Mutual information values were used to map how well the algorithm was doing, while the heat map plots visualize how confused the algorithm was against the curated topics.

Calibrating the LDA Algorithm

After the LDA code was written, the alpha value in the LDA function was adjusted and the mutual information value for each alpha was plotted. The alpha value is a free parameter that controls the smoothness of the topic-term distributions, and must be fit based on performance criteria independently for every dataset. It was shown that for this dataset, the MI value peaked around the alpha values 5 and 50. A plot is shown below of the alpha values at each exponential step of 10, where the values are peaking at 5 and 50. Near the 5 and 50 marks, where the algorithm did better, additional values of alpha were sampled to see if there were better MI values to be gained near those alphas. We chose to leave the alpha value at 50, since that seemed to be the optimal value for the AWD dataset.

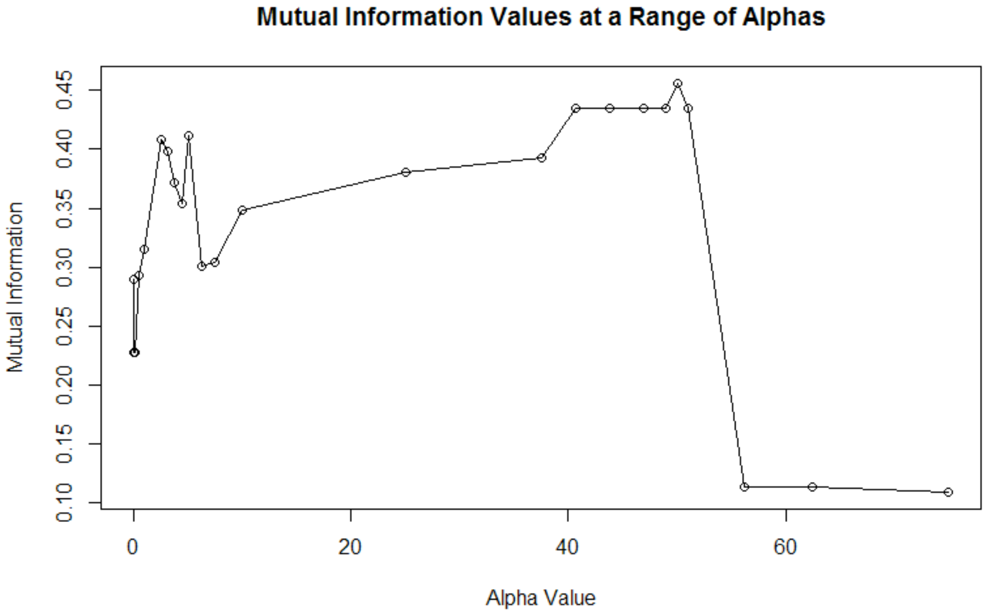


Figure 11. The alpha values of LDA on 100 documents of the AWD during the initial study of the LDA algorithm.

Training Set Optimization

Since so few of the AWD reports were used (100 out of 77,000), the algorithm was run 10,000 times to obtain a median MI value for each document. Running the algorithm so many times helped to make certain the median value would be accurate for all documents since training/testing sets were chosen from a random sample each time.

For each run of the algorithm, a sample of 50 random documents was taken to be used for training, while the rest were used in the test set. In the following example, 50 training documents and 50 testing documents were used. The R *posterior()* function was used to predict the topics for the test set after the algorithm was trained using the *lda()* function. This function calculated the probability that a test document should be in a certain topic given the prior data that was used for training.

Every time a document was used in the test set, its MI value was calculated and appended to the end of a vector of 100 lists. For this training/testing slice of 50/50, at the program's end, there was a list of approximately 5,000 LDA MI values for each AWD report in the subset. This information was recorded so that the results would show the median MI value for each document, as well as a distribution of the MI values for each. Figure 12 shows a diagram of the data structure created, and Figure 13 shows a code output example of part of the MI vector.

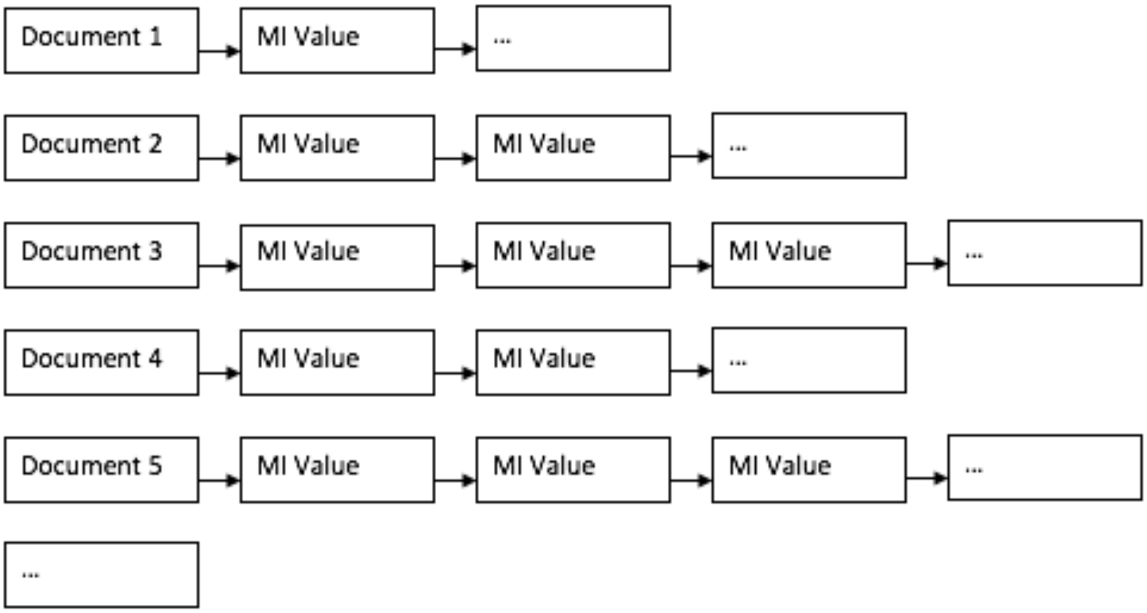
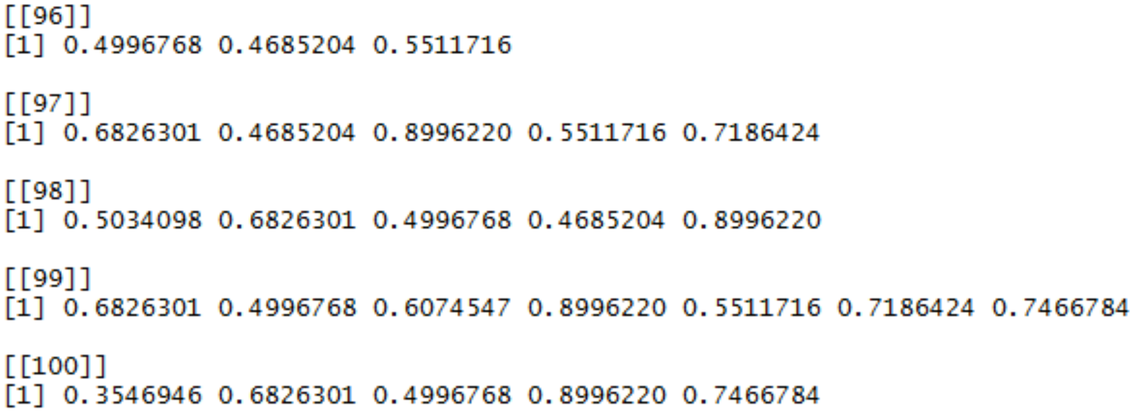


