DS-GA 1015, Text as Data
Marco Morucci
Assignment date: March 16, 2024, Due date: April 3, 2024

Homework 3

This homework must be turned in on NYU Brightspace by **6pm, April 3, 2024**, even if submission on Brightspace is open until a later time. Late work will incur penalties of the equivalent of one third of a letter grade per day late.

It must be your own work, and your own work only—you must not copy anyone's work, or allow anyone to copy yours. This extends to writing code. You may consult with others, but when you write up, you must do so alone.

Your homework submission must be a PDF or HTML report, containing all written answers and code, generated from `IPython`. **Raw `.py` or `.ipynb` files will not be accepted.** You are responsible for making sure that your homework is fully readable in .pdf format. Anything that is not readable will not be graded.

Please remember the following:

- Each question part should be clearly labeled in your submission.
- Do not include written answers as code comments. We will not grade code comments.
- The code used to obtain the answer for each question part should accompany the written answer.
- **Your code must be included in full, such that your understanding of the problems can be assessed.**
- Please make sure that code lines are not cut (by breaking up any line of code longer than 80 characters).
- Please include your **Name** and **NetID** in your Assignment
- Assignment files should be named according to the following format: TALastName_StudentLastName_hw3.pdf

---

Note All the questions in this assignment will require you to use the `gensim` python library. Make sure to read the documentation for this library (available here: https://radimrehurek.com/gensim/apiref.html)well, as it contains many functions that can quickly produced outputs needed to answer the questions in this assignment.

1. (20pts.) For this exercise you will use a database of tweets about the 2020 US presidential election (source).

   The data are available in the file `electiontweets.rds`.

(a) (2pts.) Collapse the tweets at the day-hashtag level. That is, concatenate the tweets for each day and main hashtag ("donaldtrump" and "joebiden" included in the variable hashtag). Create a table that shows how many time periods (dates) and tweets are associated with each hashtag.

(b) (2pts.) Using the day-level dataset you created in the prior question, remove non ASCII characters, solitary letters and create a document term matrix with of this subset in which punctuation and numbers are removed and set to lower case. Remove stopwords, "http","https","rt" and "t.co". Report the remaining number of features and the total number of documents in the DTM.

(c) (2pts.) Preprocessing decisions can have substantive impacts on the topics created by topic model algorithms. Make a brief (1 paragraph) argument for or against removing rare terms from a DTM on which you plan to fit a topic model.

(d) (2pts.) Fit a topic model with 10 topics using gensim. Use 1000 iteration and 5 passes. Report the bound parameter of your topic model object evaluated on the training data.

(e) (4pts.) Examine the top 10 words that contribute the most to each topic. You should save the top 10 words over all 10 topics, for later use. Next, find the two most likely topics for each document. Create a table with one column of topic numbers and one column containing the number of documents for which that topic is the most likely topic. Sort the table in decreasing order by the second column. Show this table in your answer.

(f) (2pts.) Now, using the posterior probability of each topic over all documents, determine which are the top 5 topics and give substantive labels to them (i.e. by looking at the most likely words within each of these topics). Show the top ten words for each of these topics and briefly explain your choice of labels.

(g) (4pts.) Next, find the contribution of a topic to a document from a particular hashtag ("donaldtrump" and "joebiden"), and compare the two main hashtags on particular topics. For each of the 5 topics you've named, see how their prevalence varies among the two hashtags. To do so, estimate the mean contribution of each topic over each hashtag using the topic posterior probabilities. Report this contribution for each of the topics to each hashtag. Your response should be a table. Discuss your findings.

(h) (2pts.) Finally, look at the relative fit of different k models using the function log_perplexity() on the training corpus. Re-run the model from question (d) but this time using k=5 and k=100. Compare the perplexity of the three models (k=5, k=10 and k=100). Which could be considered the best in terms of their perplexity? Discuss your findings.

2. (10pts.) We now want to see how stable these topics are, under two different topic parameter

values.

(a) (2pts.) Re-run the model from question 1 with lower priors for α (the hyper parameter for Dirichlet distribution determining the term distribution η for each topic; the default in `gensim` is $\frac{1}{k}$, where k is the number of topics) and δ (the hyper parameter for the Dirichlet distribution determining the topic distribution for each document; the default is 0.1). Set α to 0.1 and η to 0.5. Make sure to set the seed to 1234 again and run the same number of iterations and passes. Report the bound of your topic model object.

(b) (4pts.) For each topic in the new model, find the topic that is the closest match in the original run in terms of cosine similarity of the topic distribution over words. Your answer should be a table. How does changing the priors for α and η affect the results?

(c) (2pts.) Calculate the number of words in the top 10 words shared by each matched topic pair. Your answer should be a table.

(d) (2pts.)Now run two more models with different α and η values, but this time, use only 3 topics. Again, find the average number of words in the top ten shared by each matched topic pair. How stable are the models with 3 topics compared to the models with 10 topics?

3. (10pts.)The Hierarchical Dirichelet Process is designed to automatically determine the optimal number of topics for LDA on a corpus. In this question you will experiment with its implementation in the Gensim package

(a) (2pts.) Initialize and fit a HdpModel to the election tweet data (aggregate by hashtag/day, as in the previous question). Do not change any of the default hyperparameter values. Report the top 5 topics, as well as the top 10 words for each topic as identified by the model. Are they similar to the top 5 topics you identified in the previous question?

(b) (4pts.) Record the most prevalent topic for each document in the corpus, and make a histogram using the resulting data i.e., every bar should be a topic, and the bar height should be the amount/proportion of docs where that topic is the top. You may exclude topics that are never top from your plot.

(c) (4pts.) Choose one topic and give it a label based on its highest-probability words. Create a plot that shows how the prevalence of that topic varies over time for each hashtag. Substantively interpret what you see.

4. (10pts.) For this question use the `news_data.csv` file. To reduce computation time, use the *first* 1000 headlines from the "POLITICS" category.

(a) (4pts.) Obtain the document term matrix (DTM) of the corpus, removing stopwords, punctuation and lower-casing. Perform a principal components analysis (you may use any implementation of PCA you wish) on the resulting DTM and rank the words on the first principal component according to their loadings. Report the top 5 with the most positive loadings and the top 5 with the most negative loadings. Is the first principal component interpretable? If so, what would be your interpretation of what it is capturing?

(b) (4pts.) Using the LsiModel from the gensim library, estimate a latent semantic analysis model. According to the resulting term vector matrix, report the 5 nearest tokens to `immigration` and `police`. Did the model do a good job of capturing 'meaning'? In other words, do the nearest neighbors for these words make sense?

(c) (2pts.) GloVe embeddings are another way to represent words in numeric space. Gensim allows you to quickly obtain these representations from pretrained models. Using gensim.dowloader, download the `glove-twitter-25` word embeddings. Using these embeddings print the 5 most similar words to police and immigration. Are they similar to those obtained by LSA? Would you say these are more meaningful?