POL 574, Quantitative Analysis IV
Prof. Arthur Spirling
Assignment date: April 14, 2025

Homework 3

This homework must be turned in on Canvas by **5pm on Monday, April 28, 2025**. 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 `RMarkdown`. **Raw `.R` or `.Rmd` files will not be accepted.**

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.**

There are plenty of resources to get started with `RMarkdown` online, see for instance here. You can also use the `template.Rmd` template in our GitHub repo to get started. Using this template is not required.

Part 1

1. Let's try out a Support Vector Machine (SVM) classifier for the Amazon reviews data from the previous assignment. Since SVM functions are computationally intensive, restrict your analysis to a random sample of 1000 Amazon reviews (set seed to 123). Again, create a `dfm` with the following preprocessing: remove punctuation, remove numbers, set to lower case, stem terms, remove stop words, and drop terms that appear fewer than 10 times in the corpus.

   (a) Describe an advantage offered by SVM or Naive Bayes relative to the dictionary approach or wordscores in classifying positive and negative reviews.

(b) In this step, you will train SVM models with a linear kernel. Your goal is to maximize out-of-sample accuracy by fitting models with 5-fold cross-validation. Optimize over the relative size of the training and test sets, try each value from 10% to 90% (by 10s). The remaining data is the validation set. In other words, you should train 10 different models with 5-fold cross-validation. For example, the last model in this sequence uses 90% of the data for the 5-fold CV, and 10% is saved as the validation set. Report which model has the highest accuracy for out-of-sample predictions made on the validation set. In terms of accuracy, how would you evaluate the performance of this classifier?

(c) Using the model fitted over the *optimal* train-test split, produce a Receiver-Operating Characteristic (ROC) Curve with the False Negative rate on the x-axis and False Positive rate on the y-axis. Print the AUC value for the curve as well.

(d) A colleague is building a dictionary for this problem. Using the optimal SVM you identified in 1b), report the five most predictive words for a positive and for a negative review from the SVM results to help them start that process.

(e) Can we improve the performance of our SVM classifier by including bigrams in our model? To evaluate this, create a `dfm` that contains *both* unigrams and bigrams. Make sure to use the same preprocessing as in 1a). Using this data, train another SVM model with a linear kernel and 5-fold cross-validation. You can impose the same relative size of the training and test sets as before (i.e., you do not need to redo 1b). Report the accuracy of this model out-of-sample. How does this compare to your findings in 1b)?

2. Let's now try out a Random forest classifier. For this question use a random sample of 500 reviews in the Amazon dataset. Split the dataset into a training (80%) and a test set (20%) and construct a document feature matrix for each using the same preprocessing choices as before (Note: features in the test set should match the set of features in the training set).

(a) Using the `randomForest` package fit a random forest model to the training set using the package's default values for `ntree` and `mtry` (also, set the `importance` argument to TRUE). Having fitted the model, extract the *mean decrease in Gini* importance for the feature set and order from most important to least important. What are the top 10 most important features according to this measure?

(b) Using the fitted model, predict the product rating values for the test set and report the confusion matrix along with accuracy, precision, recall and F1 score.

(c) Now you will do some tuning of one the two model parameters. The package's default value for the argument `mtry` is $sqrt(\#offeatures)$. Estimate two more models, one for each of two different values of `mtry`: $0.5 * sqrt(\#offeatures)$ and $1.5 * sqrt(\#offeatures)$. As you did above, use each of the fitted models to predict the product rating values for the test set and report the respective accuracy scores. Which value

of `mtry` yielded the best accuracy?

3. **Principal Components and Clustering** This question uses (abridged) data on some US States. The dataset is named `states_data.rdata`.

   (a) Set the seed to 36. Do a $k = 2$ means clustering on the states data. Inspect the clustering vector. What states appear to be in cluster 2 as opposed to cluster 1? How would you label these clusters?

   (b) Do a $k = 3$ means clustering on the states data. Inspect the clustering vector. What states appear to be in cluster 3 as opposed to cluster 2 and 1? How would you label these clusters?

   (c) Going only by the between versus total sum of squares ratio, which model fits the data better?

   (d) Do a PCA of the data via `prcomp` with scale set to true. Plot the variance explained as a function of the number of components. Is there an elbow? Where?

   (e) Plot the states scores in two dimensions (that is, for the first two PCs). Use their names or IDs as plotting characters.

   (f) Look at the factor loadings ('rotations") for these first two PCs. What might they represent?

   (g) Produce a distance matrix (Canberra distance) of the data, and use multidimensional scaling with two dimensions. Plot the states's scores. How does this plot differ to the previous PCA projection? Which one is correct?

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

4. Applying `topicmodels`: For this exercise you will use a database of Parliamentary debates (source).

   The data are available in the file `gchq_speeches.csv`.

