Reproducing A Paper: Mining causal topics in text data: Iterative topic modeling with time series feedback

1. Overview:

We chose to reproduce the following paper:

- Hyun Duk Kim, Malu Castellanos, Meichun Hsu, ChengXiang Zhai, Thomas Rietz, and Daniel Diermeier. 2013. Mining causal topics in text data: Iterative topic modeling with time series feedback. In Proceedings of the 22nd ACM international conference on information & knowledge management (CIKM 2013). ACM, New York, NY, USA, 885-890. DOI=10.1145/2505515.2505612 (https://dl-acm-org.proxy2.library.illinois.edu/doi/pdf/10.1145/2505515.2505612)

This paper focuses on determining the casual topics in relation to time series it combines any given probabilistic topic model with time-series causal analysis to discover topics that are both coherent semantically and correlated with time series data. We iteratively refine topics, increasing the correlation of discovered topics with the time series. Time series data provides feedback at each iteration by imposing prior distributions on parameters. We have implemented this paper by using the New York Times 2000 presidential election related articles and the stock prices corresponding to that period. Based on the algorithm provided in the paper we tried finding casual topics and significant words which might have cause the election result based on the time series data - Iowa Electronic Markets (IEM)3 2000 Presidential Winner-Takes- All Market. We iteratively refine topics, increasing the correlation of discovered topics with the time series. Time series data provides feedback at each iteration by imposing prior distributions on parameters. Experimental results show that the proposed framework is effective.

This implementation can be used to find the casual topics based on the non - textual time series data.

- Team Members:
  - Heta Desai(leader) - Net-Id: hetahd2
  - Harpreet Siddhu - Net-Id: hsiddhu2

2. Implementation:

The main implementation and source files are :

- Data/BushGore.txt
- preprocessor.py
- Data/StockPrices.txt
- PythonScript.py

Methods implemented:

- **preprocessor.py:**
  - getBushGoreXmls()
    - Reads the xml files. Extracts full_text tag. Checks if the text contains keyword Gore/Bush. If it contains the words add that paragraph as a new document in new line in the BushGore.txt file along with the date of the article.
- **PythonScript.py:**
  - preprocessData()
    - Reads the text file containing the documents along with its date. Creates a dictionary with date as key and document ids with that date as the values. Reads the documents tokenizes it. Removes stop words, numbers , punctuations etc creates a word dictionary using Gensim which is in turn converted to corpus.
  - readStockPrice()
    - Reads the stock price text file and computes normalized probability.
  - runLda()
    - Calls Gensim LDA with prior provided along with number of topics and decay value.
  - calculateTS()
    - Calculates topic stream for the input model.
  - runGrangerTest()
    - Runs granger casualty test on the input 2D array and returns the result for the test.
  - isSignificant()
    - Based on the granger test checks if the input topic or word is significant.
  - findSignificantTopics()

- Finds all the significant topics using the TS and the time series data by running Granger test.
- calculateWS()
  - Calculates the word stream.
- calculatePrior()
  - Provided the significant topics this function finds the top significant words for each significant topic finds Pearson correlation and further divides into other topics and calculates the prior probabilities for each word. Also it calculates the average purity and casualty confidence.
- main()
  - The main method calls other methods step by step for :
    1. Different Mu values and plots graph for average purity and casualty confidence.
    2. Different tn and plots graph for average purity and casualty confidence.

To implement the paper we did the following:

1. Read the XML and convert to txt:

The original NY Times data was available in XML file. Each article was in a single xml file. We read the xml file searched for the keyword Bust and Gore and added the paragraphs containing those words to the text document along with its dates. Considering each paragraph as one document. The code for reading the xml file is available in **preprocessor.py** file. The XML data is available at NYTData. The generated text file is available at **Data/BushGore.txt** .

2. Extracted the Electronic Markets (IEM)3 2000 Presidential Winner-Takes- All Market dataset:

We extracted the data available on the IEM website for year 2000 residential Winner-Takes- All Market and added it to a text file available at **Data/StockPrices.txt** .

3. Preprocess NYTimes Data:

Once the data was available in txt file we processed the data to remove stop words, punctuations and numeric terms. Converted it to dictionary format and converted the dictionary to Gensim cropas which could be further used for LDA. This logic in implemented in **PythonScript.py** file inside **preprocessData()**. It returns a corpus containing tokenized words along with a dictionary with date as key and list document ids as value.

4.Calculate Normalized Probability:

From the Data/StockPrices.txt we consider the last price of each data and calculate the normalized property using the following formula:

(Gore price)/(Gore price + Bush price)

This logic is implemented in **readStockPrice(**) method in **PythonScript.py** file.

