POST-PROCESSING OF LDA FOR CLUSTER QUALITY ANALYSIS

By

Matthew Fyffe

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: COMPUTER SCIENCE

Approved:

_____

Sibel Adali, Thesis Adviser

_____

Boleslaw Szymanski, Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

April 2009
(For Graduation May 2009)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Latent Dirichlet Allocation (LDA) is a formal generative model of documents, breaking data into two tiers, documents comprised of keywords and topics comprised of distributions of keywords. LDA is implemented in most scientific papers employing information retrieval techniques, with emphasis being placed on modifying LDA or its inputs in order to improve its output.

While methods have been implemented to improve LDA's results, nothing has been done to analyze LDA's output to compare the results qualitatively. LDA clusters the data into topics but does not shed light on which topics are stronger than others. We propose a metric that executes multiple runs of LDA and compares the results to find the strongest topics in the data set.

Through simulations on both fake and real world data, we have shown that our proposed metric offers a reliable quality score for the topics.

1. INTRODUCTION

A growing area of computer science is the problem of modeling collections of discrete data such as items of text. Given large sets of data, how can we properly cluster the items into understandable sets which can be identified as having common attributes?

This issue was tackled by implementing formal generative models of documents. Latent Dirichlet Allocation (LDA), described in [1], is an example of such a model. LDA creates a two-tiered model comprised of topics and documents. The documents contain a collection of words while the topics contain a distribution of keywords. In addition, LDA produces a distribution of each document over the topics, identifying the likelihood of each topic being present in the document. To achieve this, LDA assumes words in topics are distributed by a latent dirichlet distribution and attempts to match the data in the best way possible to this distribution.

Improvements to LDA often come in the form of modifying the input to the system as seen in [2, 3]. These modifications include the stemming of words within data, the removal of filler words that are common across data, and the estimation of total number of topics by analysis of the data. Other methods add additional meta data factors to LDA's analysis itself, looking at information about the relationships between owners of the data items and the time stamps at which data was created as seen in [4, 5, 6].

While these various improvements help to improve the output of LDA in most collections, they still do not provide any sort of quality analysis of LDA's output. Some topics will be better represented in the data set or can be explained better, yet LDA does not provide any clues as to which topics these may be. Instead, research to date relies upon human analysis in which the researcher will sift through the topics and pick the ones that best prove their results. While this shows some success in their work, it includes human bias as people search for what they consider to be good topics and not necessarily what the data set values. When there are large quantities of topics, this process can be overwhelming as the researcher would be forced to go through hundreds, if not thousands of topics.

We propose a solution to this problem which analyzes the output of LDA to identify the highest quality topics of data produced. We define quality as the set of topics that are best represented in the data set and most accurately fit the input. To validate our results, we ran LDA across both randomly generated data sets with known topics and real world data sets.

In our method, we compare the results of LDA within each run, developing a similarity function $\sigma(i, j, k)$, which represents the commonality between two topics i and j within an LDA run projecting k topics. This value is generated by analyzing what keywords were found present in each topic. Topics with a high sigma value were merged, reducing entropy within the data and highlighting the most diverse topics.

We also create a function to evaluate topics for their overall strength, called the durability function, $\delta(i, j, k, l)$. This δ function compares the commonality between two topics i and j across runs k and l in a similar function to the commonality function on a single level. A high delta score means that the topic survived across multiple runs and as such was a good topic in the data set.

Using these two metrics, we were able to create a system by which LDA can be improved in its output. This system ranks the results by quality and filters out redundant topics without human review.

2. Related Work

The work that has been done to improve LDA's results thus far can be broken down into two categories: Input Modifications and Modeling Modifications. The input modifications utilize the standard LDA model with modifications altering the data being fed into it. In the modeling modifications, LDA is expanded to factor in additional variables.

One of the biggest input modifications seen is the Pre-Identification of Topic Totals (PITT) in [2]. PITT is introduced because the base version of LDA requires the number of topics in the data as an input. This modification employs an analysis of the posterior distribution of words assignments to topics, and uses the results found to conclude the number of topics in the given data set. This enhancement is important because the tester often does not know how many topics are found within the data given and it reduces the amount of variables needed by LDA.

The other major input modification involves the filtering of filler words in the input data. This modification filters out common items in the data that appear across all data sets and thus, simply convolute the model. It also removes rare words that only appear in single documents and thus, have no use in topic identification. This modification is important as it focuses the modeling on only the data that is important for the topics. Some papers incorporate more advanced versions of this filtering into the actual model for further enhancement, as described in [3].

Both [2, 3] improve upon LDAs results but there is still room for improvement. Neither of these methods does anything to differentiate the modeled topic quality. Running LDA with these modifications does not remove the need for manual labor of identifying the best topics. In addition, manual evaluation of topics introduces the challenge of human bias as to what makes a topic a good one. Do we choose topics that are most representative of the data set or those that are easiest to explain? Simulation results show that LDA output provides topics of varying quality, even when the correct number of topics is provided as an input. Thus, even though improvements were made on the input to LDA, there is still plenty of room for

improvement on the output.