Figure 12. A diagram of the data structure used to contain the MI values for each document.



```
[[96]]
[1] 0.4996768 0.4685204 0.5511716

[[97]]
[1] 0.6826301 0.4685204 0.8996220 0.5511716 0.7186424

[[98]]
[1] 0.5034098 0.6826301 0.4996768 0.4685204 0.8996220

[[99]]
[1] 0.6826301 0.4996768 0.6074547 0.8996220 0.5511716 0.7186424 0.7466784

[[100]]
[1] 0.3546946 0.6826301 0.4996768 0.8996220 0.7466784
```

Figure 13. A portion of the MI vector that was used to obtain medians for each document in the AWD subset.

Once the data structure of each document's MI values was obtained, the median was computed for each document. This list of medians was sorted from greatest to least and the documents with the lowest median values were selected as the training set that performed the best. This was based on the logic that whenever those documents were not in the test set, the mutual information values were higher on average, therefore those documents must be better for training. These selected training documents were used in the post processing step to attempt to get a higher MI. Figure 14 contains box plots that show the distribution of medians across the 50 testing documents that were used. This figure provides a visualization of how a distribution of MI values across 10,000 runs might result in a median MI value that could be used to gauge how well the document performed for training.

The post processing step utilized the same code, but it only needed to be run once. Instead of taking a random sample of 50 documents for training, only the 50 documents with the lowest mutual information values were used.

The dataset was then sliced into different training and testing sections. The slices were 10 training, 90 testing; 25 training, 75 testing; 50 training, 50 testing, 75 training, 25 testing; and 90 training, 10 testing. For each slice, the pre-processing step of 10,000 runs was done, then the post-processing step was done, where the lowest mutual information values that came from the testing data were used for training in the post-processing step. In general, the number of training documents that were used in the statistical analysis step were also used in the post-processing step. For example, if ten training values were used in the initial statistical analysis step of 10,000 runs, the ten documents with the lowest medians were taken to be used in training for the post-processing step.

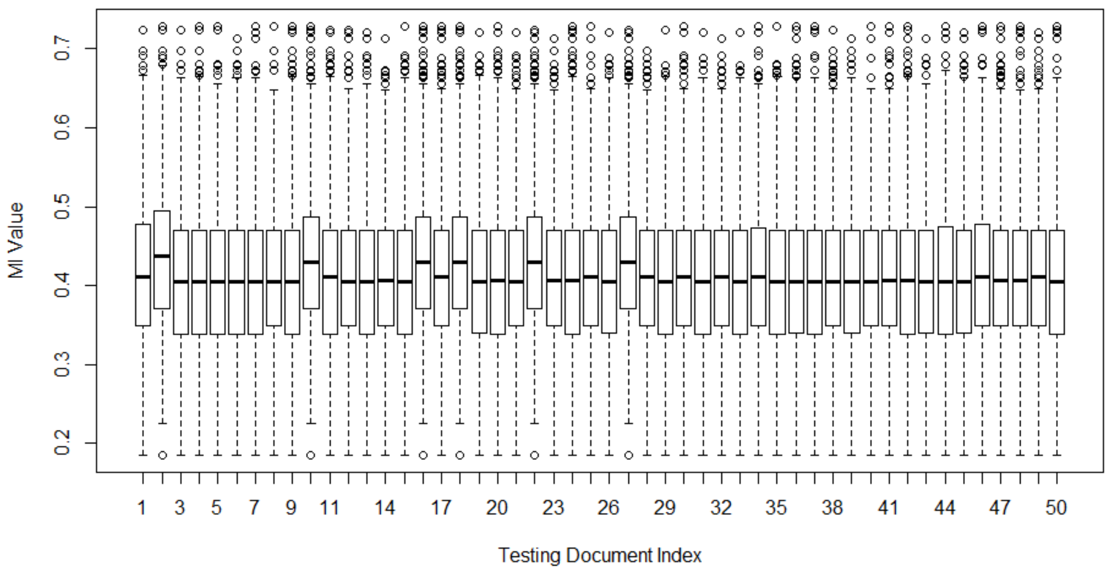


Figure 14. LDA mutual information boxplots for 50 testing documents.

LDA Partial Labeling

The semi-supervised step for LDA required the following changes to the original LDA statistical analysis code. Before the loop which ran through the algorithm 10,000 times, we drew a random "labeled" sample from the data. Even though all the data was technically labeled, the 10 random samples we drew were the only labels used to compute MI during each run. We took the sample before the looping began to simulate a real life experience, where in a large dataset, a human would desire to label as few documents as possible. Taking the sample at the beginning ensured that we only counted the labels for exactly ten percent of the data.

