Part 1: Text Classification

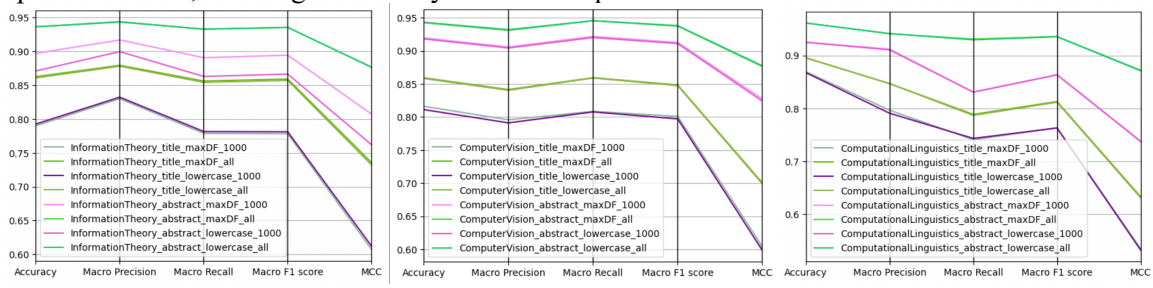
ID: 33029229 Name: SiaoHsuan, Jiang

I used max_df (to remove tokens with more than 50% document frequency) and lowercase as preprocessing methods for the statistical models. For the RNN models, I used stemming and lowercase. The other configurations are already listed in the assignment instructions, so I won't mention them here.

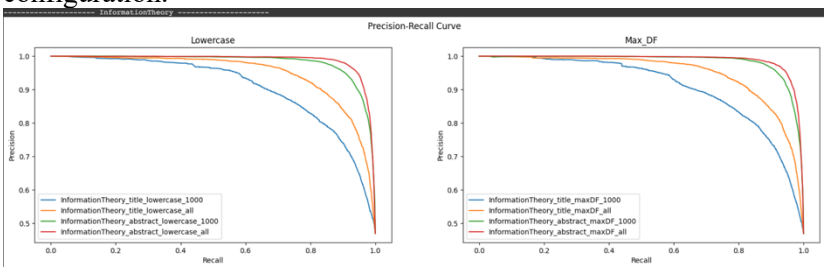
Statistical Models (Random Forest):

I have chosen to use maxDF with a threshold of 0.5 as a preprocessing step to eliminate tokens that appear in more than half of the documents. My aim is to investigate whether removing tokens with high document frequency has an impact on the prediction. In terms of lowercase, I believe that capitalized tokens usually have the same meaning as their lowercase counterparts, as capitalization is often used only because the token is the first word of the sentence.

Instead of displaying plain text, I created parallel coordinate plots to visualize the 5 scores on test data for each predicted label, enabling us to easily observe the performance across 24 statistical models.



I will discuss three comparisons: inputs, data size, and preprocessing methods. Firstly, for the data size, as expected, when all other configurations are the same, the larger the data set, the better the model performs. Secondly, for the input data, abstract performs much better than the title. All lines of the abstract scores higher than those of the title throughout three tasks, indicating that the abstract strongly dominates the title. Lastly, in terms of preprocessing methods, both methods did not show significant differences in performance (lines are overlapping), except for two pink lines in the InformationTheory task. The lighter pink line represents the maxDF configuration, which outperformed the lowercase configuration.