The other approach to improving LDA's results is by modification of the generative model. Modifications to the LDA model include the Author-Topic (AT) model from [4], Author-Recipient-Topic (ART) from [5], Role-Author-Recipient-Topic (RART) from [5], and Content-Community-Time (CCT) model from [6]. These models add additional information to LDA's model, allowing it further insight into the data it is evaluating and clustering.

AT, ART, and RART models are based on email data sets, but can be extended to general data. These methodologies factor in additional data into LDA's data set beyond just the documents' text. AT incorporates the author of the document into the model, causing LDA to evaluate a document not just by viewing its keywords, but by incorporating information about the person who created it. ART expands this further, adding the recipients of the document into the data analyzed by the model. Lastly, RART adds the role of the correspondents into the distribution, because different people writing on the same topic may communicate with different groups of people using a different vocabulary. The major factor to understand here is that the models go beyond the simple keywords found in the text and weigh other inputs in the distribution. The person sending or receiving a message, as well as their role in the company is similar to a keyword in the document, and affects the relationship between different documents.

The CCT model works in the same way. More targeted towards blogs where posts are given time stamps and the blogs may link to other related blogs, the CCT model can still be extended to cover more general data. In this model, the time at which a document was produced is added as an input to LDA. Not only do documents with similar keywords get weighted similarly, but also documents published within similar time periods. The notion of a community is also incorporated in a manner similar to the author/recipient roles described above. If the blog (author) hosting this post is associated with other blogs (recipients), these variables are important in considering documents relationships. As such, the community surrounding the document is added as a variable to LDA.

While these methods allow for topics to incorporate different types of infor-

mation, they still require the analysis of the topics after running LDA. As such, our solution can work with any of these modifications to LDA. Our solution is run after LDA completes, using the output of LDA to analyze the topics produced and to find the best ones. Thus, so long as these solutions don't modify the type of data produced by LDA, they can all be used in correlation with ours. No matter what model is used or which modifications are made to LDA's input, our solution will provide an improvement to the quality of the output.

3. Background

3.1 Terminology for LDA

In this section, we introduce the main terminology used in this thesis and explain details of LDA. Designed largely to deal with textual documents, the main concepts used by LDA are keywords, documents and topics. A keyword is a word, or in a more general context, a discrete item of data. A document is a collection of these keywords, represented in the form of a vector. This vector holds an index for every keyword found in the document. Each keyword may appear more than once in the document. The number of times the keyword appears is also recorded in the vector. The importance of the order of the words is not considered, treating documents as a bag of words.

These documents are aggregated together into a document matrix containing every document's collection of keywords which are used as input to LDA. In addition, LDA takes as input, a number which represents the predicted number of topics n in the data set. LDA processes these documents and as an output, produces two files, a .beta file (BETA) and a .gamma file (GAMMA).

The BETA file contains n lines, each corresponding to a different topic. For each topic, a vector over the keywords in the data set is given. The weight of each keyword represents the probability of that keyword being present in that topic.

The GAMMA file contains a vector for each document in the collection input to the algorithm. For each, a vector with n entries is given, one for each topic. The weight of each topic in each document represents the importance of the topic for that document. The GAMMA file values are not normalized and their magnitude depend on the length of the document.

3.2 Overview of LDA Functionality

Between reading in the input and outputing the BETA/GAMMA files, LDA processes the data. LDA receives the document vectors which it views as random mixtures over latent topics with each topic representing a distribution of words.

Taking these vectors, LDA builds a probabilistic model with three levels of computation. First, the parameters α and the number of topics k is used to create a Dirichlet distribution of topics. Then, this Dirichlet distribution is used to create a multinomial distribution of words for each document. The vector β determines the likelihood of each topic for documents in the collection. A Poisson distribution is used to determine the length of documents. Both α and β vectors need to be estimated from a given collection.

By developing LDA on a three level system, documents can be associated with more than one topic. This is caused by the fact that topics are sampled repeatedly within the documents under the given model.

LDA functions by using inference in order to compute the posterior disribution of the latent topics in the data set. A variety of approximations are used to produce this distribution, with Laplace approximation, variational approximation and Markov chain Monte Carlo being used to generate an inference. The method chosen by LDA is variational approximation, utilizing Jensen's inequality to obtain a lower bound on the likelihood of a topic. By running this inference through a sequence of iterations, the variation in the output topics can be compared with each iteration. The iterations will continue until the variation reaches a threshold at which point, LDA outputs the projected topics via the BETA and GAMMA files described above.

These results are then used by our algorithm.

