**CS410 Project Progress Report**

**Topic**: Reproducing a Paper: Mining causal topics in text data: Iterative topic modeling with time series feedback.

**Team**: PYM

**First Last email**

Pallavi Ravada [pravada2@illinois.edu](mailto:pravada2@illinois.edu)

Yash Skhwani [yashas2@illinois.edu](mailto:yashas2@illinois.edu)

Michael Bernardoni [mlb12@illinois.edu](mailto:mlb12@illinois.edu)

Tasks completed:

Data mining and cleaning of the text documents.

Data mining and cleaning of the betting probabilities

Production of the word/time slice coverage

Topic mining algorithm selection

Setup directions for the topic mining library

Topic mining on the entire corpus

Understanding of updating the topic mining model with new feedback data

Understanding the ITMTF algorithm

Tasks to do:

Code the time sequence scoring functions

Code the word analysis

Code the topic splitting

Code the introduction of feedback data

Visualization of the final data

Challenges:

We are still discussing the document/topic probabilities when new topics are added into the corpus

Detailed discussion of these steps follow:

**Detailed discussion**

**Data mining and cleansing**

**Step 1: Data mining**   
First we mined the raw xml data and produced a .txt for each document that had a paragraph with the words “Gore” or “Bush”. We only included the paragraphs with the key words, but we kept the document intact, that is if a doc had 2 paragraphs with either the word “Bush” or “Gore” the output would be one document with those 2 paragraphs.

Note this is just prep work and is not included in the project for size considerations.

**Step 2: Data cleansing - .\psla\_data\PSLAData.csv**  
For each file in the mined directory, we split the string into words. For each word we made each word lowercase, stripped out any character that was not alpha, and removed all stop words. We used stop words from: Onix Text Retrieval Toolkit Stop Word List 1: <https://www.lextek.com/manuals/onix/stopwords1.html> .

We added the results for each document in a .csv file .\psla\_data\PSLAData.csv. Each document is a row: cell 1 contains the year; cell 2 contains the month; cell 3 contains the day; cell 4 contains the cleansed text string of the document

We also created a csv file .\psla\_data\vocabulary.csv which contains unique vocabulary words in cell 1 and the count of the term in cell 2.

Step3: Data reduction - .\psla\_data\PSLAreduced.csv  
Using the vocabulary csv .\psla\_data\vocabulary.csv from step 2, we removed any word that only occurred once or twice (all words with counts over 2 were kept). We produced a csv file .\psla\_data\vocabularyreduced.csv which contains the new list of unique vocabulary words.

Using the new vocabulary, we created a new csv .\psla\_data\PSLAreduced.csv in the same form as the un-reduced csv.

**Step 3: Word coverage per time slice - .\psla\_data\wordseries.csv**  
Using the vocabularyreduced.csv and the PSLAreduced.csv we pre=processed a csv that contains the word coverage per time slice - .\psla\_data\wordseries.csv. The first row is a header row that contains the unique words in the vocabulary, this row is not used in the algorithm, but makes the file human readable. The first column in each row contains the time slice. All subsequent columns contain the word coverage during that time slice. This pre-processed file will be used in the ITMTF algorithm.

**/\* to do, add the betting data cleaning here \*/**

**Current data mining and cleansing files in the project:**

.\psla\_data\PSLAData.csv cleaned data  
.\psla\_data\vocabulary.csv cleaned data’s vocabulary

.\psla\_data\PSLAreduced.csv removed words occurring 1 or 2   
.\psla\_data\vocabularyreduced.csv removed data’s vocabulary  
.\psla\_data\wordseries.csv words counts per time slice

**Topic Mining Algorithm Selection**

The paper indicates that the PSLA algorithm was used. As such, we attempted to us PSLA. First we discovered the PSLA algorithm pypi <https://pypi.org/project/plsa/>. The algorithm worked well in our test data sets, and had excellent data visualization techniques. We identified where to add new topics in the library’s python code with the iteration feedback. However when we ran the full cleaned data, this library took over 12 hours to complete 1 model.

One of our team members wrote a PSLA algorithm in C++. The C++ algorithm was significantly faster. However, running the entire corpus caused memory issues. Time does not permit adding data swapping to disk.

Following the lead of other teams discussed on Piazza, we then selected Gensim’s LDA algorithm for topic mining <https://radimrehurek.com/gensim/models/ldamodel.html#usage-examples>. This algorithm does not have memory issues, and completes in a reasonable amount of time (under 10 min on one of team member’s home desktop).

Instructions for adding this library into an Anaconda environment is in the appendix.

**Appendix**

**Adding Gensim LDA library to an Anaconda environment**

Open Anaconda Navigator

1. Select Environments
2. Create an environment (LDA)

Open the Anaconda command prompt

1. Activate the newly created environment (Activate LDA)
2. Run: conda install nb\_conda\_kernels - Proceed Y
3. Run: conda install numpy - Proceed y
4. Run: conda install scipy - Proceed y
5. Run: pip install –upgrade scipy
6. Run: pip install –upgrade numpy
7. Run: pip install --upgrade gensim

For some reason I also had to run: python -m ipykernel install --user --name myenv --display-name "LDA"