The next consideration was to make sure that there was an adequate number of labeled test documents. Since only 10 out of 100 documents were labeled and for certain training/testing slices only 10 testing documents selected, it was necessary to check that at least two of the documents in the resulting test set were "labeled". This was due to the fact that if there was only one testing document selected that was "labeled", it would either be in the correct topic or it would not be, completely skewing the MI value. When a run of the code produced a set where less than two of the testing documents were "labeled", we simply skipped this run and calculated a new random sample of training/testing documents.

Aside from considering the case where only one testing document was "labeled", the partial labeling step was nearly identical to the statistical analysis step from above. We ran the LDA algorithm 10,000 times, taking a joint distribution matrix plot and an MI value each time. The difference was in the fact that when the *cluster_sort()* function was run, we only considered the documents that were in the "labeled" set. If a testing document was analyzed, but not in the labeled set, we assigned the MI value that was computed for only the documents that were in the testing set and also marked as labeled. For this semi-supervised approach, it was important that we run the LDA algorithm 10,000 times to be certain that the median values evened out if there were any singular discrepancies in the MI values assigned to testing documents that were not in the labeled set.

Implementing Latent Semantic Analysis

Aside from the differences in the two algorithms, nearly the same path was followed as was used for LDA to get the LSA method results. First, an initial run of LSA was done to test the algorithm. Since LSA doesn't have any parameters (e.g. alpha in LDA) to adjust, the only parameters that were analyzed were the ones that affected the k-means function. For this, the seed was set to zero using *set.seed(0)* in R, then the number of starts was set to ten. The number of starts was set to ten because during initial testing, this was the lowest value where the same cluster assignments were produced for the same slice of training and testing documents.

The goal for LSA was to return the same type of output that was returned from the LDA algorithm, which was why k-means was used for clustering. This allowed us to get the text data returned in the *LSASpace* clustered into topics, with 50 random reports chosen for training each time and the rest of the documents being placed in the test set.

The R *fold_in()* function was used to calculate the topics for the test set after the *LSASpace* was created using the training set. Folding in is typically used in the LSA method in R to map additional documents (i.e. testing documents) into an existing *LSASpace* after SVD has been done on the training set.

Initially, the document-term matrix was created using all the documents and split into training and testing just prior to the *lsa()* step being done. This was to assure that all the terms from both the testing and training sets were mapped into the *LSASpace*. After splitting the document-term matrix into training and testing sets and running the *lsa()* function, *fold_in()* was used to apply the testing set to the SVD done on the training set.

The two spaces were then mapped back into the same *LSASpace* at the index where they originally fell in the document-term matrix. This method is illustrated on a high level in Figure 15.

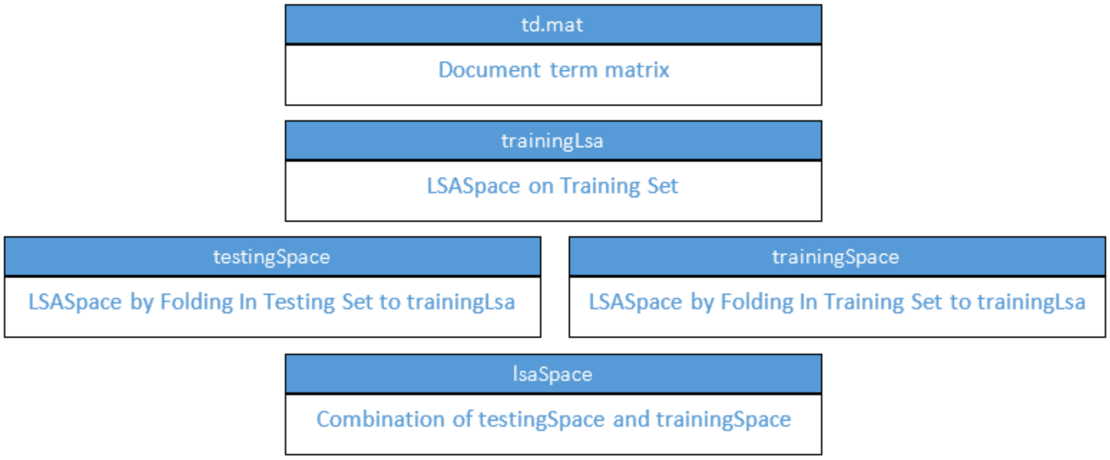
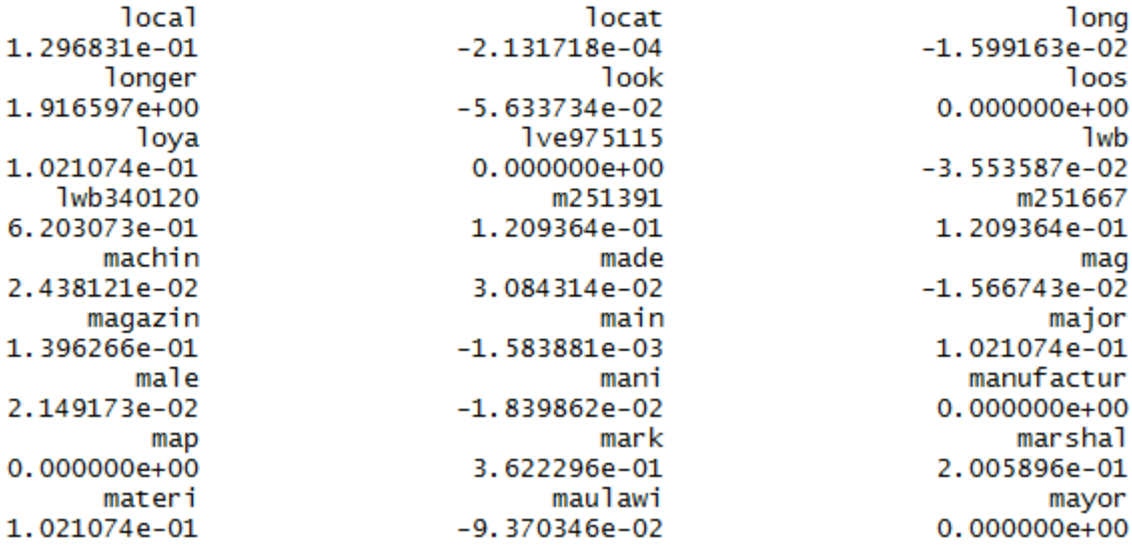

td.mat
Document term matrix
trainingLsa
LSASpace on Training Set
testingSpace
LSASpace by Folding In Testing Set to trainingLsa
trainingSpace
LSASpace by Folding In Training Set to trainingLsa
lsaSpace
Combination of testingSpace and trainingSpace

Figure 15. *Fold_in()* function overview.