3.3 Terminology for Our Solution

In our solution, a document is represented by a vector $X = \langle w_1, ..., w_n \rangle$ where each w_i is the weight of term i for that document. A topic vector is a document vector corresponding to a specific topic, given by a line in the BETA file. A run of LDA is given by a specific run with a given number of topics.

For the majority of our work, we utilized the angle distance function. Given the BETA file's topic vectors A and B, the angle distance function between those two vectors is defined as:

$$\theta(A, B) = arccos(A * B/(|A| * |B|))$$

σ is the commonality metric between topics within the same level. As such:

$$\sigma(i, j, topiclevel) = \theta(BETA_{topiclevel}[i], BETA_{topiclevel}[j])$$

Where i and j are two topics within the run of LDA in which the number of topics given as input was topiclevel and BETA$_{topiclevel}$[i] corresponds to the ith topic vector in the BETA file.

This equation is employed in our defintion of delta, however on a more generic method. For delta, a version of sigma which crosses levels is used. This metric can be seen as:

$$\delta([i, topiclevel1], [j, topiclevel_2]) = \theta(BETA_{topiclevel1}[i], BETA_{topiclevel2}[j])$$

Here, i and j represent topics while topiclevel1 and topiclevel2 represent the different runs of LDA that are being compared.

4. Algorithm

Our solution is a proposed modification to LDA in which processing is done on LDA's output. As such, it begins after LDA has finished and so, runs off of BETA and GAMMA as inputs.

Our solution is a merger of two different concepts. The first concept is the filtering of redundant topics. Duplicate topics are filtered from the output topics making the final data set cleaner. The other concept is a quality metric which evaluates the strengths of the remaining topics. Both of these concepts are combined in our algorithm with the merging of duplicate topics being weighted based upon the sigma values, and the final quality scores being output at the conclusion of the algorithm utilizing topics delta values.

4.1 Redundant Topic Filter

Utilizing the concept of sigma, the first piece of the post-processing algorithm is the redundant topic filter. This addition to LDA is used to correct a flaw in LDA's output. Even though LDA tries to find the most distinctive topics, there may still be a great deal of overlap between related topics. This overlap slows the evaluation of topics as it is hard to distinguish qualitatively between these common topics, and also because of the added overhead of the additional topics. By creating mergers, this method makes the output easier to evaluate by a human. Topics may appear statistically different from each other, which is difficult for a human to evaluate because the differences might actually be minor in the output vectors of the original algorithm.

In order to merge topics, the sigma values of each topic are used as a metric. A value close to 90 degrees means that the two topics were perpendicular and thus, had little in common between their keywords. The closer the value is to 0 degrees, the more the two topics overlap.

When comparing these topics, most will have large angles between them, upwards of 85 degrees. The reason for this is that most topics are unique and are

perpendicular to the others. This makes sense since LDA tries to create unique clusters of keywords which seem to appear in a consistent distribution and so, topics will either be distinct, or very similar to others. It is uncommon to find topics in the 50-70 degree range as that means that the topics had some overlap, which means LDA tried to create two topics of moderate similarity. However, when the input number of topics is significantly greater than the actual number, more topics may surface in this region.

It is important to note that the topics that are actually composed of the same keywords won't always be identical, since the strengths of the various keywords will differ with LDA's assignment, and so, a threshold is set. The threshold dictates at which angle of difference, the topics will still be considered the same and so, at which level they will be merged. From our simulations, it appeared that an angle of roughly 60 degrees works well as a threshold. However, a higher or lower value might be required depending upon the amount of overlap between topics in the data set and the tester's preference for granularity.

1: **function** mergeTopics(int $topicLevel$, double $threshold$)
2: Topic $T_{best}.commonValue = 90$ // Smallest angle difference between two topics
3: Topic $T_{best}.commonTopic = -1$ // Topic sharing that angle with the topic
4: **for** each topic $Ti \in topicLevel$ **do**
5: **for** each topic $Tj \in topicLevel$ **do**
6: **if** $Ti \neq Tj$ **then**
7: **if** $\sigma(Ti, Tj, topicLevel) \leq Ti.commonValue$ **then**
8: $Ti.commonValue = \sigma(Ti, Tj, topicLevel)$
9: $Ti.commonTopic = Tj$
10: **if** $Ti.commonValue \leq T_{best}.commonValue$ **then**
11: $T_{best} = Ti$
12: **if** $T_{best}.commonValue \leq threshold$) **then**
13: add $merge(T_{best}, T_{best}.commonTopic)$ to data set
14: remove T_{best} and $T_{best}.commonTopic$ from data set
15: return true
16: **else**

17: return false

1: **function** merge(Topic $T1$, Topic $T2$)
2: Topic T_{new}
3: **for** each keyword $k \in T1$ **do**
4: $T_{new}[k] = (T1[k] + T2[k])/2$
5: return T_{new}

Function mergeTopics is repeatedly invoked until no further mergers can take place. With each invoking, the system runs iteratively through the topics, comparing each topic to every other topic and accumulating a collection of sigma values for each topic. The smallest overall sigma value is selected from the collection of topics. This sigma value was a comparison between two topics in the data set which are then marked to be merged if sigma is below the threshold. If it's not below the threshold, mergeTopics halts.

