Concordia University
COMP 6721 Introduction of AI – Fall 2019
Project 2

Due Date:	By 11:55 pm Monday, November 22, 2019
Evaluation:	15% of the final mark
Late Submission:	Late submission will **NOT** be accepted.
Purpose:	The purpose of this project is to learn NLP and ML. This project can **ONLY** use python libraries Numpy, Pandas, sklearn, Nltk, Matplotlib.
Programming Language:	Python
Teams:	Maximum two students in a team. The work should be evenly distributed to each team member. Teams must submit only one copy of your project via team leader.

General Guidelines When Writing Programs:

Include the following comments at the top of your source codes

```
# ------------------------------------------------------
# Project (include number)
# Written by (include your name and student id)
# For COMP 6721 Section (your lab section) – Fall 2019
# ------------------------------------------------------
```

- Include comments in your program describing the main steps in your program. The Focus in your comments rather on the why than the how.
- Display clear prompts for users when you are expecting the user to enter data from the keyboard if needed.
- All output should be displayed with clear messages and in an easy to read format.
- End your program with a closing message so that the user knows that the program has terminated.

---

1. Hacker News Dataset

In this project, we'll work with a dataset of Hacker News fetched form kaggle.

Hacker News is a popular technology site, where user-submitted stories (known as "posts") are voted and commented upon. The site is extremely popular in technology and start-up circles. The top posts can attract hundreds of thousands of visitors.

Download the dataset available on the Moodle. This dataset contains Hacker News posts from 2018-to 2019-06 (2,709,143 posts) and each post includes the following columns:

Object ID | Title | Post Type | Author | Created At | URL | Points | Number of Comments

This project will be divided into three tasks:

1.1 Task 1: Extract the data and build the model

Write a Python program to build a probabilistic model from the training set. Your code will parse the files in the training set and build a vocabulary with all the words it contains in `Title` which is `Created At` 2018. Then for each word, compute their frequencies and the probabilities of each `Post Type` class (`story, ask-hn, show-hn and poll`). Extract the data from `Created At` 2019 as the testing dataset.

To process the texts, fold the `Title` to lowercase, then tokenize them using `re.split('\[\^a-zA-Z\]',aString)` and use the set of resulting word as your vocabulary.

For each word w_i in the training set, save its frequency and its conditional probability for each `Post Type` class: $P(w_i|story)$, $P(w_i|ask\text{-}hn)$, $P(w_i|show\text{-}hn)$ and $P(w_i|poll)$. These probabilities must be smoothed with $\delta = 0.5$.

Save your model in three text files called `model-2018.txt`. The format of this file must be the following:

1. A line counter `i,` followed by 2 spaces.
2. The word `wi,` followed by 2 spaces.
3. The frequency of `wi` in the class story, followed by 2 spaces.
4. The smoothed conditional probability of `wi` in `story` — $P(wi|story)$, followed by 2 spaces.
5. The frequency of `wi` in the class story, followed by 2 spaces.
6. The smoothed conditional probability of `wi` in `ask-hn` — $P(wi|ask\text{-}hn)$, followed by 2 spaces.
7. The frequency of `wi` in the class story, followed by 2 spaces.
8. The smoothed conditional probability of `wi` in `show-hn` — $P(wi|show\text{-}hn)$, followed by 2 spaces.
9. The frequency of `wi` in the class story, followed by 2 spaces.
10. The smoothed conditional probability of `wi` in `poll` — $P(wi|poll)$, followed by a carriage return.

Note that the file must be sorted alphabetically. For example, your files `model-2018.txt` could look like the following:

```
1 block 3 0.003 40 0.4 10 0.014 4 0.04
2 compute 3 0.003 40 0.4 40 0.034 40 0.0024
3 query 40 0.4 50 0.03 20 0.00014 15 0.4
4 system 0.7 0.003 0 0.000001 30 0.4 2 0.4
```

1.2 Task 2: Use ML Classifier to test dataset

Once you have built your model in Task 1, use it to implement and test a Naïve Bays Classifier to classify posts into their likely class in 2019 dataset. To avoid arithmetic underflow, work in $\log_{10}$ space.

Run your classifier on the 2019 testing dataset and create a single file named `baseline-result.txt` with your classification results. For each test file, `baseline-result.txt` should contain:

1. a line counter, followed by 2 spaces
2. the name of the test post `Title`, followed by 2 spaces

3. the classification as given by your classifier (the label story, ask-hn, show-hn or poll), followed by 2 spaces
4. the score of the class story as given by your classifier, followed by 2 spaces
5. the score of the class ask-hn as given by your classifier, followed by 2 spaces
6. the score of the class show-hn as given by your classifier, followed by 2 spaces
7. the score of the class poll as given by your classifier, followed by 2 spaces
8. the correct classification of post, followed by 2 spaces
9. the label right or wrong (depending on the case), followed by a carriage return.