   Note: Because topic models take a long time to run, you may save your fitted model objects or RStudio workspace to a `.RData` file after running the final version of your code to fit the models. When doing your write-up, you may comment

out the code that trains the model(s) and load the saved models from file. Please make sure the code to fit the models is clearly marked nonetheless.

(a) Collapse the speeches to the year level. That is, paste together the speeches within each year. Also create a table that shows how many documents are associated with each year. Also, create a variable indicating whether a given year is before or after the Cold War (e.g. post-1991).

(b) Remove non-ASCII characters, solitary letters and create a document term matrix of this subset in which punctuation and numbers are removed and set to lower case. Remove stopwords. Report the remaining number of features and the number of documents (years) in the annualized 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 10 topics using `LDA()`, with `method = "Gibbs"`. Set the number of iterations to 3000 to ensure that the model describes the underlying data well and set the seed to 1234 so that you can replicate your results. Report the @loglikelihood of your topic model object.

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

(f) Now, using the Bayesian point estimate for each topic, i.e. the average of 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) Next, find the average contribution of a topic to a document from a particular period (during the Cold War versus after it ended), and compare the two periods on particular topics. For each of the 5 topics you've named, see how their prevalence varies among the two periods. To do so, estimate the mean contribution of each topic over each period using the topic posterior probabilities. Report this contribution for each of the topics to each period. Your response should be a table. Discuss your findings.

(h) Finally, look at the relative fit of different k models using the function `perplexity()`. 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.

5. **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 the period in which speeches are occurring, we can use an STM (from the stm package) to model the effects of the international political context directly.

   (a) Construct a numeric binary variable from the `post-Cold War` variable in this restricted data. 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 this binary variable. 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 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 international context (the binary variable created) in which that topic is discussed?

6. **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 2023 U.S. Gubernatorial State of the State addresses, available in the `sots.rdata` object (source).

   (a) First, create a corpus that includes all speeches as separate documents. Make sure to provide the governor and year as document variables of the corpus. Create a dfm with the preprocessing choices you deem appropriate (i.e., you may want to trim features that appear in fewer than seven total documents using `quanteda`'s `dfm_trim`, to filter out terms that are used fewer than five times total, or in fewer than three documents.).

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

(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 2023 Phil Murphy and Ron DeSantis SOTS speeches to do so. That is, set `dir = c(index of Murphy speech, index of DeSantis speech)`.

(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) Generate a plot with the term on the x-axis and term fixed effects on the y-axis. Adjust the axis dimensions to ensure interpretability and readability of the plot. Describe the parameters estimated by Wordfish that lie on the axes of the plot.
Hint: see the `textplot_scale1d()` function from quanteda.

(e) **Optional:** Estimate a linear regression with the Wordfish score as the dependent variable and binary variable indicating the party of a given speech as a predictor. If we use being a Democrat as a proxy for 'liberal' ideology (in the U.S. politics sense of the word), how well did our Wordfish model do at capturing latent ideology?[2]

7. **Dimension Reduction and Semantics:** For this question use the `news_data.rds`. For this question use news data. To reduce computation time, use the *first* 1000 headlines from the "POLITICS" category.

(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 `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?

8. This question requires use of the `conText` package and `lsa`. Take a look at the user guide here. You will need GloVe embeddings which you can find here, and various items pertaining to the Levellers and texts they wrote, distributed by us.

(a) This first question uses the modern GloVe embeddings. Using the packages, report the 9 nearest neighbors (via cosine distance) for the terms "sovereign" and "rights".

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

(b) This question uses some embeddings trained on Early Modern English, which was prevalent in the 17th C when the Levellers were active. You will need `all_embed` from `embeddings.rdata` (download here). Report the 9 nearest neighbors for the same terms as above. How do they differ?

(c) Using the pretrained embeddings, and `local_transform` as your transformation matrix, produce a document-embedding matrix (`dem`) for the `peers_dfm` from `levellers.rdata`. This latter object is, essentially, a document feature matrix for Levellers and non-Levellers of the period in terms of how they used the term "peers". Then, use the `dem_group` function, setting `groups` to be defined by `leveller_expansive` variable. Finally, use the `nns` function to produce the set of 20 nearest neighbors for Levellers and non-Levellers for this period, for the term "peers". You can set `candidates` to `peers_lev_nonlev@features`. What are the nearest neighbors? How do Levellers compare to non-Levellers?

(d) Finally, using `the_tokens` from the `tokens.rdata` object as your data, do an embedding regression. Specifically, you want to see if "peers" varies in use for the covariate `leveller_main`. Use the embedding matrix and local transform as above, and do 1000 permutations. Set `case_insensitive` to true. Do the Levellers and non-Levellers differ, statistically, in terms of how they use the term "peers"? How do you know? Report the regression results.