When two topics are marked to be merged, they are combined by the method merge(). Merge creates a new topic which contains the average value for each keyword between the two topics. It then removes the two merging topics from the topic list, replacing them with the new combined topic.

Using this metric, it is possible not only to remove redundant topics, but also to produce stronger, higher quality topics. The reason for this improvement is that the common strong keywords between the two topics will maintain their strength, while keywords of weak to moderate strength in one topic but very low presence in the other topic will be softened by their conflicting values. Thus, this method highlights the keywords that LDA was certain of and allows them to appear stronger in their topics.

However, this method of merging can produce some potentially poor results. First, if one of the merging topics is of a poorer quality than the other, then merging the two topics will actually dilute the quality of the topics output. Second, it unfairly gives equal weight to both topics, even though one may be qualitatively better than the other. Additional enhancements can be taken to avoid such dilution, which are detailed later on.

4.2 Topic Quality Evaluation

The other facet to the output analysis is quality evaluation. In this aspect of our solution, we are able to output qualitative scores that rank each of the topics. By doing this, our solution is able to remove the human bias from evaluating the output, and to distinguish the best topics.

To create this analysis, the algorithm utilizes delta values. After running multiple runs of LDA with varying topic predictions, the outputs of each of these runs are compared. The algorithm finds the delta values between every topic i and every other topic j from the different LDA runs. The minimum delta values for topic i across every run are averaged together to produce topic i's overall quality score.

The algorithm for this process is defined as:

1: **function** GetBestDelta(i, $basetopiclevel$)
2: **for** each TopicLevel $tlevel \in TopicLevels$ **do**
3: **if** $tlevel \neq basetopiclevel$ **then**
4: **for** each topic $Tj \in tlevel$ **do**
5: $comval = \delta(i, j, basetopiclevel, tlevel)$
6: $topicval[tlevel] = \min(comval)$
7: $BestDelta = \text{avg}(topicval)$

These BestDelta values are considered to represent a topic's quality because they show the topic's resistance to change. The reason for their representation of change can be seen by looking at their meaning within LDA. Within each iteration of an LDA run, topics are generated which are characterized by a distribution of words. These topics are compared to the documents and best fits are attempted. These topics are modified each iteration based upon their quality of fit until the amount of change between iterations is below a certain value. Because of this, there is no guarantee as to what topics will be created and so, the topics that appear most often in LDA runs are the strongest topics that arose regularly when fitting the documents to generated topics. Those topics that appear frequently were probabilistically stronger than their counterparts and are overall higher quality topics in the original data set.

At the end of this analysis, the algorithm produces a vector representing each topic's quality where quality is defined as that topic's survival rate.

4.3 Redundant Topic Filter with Quality Evaluation for Merging and Error Detection

As mentioned earlier, topic consolidation using simple merging can cause some loss in quality. The merged topics may be of unequal qualities and yet the merger ignores this, thereby enhancing the features of the poorer topic. As a result, the newly created topic may actually be worse than the existing two topics. Because of this, quality evaluation is merged with topic filtering in order to create a single algorithm that can utilize the best of both methods.

Building off of the standard topic filter, the first change introduced is that when merging the two topics, the topics quality metric, BestDelta, can be used as a weight to determine the topic's contributions to the new topic. The topic with the higher quality will contribute more to the new topic. In order to do this, the first topic, i, is given a weight of BestDelta(j) divided by the sum of GetBestDelta(i) and GetBestDelta(j). The same thing is done for the second topic, j, except with the numerator being swapped with GetBestDelta(i). The reason that i's weight uses j's quality score in the numerator is that the better topic has a smaller weight and so, this serves to give the better topic a larger value.

Using these weights, the positive attributes of the stronger topic will maintain their strength, and the weaker topic will simply reinforce the strongest attributes, making the new topic of a high quality.

In addition, in order to prevent the instance where the topic might actually be worse than the originals, and thus, cause a dilution in the quality of the system, a check is done after the merger is complete. The quality of the new topic is compared to the qualities of each of the non-merged topics. If the new topic is of a higher quality, then it is kept. Otherwise, it is thrown out and the two non-merged topics are kept, with a marking that they should not be merged to avoid falling into an infinite loop of attempted mergers. By making these additions, the topic consolidation is able to avoid the creation of poorer quality topics and is able to

place greater emphasis on higher quality topics when merging.

One final addition is a history of comparisons. Before any mergings are done, the original differences between topics are stored. Using these values, the algorithm compares topics before merging them to find out the difference in their values prior to all of the mergers. If the values prior to the mergers execution is above a certain threshold, the merger is not executed. The reason for this addition is to avoid the case in which a sequence of mergers causes unrelated topics to be homogenized to the point where they look alike and are merged.

