Team Buddie Final Documentation

Each team must submit the software code produced for the project along with a written documentation. The documentation should consist of the following elements:

1) An overview of the function of the code (i.e., what it does and what it can be used for).

main.py compiles code above to automate the process of running PLSA without priors first and then with priors until we get the convergence we want (0.95+). This is to determine the correlation between US Stock data and New York Times articles per date. If the file structures match up, we can replace either the stock data or articles with other data sources to get the correlation between those two data sets also.

This project is an attempt to reproduce the following paper:
https://dl-acm-org.proxy2.library.illinois.edu/doi/10.1145/2505515.2505612

2) Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

- *plsa_without_prior.py*: initial run of PLSA without any priors
- *plsa_with_prior.py*: subsequent PLSA runs with priors determined from Granger and Pearson tests
- *word_retriever.py*: retrieves various info required such as word frequency per day
- *analysis.py*: contains code for running the Granger and Pearson coefficient tests
- *analysis.ipynb*: A jupyter notebook containing the same code in analysis.py allowing for more exploration and changes as needed. We utilized this notebook to explore our CSVs, view dataframes, and run our analysis.
- *main.py*:

1. Retrieve and normalize stock data
2. Initially run PLSA without prior, and run the analysis (Granger and Pearson coefficient tests) to retrieve priors
3. Using the priors retrieved above, iterate with the PLSA with prior until the desired convergence is achieved.

3) Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

How to run the project:

- Pull from github repo https://github.com/97agupta/CourseProject
- Install the necessary libraries, with the most notable one being
    - pip install plsa
- Place NYT corpus in the correct folder format
    - Can be retrieved from https://catalog.ldc.upenn.edu/LDC2008T19
- To run the iterative code, run 'python3 main.py'
- To run sections of the iterative code, run 'python3 <filename>.py'

Sidenote:

- After running the granger test, the F values for different topics need to be manually inspected and relevant topics need to be parsed out. These are then used to pull top words which are used for a pearson test.
- Running the PLSA as written can take multiple hours, so we have provided the end results for our 1st iteration in the /data folder for your viewing

4) Brief description of contribution of each team member in case of a multi-person team

- Katie
    - Ran initial PLSA algorithm without any priors
    - With the relevant topics retrieved from Granger test, retrieved the top 20 relevant words per topic and their frequencies per date.
- Aman:
    - Parsed Iowa Stock Exchange data
    - Used external time series & PLSA topics to write functions for a Granger test to find relevant topics
    - Used external time series & top words per topic to write functions for a Pearson test to create sub-topics from our topics with positively and negatively correlated words for use as a prior
- Sandeep
    - The PLSA library we were using did not have a provision to take priors and use that as part of the algorithm. So extended the library to accept priors and use them as part of the M-step of PLSA using numpy einsum.
    - Read the priors as input from csv of the previous steps, filter only the words that exist in the particular day's corpus and feed them into the PLSA step and do multiple iterations.