DS-GA 1015, Text as Data
Prof. Arthur Spirling
Assignment date: April 10, 2019

Homework 3

This homework must be returned to Pedro Rodriguez' mailbox (2nd floor, 19 West 4th Street) by **5pm, May 3, 2019**. 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 them 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 in one of the following formats: (1) A set of answers and a clearly commented `R` code appendix (**use comments to identify code relevant to each answer you produced**), (2) A report consisting of clearly marked answers, each accompanied by the relevant code (e.g., a report generated using `rmarkdown`, `knitr`, or similar). **In either case, your code must be included in full, such that your understanding of the problems can be assessed. Homework that does not conform to these guidelines will not be graded.**

For the following exercises, it is recommended that you use the following packages: `topicmodels`, `lda`, and `stm`.

---

1. **Applying `topicmodels` to the news corpus**:

   (a) To decrease the time it takes to fit a topic model, we will limit our analysis to a subset of the immigration corpus. Create a subset of `data_corpus_immigrationnews` that only contains articles from the following news sources: `telegraph`, `guardian`, `ft`, `times` and `sun`. Create a table that shows how many documents are associated with each newspaper.

   (b) Create a document term matrix with your new immigration corpus in which punctuation and numbers are removed and words are stemmed and set to lower case. Also, remove a custom set of stopwords `custom_stopwords` (available on GitHub) that is relevant to this particular data set. Finally, use `quanteda`'s "dfm_trim" to remove words that occur fewer than 30 times or in fewer than 20 documents. Report the remaining number of features and the total number of documents in the DFM.

   (c) 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 dfm on which you plan to fit a topic model.

   (d) Fit a topic model with 30 topics using `LDA()`, with `method = "Gibbs"`. Increase the number of iterations to 3000 to ensure that the model describes the underlying data well and set the seed to 10012 so that you can replicate your results. Report the @loglikelihood of your topic model object.

(e) Examine the top 10 words that contribute the most to each topic using `get_terms()`. Find the most likely topic for each document using `topics()`. Rank topics according to the number of documents for which they are the most likely topic and label the top five (i.e. by looking at the most likely words withing each of these topics). Explain your choice of labels. You should save the top 10 words over all 30 topics, for later use.

(f) Examine the topics that contribute the most to each document, using the code from recitation to visualize the top two topics per document for the Sun and the Time with separate graphs for each newspaper. Make sure that the documents are sorted by day of publication (the "day" variable in the `data_corpus_immigrationnews` corpus). Discuss your findings.

(g) Finally, we can find the average contribution of a topic to an article from a particular newspaper, and compare newspapers on particular topics. For each of the 5 topics you've named, see how their prevalence varies among the different newspapers. To do so, estimate the mean contribution of each topic over each newspaper. Report the contribution of each of the top 5 topics to each of the 5 newspapers. Discuss your findings.

2. **Topic stability:** We want to see how stable these topics are, under two different topic parameter values.

   (a) Re-run the model from question 1 with a different seed. Report the @loglikelihood of your topic model object.

   (b) 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.

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

   (d) Now run two more models, but this time, use only 5 topics. Again, find the average number of words in the top ten shared by each matched topic pair. How stable are the models with 5 topics compared to the models with 30 topics?

3. **Topic Models with covariates:** The Structural Topic Model (STM) is designed to incorporate document-level variables into a standard topic model. Since, we have information about both the newspaper and the date of the articles, we can use an STM (from the `stm` package)

to model the effects of these covariates directly.

(a) Using only articles from the Sun and Times, construct a numeric date variable from the "day" variable in the immigration news corpus. Use what preprocessing you believe to be appropriate for this problem. Discuss your preprocessing choice.

(b) Fit an STM model where the topic content varies according to this binary variable, and where the prevalence varies according to both this binary variable and the spline of the date variable you've created. Be sure to use the spectral initialization and set k=0, which will allow the STM function to automatically select a number of topics using the *spectral learning* method. Keep in mind that this function is computationally demanding, so start with the minimum threshold document frequency threshold set to 10; if your computer takes an unreasonably long time to fit the STM model with this threshold, you can raise it to as high as 30.
Report the number of topics selected in the fitted model. Also report the number of iterations completed before the model converged.

(c) Identify and name each of the 5 topics that occur in the highest proportion of documents using the following code:[1]

```
    plot(fit.stm, type = "summary")
```

(d) Using the visualization commands in the `stm` package, discuss one of these top 5 topics. How does the content vary with the paper discussing that topic? How does the prevalence change over time?

4. **Non-Parametric Scaling - Wordfish:** Recall that the Wordfish algorithm allows us to scale political texts by a latent dimension. We will apply this function to analyze the UK manifestos.

(a) First, create a corpus that is the subset of the `data_corpus_ukmanifestos` that contains only speeches by the Conservative ('Con') and Labor ('Lab') parties.

(b) Quanteda's implementation of Wordfish, `textmodel_wordfish`, requires that we provide the indices for a pair of documents to globally identify θ, the latent dimension of interest (e.g. lower values of θ = more Liberal, higher values of θ = more Conservative). In this case, we are looking to estimate the latent left-right ideological dimension. Use the indices of the 1979 Labor and Conservative manifestos to do so. That is, set dir = c(index of 1979 Labor manifesto, index of 1979 Conservative manifesto).

[1]`fit.stm` Represents the output of the STM model you fit in the preceding question.

(c) Which of the documents is the most left wing? Which is the most right-wing? Are these results surprising? Why or why not?

(d) Re-create the "guitar plot" from recitation. Describe the parameters estimated by Wordfish that lie on the axes of the plot.

(e) **Optional:** Estimate a linear regression with the Wordfish score as the dependent variable and binary variable indicating whether or not a Manifesto was from the Labor party. Include a binary control variable for each manifesto. If we use being Labor as a proxy for 'liberal' ideology (in the American sense of the word), how well did our Wordfish model do at capturing latent ideology?[2]

5. **Burstiness:** Here we evaluate the burstiness of several words using the `news_data` corpus of news headlines. To evaluate burstiness we will use the `bursts` package and the user-written function `bursty` from recitation that visualizes the results. You can download the `news_data` corpus data from GitHub.

   (a) Create a corpus. For each of the words "trump", "korea", and "afghanistan" use the `bursty` function to visualize the burst period(s) and levels. Also, for each of the plots include a brief interpretation about what the timing and level of the burst may indicate.

6. **Dimension Reduction and Semantics:** For this question use the `news_data.rds` (on GitHub). To reduce computation time, use the first 1000 headlines.

   (a) Obtain the document feature matrix (DFM) of the corpus, removing stopwords, punctuation and lower-casing. Perform a principal components analysis on the resulting DFM 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) Using the `lsa` package, estimate a latent semantic analysis model. According to the resulting term vector matrix, report the 5 nearest tokens to `korea` and `corruption`. Did the model do a good job of capturing 'meaning'? In other words, do the nearest neighbors for these words make sense?

   (c) Load the pretrained GloVe embeddings provided. Using these embeddings, find the nearest neighbors to `korea` and `corruption`. Do these make sense? How do they compare to the nearest neighbors using LSA (keep in mind, these embeddings were estimated on a much larger corpus comprising Wikipedia and Google News articles)?

[2] If it did well, then our proxy variable for ideology should be significant at at least a 5% level.