**CBA Batch 12**

**TABA Group Assignment**

**Assignment Instructions:**

1. Coding Scheme for this assignment is 2N-b.
2. This is a group assignment. Team size should be maximum 3.
3. Submit the individual files (Word, PDF etc.) on LMS. Do not zip the files.
4. ZIP Files will not be evaluated.
5. If there are any code files kindly copy paste the codes in the word file and upload both the word file and the code files. For example: if you have the code file “sample.R” then copy paste the code of “sample.R” in the word file say “sample.docx” and upload both the files “sample.R” and “sample.docx”.

**Q1: Topic Model Simulation**

Perform a simulation experiment along the lines of what we did in class (based on the tidytext book example). Use the following steps as a guideline.

Step 1: Choose 4 very different domains and scrape data from there (e.g., Wiki pages on Cricket World Cup 2019, Iron Man, Justin Bieber or Narendra Modi etc). Ensure the corpus has adequate length in each case.

Step 2: Break each Wiki page into sentences using a sentence tokenizer.

Step 3: Now construct simulated documents consisting of a few sentences from page 1, a few from page 2 and so on. Each document should be 8 sentences long. For instance, suppose there are 8 sentences in each simulated document, then you could choose the distribution of sentences in the following way:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Simulated Document # | Corpus 1: Cricket World Cup Wiki Page | Corpus 2: Iron Man Wiki Page | Corpus 3: Justine Bieber Wiki Page | Corpus 4: Narendra Modi Wiki Page |
| Doc 1 | 2 sentences (25%) | 2 sentences (25%) | 2 sentences (25%) | 2 sentences (25%) |
| Doc 2 | 4 sentences (50%) | 0 sentences (0%) | 2 sentences (25%) | 2 sentences (25%) |
| Doc 3 | 0 sentences (0%) | 8 sentences (100%) | 0 sentences (0%) | 0 sentences (0%) |
| Doc 4 | 0 sentences (0%) | 0 sentences (0%) | 4 sentences (50%) | 4 sentences (50%) |
| Doc 5 | Etc |  |  |  |

Note above that the sentences in each document could appear in any order. Randomly choose sentences from each wiki page and insert them to create simulated documents. Create at least 50+ simulated documents containing 8 sentences each in this manner.

Step 4: Now run an LTM on this simulated corpus with K=4 topics. Obtain the factor score and factor loading matrices from the topic model output.

Step 5: For each document, check the topic score for each topic. See how closely it matches the percentages we calculated in the table. For example, does ‘Doc 1’ have around 25% topic score for topic 1? And so on. One could compute the mean squared deviation for each document between the recovered topic scores and the constructed topic proportion.

Step 6: Bonus points - Draw a confusion matrix of the 4 topics versus the 4 wiki pages to see if they were correctly or wrongly classified.

**Deliverable**: Build a markdown showing your workflow, results and discussion. Put the markdowns and the word files and upload to the LMS link within the deadline. Remember to write the names and Student ID numbers of all students in the group.

**Q2. Text Classification Simulation**

**Step 1:** Use the same documents as created in Q1.

**Step 2:** Split the documents into training and test dataset with 70:30 ratio.

**Step 3:** Implement the text classification methods explained in Session 5 on training dataset and calculate the accuracy percentage for each method on the testing dataset.

Deliverable: Build a jupyter notebook showing your workflow, results and discussion. Upload the jupyter notebooks and word files to the LMS link within the deadline. Remember to write the names and Student ID numbers of all students in the group.

**Q3: Building a Shiny App around the UDPipe NLP workflow**

Recall sessions 4 and 5. In session 4, we learned different techniques for natural language processing using R and Python. In session 5, we learned how to build shiny apps. In this group assignment, you have to build a shiny app using ***“udpipe”*** R package.

Shiny app should have these features.

A – Should be able to read any text file using standard upload functionality.

B – English language model should be included within the app.

C – User should be able to select list of Universal part-of-speech tags (upos) using check box for plotting co-occurrences. List of upos required in app –

adjective (ADJ)

noun(NOUN)

proper noun (PROPN)

adverb (ADV)

verb (VERB)

Default selection should be adjective (ADJ), noun (NOUN), proper noun (PROPN). Based on upos selection, filter annotated document and build co-occurrences plot

D – App should have at least 4 output tabs. Details of output tab is as follows –

1. First tab should describe your app.
2. Second tab should display a table of annotated documents (use *udpipe\_annotate* function from udpipe) as data frame. Drop sentence column from the data frame. Use dataTableOutput function for displaying the output in shiny. Show only about a 100 rows of the annotated DF in the app and give user an option to download the full file as a .csv.
3. Third tab should display two wordclouds, one for all the nouns in the corpus and another for all the verbs in the corpus
4. Fourth tab should display a plot of top-30 co-occurrences at document level using a network plot as mentioned in point C.

You are free to make your app more useful by adding more widgets, backgrounds, features etc. (bonus points). You can use session 4 & 5 codes if you want.

**Deliverable:** Upload your files to Github and see if you can source the app using Rstudio. If so, provide the code to runGithub() the same.

In addition, upload the codes of ui.R and server.R files to LMS within deadline. Remember to write the names and Student ID numbers of all students in the group.