The new version of merge looks as follows:

1: **function** merge(Topic $T1$, Topic $T2$)
2: **if** $T1.OriginalDistance(T2) \geq MergerThreshold$ **then**
3: Prevent $T1$ and $T2$ from merging again, return
4: Topic $Tnew$
5: Double $T1Quality$ = GetBestDelta($T1$, baselevel)
6: Double $T2Quality$ = GetBestDelta($T2$, baselevel)
7: Double $T1Weight = T2Quality$ / $(T1Quality + T2Quality)$
8: Double $T2Weight = T1Quality$ / $(T1Quality + T2Quality)$
9: **for** each Keyword $k \in T1$ **do**
10: $TNew[k](T1Weight * T1[k] + T2Weight * T2[k])/2$
11: **if** $GetBestDelta(TNew, baselevel) \leq T1Quality$ **then**
12: **if** $GetBestDelta(TNew, baselevel) \leq T2Quality$ **then**
13: Replace $T1$ and $T2$ with $TNew$
14: **else**
15: Ignore $TNew$, Prevent $T1$ and $T2$ from merging again

4.4 Overall Solution

In summation, our algorithm is able to take the output of LDA with any modifications and provide a qualitative analysis of it. It first consolidates the output by merging similar topics while utilizing their qualities evaluated across runs to dictate the weights of the mergers and strengthening their strongest keywords. It then takes this improved data set and compares it to similar runs of LDA using the

same data set but variable input parameters. Using these comparisons, it ranks the topics by their ability to survive with varying inputs to provide a qualitative metric of LDA's output.

5. Simulation Results

To test the quality of our proposed algorithm, we ran simulations on three different data sets. The first two data sets were fake topical data distributed randomly, and fake image data with a dirichlet distribution. These data sets were used because of our knowledge of their latent structure and so, the actual success of the algorithms can be easily evaluated. The third data set used was the data from the Proceedings of the National Academy of Sciences (PNAS) conference for 1991 to 2001 found at [7]. This data provides a real world data set with which we can compare our solution to others.

Using these data sets, we compared the performance of our solution to PITT described in [2]. However, it's important to remember that our solution can be augmented onto any of the modifications to LDA. Our reasoning for comparing it to PITT is that it enables us to highlight some of the flaws inherent in LDA and to show how our solution can improve the results on a general basis.

To evaluate the performance of our solution, we have used a variety of metrics. For the generated data, we compare the percentage of correct topics found within the data set. We feel this metric highlights the algorithm's ability to find the correct topics within the data set.

Using the real PNAS data, it is not as easy to evaluate because there are no well-defined topics for the data set. As such, coming up with a comparison between our solution and PITT is difficult. To evaluate the results, we looked at the relationship between the papers that were identified as being in each topic, and the way in which other resources categorized these papers. Papers categorized as dealing with similar subject matter are considered to have a strong relationship and so, topics containing papers of similar material are ranked as stronger than those without.

5.1 Fake Topical Data

The first data set we used in our simulations was a set of fake documents. The documents were generated with two topics each in a data set that was comprised of sixteen distinct topics. Each topic has 100 different keywords. When generating documents, each document is assigned 20 keywords from each of the two topics found in it and an additional 25 keywords randomly selected from the entire set of keywords.

Using this data, we ran LDA with a prediction of 12, 16, and 20 topics. Because the correct number of topics is 16, we treated this run as the solution as found by PITT since it provides the correct input number of topics. All three runs were used as inputs to our solution.

Table 5.1: Topic Finding Accuracy for Fake Documents

Method	Topics Found																Correct
OURS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	100
PITT	1	2	3	4	5	7	8	9	10	11	12	13	15	16	4/12	6/14	87.5

As seen in Table 5.1, using general LDA under PITT, most topics identified are correct. 14 of the 16 unique topics are identified correctly. However, there are two additional topics that contain a mix of two or more topics. As a result, somebody analyzing this collection of data without innate knowledge about the correct topics would be misled into believing these two incorrect topics were valid. In addition, there is nothing to distinguish these topics from the other topics, and so, they are considered just as strong as every other topic.

With our solution, data from runs using 12 and 20 topics was also used. These other runs produced different topic results, some of which had greater presence of topics that weren't found in the 16 topic run. While each of the data sets contained some inaccuracies, it also revealed real topics because it produced topics with unique real topics 6 and 14 which did not appear alone in the original set.

This data shows that all 16 unique topics were produced through the consolidations. As such, by adding our algorithm, the two wrong topics are filtered out and the correct two topics surface. Thus, 100 percent accurate results were obtained

using our results, whereas LDA with the correct number of topics given as input could only provide 87.5 percent accuracy.

