UNIVERSITÉ Concordia UNIVERSITY

COMP 6721

Project Assignment - 2

Spam Detector

Submitted to

Prof. Dr. René Witte

Team name

FL-G08

TA and Marker

Ms. Fathima Nihatha

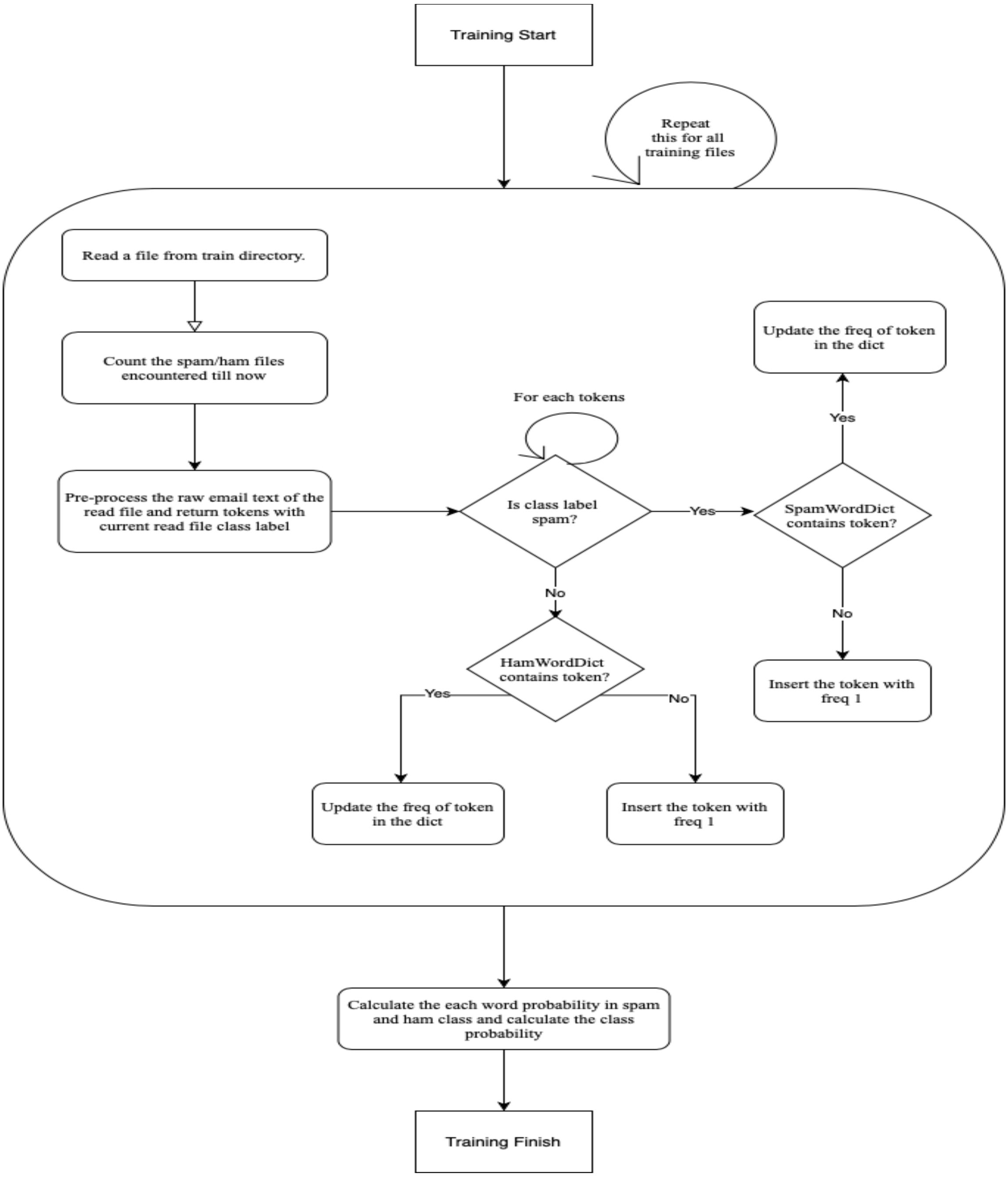
Team members

Apoorv Semwal	40083939
