Homework 1: Corpus Analysis

Brayden Cloutier

University of Michigan-Flint

ARI-525: Natural Language Processing

2/18/2025

GitHub project link: https://github.com/BraydenAC/ARI-525-Homework-1

Dataset

For this project, I chose to use a collection of chapters from two famous classic authors as my categories; H. G. Wells and Leo Tolstoy. I chose these subjects because what makes many of these old authors great are their distinct styles and personal prose, and I wanted to see whether I could reflect that through NLP.

To collect this data, I first downloaded numerous works from Project Gutenburg for both authors; in particular, for Tolstoy I chose War and Peace. With the books downloaded, I used regex to split the books into individual chapters and manually removed items such as appendices and tables of contents. Some additional information about this dataset is in the below table:

	H. G. Wells	Leo Tolstoy
Number of books	6	8
Num of resulting chapter documents	119	167
Avg tokens per doc	258542	562159

Originally, Tolstoy's folder had well over 300 documents, so I ended up deleting almost 200 in order to try and balance the scales.

Methodology

Preprocessing: I began by building a script to preprocess the categories individually into tokenized csv files. This process was fairly simple, and I elected to add three of my four preprocessing steps from Homework 0 as a function, only leaving out stopword removal so I could have something to try during the experimentation phase. Some significant libraries I used during this phase included pandas and nltk.

Bayes: With BoW_HGWells.csv and BoW_Tolstoy completed, I uploaded them into the new file Bayes.py to build my log-likelihood rankings. This part went relatively smoothly, not using any new libraries beyond the preprocessing step. After a bit of fine-tuning, results were being displayed successfully. Of note for this part is that it was my first real foray into using defaultdicts in python.

LDA: During this step, I ran into several issues. While in the bayes section worked perfectly well with a dense matrix representation of the tokenized data, attempting the same method with the combined dataset resulted in exponentially longer wait times and multiple sigkill errors. To combat this, I rebuilt the preprocessing script to be combined with the new LDA script and attempted to implement sparse scr_matrixes to hold the data instead of pandas dataframes. After a good deal of bug fixing, I managed to get the

program to run all the way through, but the result was unsatisfactory. While I will talk more about these results in the below in the appropriate section, the summary is that the words that showed up in the graph were seemingly unrelated and highly uncommon. After determinining that the problem likely lay in my token-to-id-to-token mapping, I ended up creating a new file called newLDA.py and remaking the entire program from the ground up.
While the resulting program was much more well-structured and clear than its predecessor, when I tried to run it the program froze my computer, then crashed it. With some doing, I found that the problem arose in the .prepare function that takes the model and the corpus and turns it into a real graph. Ultimately, I was not able to actually fix the problem, and my results will be based off the original script. However, I was able to implement switching between binary and count-based data.
For the LDA model, I used gensim from pyLDAvis. Sparse data storage was implemented via scipy. Finally, nltk was again used during the preprocessing stage.

Results and Analysis

Bayes:

