COL 774: Assignment 2 (Semester I, 2023-24)

Due Date: 11:50 pm, Wednesday Oct 4, 2023. Total Points: 30 +

Notes:

- This assignment has two main parts - text classification using Naïve Bayes (Part 1), and Classication using SVMs (Part II). Part I is being released now, and Part II will be released after the Minor exams. Both parts will be due at the same time.
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- Do not submit the datasets. Do not submit any code that we will be providing to you for processing.
- Include a single write-up (pdf) file which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You should use Python for all your programming solutions.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you have submitted as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. We will also include submissions from previous years since some of the questions may be repeated. Any cheating will result in a zero on the assignment, a penalty of -10 points and possibly much stricter penalties (including a fail grade and/or a DISCO).

1. **(30 points) Text Classification**
   In this problem, we will use the Naïve Bayes algorithm for text classification. The dataset for this problem is the Coronavirus tweets Dataset. Given a tweet related to coronavirus, the task is to predict the sentiment of the review. There are three possible sentiments (labels) - positive, neutral or negative. You have been provided with a subset of the Coronavirus tweets Dataset, with the training and validation splits containing 37,864 reviews (samples) and 32,93 reviews respectively. We will be testing your code on a held-out test split. Data is available at this link.

(a) (10 points) Implement the Naïve Bayes Multiclass classification algorithm to classify each sample into one of the given three categories. You should implement the model where for each word position in a document, we generate the word using a single (fixed across word positions) Multinoulli distribution.

   i. Report the accuracy over the training as well as the validation set.
   ii. Read about word cloud. Construct a word cloud representing the most frequent words for each class.

   Notes:

   - Make sure to use the Laplace smoothing for Naïve Bayes (as discussed in class) to avoid any zero probabilities.
   - You should implement your algorithm using logarithms to avoid underflow issues.
   - You should implement Naïve Bayes from the first principles and not use any existing Python modules.

(b) (2 points)

   i. What is the validation set accuracy that you would obtain by randomly guessing one of the categories as the target class for each of the reviews (random prediction)?
   ii. What accuracy would you obtain if you simply predicted each sample as positive?
   iii. How much improvement does your algorithm give over the random/positive baseline?

(c) (2 points) Read about the confusion matrix.

   i. Draw the confusion matrix for your results in parts (a) and (b) above (for both training and validation data).
   ii. For each confusion matrix, which category has the highest value of the diagonal entry? What does that mean?

(d) (4 points) The dataset provided to you is in the raw format i.e., it has all the words appearing in the original set of articles. This includes words such as 'of', 'the', 'and' etc. (called stopwords). Presumably, these words may not be relevant for classification. In fact, their presence can sometimes hurt the performance of the classifier by introducing noise in the data. Similarly, the raw data treats different forms of the same word separately, e.g., 'eating' and 'eat' would be treated as separate words. Merging such variations into a single word is called stemming. Read about stopword removal and stemming (for text classification) online.

   i. Perform stemming and remove the stop-words in the training as well as the validation data.
   ii. Construct word clouds for both classes on the transformed data.
   iii. Learn a new model on the transformed data. Report the validation set accuracy.
   iv. How does your accuracy change over the validation set? Comment on your observations.

(e) (6 points) Feature engineering is an essential component of Machine Learning. It refers to the process of manipulating existing features/constructing new features in order to help improve the overall accuracy of the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature.

   i. You can read here about Bi-grams. Use word-based bi-grams to construct new features. You should construct these features after doing the pre-processing described in part d(i) above. Further, these should be added as additional features, on top of existing unigram (single word) based features. Learn your model again, and report validation set accuracy.

ii. Come up with at least one additional set of features to further enhance your model. Learn the model after doing pre-processing as described in part d(i) above, and report the validation set accuracy.

iii. For both your variations tried above, compare with the validation set accuracy that you obtained in parts (a) and parts (d). Do the additional set of features constructed in each case help you improve the overall accuracy? Comment on your observations.

(f) (6 points) In Domain Adaptation, given a model trained on data from the *source domain*, we apply it on data from the *target domain*. The idea is to exploit the similarity between source and target domains, and leverage the parameters learned on the target domain, making up for lack of sufficient training data in the target domain. In our setting, we will assume the source domain is coronavirus tweets, and the target domain is represented by another dataset of general purpose tweets. You are given subsets of different sizes of the training data (in the target domain), varying from $1\%, 2\%, 5\%, 10\%, 25\%, 50\%, 100\%$ of the total training set.

i. Learn a Naïve bayes model, on entire source training data, combined with the partial target training data for each of the splits above. Your vocabulary should be constructed on the combined dataset. You should use the model in part (d) above, i.e., after removing stopwords, and performing stemming but before doing any additional feature engineering. Compute (target) validation set accuracy for each split.

ii. Do the same thing as in the part above, but without adding any data from source domain. This represents learning from scratch on the target data. Compute the accuracy on target validation set for each of the splits.

iii. Plot on a single graph the validation set accuracy for the two algorithms described above, as we vary the size of target training data.

iv. What do you observe? Explain.

2. **Support Vector Machines:** Will be added soon..