Pattern Recognition and Machine Learning: 3^{rd} Assingment

Utsav Dey (CS20S009) • Arup Das (CS20S016)

1 Aim

Build a spam classifier for emails using labelled data.

2 Data set

A correct set of labelled data is very important for this task.

We used the Enron data set [1] as our base data set. The data-set has a large amount of emails, we used only 2000 spam email and 2000 non spam emails for training purpose. The Enron data set however did not contain any email related to Indian context , so we added a few emails to our data set on our own. For testing accuracy of our models we used another 200 spam and 200 non spam email from Enron data set.

Our final data sets can be downloaded from here .

3 Models we considered

1. Naive Bayes: This is very simple to implement. The problem with this model is that it takes into account only presence or absence of a certain word. However taking look into real world data we are missing the important information given by frequency of words. Some word may have more frequency in a spam email, others may not.

2. Decision Tree with Ada-boost: Next we decided to build this. Each stumps will contain the frequency of word. However this idea quickly became infeasible for us because the number of features were a lot. Finding the best feature value pair for a stump required a lot of computation and we needed a bunch of stumps. When given a lot of data points the training time exceeded a day and we aborted the process mid way.

3. SVM: This is the model we finally used. It used the frequency of words to train on. The model took fairly low amount of time on our data set.And this gave a good result. We however experimented with various Kernels. We chose SVM with an RBF kernel with regularization parameter C=3 after hyper-parameter tuning.

[1] http://www2.aueb.gr/users/ion/data/enron-spam/

4 Building Classifier

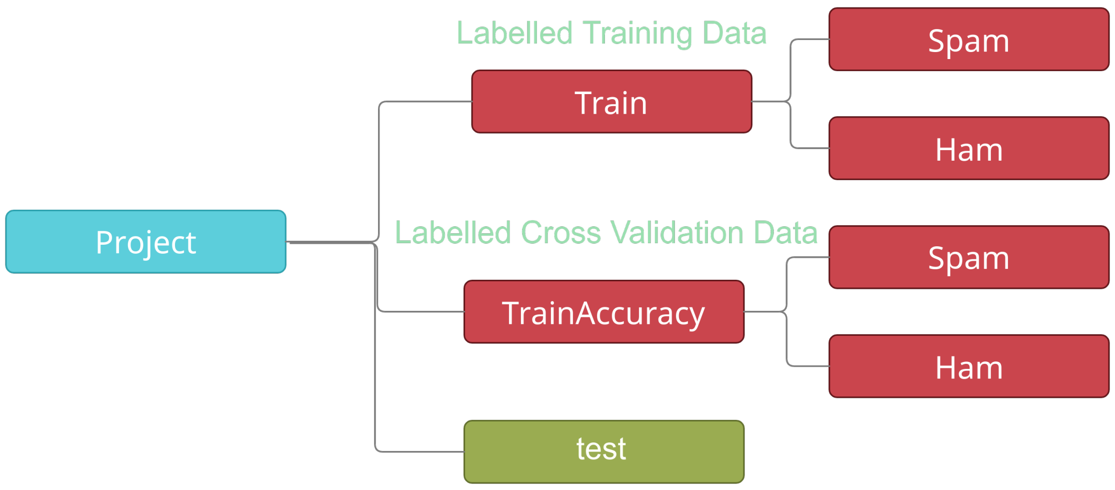

Figure 1: Folder structure

First defining the **folder structure**. The program reads the data for training from the **maroon** colored folders. It decided the label of the data depending on the folder it is reading the data from. The 'test' folder marked in **green** contains emails that will be classified using our classifier.Figure 1

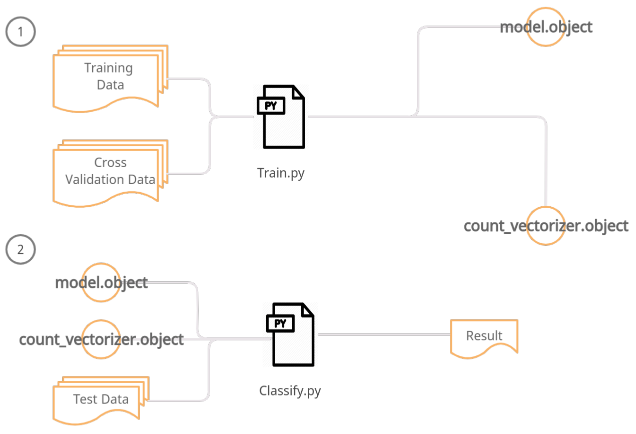

Figure 2: Train and Test Pipeline

This is our pipeline for testing and training. There is a file named 'Train.py' that takes in data and outputs the best possible learner after cross validation. The model and count vectorizer objects are stored as .object file in the disk for further use.

The file 'Classify.py' uses the outputs from 'Train.py' and test data, where we store the file we want to classify. So in summary for using the classifier use only the classify.py

For SVM we have used the sklearn library.