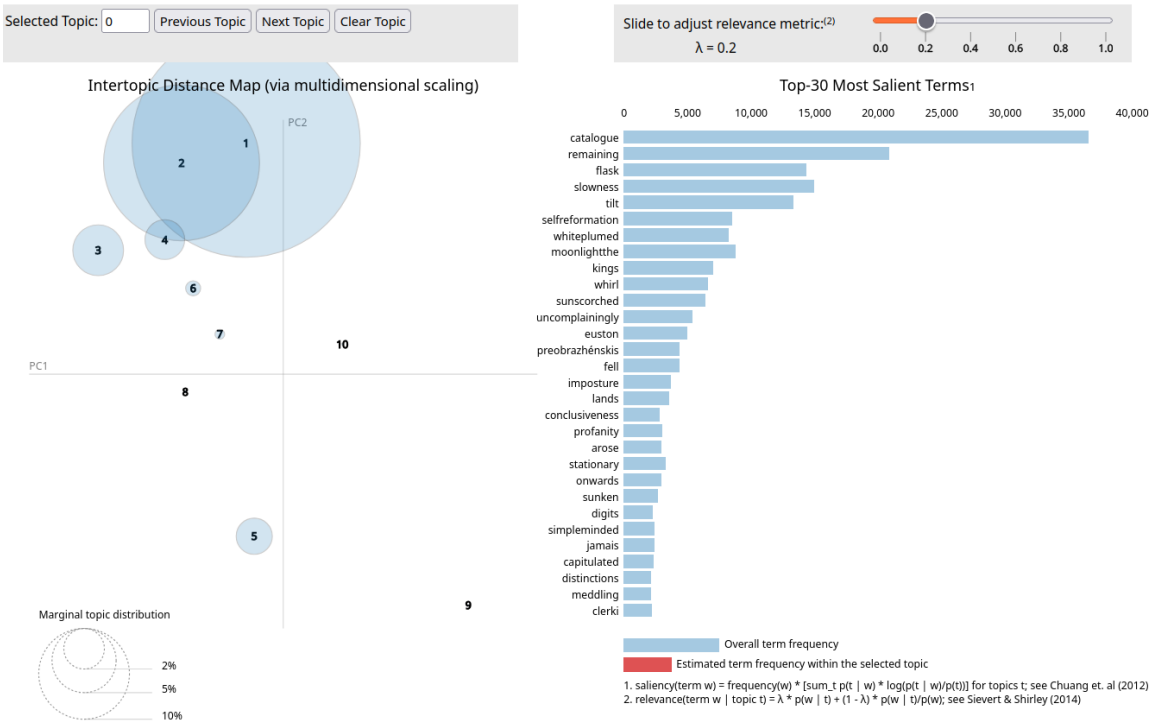
	Leo Tolstoy Values	H. G. Wells Values
1	prince: 6.821982198986033	graham: 6.893018283117246
2	pierre: 6.76376614358795	towards: 6.686986894244236
3	natásha: 6.273273937550749	kemp: 6.0799938252978025
4	andrew: 6.223568441509628	martian: 6.001522209856307
5	princess: 6.105304521015894	montgomery: 5.899557283199619
6	rostóv: 5.8502726163348555	ostrog: 5.747288071472066
7	nicholas: 5.717538990007131	marvel: 5.459605999020285
8	mary: 5.694957385310421	grey: 5.2560070437790465
9	emperor: 5.573701769308146	mrs: 5.245196127674831
10	countess: 5.460741340311693	grahams: 5.1422481584223885

Many of the words showing up were character names, which makes sense. After all, one author isn't going to use the characters of another. Tolstoy has multiple stations of royalty appear, fitting with the subject of the War and Peace series. Of note within Wells's documents is the appearance of "martian", which I can recognize as a common occurrence in more than one of his books, most significantly the War of the Worlds series. In the end, I'm just glad that my results aren't spammed by stopwords despite the absence of stopword removal in preprocessing, and that the overall list has meaning individual to its respective category.