5.2 Fake Image Data

The second data set we used in our simulations was a set of fake 5 X 5 images. This test was borrowed from the results in [2]. As described in their paper, the images were constructed by defining 10 topics which represented the horizontal and vertical segments of the image. Using this structure, 100 pixels were sampled from a dirichlet distribution of the topics.

Extending the work done in that paper, we created a dirichlet distribution with unbalanced weights so that real topics 1, 2, 6 and 7 were given double the weight of the other topics. This was done so that some topics would be stronger than others and thus, we could identify how this affects their appearance in our data set. 2000 sets of images were then drawn and used as documents as input for the algorithms. Using this data, we ran LDA with 5, 10, 15 and 20 topics projected. Because there were ten actual topics, PITT was considered to be the results of LDA with ten topics, while our solution utilized the results of every topic level in its identification.

Table 5.2: Topic Finding Accuracy for Fake Images

Method	Topics Found										Correct
OURS	1	2	3	4	5	6	7	8	9		90
PITT	1	2	3	5	6	7	8	9	10	4/7	90

Looking over the results, you can see several things. First, despite the fact that some topics were given a bias to make them better than others, there is no way to tell simply by looking at the output results. Second, in PITT, one of the topics was a combination of topics 4 and 7, so one topic was inaccurate in the data set.

These results also introduce the first potential problem with our algorithm. One topics is missing from our dataset, which was caused by the merging process. When handling mergers, the threshold value can produce some poor mergers. In this case, two topics of reasonable strength holding unique keywords have a common

angle of 59 degrees, causing the two topics to fall below the 60 degree threshold that we decided upon. Their merger hides one of the topics, causing the produced topic to be strong, but the loss of one of them.

While this is a minor issue, it can be avoided by manipulating the threshold value to see what works best for your data set.

The other facet of our algorithm is its ability to produce quality values associated with each topic in the data set. Below are the output qualities for each of the corresponding real topics in the data set.

Table 5.3: Quality Evaluation by Our Solution

Topic	1	2	3	4	5	6	7	8	9
Quality	4.29	20.69	24.89	42.71	28.97	8.04	11.57	19.64	26.59

Reviewing these qualities, the strongest topics appear to be topic 1, 6, 7, 8, and 2 in that order. The three best topics 1, 6, and 7 are three of the four topics that were given a biased weight, and thus, were three of the four strongest topics in the data set. The fourth topic given the bias was topic 2, which was ranked 5th out of the 8 topics. Thus, our ranking system was able to identify the strongest topics with reasonable certainty and to assign them appropriately higher quality scores than the weaker topics.

5.3 PNAS Data Set

The final data set used was the PNAS conference papers from 1991 to 2001. To generate the data set, each abstract and title was used as the keywords of a document. The total data set amounted to 20,551 keywords in 28,000 abstracts. Keywords were required to be found in two or more abstracts, and all stop words were removed from the data set. In addition, hyphens and other delimiting characters were removed from words.

This data set was used in [2] and served as a good dataset to identify how well the system performs with an estimate of the number of topics. Through [2], the data set was identified as having 300 topics and so, for our system we ran LDA over the set with 250, 300 and 350 topic projections. The 300 topic prediction was used

as the hub results for our system, comparing these results to the two different topic levels in order to generate a quality score. In addition, where a merger threshold of 60 was used in previous simulations, the threshold was lowered to 50 for the PNAS testing due to the reduced granularity of such a large data set.

Running our algorithm over this data set, the output quality results were:

Table 5.4: PNAS Quality Evaluation by Our Solution With 300 Topics

Best 5	18	97	17	135	24
Value	4.89	5.25	6.04	6.39	6.51
Worst 5	195	284	91	172	227
Value	71.26	71.40	72.10	72.56	78.89

From this output, it appears that our solution clearly identifies a difference in quality from the best topics to worst topics. The challenge however, is finding a method to compare our findings to the actual quality of these topics. The method used in [2] was to review the topics and identify the types of keywords that were found. This is fine, however, it can be a burdensome task when forced to sift through 300 or more topics evaluating each of their keywords. It also requires a knowledge of the subject matter to be sure you properly review the data. To create a non-biased system to evaluate topics, we generated a scoring system utilizing the top twenty documents found in each topic.

This score was produced by taking the top twenty documents found in each topic, and looking them up on Web of Science found at [8]. From [8], the papers citing the document can be found and the subjects covered by these papers are available. We take the subjects covered by the papers citing each document and build a list of the set of subjects relating to each document. We then compare the subjects covered by each of the top 20 documents and identify which three subjects appear most frequently.

We create a vector for each document which holds a one for each of those top three subjects it has present. The vectors are compared and the most frequently appearing subject distribution vector is identified as the base. Then, the distance is calculated between every distribution vector and this base vector. These values are aggregated together to create a correlation score for each topic's set of documents.