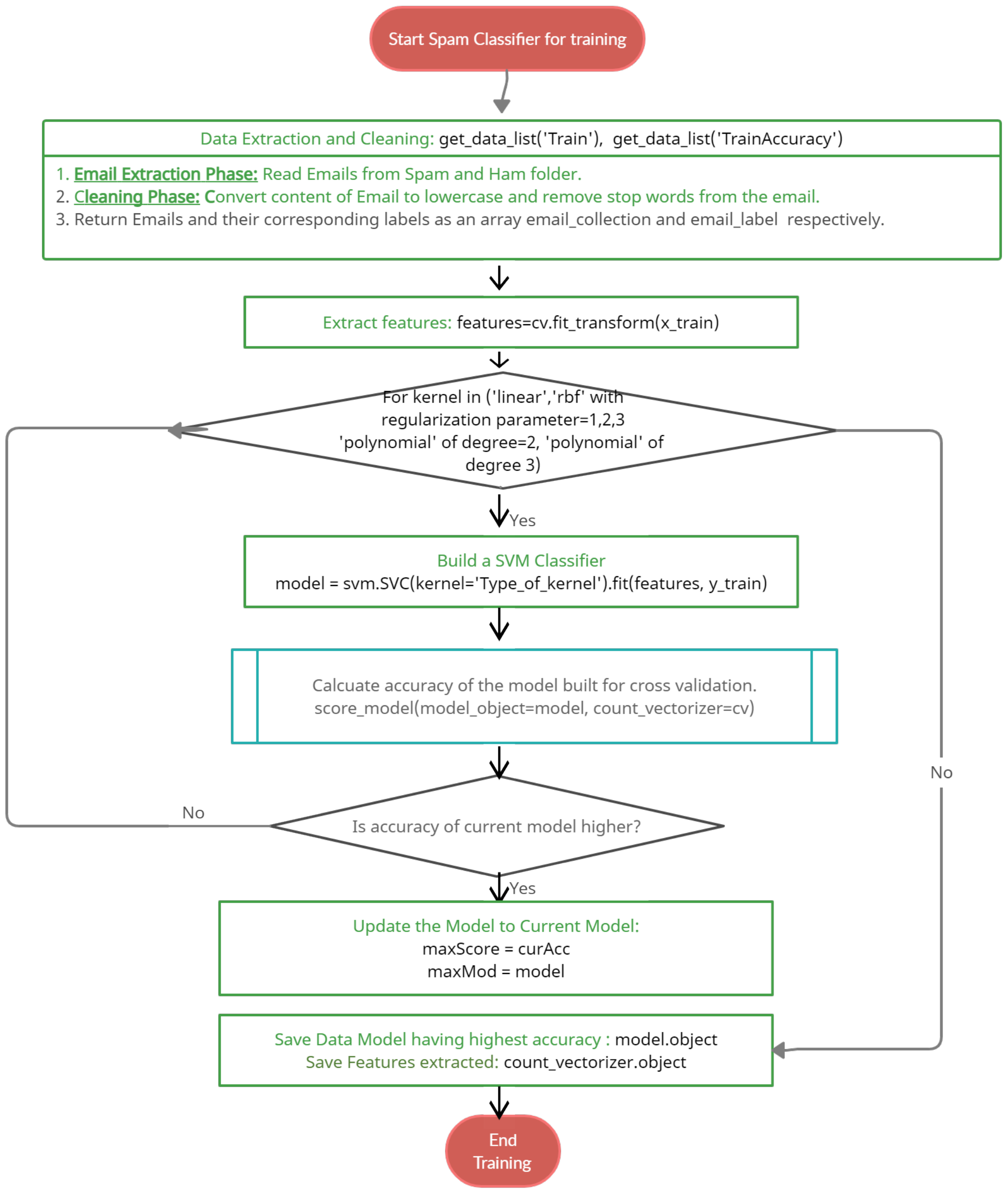
Start Spam Classifier for training
Data Extraction and Cleaning: get_data_list('Train'), get_data_list('TrainAccuracy')
1. Email Extraction Phase: Read Emails from Spam and Ham folder.
2. Cleaning Phase: Convert content of Email to lowercase and remove stop words from the email.
3. Return Emails and their corresponding labels as an array email_collection and email_label respectively.
Extract features: features=cv.fit_transform(x_train)
For kernel in ('linear','rbf' with regularization parameter=1,2,3 'polynomial' of degree=2, 'polynomial' of degree 3)
Yes
No
Build a SVM Classifier
model = svm.SVC(kernel='Type_of_kernel').fit(features, y_train)
Calcuate accuracy of the model built for cross validation.
score_model(model_object=model, count_vectorizer=cv)
Is accuracy of current model higher?
No
Yes
Update the Model to Current Model:
maxScore = curAcc
maxMod = model
Save Data Model having highest accuracy : model.object
Save Features extracted: count_vectorizer.object
End Training

Figure 3: Model training and Verification flowchart

Figure 3 gives a pictorial description of how our model is trained and cross validated.Labelled data for training is taken from **Train** folder and labelled data for cross validation is taken from

TrainAccuracy folder.