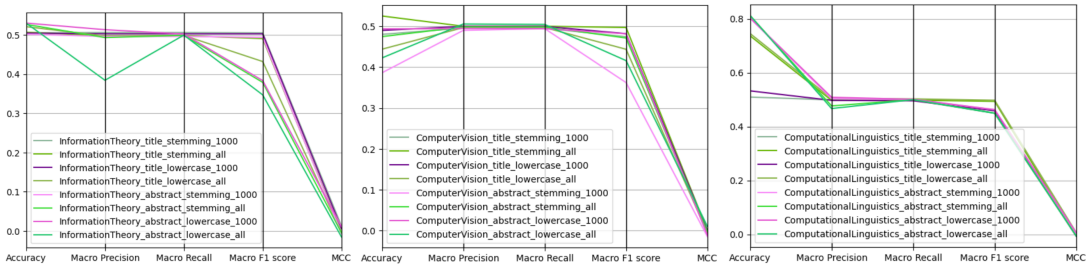


As preprocessing methods were found to perform similarly, their precision-recall curves would overlap with each other. To address this, I plotted the curves by grouping the preprocessing methods (graph on the left).

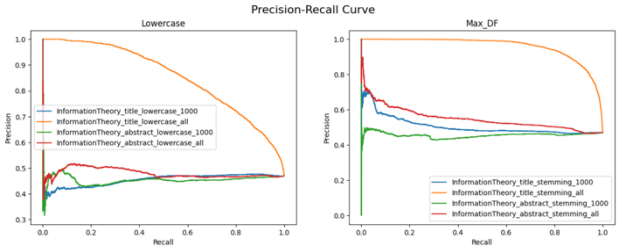
The precision recall curve indicates that the closer the curve is to point (1,1), the better the model performance. Based on this, we can conclude that the red curve performs the best, followed by the green, orange, and blue curves. This result is consistent with the previous finding that larger data sizes and abstract inputs outperform smaller data sizes and title inputs. Although the plots for the other two tasks are not shown here, their results are very similar to this one.

RNN Models:

I have chosen to use stemming data and lowercase as preprocessing methods. Stemming process can reduce some derived words to their root or base form. This is done by removing the suffixes like cats to cat that allows for more efficient analysis.

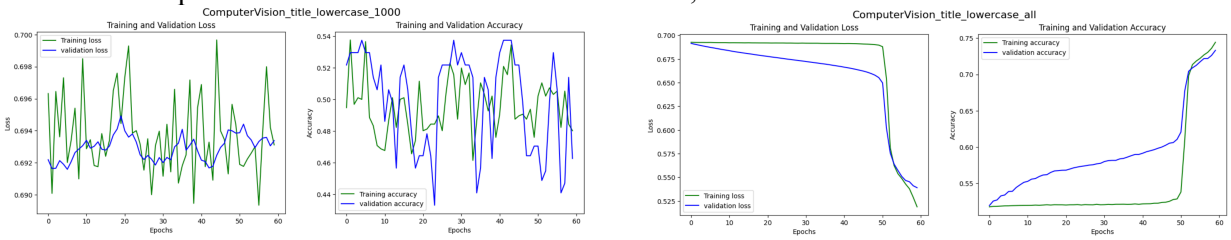


Each model is trained in 60 epochs. The results are like statistical models, there are three comparisons. Firstly, regarding the data size, when all other configurations are kept the same, a larger data set leads to better model performance on test data. Secondly, concerning the input data, the abstract performs better than the title in ComputerLinguistics, while in ComputerVision, the abstract performs slightly worse than the title. Lastly, with regards to preprocessing methods, there is no significant difference between them.

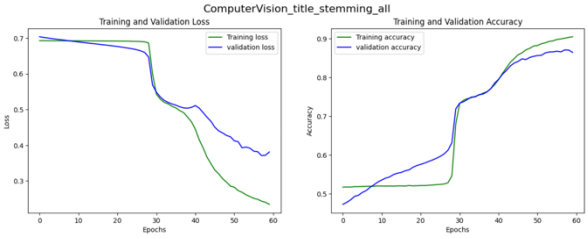


In terms of precision recall curves, all three task labels showed flat curves for the models, except for title in lowercase with all data size. A flat curve indicates that the model is not performing well and may require further tuning and optimization. Therefore, to evaluate their performance, I traced and visualized the training accuracy and loss for each epoch and for each model.

