[redacted]

[redacted]

Purpose:	The purpose of this project is to learn NLP and ML. This project can **ONLY** use python libraries ***Numpy, Pandas, sklearn, Nltk, Matplotlib, math and sys***.
Programming Language:	Python

[redacted]

General Guidelines When Writing Programs:

Include the following comments at the top of your source codes

```
# ------------------------------------------------------
# Assignment (include number)
# Written by (include your name and student id)
# For COMP 472 Section (your lab section) – Summer 2020
# ------------------------------------------------------
```

- 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 assignment 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 and each post includes the following columns:

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

This assignment 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 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: $P(w_i|story)$, $P(w_i|ask\_hn)$, $P(w_i|show\_hn)$ and $P(w_i|poll)$. These probabilities must be smoothed with $\delta = 0.5$.

Save your model in the text file named `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 `ask_hn`, followed by 2 spaces.
6. The smoothed conditional probability of `wi` in `ask_hn` – $P(wi|ask\_hn)$, followed by 2 spaces.
7. The frequency of `wi` in the class `show_hn`, followed by 2 spaces.
8. The smoothed conditional probability of `wi` in `show_hn` – $P(wi|show\_hn)$, followed by 2 spaces.
9. The frequency of `wi` in the class `poll`, followed by 2 spaces.
10. The smoothed conditional probability of `wi` in `poll` – $P(wi|poll)$, followed by a carriage return.

Your file must be sorted alphabetically. For the four different `Post Type` class, ask_hn and show_hn should be considered as two words (ask_hn and show_hn) not 4 words in your vocabulary. 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 ask-hn 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 show-hn 0.7 0.003 0 0.000001 30 0.4 2 0.4
```

Please note:
1. The values presented above is an example not based on the real data.
2. Your program should be able to output a file named “vocabulary.txt”, which should contain all the words in your vocabulary. If the title includes some words, which you think is not useful for classification, you can remove it from you vocabulary, but you need put all the removed words in an independent file named “remove_word.txt”.

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
(Please note the values presented above is an example not based on
the real data)
```

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 1: 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 2: 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 3: 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 both performance of the classifiers against the number of words left in your vocabulary as two graphs in your program.

2. Deliverables

2.1 Demos

[redacted] **During the demo, you will receive a new dataset in the same format with the same four class types that only contains training sample (less than 100 samples), and your code should be able to generate the new model for the new training set without any modification within 1 minute**. [redacted]

2.2 Grading Scheme

Grading Criteria	Description	Points
Task 1 (8 pts)	Build the model	4 pts
	Output model data file	2 pts
	Output vocabulary and remove_word file	2 pts
Task 2 (2 pts)	Baseline result, format	2 pts
Task 3 (8 pts)	Experiment 1 (including new model 1pt and output files 1.5pts)	2.5 pts
	Experiment 2 (including new model 1pt and output files 1.5pts)	2.5 pts
	Experiment 3 (including new model 1pt and two output graphs 2pts)	3 pts
Code Quality	Necessary comments, readability and clarity	1 pt
README.txt	Instructions on how to run your program and list of libraries	1 pt
Total		20 pts