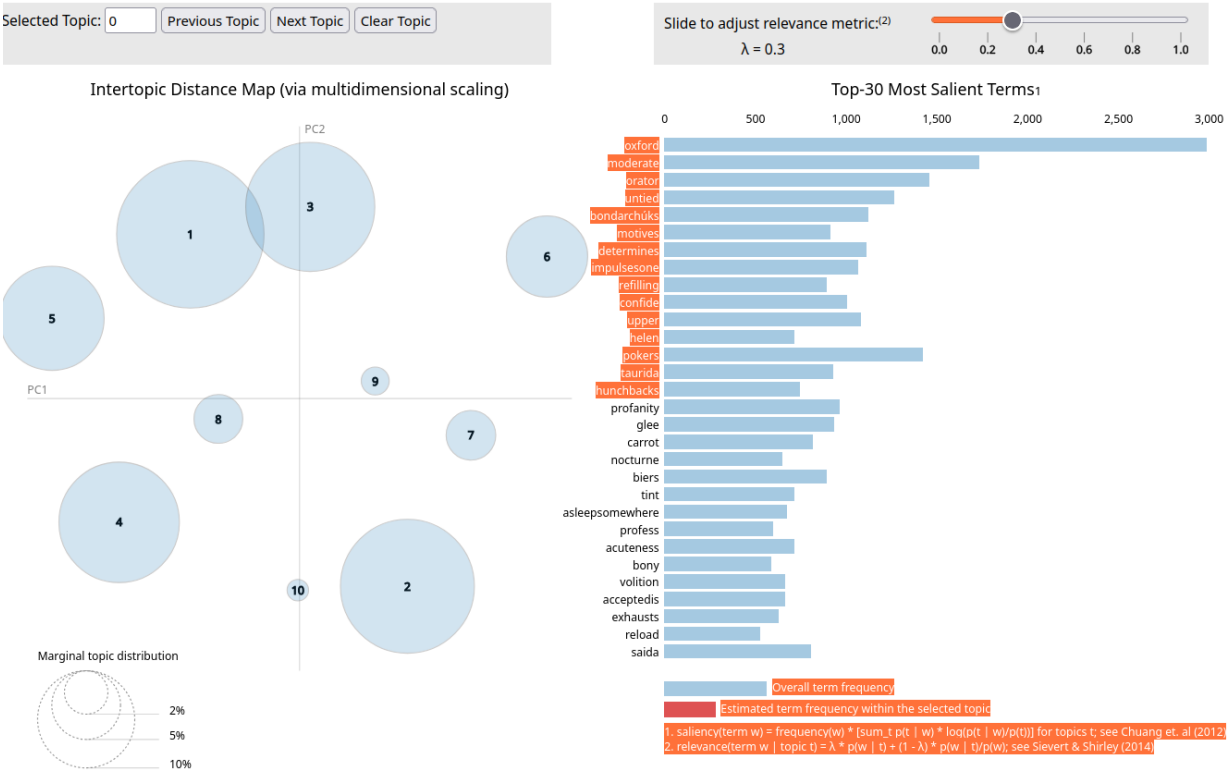
Topic Modeling:
Unfortunately, due to the issues outlined within the methodology section I was unable to get the LDA outputs mapped to the proper wording, making it impossible for me to conclusively assign labels to

each category. In order to try and gain some value out of my output, I will display my results from before and after stopword removal, and try to make observations as to how the counts and displayed tokens changed.