Let's consider ComputerVision_title_lowercase_1000 and ComputerVision_title_lowercase_all as examples, as showing in the graphs below. The plot on the left shows that, probably due to the data size, the model with 1000 data points did not show any significant improvement in performance with the increase of epochs. Therefore, it is pointless to increase the parameter of the epoch for this model. However, the model with all data showed a significant improvement in both the training and validation sets when the epochs were increased. The losses decreased, and the accuracies increased.



Nevertheless, on the right graph, if we observe a significant divergence between the train and validation lines, it can be concluded that the model is overfitting as the training error is decreasing and accuracy in increasing, but the validation error and accuracy are not improving.



Statistical and RNN Models:
To sum up, statistical models perform much better and are more stable than RNN models. The accuracies of statistical models can easily reach over 80%, while only a few RNN models in Computational Linguistics can achieve that level. Furthermore, most precision recall curves of RNN models look unsatisfactory. Compared to statistical models, neural network methods are much more complex. RNN models could perform much better if we tune the models in a more professional way.

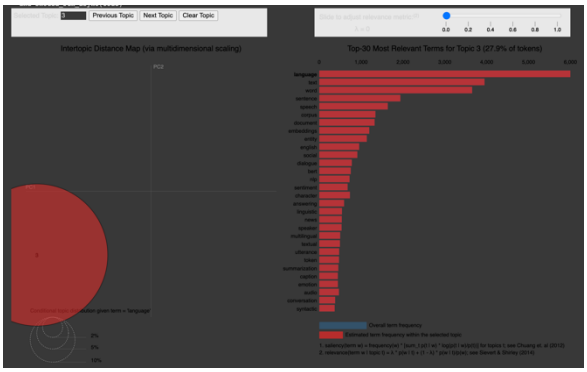
Part 2: Topic Modelling

In this section, I chose to use bigram and non-bigram as my preprocessing methods, as these have been shown to be effective in previous studies. I created four models, with configurations specified by suffixes in the variable names. For example, model_1000_T represents a model of 1000 data size and bigram=True, while model_20000_F represents a model of 20000 data size and bigram=False.
To create the models, I followed the tutorial materials and used parameters such as chunksize=2000, passes=20, and iteration=400. However, I made some modifications to the NUM_TOPICS parameter, setting it to 3 to reflect the fact that each raw in the training data had been labeled to one of three topics. This allowed the original train data to provide more information and criteria to evaluate our cluster methods.

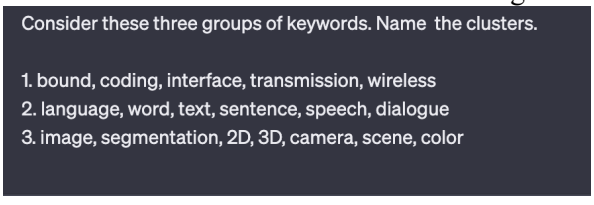
The four topic distributions appear quite similar, with three distinct clusters that do not overlap. However, the plot reveals that the two clusters on the left side of the axis are closer to each other than to the cluster on the right side, indicating a higher degree of relatedness between the former two clusters.

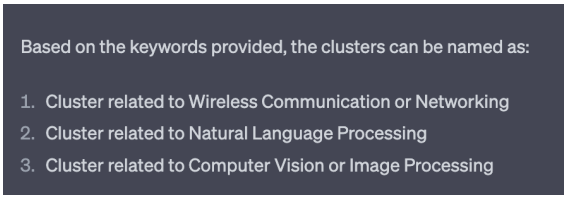
When we take a closer look at the interactive chart on the left, we can see that by setting lambda to 1, we can find out how likely a word is to appear in a topic. For example, when we hover over the word "it" on graph A, we can see that the radius of three clusters are similar in length, meaning that the word is equally possible to appear in any of those three topics. This makes sense because other common words like "I", "can", and "the" would also have similar results since they are commonly used in many types of writing.