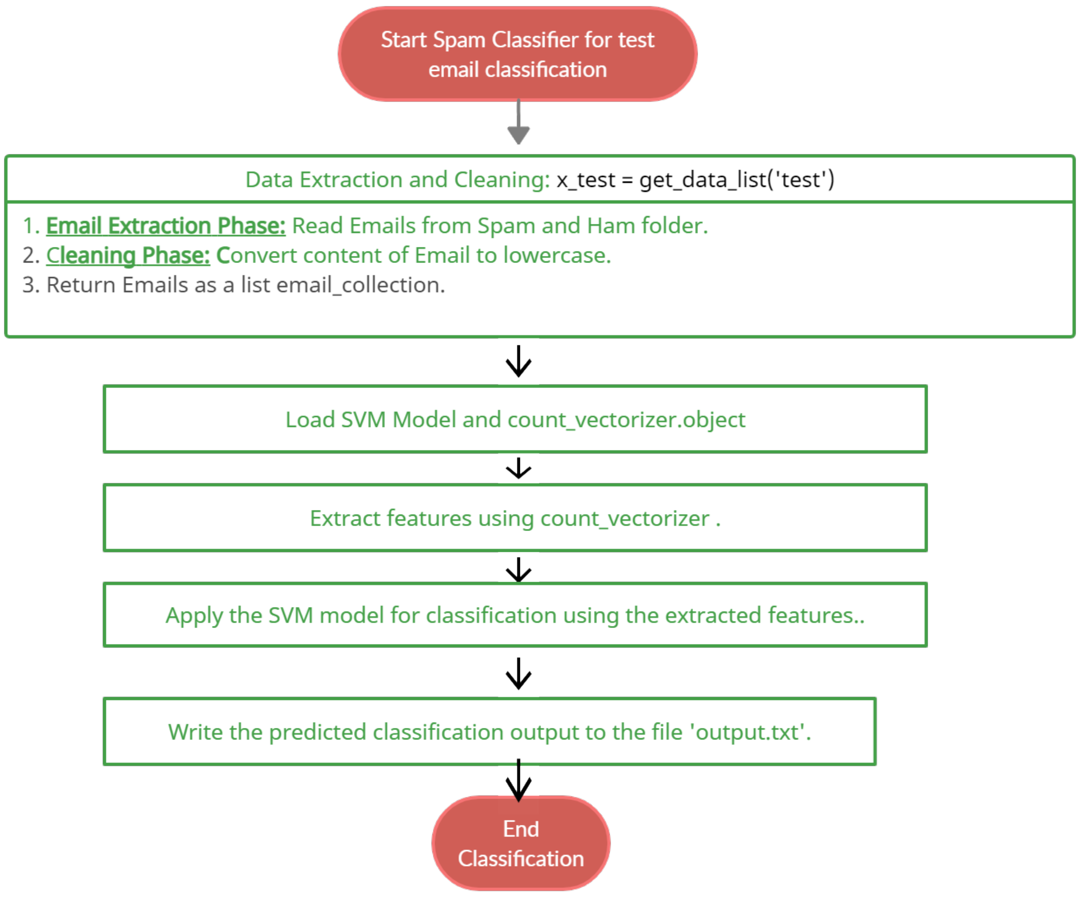
Start Spam Classifier for test email classification
Data Extraction and Cleaning: x_test = get_data_list('test')
1. Email Extraction Phase: Read Emails from Spam and Ham folder.
2. Cleaning Phase: Convert content of Email to lowercase.
3. Return Emails as a list email_collection.
Load SVM Model and count_vectorizer.object
Extract features using count_vectorizer .
Apply the SVM model for classification using the extracted features..
Write the predicted classification output to the file 'output.txt'.
End Classification

Figure 4: Test Email classification flowchart

Figure 4 gives a pictorial representation of how a test email is classified using the model that we have built earlier.

5 Feature Extraction

We used frequency of each words in the training set as our feature. We ignored stop words from English language, for example 'the' is a stop word. We did this because these words do not provide any extra information to emails, in fact as they are very common they may lead to misclassification of emails.

6 Hyper-Parameter Tuning and Cross validation

For cross validation purpose we randomly selected 200 spam and 200 non spam emails from same source and built a data repository. Since both training data and cross validation data were selected through a random process (we did only once and fixed it) , the cross validation results gives important clue about performance of a model.

We tuned hyper-parameter "C" for RBF by cross validating SVM model with the cross validation data set that we have built earlier. And then we took the one with best accuracy.

7 Cross validation result

Below table shows the result we get after cross validating our various SVM models using the cross validation data.

Kernel Used	***Percentage of Accuracy***
Linear Kernel	95.51
RBF Kernel with C=1	95.01
RBF Kernel with C=2	96.00
RBF Kernel with C=3	97.07
Polynomial Kernel, degree=2	73.31
Polynomial Kernel, degree=3	60.84

Conclusion: Using SVM with a RBF kernel with regularization parameter [2] C=3 gives the best result among the ones we tried. And we use that for our classifier.

8 Output

To perform classification run 'Classify.py', copy **count_vectorizer.object** and **model.object** files from the zip to the same folder as the code file before running it. We have written the output to a file -'output.txt' and to console.

[2]Regularization Parameter denotes our tolerance to misclassification. Higher value denotes lower level of tolerance.