5. Implement LDA:

For probabilistic topic modeling we used Gensim LDA. The implementation can be seen in the **runLda() method in PythonScript.py** file. The LDA method available on Gensim is :

Gensim.models.LdaModel(corpus,id2word,alpha,eta.num_topics,passes,decay). It has many more parameters but for our implementation we just require these:

- Corpus : The corpus we generated in the preprocess step.
- Id2word : Dictionary generated in the preprocess step.
- Eta : The prior which is to be fed.
- Decay : The mu value that we will use to control how much the prior weighs.

The LDA would return a model.

6. Calculate TS:

Once we have run the LDA in the corpus we will calculate topic stream using our NY Times text data. TS is the sum of topic document probability for a particular date. This logic is available in **calculateTs(model,datedocument,number_topics,corpus)** method. This method returns and array with dimension number_of_days X number_topics.

7. Find significant topics:

Based on the TS we obtained in the previous step and the time series data we run Granger Casualty test and determine the significant topics which have significance level greater than 95. This logic is available in **findSignificantTopics(TS,goreNormProbability,datedocument,number_topics).**

8. Calculate WS:

Similar to TS we have calculated WS i.e the word stream. The logic can be found in **calculateWS(corpus,datedocument,vocabulary_size) method.**

9. Calculate Prior:

Once we have calculated WS and found the significant topic next we need to generate the prior. To generate the prior we need to find the significant words in the significant topics found in the previous step. To find the significant words we again run the Granger Casuality test for words and time series data. We would not use all the significant words but instead we'll only take the top significant words whose sum of probability is <= probM. We have fixed the probM values to 0.1 after trying many values. Once we find the significant words for that topic using Pearson Correlation test we determine if the word has positive impact or negative impact. Based on that we divide all the words into 2 topics one with positive words and other with negative words. If one category contains very few words as compared to the other we ignore the formar category. Once we have divided the topic we calculate the prior using following formula:

$$'\varphi w = sig(C,X,w)-\gamma \,/\, sum(sig(C,X,wj)-\gamma)$$

After calculating the prior we feed back this to the LDA model. This logic can be found in the following method:

calculatePrior(WS,significantTopics,model,datedocument,goreNormProbability,vocabulary_size,number_topics,dictionary).

This function also calculates the following:

1. Average Purity
2. Average Confidence

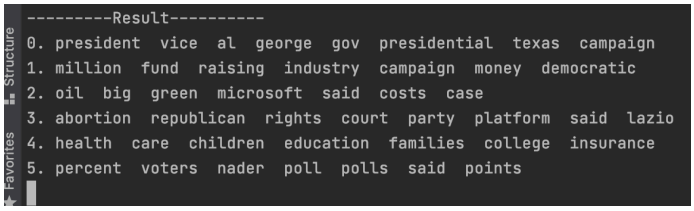
All these steps(5-9) are performed for different Mu values ranging in (0.6,0.7,0.8,0.9,1) with tn increasing in each iteration till it reaches 30.
Similarly keeping Mu constant i.e 1.0 we try running the same algorithm for different tn with
The following values for tn:

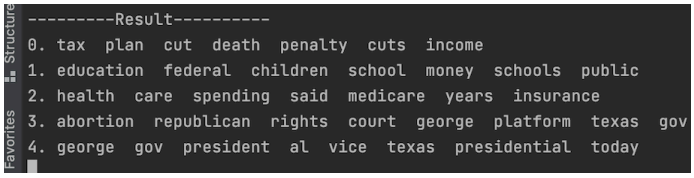
1. 10
2. 20
3. 30
4. 40
5. {10,20,30,30,30}

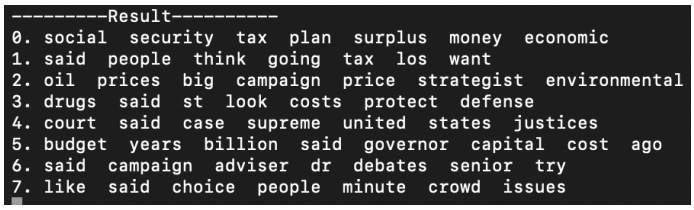
We calculate average purity and confidence for each iteration for each value and at the end plot a graph similar to that in the paper.