However, there is a limitation to setting lambda to 1, so I recommend setting it to 0 instead because it provides more useful information. By setting lambda to 0, we can see how exclusively a word belongs to a certain topic on the right graph. For instance, when we hover over the word "language", we can see that the other two clusters are not even visible on the chart, which means this word rarely appears in those two topics. This also means that when the word "language" appears in a paragraph, it is highly likely that the paragraph will be labeled to the hovered cluster. I use some keywords to name the cluster.



As it's allowed to use chatGPT in this assignment, from model_20000_F, I copied some top exclusive keywords to chatGPT to seek advice on naming the clusters.





Consider exclusive keywords (lambda = 0) and the articles, we got names of the topics as follow:

- Wireless Communication System Topic:
  - Keywords: bound, coding, interface, transmission, wireless
  - Exemplars: Since radio signals carry both energy and information…, Suppose Alice wishes to send messages to Bob through a communication channel…
- NLP Topic:
  - Keywords: language, word, text, sentence, speech, dialogue
  - Exemplars: Neural text matching models have been widely used in community question answering information retrieval and dialogue…, Unlike sentences words rely on their contexts to express concrete…
- Vision Topic:
  - Keywords: image, segmentation, 2D, 3D, camera, scene, color
  - Exemplars: advancement of deep convolutional neural networks (CNN)…, for a wide range of vision tasks outperforming traditional machine learning (ML) methods…

By considering the exclusive keywords and exemplars, we have gained a significant amount of meaningful information. For instance, the term "convolutional neural networks" is a commonly used method in image recognition, while the word "dialogue" is more closely associated with the NLP topic. These examples highlight the usefulness of identifying exclusive keywords and exemplars in characterizing the topics and interpreting the results of the analysis.

Very similar to the labels in original data, System topic can be compared to "InformationTheory", NLP topic can be compared to "ComputationalLinguistics", and Vision topic can be compared to "ComputerVision".

Upon examining the original training data, I tried to evaluate the performance of my clusters. I have assumed that the three label columns in the original train data represent the "correct" labels for the dataset. I compare cluster labels to original labels for each model to test the accuracy. **After performing LDA modeling on the abstract, I found that all four of the models clustered the data well.** Surprisingly, even the model with the lowest accuracy (which was a bigram model trained on only 1000 samples) had an accuracy of over **84%**. In contrast, the highest accuracy was achieved by a non-bigram model trained on 20000 samples, which achieved an accuracy of **90%** (as can be seen in the graph below). This finding was somehow expected, as it is well known that larger datasets can improve the performance of machine learning models. Specifically, the models trained on 20000 samples achieved at least 2% higher accuracy than the models trained on only 1000 samples. However, interestingly, **the addition of bigram tokens to the documents did not improve the performance of the models**, as I initially thought it would. In fact, adding bigrams actually decreased the accuracy of the models in both the 1000 and 20000 sample cases. This suggests that adding bigrams to the documents only added noise to the clustering process, rather than providing additional useful information.

```
print(f"Accuracy of n=1000 bigram model:{cluster_accuracy(document_topics_df=document_topics_df_1000_T)}")
print(f"Accuracy of n=1000 non bigram model:{cluster_accuracy(document_topics_df=document_topics_df_1000_F)}")
print(f"Accuracy of n=20000 bigram model:{cluster_accuracy(document_topics_df=document_topics_df_20000_T)}")
print(f"Accuracy of n=20000 non bigram model:{cluster_accuracy(document_topics_df=document_topics_df_20000_F)}")
```

```
/usr/local/lib/python3.9/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not
  and should_run_async(code)
Accuracy of n=1000 bigram model:0.845
Accuracy of n=1000 non bigram model:0.864
Accuracy of n=20000 bigram model:0.888
Accuracy of n=20000 non bigram model:0.90695
```

A. Clustering Accuracy