The formula used for the comparisons is as follows:

$$\sum_{doc=0}^{20} \sqrt[2]{((distr[doc]_x - baseDistr_x)^2 + (distr[doc]_y - baseDistr_y)^2 + (distr[doc]_z - baseDistr_z)^2)}$$

The results of this system can be seen for the five strongest and the five worst topics rated by our system under a 300 topic projection:

Table 5.5: PNAS Quality Evaluation by Web of Science Method With 300 Topics

Best 5	18	97	17	139	24	Mean	Median
Value	10	7.83	8.88	11.41	4	8.42	8.88
Worst 5	195	284	91	172	227	Mean	Median
Value	17.07	14.02	15.29	14.66	14.85	15.18	14.85

Reviewing these results, our metric's success is clear. We were able to analyze the topics by hand using the non-bias of each document's past citations and found the results to line up perfectly. From the Web of Science results, you can see that the worst topics predicted by our system have a mean quality score that is nearly double that of the best topics. Going further, the worst best topic is still three points better than the best worst topic. From these results, it is evident that we've found a method to provide a non-biased qualitative analysis of LDA output.

6. Future Work

From our simulations, the success of our method is clear. However, an area that has yet to be explored is the influence of keywords vs documents on the quality of topics. In our algorithm, we analyze the keywords found in topics as defined by the output BETA file from LDA. However, LDA also produces a GAMMA file defining the documents found in topics. Both of these allocations define the topics differently and represent different characteristics of the topics. The keywords show the topic's composition at the finest level, while documents show a higher level representation of the topic.

To explore the strength of using a document metric, the same algorithm used in our research can be explored. The only variation is that the distance between document vectors would be used instead of the distance between keyword vectors. Running tests using this similar method could shed light on whether or not there is any qualitative difference between using documents or keywords as the base for topics.

A second area of exploration is the influence of the threshold value on the strength of mergers. The value is currently decided upon manually, which can cause inaccurate mergers when set too high. However, every data set requires a different threshold value as different data sets have different levels of innate granularity in their topics. As such, it would be nice to separate this step from the user and create a method of analyzing the data set to find the proper threshold.

7. Conclusion

In conclusion, we've identified a method in which the results of Latent Dirichlet Allocatoin can be analyzed and assigned qualitative values. Prior to our work, there has never been a proposed method in which the results could be evaluated and compared. Instead, the results had to be reviewed by a person and to be assigned qualitative value based upon a person's own view of the system and their interpretation of the results. Our method provides a non-biased source that is capable of taking in LDA's entire output and to find the optimal results.

Using fake data, we were able to prove our metric's merit on a system where we knew the correct results. When dealing with fake topical data, our system was able to identify every latent topic in the data set, whereas LDA unassisted only identifies 87.5 percent of the correct topics. With the fake image data, our system correctly ranked the topics by their strength, placing three of the four strongest topics above the others, and ranking the last strongest topic fifth in the set of eight. These results prove our abillity to extract the best results from LDA's output and to order them in a way that reflects their quality.

When testing on the PNAS real world data, our metric reinforced its ability to find the best results by correctly ranking topics in a manner that coincided with the results produced by comparing the clustered document's subject matters in an external database. This show's our metric's ability to scale with larger data sets and to apply to real world data.

Possible future work built off of our method is the perfection of the data set's merging threshold value. Currently, the tool requires the user to provide the threshold value at which topics are merged. However, these mergers can potentially remove good topics when the threshold is set too high. By generating a method of analyzing the data to create a threshold value that fits with the type of data being analyzed, the system could avoid these potential issues.

LITERATURE CITED

[1] Blei, D. M., Ng, A. Y., and Jordan, M. J. (2003). Latent Dirichlet allocation. Journal of Machine Learning Research, 3, 993–1022.

[2] T. Griffiths and M. Steyvers. Finding scientific topics. PNAS Colloquium, 2004.

[3] Integrating Topics and Syntax, Thomas L. Griffiths, Mark Steyvers, David M. Blei, Joshua B. Tenenbaum, In: Advances in Neural Information Processing Systems, 17

[4] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In Conference on Uncertainty in Artificial Intelligence (UAI), 2004.

[5] A. McCallum, A. Corrada-Emmanuel, and X. Wang. The author-recipient-topic model for topic and role discovery in social networks. 2004.

[6] A. Qamra, B. Tseng, and E. Chang. Mining blog stories using community-based and temporal clustering. In Proceedings of the Fifteenth International Conference on Information and Knowledge Management (ACM CIKM' 06), 2006.