Figure 16 shows a snippet of the output LSA produces. This snippet is a few rows of a single column, or document, returned after the *LSASpace* was created. The weight of each element in the figure below corresponds to the number of times it appeared in the document. What can also be seen in Figure 15 is the stemming of the words. For instance, the word “locat” below could mean that “locate”, “locating”, or “located” appeared in the *LSASpace*.



```
       local                  locat                   long
1.296831e-01          -2.131718e-04          -1.599163e-02
      longer                   look                   loos
1.916597e+00          -5.633734e-02           0.000000e+00
        loya              lve975115                    lwb
1.021074e-01           0.000000e+00          -3.553587e-02
   lwb340120                m251391                m251667
6.203073e-01           1.209364e-01           1.209364e-01
      machin                   made                    mag
2.438121e-02           3.084314e-02          -1.566743e-02
     magazin                   main                  major
1.396266e-01          -1.583881e-03           1.021074e-01
        male                   mani             manufactur
2.149173e-02          -1.839862e-02           0.000000e+00
         map                   mark                marshal
0.000000e+00           3.622296e-01           2.005896e-01
      materi                maulawi                  mayor
1.021074e-01          -9.370346e-02           0.000000e+00
```

Figure 16. An excerpt of LSA output from one run of the algorithm.

Then, a distance matrix was computed from a text matrix version of the *LSASpace*, and weighting was done using the R function *cmdscale()*. This function returned a points matrix of five columns, where the rows contain coordinates of the points representing similarities in the data. This matrix was put into the k-means clustering function, which returned a list of documents corresponding to the topics in which they were placed.

The topics were then put into the *cluster_sort()* function to ensure the topics would not vary solely due to different topic values assigned. For example, the topic numbers (1, 2, 3, 4, 5) would be assigned at random, so there was no guarantee the same logical topic would be assigned the same topic number each time. See Figure 7 for the code for *cluster_sort()*.

The joint distribution matrix plots for the LSA algorithm were created as described above for LDA (see Figures 8-10).

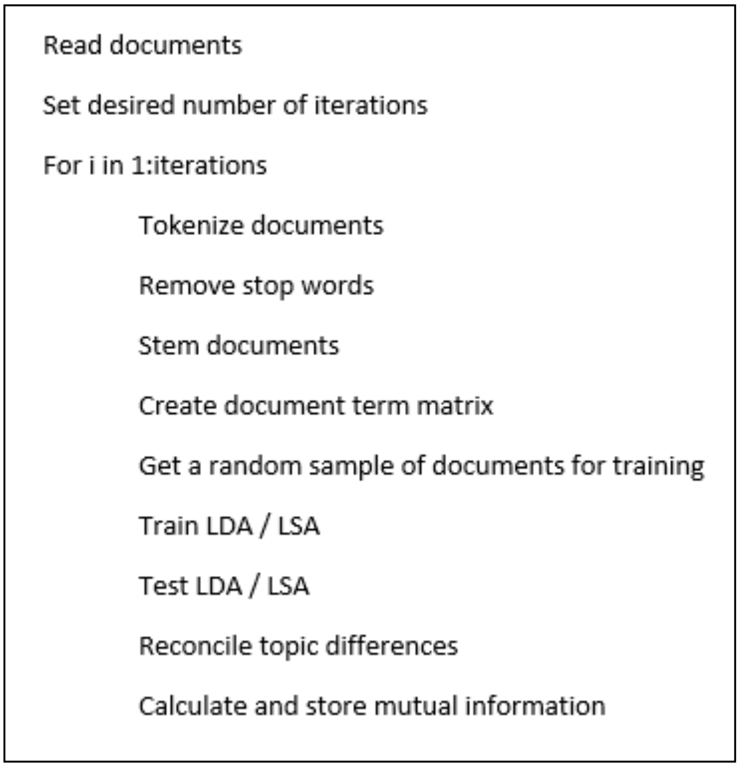
LSA Partial Labeling

When completing the LSA partial labeling code, it was necessary to ensure that the randomly selected testing data contained at least two instances of "labeled" data, identical to the method explained in the LDA section above. For LDA, the probability that the testing documents selected would be in different topics was high, since LDA tended to choose varied topic assignments. For LSA, however, this was not the case. LSA tended to group most of the documents into one topic, only assigning a few documents (usually 4 out of 100) to the other topics. Therefore, the testing documents that were considered "labeled" and fell into the randomly selected test set needed to be checked to make sure they were also in different topics. Frequently, all of the testing documents that were in the "labeled" set were all in the same topic assignment, so those were skipped and a new random selection was calculated.

In all other ways, the LSA partial labeling code mirrored the method that was done in the LDA partial labeling code. The MI values were calculated only from the human-labeled set that were included both in the randomly selected testing set and in the preselected labeled set that was done at program start. Whatever value of MI that was computed for those was applied to all items in the testing set. It was hoped that over 10,000 runs, enough MI values would be calculated for each document that the overall median would be accurate.

Comparing the Two Algorithms

Figure 17 contains the broad program logic that was used for both LDA and LSA. This was considered to be the statistical analysis step. The number of iterations was always set to 10,000.



```
Read documents
Set desired number of iterations
For i in 1:iterations
        Tokenize documents
        Remove stop words
        Stem documents
        Create document term matrix
        Get a random sample of documents for training
        Train LDA / LSA
        Test LDA / LSA
        Reconcile topic differences
        Calculate and store mutual information
```

Figure 17. Program logic for calculating mutual information medians for LDA and LSA.

As an overview, after the 10,000 runs of each algorithm were complete, the documents with lower mutual information values were considered to be better training documents, since the mutual information values were higher when they were used for training. With this information, a training set of the best documents were used to do one final run of each algorithm. The median value from this run was put over a boxplot of the 100 previous median values to gauge performance of the post processing step.

