Milestone II – Final Project (CS 5100)

Building a Spam Filter using Machine Learning

Team: Amey Arya & Nikhita Singh

Steps taken after Milestone I:

As already mentioned, we are using the Ling-Spam Dataset. Until milestone I, we had finished cleaning the dataset (cleaning from interpunctuation and special character). After we got the data prepared, our next step was to create a dictionary of words. The main motive of this was to use these words as a feature to determine whether the given email is spam or non-spam.

Generating Dictionary:

There are a lot of possible ways to choose the words to create a dictionary. In our case, we chose the first 2500 most frequent words across all the emails. The algorithm used for this is as follows:

1. Load and concatenate all the emails in one string.
2. Split the emails by white space so we can count the number of occurrences of each of the words.
3. Build cell array of unique words and counts.
4. Save the word count in dictionary.txt file.

After creating the dictionary, it looks something like this:

1. email 2163

2. order 1648

3. address 1645

4. language 1534

5. report 1384

6. mail 1364

…

Generating features:

The gist of this step is to generate features so that the resulting structure is ready to apply the three algorithms namely, Naïve Bayes, Random Forest and Decision Tree. The dictionary that we already created contains all the 2500 words (features) based on which we created the prediction model. Here we counted the number of occurrences of each word from the dictionary in the emails.

A sample of this generated feature is as follows:

1 7 1

1 12 2

1 19 2

1 22 1

1 25 1

The first column is the document sequence number, the second is the sequence number for word in the dictionary and the third column is the frequency of that word in a single email. For example, the first row means that document is one which according to our dataset is 3–380msg4.txt. We segregated this data into 2 text files, namely, features_train.txt (containing structured data for training the model) and features_test.txt (containing structured data for testing the model).

Generating the Machine Learning Model:

Since, now we had set up the structured data set up, we used our three algorithms so that we get the prediction model.

Naïve Bayes Classifier:

The Naïve Bayes algorithm is a simple probabilistic classifier that calculates a set of probabilities by counting the frequency and combination of values in each dataset. Naïve Bayes classifier use bag of words features to identify spam e-mail and a text is representing as the bag of its word. The bag of words is always used in methods of document classification, where the frequency of occurrence of each word is used as a feature for training classifier. This bag of words features is included in the chosen datasets. Its formula is simple. It counts the number of occurrences of a w (in our case one word from the dictionary) in all the c (sum of all occurrences of the dictionary words in ether spam or non-spam emails depending for which one we are estimating the probability). V — is the number of words in the dictionary. This probability will be calculated separately first on spam and then on non-spam emails.

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

We implemented the Naïve Bayes algorithm on our structured dataset.

Random Forest Classifier:

The Random forest is a meta-learner which consists of many individual trees. Each tree votes on an overall classification for the given set of data and the random forest algorithm chooses the individual classification with the most votes. Each decision tree is built from a random subset of the training dataset, using what is called replacement, in performing this sampling. That is, some entities will be included more than once in the sample, and others won't appear at all. In building each decision tree, a model based on a different random subset of the training dataset and a random subset of the available variables is used to choose how best to partition the dataset at each node. Each decision tree is built to its maximum size, with no pruning performed. Together, the resulting decision tree models of the Random forest represent the final ensemble model where each decision tree votes for the result and the majority wins. Calculating the total spam score which is the minimum score required to mark a message as spam. If category is zero email classified as non-spam & if category is one email classified as spam.

The random forests pseudocode for prediction or classification task that we used is as follows:

1. InputX: number of nodes
2. InputN: number of features in the Email Message
3. InputY: number of trees to be grown
4. while termination conditions is not true do
5. Select a self-staring Email Message S from the training corpus Y
6. Create tree R from the selected self-starting Email Message S
7. Choose n features arbitrarily from N; where $n \ll N$
8. Compute the optimal dividing point for node d among the n features
9. Divide the parent node to two offspring nodes through the optimal divide
10. Execute steps 1–3 till the maximum number of nodes (x) is created
11. Create your forest by iterating steps 1–4 for Y number of times
12. end while.
13. generate result of every created trees $\{R_t\}_1 Y$
14. use a new Email Message for every created trees beginning at the root node
15. designate the Email Message to the group compatible with the leaf node.
16. merge the votes or results of every tree
17. return Final Email Message Classification (Spam/Non-spam email) group having the highest vote (G).
18. end

Decision Tree Classifier:

Decision tree learning is a method commonly used in data mining. The goal is to create a DT model and train the model so that it can predicts the value of a target variable based on several input variables. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the

leaf. A tree can be "learned" by splitting the source set into various subsets based on attribute value prefixed. This process is repeated on each derived subset in a recursive manner which is called as recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The pseudocode we referred for implementing the Decision Tree classifier on our dataset is as follows:

1. Input Email Message dataset.
2. Compute entropy for dataset.
3. while condition do
4. for every attribute/feature
5. calculate entropy for all categorical values
6. take average information entropy for the current attribute.
7. calculate gain for the current attribute
8. pick the highest gain attribute
9. end for.
10. end while.
11. return Final Email Message Classification (Spam/Non-spam email)
12. end

Steps further:

So, till date we have executed the three classifiers on our structured data and successfully trained the model. Our next action item will be to test this model.

We will further analyze the accuracy of each classifier and represent it graphically against each other to figure out which classifier gives better results.