1. **Abstract**
   1. Progress Report 1
2. **Background**
   1. What is it used for?
      1. What problem does it address?
         1. “generative probabilistic model for collections of discrete data such as text corpora” (LDA, 994)
         2. “The goal is to find short descriptions of the members of a collection that enable efficient processing of large collections while preserving the essential statistical relationships that are useful for basic tasks such as classification, novelty detection, summarization, and similarity and relevance judgments.” (LDA, 994)
   2. Description of the algorithm
      1. Generative process assumed for each document in a corpus (LDA, 996)
      2. Intractable posterior, therefore it is necessary to use an approximating method such as variational inference together with EM
   3. Applications of the algorithm
      1. Most well known in order to analyze text data
         * Determining and assigning documents in a corpora to topics
      2. Can also be used in other problems that have a structure similar to the term-document-topic structure
         1. Movie example in paper – data set in which users indicate the movies that they prefer, thus the users can be thought of as a document and the movies are the words, topics would classify the different movie choices
         2. Upon seeing some of the movie choices, be able to predict the held out set, this could potentially be used as part of a recommendation program
      3. Use in our research….lol
         1. There are many situations that LDA can be applied to
         2. Analyzing large amounts of text data, such as customer reviews to determine where people might be unhappy
   4. Other Methods/Algorithms
      1. Tf-idf matrix, latent semantic indexing, probabilistic LSI model
         1. “need to consider mixture models that capture the exchangeability of both words and documents” (994)
      2. Advantages
         1. Performs better in perplexity measurement
         2. Other techniques prone to overfitting
         3. Performs better in perplexity measurement
      3. Disadvantages
         1. “Exact inference is intractable for LDA” (LDA, 1014)
         2. Lots of steps to converge
            1. E step, M step
         3. Need to define the number of topics before hand
3. **Implementation**
   1. Brief Description
      1. Used Python together with Jupyter
      2. LDA function takes a word matrix, output from the make\_word\_matrix function, and implements the algorithm found on PAGE NUMBER of Blei et. Al
         1. Arguments Taken
            1. Number of topics – determined by the user
            2. Output from make\_word\_matrix

Make\_word\_matrix takes a corpus as the input

Outputs:

c is a matrix where the rows are each word in the document and columns are the unique words in the corpus, thus a 1 in cell [1,1] means that the word in the column appears in the first document

wordOrder is the unique words in the corpus

M is the number of documents in the corpus

* + - * 1. Tolerance
        2. Need to split
    1. Also a function that returns the top number of words for each topic
  1. Testing – “Base cases”
     1. Number of topics is an integer greater than zero
     2. Check that corpus is not empty and that each element of the corpus was a string
     3. Tolerance is non-negative and NeedToSplit argument is 0 or 1
  2. Compare with Python Implementation
     1. Potential issue with the convergence stuff

1. **Speed –Up**
   1. Text used for this implementation
      1. Clinton’s State of the Union addresses from 1993-1996
      2. Found in the nklt package
   2. Profiling of the Initial Code
      1. Vectorization used from the beginning
      2. Creating the word matrix – 1.095 seconds
      3. LDA – 88.676 seconds
      4. Bottlenecks occur at the E step and the M step
         1. E step total time – 10.108 seconds
         2. M step total time – 61.507 seconds
   3. How to make this faster…
      1. Identified some opportunities in the initial code to reduce computations
         1. Particularly in the make\_word\_matrix function
         2. Numba JIT Compilation
   4. Profiling of Optimized Code
      1. Creating the word matrix – 0.718 seconds
      2. LDA – 83.555 seconds
      3. Did the bottlenecks improve?
         1. E step total time -
         2. M step total time –
2. **State of the Union Results**
   1. State of the Unions
      1. How do Presidents’ topics change during their years in office?
      2. How are they similar to one another?

1. **Collaborative Filtering**
   1. User rating data from MoveLens.com (link)
      1. Users rate movies on a 1-5 scale
      2. Preferred movies were defined to be those for which the user rated 4 or 5
      3. Filtered the data set to users who rated at least 50 preferred movies
   2. Method
      1. Trained the model on a random set of 2535 users (about 75% of users who met description above) and got values for the parameters
      2. For the remaining users in the test set, one preferred movie is held out and the likelihood is found for the remaining possible movies
   3. Clustering with the movie data
      1. What are some of the movies that get grouped together?
   4. Performance
      1. How many movies were predicted correctly?
      2. How many users had a high likelihood assigned to the movie (within certain amount of the highest likelihood movie)?
2. **Areas of Improvement**
   1. Other ways to estimate the posterior could be more computationally feasible or faster
   2. Investigate literature on ways to select the number of topics – this is in the R package
   3. Other non-text related problems
3. **References**
4. David M. Blei, Andrew Y. Ng, and Michael I. Jordan, *Latent Dirichlet Allocation*, Journal of Machine Learning Research 3, 2003, pg. 993-1022.
5. Max Sklar, *Fast MLE Computation for the Dirichlet Multinomial*, May 2014, <http://arxiv.org/pdf/1405.0099.pdf>. (accessed April 14, 2016)