For example, your result file could look like the following:

```
1  Y Combinator  story  0.004 0.001 0.0002. 0.002 story right
2  A Student's Guide  poll 0.002 0.03 0.007 0.12 story wrong
```

1.3 Task 3: Experiments with the classifier

Tasks 1 & 2 above will constitute your experiment 1, or baseline experiment, and you will perform variations over this baseline to see if they improve the performance of your classifier.

1.3.1 Experiment 2: Stop-word Filtering

Download the list of stop words available on Moodle. Use the baseline experiment and redo tasks 1 and 2 but this time remove the stop words from your vocabulary. Generate the new model and result files that you will call `stopword-model.txt` and `stopword-result.txt`.

1.3.2 Experiment 3: Word Length Filtering

Use the baseline experiment and redo tasks 1 and 2 but this time remove all words with length ≤ 2 and all words with length ≥ 9. Generate the new model and result files that you will call `wordlength-model.txt` and `wordlength-result.txt`.

1.3.3 Experiment 4: Infrequent Word Filtering

Use the baseline experiment, and gradually remove from the vocabulary words with frequency= 1, frequency ≤ 5, frequency ≤ 10, frequency ≤ 15 and frequency ≤ 20. Then gradually remove the top 5% most frequent words, the 10% most frequent words, 15%, 20% and 25% most frequent words. Plot the performance of the classifiers against the number of words left in your vocabulary.

1.3.4 Experiment 5: Smoothing

Use the baseline experiment, and change the smoothing value gradually from $\delta=0$ to $\delta=1$ in steps of 0.1. Plot the performance of the classifier against the smoothing value.

2. Deliverables

2.1 Evaluation Scheme

Students will be given individual grades that will be a function of the team grade and the peer-evaluation.

At the end of the project, all team members will fill out a peer-evaluation form to evaluate the contribution of each team member. The grade of a student will be a function of his/her team grade and the peer evaluation received.

2.2 Demos

Project 2 will be demonstrated on the lab machines. You will not be able to demo on your laptop. Regardless of the demo time, you will demo the program that was uploaded as the official submission on or before the due date. The schedule of the demos will be posted on Moodle. No special preparation is necessary for the demo (no slides or presentation needed). **During the demo, you will receive a new dataset in the same format that only contains training sample, and your code should be able to generate the new model for the new training set without any modification**.

2.3 Report

Please note that this is a project, not an assignment. The difference is that is open-ended and in addition to the required work started above, you are expected to be creative, perform additional experiments, comparisons, and analysis. This means that in addition to stating *what* you did, you should also describe *why* you did it that way and what other options were available and why they were not retained.

Your report should have a reference section (not included in the page count) that properly cites all relevant resources that you have consulted (books, papers, websites, etc.), even if it was just to inspire you. Failure to properly cite your references constitutes plagiarism and will be reported.

Requirements:

Your report should be 5-6 pages (without references and appendix) and use the template provided on Moodle. The report should contain at least the following:

1. ½ page: Introduction and technical details.
2. 1 page: Show and analyze the results of the baseline experiment (exp. #1). Give a table of results showing the accuracy, precision, recall and F1-measure for each class, as well as a confusion matrix. Analyze, compare and discuss these results from each year.
3. ½ pages: Results and analysis of the stop-word filtering experiment (exp. #2).
4. ½ pages: Results and analysis of the word-length filtering experiment (exp. #3).
5. ½ pages: Results and analysis of the infrequent word filtering experiment (exp. #4).
6. ½ pages: Results and analysis of the smoothing experiment (exp. #5).
7. 1 page: Compare and discuss the results of the 4 experiments.
8. ½ to 1 page: If you were to continue working on this project, what do you feel would be interesting to investigate? Are there questions that you would like to investigate more, if you had the time and the energy?
9. Not included in the page count: References, Appendix, Figures, Tables, and pseudocode.

Submitting Project 2

Your submission should include the following documents:

1. Read and sign the expectation of originality form (available on Moodle or at https://www.concordia.ca/content/dam/ginacody/docs/Expectations-of-Originality-Feb14-2012.pdf)
2. Create a README.txt file, which will contain specific and complete instructions on how to run your program on the lab computers. If the instructions in your file do not work or incomplete, you will not be given the benefit of the doubt.
3. Create one zip file, containing all your code, the README.txt, technical report, the expectation of originality and all the output files from your code.
4. Name your zip file: ***P2_ StudentID.zip*** if there is only one student in the team;
   P2_StudentID1_StudentID2.zip if there are 2 students in the team.
5. Have the team leader upload the zip file on the Moodle webpage.

   Note: please check your course Moodle webpage on how to submit the project and follow the instructions to submit your project. Wrong submission files and codes will not be accepted.
6. In addition, print your report and submit a paper copy to your TA when you give your demonstration.

Have fun!!