The two algorithms were simple to compare, due to the fact that a single mutual information value was returned for both. A higher mutual information value meant a better result when each algorithm was compared to the human-labeled data. In addition to viewing the MI value, each algorithm output a heat map plot for each run of the program, showing a five by five grid of the human-labeled data versus the machine-labeled data (see Figures 8-10).

CHAPTER IV

RESULTS

The method behind the approach used in this study yielded definitive results. For each run of the post-processing step for both LDA and LSA, the MI value was higher than in the statistical analysis step that preceded it. In the semi-supervised approach that followed, this held true as well. A short overview of the results follows.

The full method consisted of analysis on 100 randomly selected documents out of approximately 77,000 reports from the AWD. The 100 report subset was curated by hand, each document being placed into one of five topics. The statistical analysis step was done using all the labels, and median values of MI were gathered. The post-processing step proved that the MI value over 10,000 runs of the algorithm improved using training documents selected during the statistical analysis step.

The semi-supervised step used only ten percent of the human labels. Mutual information values were assigned by the algorithm to the testing documents, but only using the ten labels that were pre-selected randomly before the beginning of the program run. The post-processing step showed that the MI value rose using only ten percent of the human labels. Using two separate topic modeling algorithms, LDA and LSA, with the semi-supervised approach, we were able to show that having a human partially label ten percent of the AWD subset gained significant improvement in MI.

Figure 18 shows a line plot comparing LDA and LSA median MI values during the statistical analysis steps and the post-processing steps when all 100 of the labels were used. The plot shows that median MI values using the statistically selected training documents were always higher. The post-processing MI medians can be seen in the plot below by viewing the blue line for LDA and the green line for LSA.

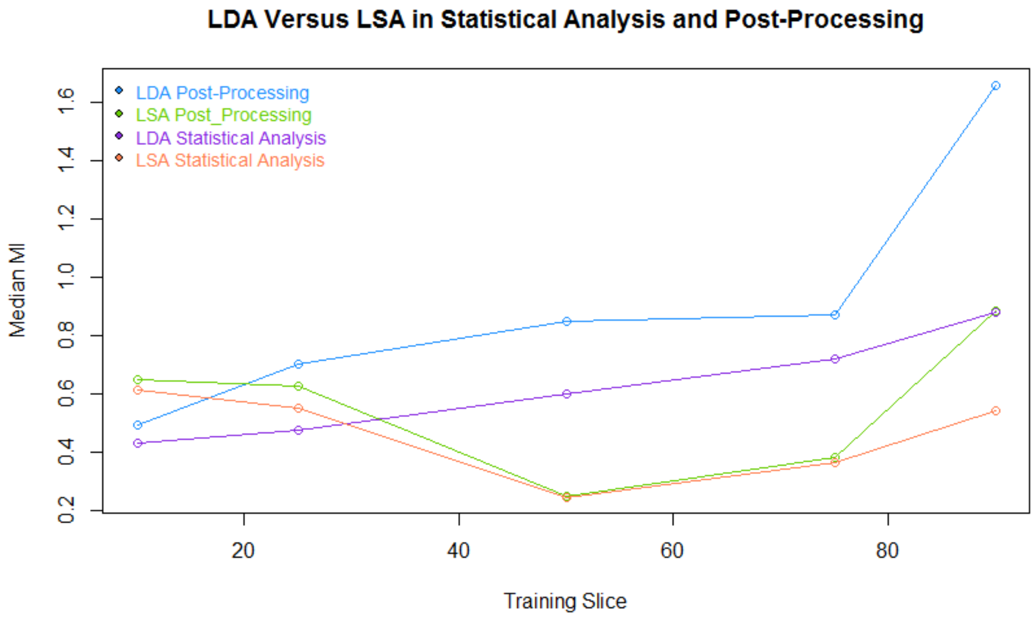


Figure 18. Line plot of LDA and LSA median mutual information values during statistical analysis and post-processing with training sets of 10, 25, 50, 75, and 90 documents.

Examining the median values on the line plots in Figure 18 yields the following conclusions. Both algorithms undoubtedly performed better when the 90-slice of training values was used. LDA in particular had very predictable results with regard to the median value rising as more training values were utilized. LSA had a dip between the 10 and 25 slices and the line chart resembles a U-curve. After passing the 50-slice of training data, the median for LSA began to rise again to peak at the 90 slice.

When compared to each other, we see that initially, LSA performed better than LDA, viewing the orange and green LSA lines in Figure 18. This is likely due to the fact that when examining the actual topic assignments LSA produced, we saw that LSA almost always grouped every item into one topic, only assigning a few documents into the remaining topics. When compared against the human labeled topic assignments, this aligns moderately well.

Looking back at the pie chart of hand-curated topics (see Figure 6), we recall that most of the documents fell into topics one and two. If LSA confused the wording in topics one and two, putting the two groups together, this is highly likely the reason that LSA achieved such a high mutual information value for the 10-slice training set.

LDA in particular had a high rate of change between the statistical analysis step and the post-processing step, suggesting it benefitted more greatly from the statistical bootstrapping than did LSA. Figure 19 shows the improvement in MI in the two steps for each algorithm.

Regardless of how LDA and LSA performed against one another, the most important result to consider is how they performed against themselves in the semi-supervised statistical analysis step versus the post-processing step. There was not a single case where the post-processing step did not outperform the statistical pre-analysis. This makes the case that selecting the best training data via quantitative mutual information values is a viable way to improve performance in these two algorithms.