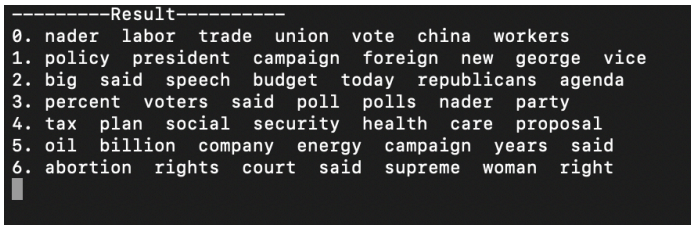
- **Result:**

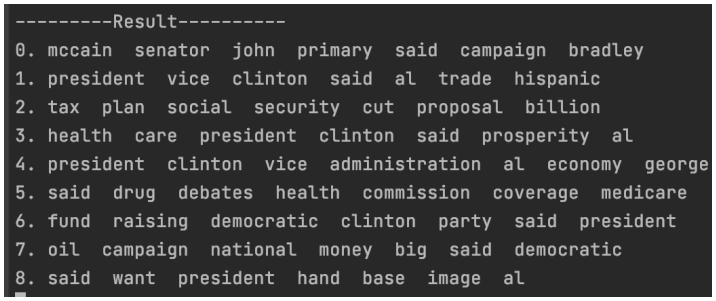
In the Result section we output the top words for significant topics from iteration **5** with **Mu = 1** and **tn = 30.** For these values for multiple runs we get the following outputs:

