Code Structure - https://github.com/mfranklin128/lda
The LDA code is broken into three separate modules:

1. Data Preprocessing - various manipulations to take a corpus of text documents and format it properly for the LDA library, and also to implement a set of pre-processing techniques meant to reduce “noise” in the dataset and magnify the significant text.
2. Run LDA - simply launch the Blei LDA library with configurations according to user preference.
3. Output Processing - various tools to process the output of the LDA procedure into usable information, such as topic words and topic prevalence.

The LDA code used is the version implemented by Blei et al [cite], written in C. The code is packaged in the repository, but installation constitutes one of the main dependencies. The processing code is written in Python. There are a few hard-wired *nix-specific constants (e.g., forward-slashes for file paths), but could be adjusted to run on a Windows platform.

There are also a few hard-wired arrangements for use on the specific dataset and for testing (e.g., assuming over 1500 documents, and taking a 500-document slice in order to fit the corpus into RAM).

Data Preprocessing
Currently there is one technique packaged to do data preprocessing. This is a script that provides the following abstraction:

> Given a directory containing the corpus in the form of “one text file per document”, produces a representation of that corpus in the format for input to the Blei LDA library. Specifically, each document in the corpus is represented by one line in the LDA input file; each line is a list of [*index*]:[*count*] pairs, where *index* references a separate dictionary of words used in the corpus and *count* refers to the number of times it appears in that document.

In simpler terms:

> Data preprocessing takes a folder with your files, and converts it to the right format for LDA.

It can also apply several standard filters:

- Remove all words that consist solely of digits
- Remove single-letter words
- Convert the documents to case-insensitive
- Remove stopwords (e.g., prepositions and articles, taken from a standard list)
- Stem the words (e.g., “run”, “running”, and “ran” all become “run”)

- Only include the ten-thousand most common words

These filters are applied in order to remove contents of the documents that do not hold semantic value. Some of the techniques are intended to remove errors resulting from optical character recognition (OCR) of hand-written documents, such as the removal of single-letter words.

The code is easily extensible; by adding a filter in /data_preprocessing/utils/ and adding it to the filters listed in /data_preprocessing/utils/defaults.py.

Run LDA

The LDA automation portion is relatively straightforward. It simply reads configurations from a file and from the command line, and calls the Blei LDA library. It is configurable through configs.txt in the top-level directory.

Output Processing

This module runs some of the general analyses used in the project, though some other one-off utility scripts are not included (e.g., taking the tagged CSV files and sorting documents on that basis, since the CSV was of a specific format). The two tool included are:

- print_topics.py, which prints the top words from each topic. As with other components, it uses a set of default configurations which can also be specified.
- topic_proportions.py, which produces a set of statistics about the topic distributions. This is especially useful for the techniques used in this project, in which these statistics can be compared for different subcategories of the corpus to find differences between those categories.