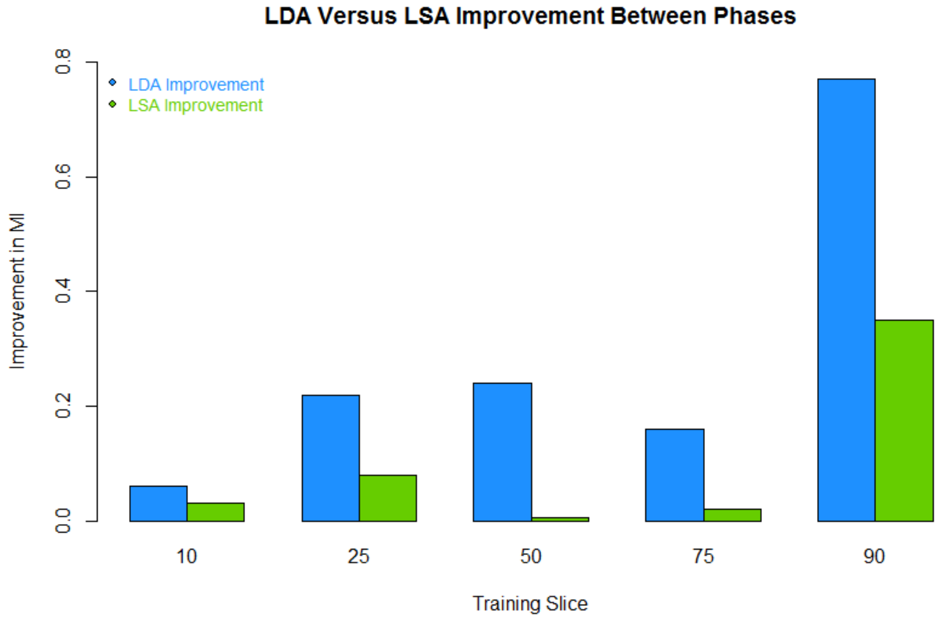


Figure 19. Improvement between statistical analysis phase and post-processing phase for LDA and LSA.

For the semi-supervised step, LDA and LSA individually performed better than they had in the partially labeled statistical analysis step, suggesting that using only some of the labels for these algorithms provides performance much like using all of the labels. Figure 20 compares LDA and LSA in the partial labeling step versus its post-processing. It is worthwhile to note that the training slices for LSA do not follow the pattern established in the previous plots, as only training slices of 10, 25, 50, and 75 are present. The 90 training slice has been left out due to the way in which LSA tended to categorize the documents. Since LSA always grouped almost all documents into the same topic when running the statistical analysis step, it was nearly impossible to generate a sample of ten testing documents that were in two different topics, which was needed when calculating MI. LSA needed to frequently resample to find such a state, making it extremely time consuming to run. In 24 hours, LSA had completed less than 100 runs of the needed 10,000. For this reason, the 90 training slice for the LSA partial labeling step was not completed.

It is also worthwhile to compare the results from Figure 18 with Figure 20. The fully-labeled analysis (Figure 18) greatly outperformed the partially-labeled analysis (Figure 20). This is to be expected, as the algorithm is functioning with only ten percent of the information in the semi-supervised step as it had in the fully-labeled step.

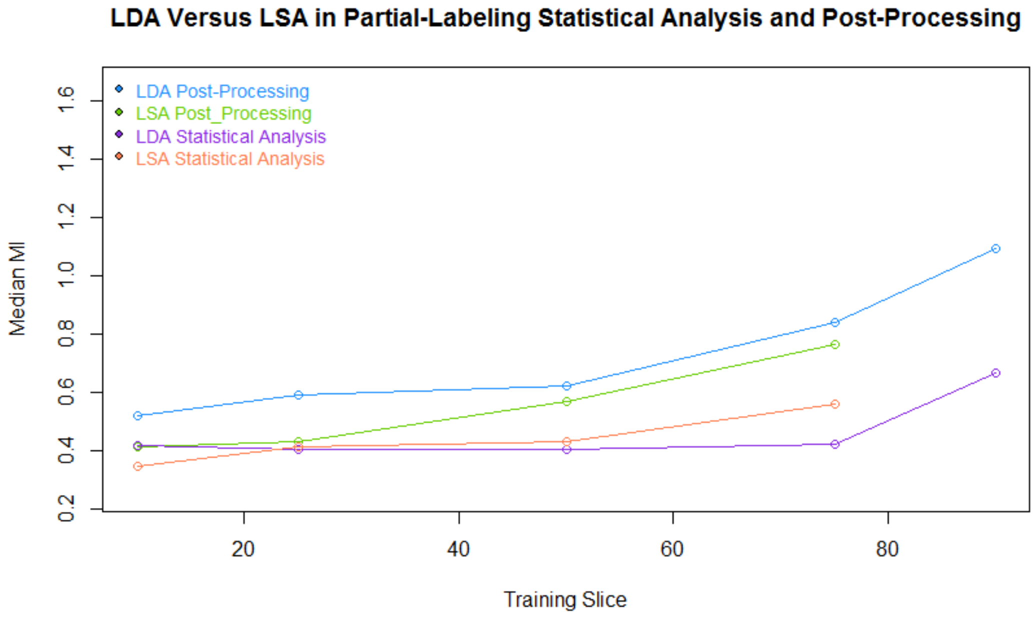


Figure 20. Line plot of LDA and LSA median mutual information values in the partial labeling and post processing stages.

We also show a bar chart below in Figure 21 of the improvement in LDA and LSA MI values when run in the post-processing step versus a semi-supervised statistical analysis that used only 10% of the labels. As a reminder, the 90 training slice for LSA is not present due to the exorbitant length of time that slice required to run. We see again that LDA seemed to benefit more greatly than LSA by the used of this approach.

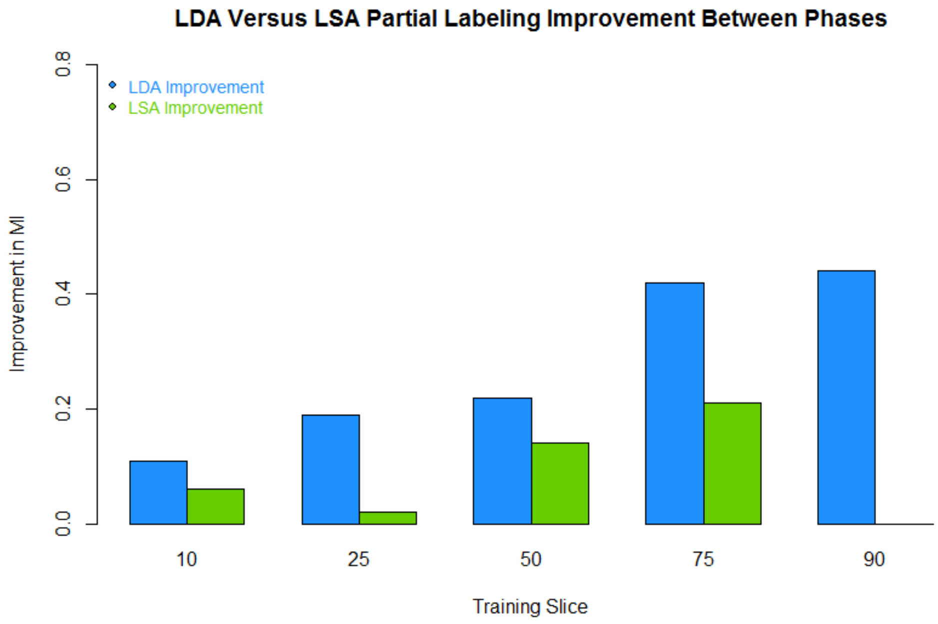


Figure 21. Improvement between statistical analysis phase and post-processing phase for LDA and LSA during the partial labeling step.