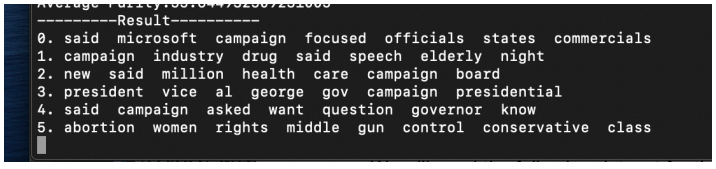


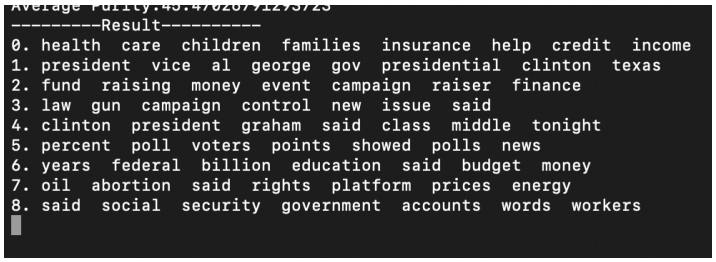




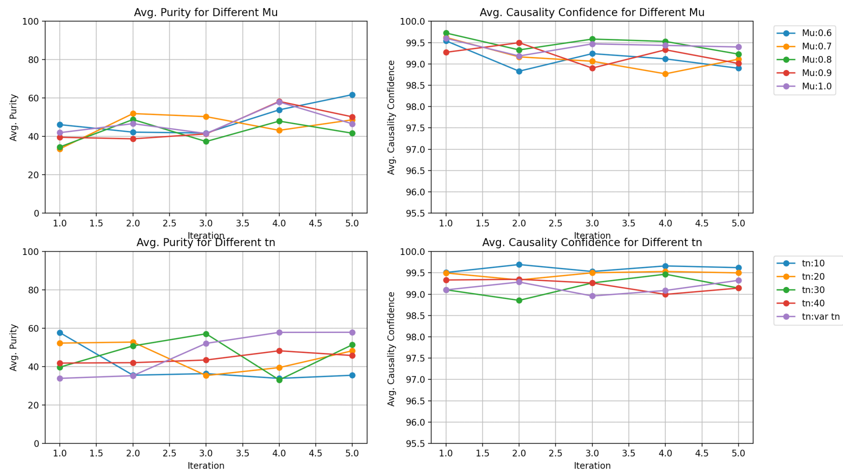


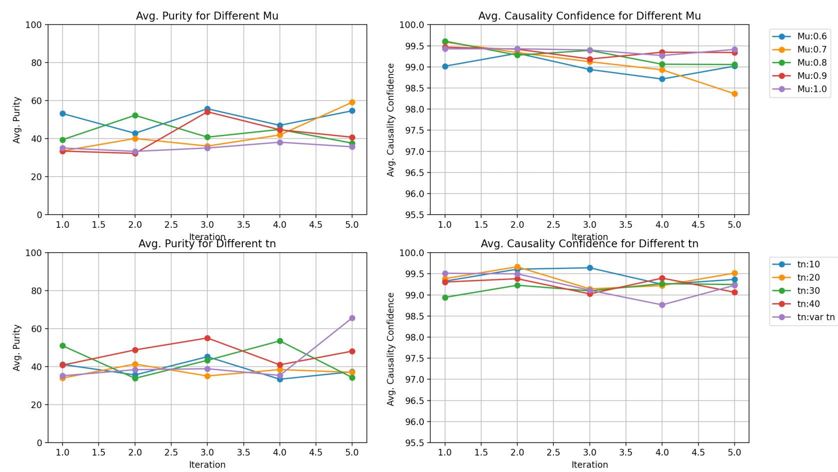


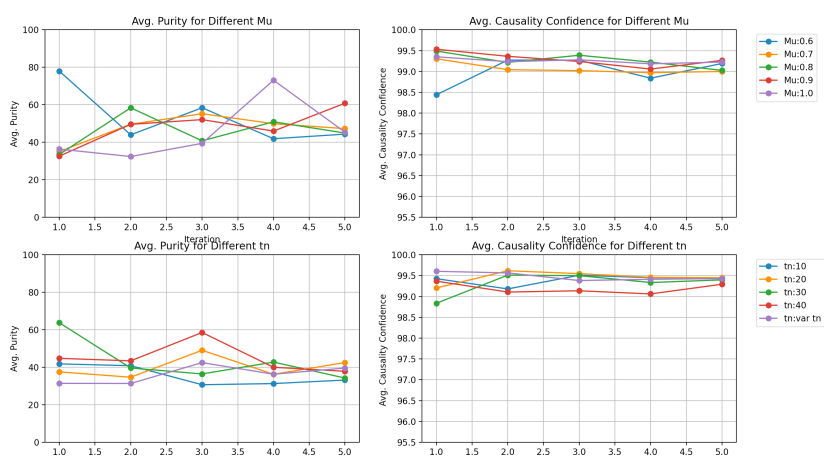




Average Casuality Confidence and Purity plot for several runs:







If we see the result we observe that some words that we observed after running the code are same as the one observed in the paper. The plots look similar to that in the paper. There are some variations seen in the plot as well as the words as there are few deviations from the paper.

- **Deviations From the paper:**
  - Probabilistic Topic Modeling : In the paper the authors have used the Lemur PLSA but in our implementation we have used Gensim LDA.
  - Mu : In paper Mu is the value that controls the weightage of the prior. Instead of using the mu values provided in the paper we considered decay as the Mu parameter. Decay in the LDA model works similar to mu and is used to control the rate of forgetting the previous values.
  - Lag : We tried 2 approaches for selecting the best lag for words. One was to use the same lag as the topic and the other was to take the best lag for word independent from that of the topic. When comparing the both we found that the second approach gave better results.

We completed the entire implementation of the paper and got results similar to that of the paper. There are words that were found in the implementation which are not seen in paper but are some important election related words. The difference in the plots and the output words between the implementation and the paper might be because of the above deviations mentioned.

3. How to run the code:

To run the code in local we may need the following files:

- Data/BushGore.txt
- preprocessor.py
- Data/StockPrices.txt
- PythonScript.py

- The directory structure should be the following:
  - Data
    - StockPrices.txt
    - BushGore.txt
  - PythonScript.py
  - preprocessor.py

Since the XML file is already processed we don't need preprocessor.py file. The code is written using python so first python is needed to be installed on the machine to run the code. Next following python packages need to be installed in the machine:

- Gensim
- nltk
- numpy
- Itertools
- scipy
- statsmodels
- matplotlib

Modules used:

- Gensim LDA:
  - The documentation for this can be found at the following website:
    - https://radimrehurek.com/gensim/models/ldamodel.html?highlight=lda#module-gensim.models.ldamodel
- Gensim preprocessor:

    - To preprocess the text data and remove stop words and tokenize the text used this Gensim module. The documentation for this model can be found at the following site:
        - https://radimrehurek.com/gensim/corpora/textcorpus.html
- Granger Casuality Test:
    - https://www.statsmodels.org/stable/generated/statsmodels.tsa.stattools.grangercausalitytests.html
- Pearson Correlation Test:
    - https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html

Once all python modules are installed just need to traverse to the directory containing the PythonScript.py file and run the following command:

python3 -W ignore PythonScript.py

4. Team Member Contribution:

Steps to be completed	Team Member	Time Spent
Project Proposal	Heta Desai	1.5 hour
Progress Report	Harpreet Siddhu	1.5 hour
Initial Analysis of Paper	Heta Desai & Harpreet Siddhu	4 hours
Read XML file and generate text document	Harpreet Siddhu	2 hours
Create text file for stock prices	Harpreet Siddhu	1 hour
Preprocess text data	Heta Desai	1.5 hours
Calculate normalized probability	Harpreet Siddhu	1 hour
Topic modeling	Heta Desai & Harpreet Siddhu	6 hours
Calculate TS	Heta Desai	2 hours
Calculate WS	Harpreet Siddhu	2 hours
Find Significant Topics	Heta Desai	3 hours
Find Significant Words	Heta Desai	2 hours
Calculate Prior	Heta Desai	4 hours
Calculate average casualty confidence	Harpreet Siddhu	2 hours
Calculate average purity	Heta Desai	1 hour
Plot Graphs	Heta Desai & Harpreet Siddhu	3 hours
Integrate all the logic	Heta Desai	3 hours
Final Document Creation	Heta Desai	2.5 hours
Final Presentation Creation	Harpreet Siddhu	2.5 hours