Web Science: Assignment #8

Alexander Nwala

Mohd. Nauman Siddique

Sunday, April 14, 2019

Contents

Problem 1 **3**

Problem 2 **3**

Problem 3 **4**

Problem 4 **5**

	Spam	**Not Spam**
Spam	3	7
Not Spam	0	10

Table 1: Classication results on testing dataset

Problem 1

Create two datasets; the first called Testing, the second called Training.
The Training dataset should:

1. consist of 10 text documents for email messages you consider spam (from your spam folder)
2. consist of 10 text documents for email messages you consider not spam (from your inbox)

The Testing dataset should:

1. consist of 10 text documents for email messages you consider spam (from your spam folder)
2. consist of 10 text documents for email messages you consider not spam (from your inbox)

1. Upload your datasets on github
2. Please do not include emails that contain sensitive information

SOLUTION

I created a dataset for the email classififcation problem with training and testing data each folder having 10 spam and non-spam emails. The dataset has been uploaded to the Github.

Problem 2

Using the PCI book modified docclass.py code and test.py (see Slack assignment-8 channel) Use your Training dataset to train the Naive Bayes classifier (e.g., docclass.spamTrain()) Use your Testing dataset to test (test.py) the Naive Bayes classifier and report the classification results.

SOLUTION
I resued the code from test.py and trained the classifier on the data set using function *sample_train()*. For the purpose of classifying documents we can call function *classify_document()* with file path which needs to be classified.
On running my classifier on my testing data using function *calculate_confusion_matrix()*. Table 1 shows results on testing the classifier on the data set.

```
def classify_emails():
    cl = Assignment8.docclass.naivebayes(Assignment8.docclass.getwords)
    cl.setdb('SpamClassifier.db')
    sample_train(cl)
    testing_data = "/home/msiddique/WSDL_Work/WebScience/Assignment8/Dataset/testing/"
    # classify_document(cl, testing_data + "/spam/" + "2.txt")
    calculate_confusion_matrix(cl)
```

Problem 2 continued on next page...

	Spam	**Not Spam**
Spam	3 (TP)	7 (FP)
Not Spam	0 (FN)	10 (TN)

Table 2: Results for confusion matrix on testing dataset

```
def sample_train(cl):
    training_data = "/home/msiddique/WSDL_Work/WebScience/Assignment8/Dataset/training
        /"
    for dir, path, files in os.walk(training_data):
        for file_name in files:
            with open(dir + "/" + file_name, "r") as file_object:
                file_content = file_object.read()
                print(file_content)
                file_tag = dir.split("/")[-1]
                if file_tag == "notspam":
                    file_tag = "not spam"
                cl.train(file_content, file_tag)


def classify_document(cl, file_path):
    with open(file_path, "r")as file_object:
        file_content = file_object.read()
        return cl.classify(file_content)
```

Problem 3

Draw a confusion matrix for your classification results (see: https://en.wikipedia.org/wiki/Confusion_matrix)

SOLUTION

I used the *calculate_confusion_matrix()* function to classify the testing documents. Table 2 shows the results for confusion matrix.

```
def calculate_confusion_matrix(cl):
    fn = 0
    fp = 0
    tn = 0
    tp = 0
    testing_data = "/home/msiddique/WSDL_Work/WebScience/Assignment8/Dataset/testing/"
    for dir, path, files in os.walk(testing_data):
        for file_name in files:
            classifier_result = classify_document(cl, dir + "/" + file_name)
            file_tag = dir.split("/")[-1]
            if file_tag == "notspam":
                file_tag = "not spam"
            if file_tag == "spam" and classifier_result == "spam":
                tp += 1
```

```
            elif file_tag == "spam" and classifier_result == "not spam":
                fp += 1
            elif file_tag == "not spam" and classifier_result == "spam":
                fn += 1
            elif file_tag == "not spam" and classifier_result == "not spam":
                tn += 1

            print(classifier_result)
    print("TP: " + str(tp))
    print("FP: " + str(fp))
    print("TN: " + str(tn))
    print("FN: " + str(fn))
    print("Recall: " + str(tp/(tp + fn)))
    print("Precision: " + str(tp / (tp + fp)))
    print("Accuracy: " + str((tp + tn) / (tn + tp + fn + fp)))
```

Problem 4

Report the precision and accuracy scores of your classification results (see: https://en.wikipedia.org/wiki/Precision_and_recall)

SOLUTION

$Recall = \frac{TP}{TP+FN}$
$Recall = \frac{3}{3+0}$
$Recall = 1$

$Precision = \frac{TP}{TP+FP}$
$Precision = \frac{3}{3+7}$
$Precision = 0.3$

$Accuracy = \frac{TP+TN}{TP+TN+FN+FP}$
$Accuracy = \frac{3+10}{3+10+0+7}$
$Accuracy = 0.65$