Ayush Dave	40080515
Mayank Jariwala	40075419

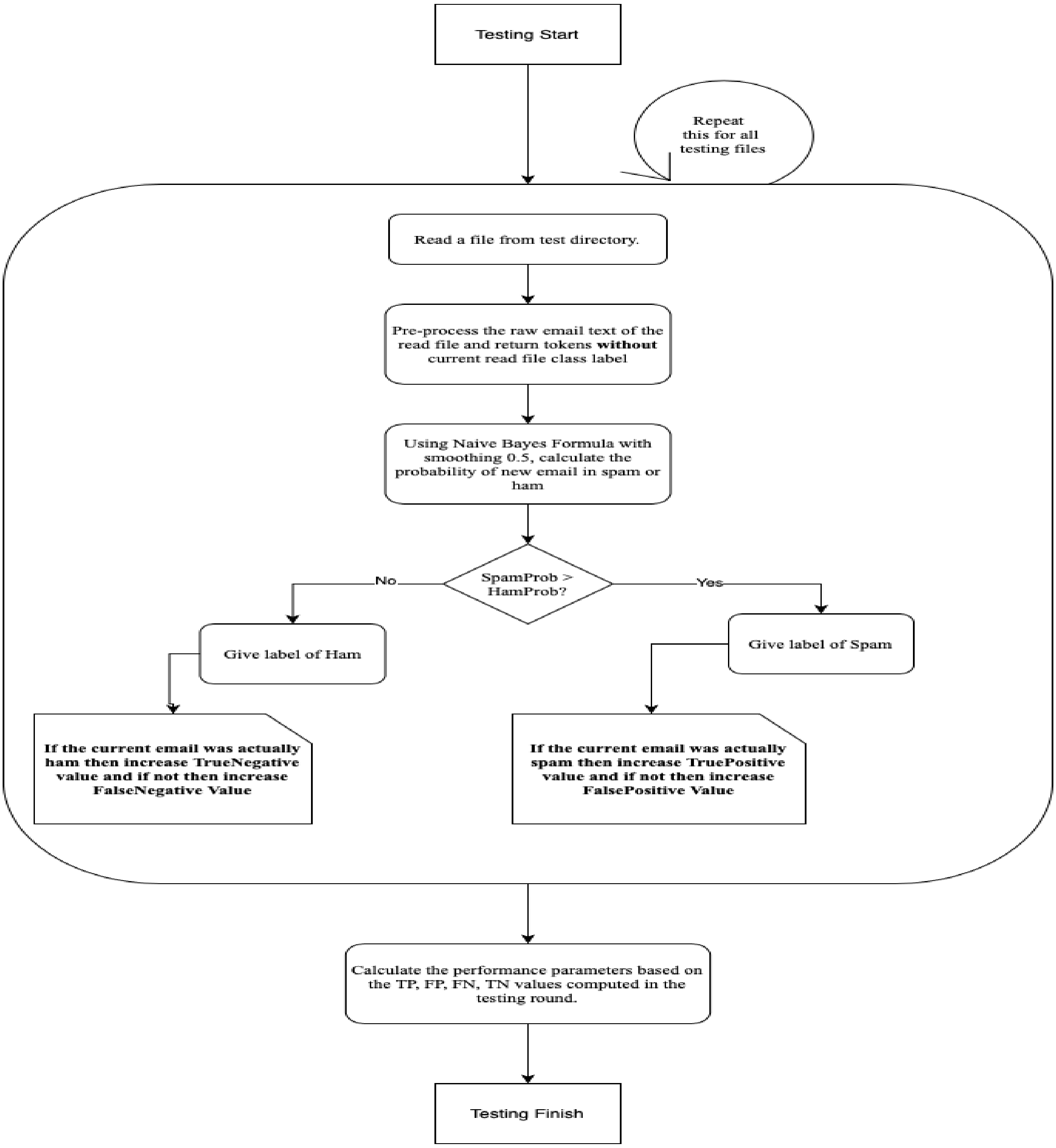
Analysis

- **Flow Diagrams:** For **Evaluating** our Spam Classifier we tried both the **Unigram approach (with 2 different REGEX explained below)** and a **Bigram approach**.