[7] Proceedings of the National Academy of Sciences. http://www.pnas.org/

[8] Web Of Science. http://isiknowledge.com/

[9] Lightspeed Matlab Toolbox. http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/

[10] FastFit Matlab Toolbox. http://research.microsoft.com/en-us/um/people/minka/software/fastfit/

APPENDIX A
Analysis Code

A.1 Code for Fake Topic Data Generation

```
public static void generateDocWithFiller(int i, int j, int numTopics,
      int numKeywordsPerTopic, int keywordsPerTopPerDoc, int numFillerWords,
      Random generator, FileWriter out)
{
  int[] docVector ;
  int lowIndex, highIndex, numKeywords, newIdx ;

  docVector = new int[numTopics*numKeywordsPerTopic] ;

  for (int k=0; k < numTopics*numKeywordsPerTopic ; k++)
    docVector[k] = 0 ;

  //generate topic i
  lowIndex = numKeywordsPerTopic*(i-1);
  highIndex = numKeywordsPerTopic*i - 1;

  numKeywords = 0 ;
  while (numKeywords < keywordsPerTopPerDoc)
  {
    newIdx = lowIndex + generator.nextInt(numKeywordsPerTopic) ;
    if (docVector[newIdx] == 0) {
      numKeywords++ ;
      docVector[newIdx] = 1 ;
    }
  }
```

```
//generate topic j
lowIndex = numKeywordsPerTopic*(j-1);
highIndex = numKeywordsPerTopic*j - 1;

numKeywords = 0 ;
while (numKeywords < t)
{
  newIdx = lowIndex + generator.nextInt(numKeywordsPerTopic) ;
  if (docVector[newIdx] == 0)
  {
    numKeywords++ ;
    docVector[newIdx] = 1 ;
  }
}

//generate filler words
lowIndex = 0;
highIndex = numTopics * numKeywordsPerTopic - 1;

numKeywords = 0 ;
while (numKeywords < numFillerWords)
{
  newIdx = lowIndex + generator.nextInt(highIndex) ;
  if (docVector[newIdx] == 0)
  {
    numKeywords++ ;
    docVector[newIdx] = 1 ;
  }
}

numKeywords = 0 ;
```

```
    try
    {
      for (int k=0 ; k< numTopics*numKeywordsPerTopic ; k++)
      {
        if (docVector[k] == 1)
          numKeywords++ ;
      }
      System.out.print(numKeywords +  " " ) ;
      out.write(numKeywords +  " ");
      for (int k=0 ; k< n*s ; k++)
      {
        if (docVector[k] == 1)
        {
          System.out.print((k) + ":1 ") ;
          out.write((k) + ":1 ");
        }
      }
      System.out.println(" ") ;
      out.write("\r\n");

    }
    catch(IOException e){
    }
  }
```

A.2 Code for Fake Image Data Generation

A.2.1 Matlab Code Generating Image Distributions

Using [9] and [10] for Dirichlet function calls.

```
a = [2,2,1,1,1,2,2,1,1,1]
dirichlet_sample(a,2000)
```

A.2.2 C Sharp Code for Document Generation

```
static void Main(string[] args)
{
    int totTopics = 10;
    List<Topic> topics = new List<Topic>();
    int totTopSq = (int)Math.Pow(totTopics / 2, 2);

    for (int i = 0; i < totTopics / 2; i++)
    {
        Topic tempTopic = new Topic(i);
        for (int j=totTopics / 2 * i; j < totTopics / 2 * (i + 1); j++)
        {
            tempTopic.addWordToTopic(j);
        }
        topics.Add(tempTopic);
    }
    for (int i=totTopics / 2; i < totTopics; i++)
    {
        Topic tempTopic = new Topic(i);
        for (int j=i % (totTopics / 2); j < totTopSq; j+=totTopics / 2)
        {
            tempTopic.addWordToTopic(j);
        }
        topics.Add(tempTopic);
    }

    List<List<double>> probDistributions = readDistr("topDistr.txt");
    List<Document> documentSet = new List<Document>();
    Random r = new Random((int)DateTime.Now.Ticks);
    for (int i=0; i < 2000; i++)
    {
```

```
int curTopic = 0;
double minProb = 0;
double maxProb = 0;
foreach (Topic t in topics)
{
    minProb = maxProb;
    maxProb = minProb + probDistributions[i][curTopic];
    t.resetWords();
    t.setStartingProbability(minProb);
    t.setMaxProbability(maxProb);
    curTopic++;
}
// Generate topics, add words to document

for (int j=0; j < 100; j++)
{
    double probability = r.NextDouble();
    foreach (Topic t in topics)
    {
        if (t.isInRange(probability))
        {
            t.addTopicPresence();
        }
    }
}

Document tempDocument = new Document(totTopSq);
foreach (Topic t in topics)
{
    Hashtable wordsToFrequency = t.getWords();
    foreach (int wordNumber in wordsToFrequency.Keys)
```

```
            {
                for (int j=0; j < (int)wordsToFrequency[wordNumber]; j++)
                {
                    tempDocument.addWord(wordNumber);
                }
            }
        }
        documentSet.Add(tempDocument);
    }
    writeResults(documentSet);
}
```