Figure 22 shows the results in boxplot form. These boxplots demonstrate that the one run done with the training values selected from the partially labeled 10,000 runs has a median value much greater than any value from the statistical analysis step. The red dot is the MI value from the post-processing run using the training values from the statistical analysis step of partially labeled values. Boxplots of LDA and LSA for each slice of training/testing sets are shown, except for the LSA 90/10 training/testing slice, which took too long to run.

These plots demonstrate the rarity of randomly sampling to get a set of training documents that produces a high MI value for the 100-document subset. Even the outliers from each set of values usually do not approach the MI value for the post-processing represented by the red dot. Only in the 25 training slice of LSA does the MI value from the post-processing step come close to the highest outlier in the median MI values from the statistical analysis step.

The boxplots in Figure 22 were generated by taking the median MI for each document in the statistical analysis step. That means they are generated using 100 values each, whereas the red dot is the single MI value obtained from the post-processing step.

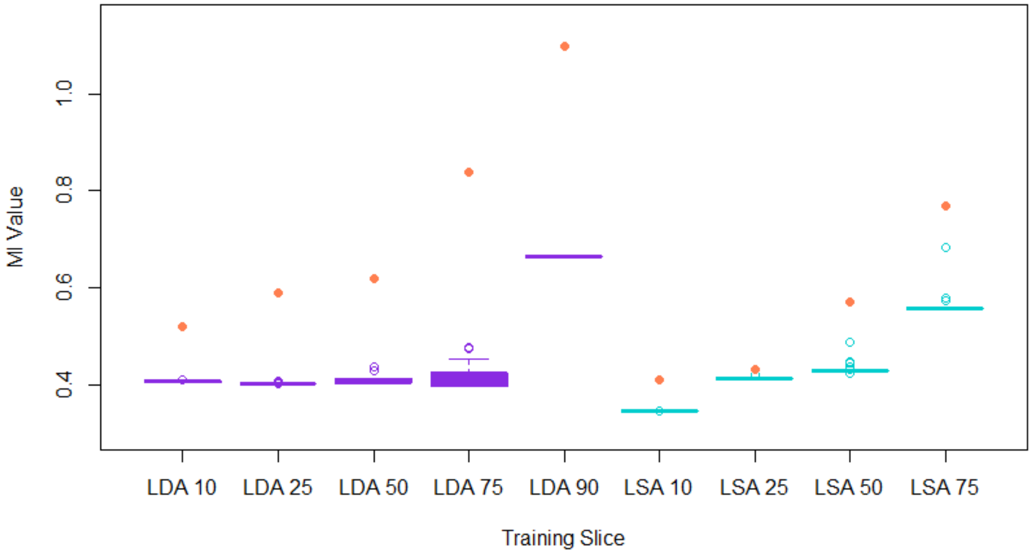


Figure 22. LDA and LSA partial labeling box plots for all sample training/testing sets.

CHAPTER V

DISCUSSION

Statistical analysis is common among supervised machine learning algorithms, but in the past has not frequently been able to be used for unsupervised algorithms such as LDA and LSA. This is due to the difficulty in figuring out a quantitative method by which unsupervised algorithms can be measured. Our results have shown promise using a semi-supervised approach. By statistically analyzing the data in the AWD, we have shown that mutual information values get higher when just ten percent of the data is labeled by humans. The human labeled data is used as the measure by which the unsupervised algorithms are calibrated and training set optimization allows the algorithms to perform in as human-like a manner as possible.

We have shown that this method is effective on two different text modeling algorithms, LDA and LSA. Both of these methods have been proven in prior research to be useful for categorizing unsorted and non-curated data when using them in their natural, unsupervised states. We have used both of them in a semi-supervised way to attempt to improve upon their performance.

LDA seemed to benefit more greatly from the semi-supervised approach. Since it initially put the documents into more varied topics, we suspect that it was able to better hone in on which topics were more accurate after the better training documents were gathered together. Since LSA tended to group most of the documents into one topic, it had less room to improve.

LSA initially had higher MI values, but they did not rise greatly as more and better training documents were selected. The MI values did rise consistently after the post-processing step, but at a less steep rate than LDA. This is likely due to the fact that LSA was often categorizing most of the documents into the same topic. However, the LSA also benefitted from the training data being filtered into which documents produced higher MI values.

This work should improve the methods used by journalists in the way that data from data leaks is analyzed and given to the public. By allowing a data scientist to use unsupervised machine learning algorithms such as LDA and LSA in a semi-supervised way, more information might be gleaned from the results of the topic modeling. Combining human curation of datasets with machine learning methods and tracking the results with statistical analysis provides a way to monitor how well previously unsupervised algorithms are sorting the data.

A drawback of this technique is that having humans pre-label the data gives the potential for the data to be mislabeled. Any mislabeling of the data at the beginning of the program run would skew the pruning of the algorithm, providing erroneous results.

In the future, we would like to compare LDA and LSA to a null model to see how well they are doing against topics that are randomly sorted instead of comparing them to human sorted documents. This has the potential to validate that the algorithms are categorizing reports as a human would since performance on randomized data is, by contrast, hypothesized to be completely random as well.

In addition, it might be helpful to change the number of topics we ask the algorithms to use. In particular, the fact that LSA mainly put the documents into one topic each time could mean one of two things. It could mean that the documents use similar wording and LSA is accurately putting them into topics, just too broadly. It also could mean LSA needs to be told to use more topics, therefore increasing the granularity of the clustering.