  Finally it was the **Unigram Approach with Regex-2** which gave us better results compared to other options.
  Training Flow:


Training Start
Repeat this for all training files
Read a file from train directory.
Count the spam/ham files encountered till now
Pre-process the raw email text of the read file and return tokens with current read file class label
For each tokens
Is class label spam?
Yes
No
SpamWordDict contains token?
Yes
Update the freq of token in the dict
No
Insert the token with freq 1
HamWordDict contains token?
Yes
Update the freq of token in the dict
No
Insert the token with freq 1
Calculate the each word probability in spam and ham class and calculate the class probability
Training Finish

Testing Flow:


Testing Start
Repeat this for all testing files
Read a file from test directory.
Pre-process the raw email text of the read file and return tokens **without** current read file class label
Using Naive Bayes Formula with smoothing 0.5, calculate the probability of new email in spam or ham
SpamProb > HamProb?
No
Yes
Give label of Ham
Give label of Spam
If the current email was actually ham then increase TrueNegative value and if not then increase FalseNegative Value
If the current email was actually spam then increase TruePositive value and if not then increase FalsePositive Value
Calculate the performance parameters based on the TP, FP, FN, TN values computed in the testing round.
Testing Finish

- **Problem formulation for training the Naïve Bayes Classifier:** Every single file provided within Training dataset **(1000 HAM and 997 SPAM),** was passed through a regular expression to split the file string and arrive a vocabulary of unique words. For splitting the file string we started our development with the regex provided in the problem statement, **'[^a-zA-Z]' i.e. split at every non-alphabet character.** However, we also tried checking if we could arrive at better results by slightly tweaking the regex to '**W**' which is equivalent to **'[^a-zA-Z0-9_]' i.e. split at every non-alphanumeric character (except underscore).** We would be showing results for both.

- **REGEX-1: [^a-zA-Z]**

- **Vocabulary Size: 59251 Unique Words**

- **Confusion Matrix over Test Dataset:** Considering **SPAM as a positive class** and **HAM as the negative class**.

Test Files = 800	**Predicted (Spam)**	**Predicted (Ham)**	
Actual (Spam)	TP=364	FN=36	*400*
Actual (Ham)	FP=7	TN=393	*400*
	371	*429*	

- **Evaluation Metrics:** Based on the above confusion matrix values following evaluation metrics were calculated.

Accuracy: TP + TN / (TP + TN + FP + FN)
Recall: TP / (TP + FN)
Precision: TP / (TP + FP)
F1-Score: 2 * Recall * Precision / (Recall + Precision)

Evaluation Metric	**Value**
Accuracy	94.625
Precision	98.113
Recall	91.0
F1-Score	94.423

- **REGEX-2: [^a-zA-Z0-9_] with unigram approach**

- **Vocabulary Size: 63480 Unique Words**

- **Confusion Matrix over Test Dataset:**

Test Files = 800	**Predicted (Spam)**	**Predicted (Ham)**	
Actual (Spam)	TP=366	FN=34	*400*
Actual (Ham)	FP=5	TN=395	*400*
	371	*429*	

- **Evaluation Metrics:**

Evaluation Metric	Value
Accuracy	**95.125**
Precision	**98.652**
Recall	**91.5**
F1-Score	**94.942**

- **REGEX-2: [^a-zA-Z0-9_] With Bigram Approach:**
  ***Note:** Code for this has not been commited to the master branch and can be seen in a different branch – **'bigram' -** https://github.com/apoorvsemwal/IntroToAI-SpamDetector/tree/bigram.

- **Evaluation Metrics:**

Evaluation Metric	Value
Accuracy	14.375
Precision	0
Recall	0
F1-Score	NA

We obtained **best accuracy** for **Unigram Approach** with **Regex-2 - [^a-zA-Z0-9_] - 95.125%**

Overall in general the model was able to show very high Precision signifying that there is very high possibility of a mail being SPAM if our model predicts it to be a SPAM. This is desirable as we do not want our model to predict SPAM for a genuine mail.

However a slightly lower Recall signifies that in some 9% of the cases model might not classify a mail as SPAM at all while the mail was actually a SPAM.

References

1. https://stackoverflow.com/questions/28931224/adding-value-labels-on-a-matplotlib-bar-chart
2. https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string

Instructions to run the project:

- Download/Clone the Project Repo to your local machine – [IntroToAI-SpamDetector](https://github.com/apoorvsemwal/IntroToAI-SpamDetector.git) or access it from Google drive – (https://drive.google.com/open?id=1hFeO5xocprJfMTZcDSfcwEt-uOsAlrHS)
- Navigate to **'\IntroToAI-SpamDetector\src'** in your terminal
- Run CMD:

  python launcher.py
- Check results folder **'\IntroToAI-SpamDetector\results'**