1. **Background**

Latent Dirichlet Analysis (LDA) is a method used to model the generative process of creating discrete data such as text corpora. LDA can take a large amount of data and create descriptions of that data. This method reduces the size of the data to these short descriptions while still maintaining the relationships necessary to carry out various inferences. The algorithm assumes a generative process for the creation of documents in a corpus D using N words. The topic for a particular word, $z\_n$, is modeled as Multinomial($\theta$), where $\theta \sim Dir(\alpha)$. Then, each word, $w\_n$, is chosen from a multinomial probability that is conditioned on the topic $z\_n$, P($w\_n | z\_n, \beta)$. (Blei et al., pg. 996)

LDA is most well-known for its application in the analysis of text data. It is used to create topics for documents, classify documents based on these topics and to determine which documents are similar to one another. The algorithm can also be used in other problems that have a similar structure to the document generating method described above. For example, in Blei et.al, 2003, the authors used a data set where web site users provide information about movies they enjoy. In this example, the users are analogous to the documents and their preferred movies are “words”. Topics can be found by determining similar movies.

There are multiple alternative algorithms that can be used for text analysis. These include the term-frequency inverse-document-frequency matrix, latent semantic indexing (LSI) and the probabilistic latent semantic indexing (pLSI) model. In Section 7 of the paper, the authors present the results of document modeling and document classification. In this section, the authors’ results show that their implementation of LDA performs better than competing methods in terms of perplexity measure, where better generalization performance is defined by a smaller perplexity measure.

Perplexity measure is defined as:

\begin{center}

Exp(-\frac{\sum\_{d=1}^M log p(\textbf{w}\_d}{\sum\_{d=1}^M N\_d})

\end{center}

In addition, this section demonstrates that other methods are prone to overfitting. As the number of topic increases, some of the alternative algorithms induce words that have small probabilities. This occurs because the documents in the corpus are divided into more collections. This can result in the perplexity measure becoming very large for these alternative algorithms. The problem comes from the requirement that the topic proportions in a future document must be seen in at least one of the training documents. On the other hand, LDA does not have this overfitting problem. For more detail, see Section 7 of Blei et al.

A disadvantage of LDA is that exact inference of the posterior is impossible as a result of intractability. Thus, it is necessary for the user to implement an approximating technique such as variational inference, a Gibbs sampler, or another technique.

1. **Implementation**
2. **Code**

The LDA algorithm was implemented using the Python programming language in the Jupyter notebook environment. The LDA function requires four arguments. The number of topics, k, must be specified. In addition, the output from the make\_word\_matrix is necessary in the LDA function. This function takes the corpus as an input and returns three things: a matrix where each word in the document is a row and the columns are the unique words in the corpus, the list of words unique to the corpus, and the number of documents in the corpus. The third argument necessary for the LDA function is the tolerance for convergence. Finally, an indicator value of the form of the corpus documents is required for the LDA. This value is 0 if the documents are a list of strings, and the value is 1 if the documents are just one long string. In addition, we created a function to return a specified number of words for each topic.

1. **Testing and Base Cases**

In addition to the implementation described above, various checks have been included in the function to prevent incorrect arguments. One check is that the number of topics specified by the user must be greater than one. It is not possible to have zero or less than zero topics, and one topic would just be described by the entire document. In addition, there is a check in place to ensure that the corpus is not empty and that each element of the corpus is a string. This prevents the user from receiving an error that the input is not a string. Lastly, there are checks to ensure that the tolerance is greater than zero and that the entry for needToSplit is either zero or one. These checks print messages that inform the user of why their input was problematic.

1. **Profiling Code**

We used President Bill Clinton’s State of the Union Addresses to profile the code. A dictionary was created of his State of the Union Addresses from 1993-1996. These documents can be found in the nklt package. The initial creation of the code utilized vectorization. The creation of the word matrix function took 10.95 seconds, whereas the LDA function took 88.676 seconds. The bottlenecks in the functions occurred at the expectation and maximization steps. These steps took 10.108 and 61.507 seconds respectively.

Making Faster

1. **Areas of Improvement/Further Research**

One area in which we would like to expand our research and improve our project is in the approximation of the posterior. In the implementation for this project, variational inference was utilized to estimate the intractable posterior distribution. However, there are other methods that can be used to estimate the posterior. These include Gibbs sampling and Metropolis-Hastings. It is possible that these approximations might increase the speed of the implementation.

Another area to expand on is the determination of the number of topics. This is an ongoing area of debate and research at the moment. The R implementation of LDA provides different measures to determine the number of topics, and these are something that we can look into adding to our code.

In addition, we want to continue to use the algorithms to situations beyond text corpora. The movie data example in this paper is one example of how LDA can be used for something beyond text data. The algorithm can be used for any situation that has the same structure as text corpora.

**References**

1. David M. Blei, Andrew Y. Ng, and Michael I. Jordan, *Latent Dirichlet Allocation*, Journal of Machine Learning Research 3, 2003, pg. 993-1022.
2. Max Sklar, *Fast MLE Computation for the Dirichlet Multinomial*, May 2014.
3. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872