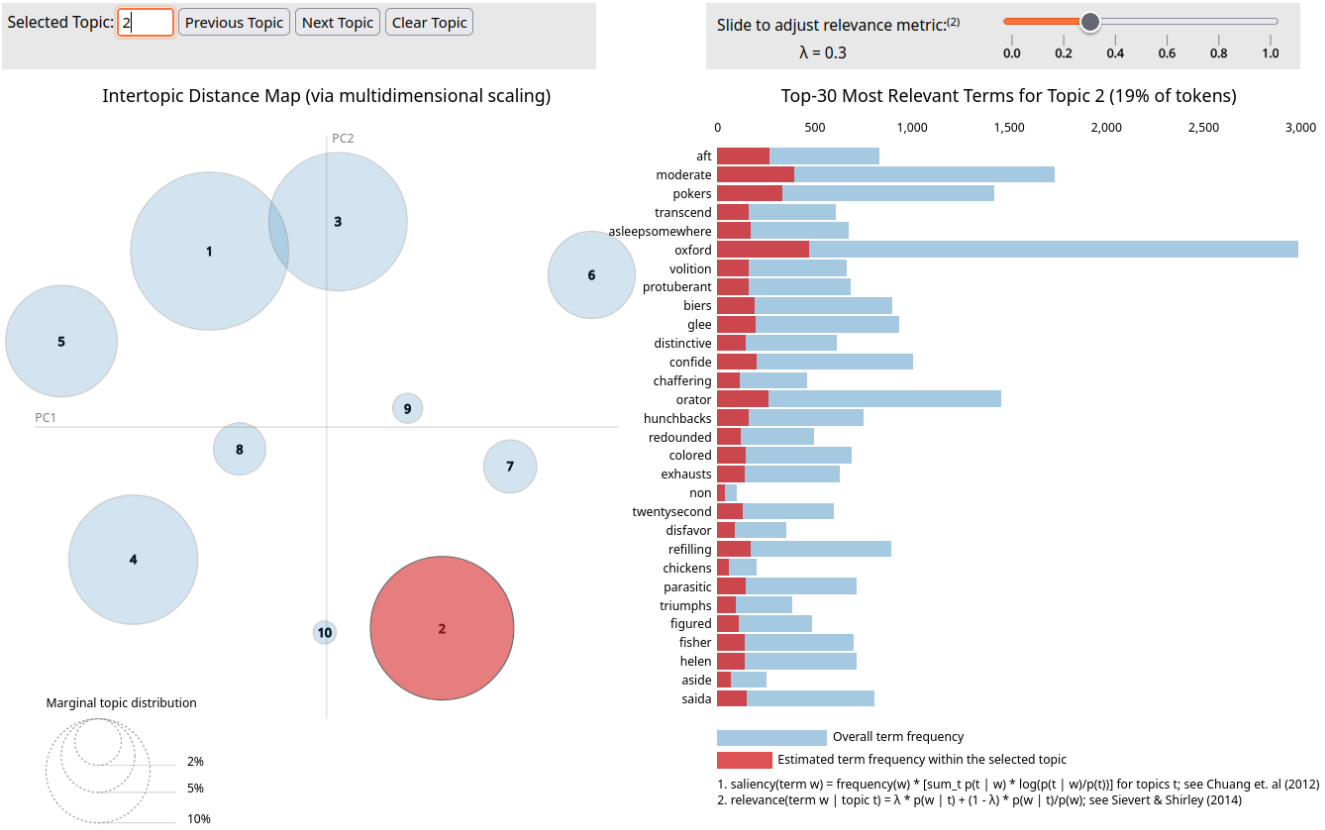
Without Stopword Removal:



With Stopword Removal:



In the first graph, all major topics seem to be clustered together. When I hover over each one, they all seem to be completely dominated by the top three overall tokens, leading me to believe that they are stopwords such as a, the, and and. This is further reinforced by the frankly ridiculous counts attributed to them, with the highest being over 30,000. On the other hand, the graph of after stopwords displays a much more spread out distribution, and much more stable overall token counts. Each topic seems to have its own unique set of words, and I am confident that with the correct word associations in place creating well defined topic labels would be simple. An example of one of these unique topic distributions can be found below.



As can be seen here, most words appearing within this list are quite different from the general Top-30 shown further above, and the list is a more natural spread of lower frequency tokens.

If I was to create categories from this graph, I would choose to take topic 1-6, as they seem to have a well spread-out sampling with unique tokens for each.

Discussion

1. In this homework assignment I displayed the value of creating visualizations for NLP data, demonstrating that even if you cannot see every detail that should be available you can still pull

inferences out of a good graph. I showed that in the case of large corpuses of english literature, stopwords can overwhelm detailed results. The final graphical outputs implied that even taking just two authors, well defined subjects can be extracted and differentiated even at a glance.

2. For this paragraph, I will explain the two main things that I learned during this assignment. Firstly, creating the Bayes.py script really helped me get the lessons involved in calculating log-likelihoods into my long-term memory, as I ended up needing to watch the related lectures multiple times and try multiple iterations. It also gave me the chance to really start learning how to add defaultdicts into my code, a concept that I hadn’t really run across before this point. Secondly, the creation of the LDA script really pushed me to learn to keep my code well structured. At multiple points throughout my debugging process did I find myself confused by my own code; a problem whose solution should be preventative in nature. I did like meshing lists with defaultdicts, it was surprisingly intuitive when I got into the swing of things.

While it was disappointing that the program did not go to plan, I believe that I have managed to gain multiple new tools that I can bring into future labs, most notably defaultdict data structures. Hopefully, I can use the lessons learned here to not fall into pitfalls in future assignments.