Due to the massive amount of report summaries included in the AWD, it was quite difficult to run this analysis on the full dataset. With enough time, however, it would be interesting to curate ten percent of the entire dataset, approximately 7000 reports, and let the analysis run with all of the data to verify that the method holds true. That would most likely entail adding more topics, as it is not presumed that the 77,000 records contain only the five topics discussed in this study.

Future work could also include combining the LDA method with point process modeling for a predictive approach to the data contained in the AWD. The point process modeling approach was able to track intensity of conflict in space and time, so choosing LDA documents that increased the effectiveness of the geospatial and temporal analysis might glean some interesting results. That could be done using the semi-supervised method in this study to filter the training data by documents that were shown to be better at clustering documents based on a certain attribute that was determined to be helpful for the point process method.

Another method of combining the two methods would be to access some quantitative data generated by the LDA step, such as the partitioned topics and add them to the PP modeling approach. For instance, the PP modeling was done by finding the distance to each city that was listed as a latitude and longitude point, then mapping the intensity to that location. Having information included from the LDA step would also make this process more precise, since modeling the topics in advance gave more details about the type of fatality that occurred in most instances.

BIBLIOGRAPHY

[1] T. K. Landauer, P. W. Folt, and D. Laham, “An introduction to latent semantic analysis,” *Discourse Process.*, vol. 25, no. 2, pp. 259–284, 1998.

[2] F. Qian, A. Pathak, Y. C. Hu, Z. M. Mao, and Y. Xie, “A case for unsupervised-learning-based spam filtering,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 1, pp. 367–368, 2010.

[3] M. K. Petersen and A. Butkus, “Modeling emotional context from latent semantics,” *ACM Int. Conf. Proceeding Ser. Vol. 291*, p. 3, 2008.

[4] V. Ha-Thuc, Y. Mejova, C. Harris, and P. Srinivasan, “A relevance-based topic model for news event tracking,” *Proc. 32nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR ’09*, no. 1, p. 764, 2009.

[5] D. Newman, C. Chemudugunta, and P. Smyth, “Statistical entity-topic models,” *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’06*, p. 680, 2006.

[6] E. Diaz-aviles, M. Georgescu, A. Stewart, and W. Nejdl, “LDA for On-the-Fly Auto Tagging,” *Evaluation*, pp. 309–312, 2010.

[7] E. Yao, G. Zheng, O. Jin, S. Bao, K. Chen, Z. Su, and Y. Yu, “Probabilistic text modeling with orthogonalized topics,” *Proc. 37th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR ’14*, pp. 907–910, 2014.

[8] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie, “Improving LDA Topic Models for Microblogs via Tweet Pooling and Automatic Labeling,” *Proc. 36th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 889–892, 2013.

[9] H. Misra, O. Cappé, and F. Yvon, “Using LDA to detect semantically incoherent documents,” *Proc. Twelfth Conf. …*, no. August, pp. 41–48, 2008.

[10] D. Du, L. Yu, and R. R. Brooks, “Semantic Similarity Detection For Data Leak Prevention,” *Proc. 10th Annu. Cyber Inf. Secur. Res. Conf. - CISR ’15*, pp. 1–6, 2015.

[11] I. Bhattacharya, “Query Classification using LDA Topic Model and Sparse Representation Based Classifier,” *Adv. Intell. Syst. Comput.*, vol. 469, 2016.

[12] Q. Diao, J. Jiang, F. Zhu, and E. Lim, “Finding Bursty Topics from Microblogs,” *Acl*, no. July, pp. 536–544, 2012.

[13] T. Rusch, P. Hofmarcher, K. Hornik, and R. Hatzinger, “Modeling Mortality Rates In The WikiLeaks Afghanistan War Logs,” *Knowl. Creat. Diffus. Util.*, no. September, 2011.

[14] J. Chang, S. Gerrish, C. Wang, and D. M. Blei, “Reading Tea Leaves: How Humans Interpret Topic Models,” *Adv. Neural Inf. Process. Syst. 22*, pp. 288--296, 2009.

[15] H. M. Wallach, D. Mimno, and A. Mccallum, “Rethinking LDA : Why Priors Matter,” *Adv. Neural Inf. Process. Syst. 22*, vol. 22, no. 2, pp. 1973–1981, 2009.

[16] J. O ’loughlin, F. D. W. Witmer, A. M. Linke, and N. Thorwardson, “Peering into the Fog of War: The Geography of the WikiLeaks Afghanistan War Logs,” *Eurasian Geogr. Econ.*, vol. 51, no. 4, pp. 472–495, 2010.

[17] A. Zammit-Mangion, M. Dewar, V. Kadirkamanathan, and G. Sanguinetti, “Point process modelling of the Afghan War Diary.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 109, no. 31, pp. 12414–9, 2012.

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet Allocation,” *J. Mach. Learn. Res.*, vol. 3, no. 4–5, pp. 993–1022, 2012.

[19] D. Blei, L. Carin, and D. Dunson, “Probabilistic topic models,” *IEEE Signal Process. Mag.*, vol. 27, no. 6, pp. 55–65, 2010.

[20] M. Brehmer, S. Ingram, J. Stray, and T. Munzner, “Overview: The Design, Adoption, and Analysis of a Visual Document Mining Tool For Investigative Journalists,” *IEEE Trans. Vis. Comput. Graph.*, vol. 2626, no. c, pp. 1–1, 2014.

[21] T. Rusch, P. Hofmarcher, R. Hatzinger, and K. Hornik, “Model trees with topic model preprocessing: An approach for data journalism illustrated with the Wikileaks Afghanistan war logs,” *Ann. Appl. Stat.*, vol. 7, no. 2, pp. 613–639, 2013.

[22] Jurka, Timothy P. "Getting Started With Latent Dirichlet Allocation Using Rtexttools + Topicmodels". *RTextTools: a machine learning library for text classification*. N.p., 2011. Web. 7 June 2016.

[23] J. L. Phillips, M. E. Colvin, and S. Newsam, “Validating clustering of molecular dynamics simulations using polymer models,” *BMC Bioinformatics*, 2011.