eYSIP-2018

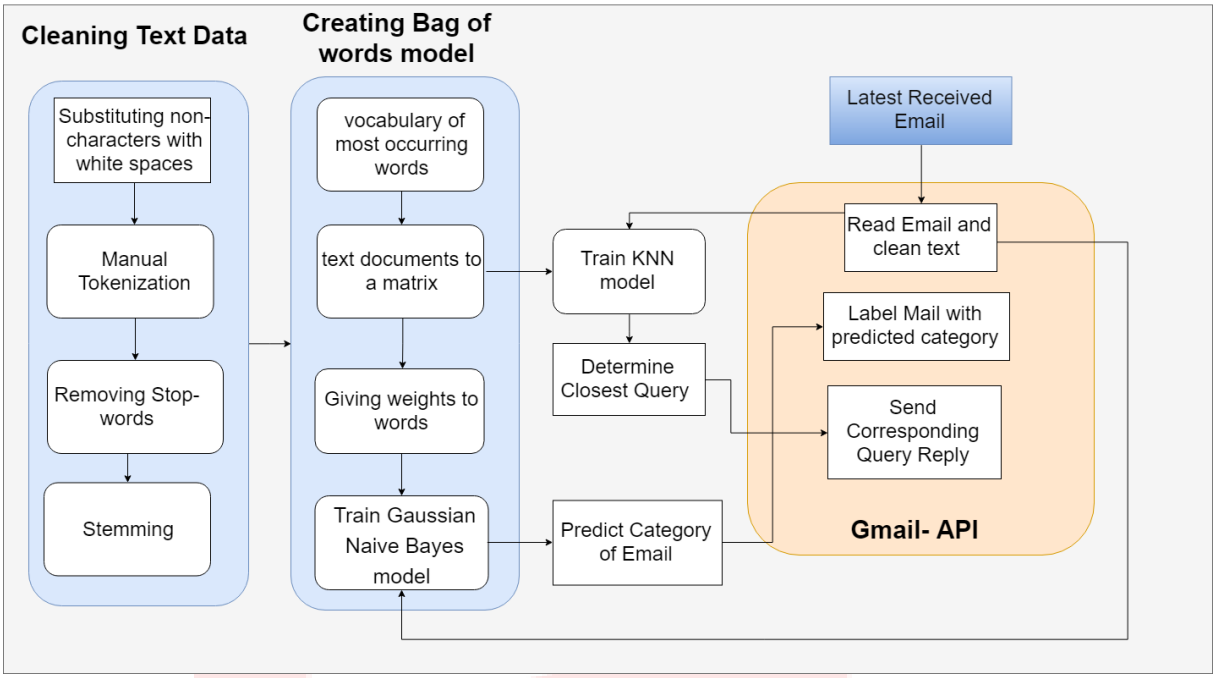
Automated Reply Tutorial

Authors:
Swapnil Masurekar
Abhishek Sharma

Duration of Internship: 21/05/2018 – 06/07/2018

2018, e-Yantra Publication

0.1 Overview



During e-YRC help-desk and piazza receive lot of queries regarding certificate, battery, etc. Many queries are similar and are subject to same replies, so our code finds the most similar queries amongst the queries already present and send the corresponding reply. Even if the similar query is not present in the data, the mail is labeled correctly with the category to which the query belongs to, so its efficient for the user to reply to these remaining queries. Entire system flow is explained briefly in Figure: Automated-Reply Flowchart

0.2 Installation Instructions:

The list of libraries required for each application are:

- Natural Language Toolkit v3.2.5
- *Dependencies for using Gmail-API:* apiclient, httplib2, outh2client
- Scikit-Learn v0.19.1
- Pandas v0.22.0

For installation instruction refer: Installation Instruction

0.3 Dataset Description

The Piazza_dataset.txt includes the piazza queries related to certificate, batteries, etc and their corresponding replies.

0.4 Importing Libraries:

All the required are shown below, later more libraries would be imported as per requirement

Python code snippet:

```
from __future__ import division, print_function
print("Importing Libraries....")
import re, numpy as np
#nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from collections import Counter
from apiclient import errors
from apiclient.discovery import build
from httplib2 import Http
from oauth2client import file, client, tools
import base64
from email.mime.text import MIMEText
from sklearn.externals import joblib # to save the model
```

0.5 Creating Bag of words model:

0.5.1 Reading the dataset:

We read the dataset by pandas library,

Python code snippet:

```
print("Reading Dataset....")
import pandas as pd
filename="Piazza_Dataset.txt"
Dataset=pd.read_csv(filename, sep='\t', lineterminator='\r')
corpus_piazza_queries=Dataset['QUERIES']
query_class_og=Dataset['CATEGORY'].tolist()
```

0.5.2 Cleaning the dataset:

Cleaning the dataset includes:

1. Substituting non-characters with white spaces
2. Manual Tokenization: Split into words by white space
3. Removing Stop-words
4. Text Normalization - Stemming

This done by the following function, input of the function is list of strings and output is list of strings after cleaning

Python code snippet:

```
def clean_dataset(Dataset):
# Cleaning the dataset: Removing punctuations and stopwords
    corpus = []
    for i in range(0, len(Dataset)):
                        # Creating corpus
        review = re.sub('[^a-zA-Z?]', ' ', Dataset[i])
                        # Removing punctuations
        review = review.lower()
                        # Lower-case
        review = review.split()
                        # Splitting sentence into list of words (
    tokenization)
        ps = PorterStemmer()
        review = [ps.stem(word) for word in review if not word
    in set(stopwords.words('english'))]
        review = ' '.join(review)
                        # Joining the words again
        corpus.append(review)
    return corpus
```

Now, Clean the dataset:

Python code snippet:

```
corpus_piazza_stemmed=clean_dataset(corpus_piazza_queries)
```

Input output example of for clean dataset function is as shown,

Python code snippet:

```
Input examples:["I haven't received the dataset, please help.."]
        ["The battery is not working properly"]
        ["The batteries is not working properly"]
Output examples: ['receiv dataset pleas help']
        ['batteri work properli']
        ['batteri work properli']
```

From the above examples you can note that while predicting the category of the query the words "battery" and "batteries" should be taken as same word while creating bag of words model (as they correspond to same meaning). Hence, we do stemming to represent such words as same word. eg: "batteri". corpus_piazza_stemmed includes all cleaned version of the original dataset. To know more about stemming refer this link: Stemming

0.5.3 Creating Vocabulary for bag of words model:

We need to make vocabulary of 150 most occurring words in the entire dataset. These words would then become the independent 1D vectors for fitting the classification model. Example of bag of words is shown here: Bag of words

Python code snippet:

```
# Creating Vocabulary
vocab=[]
for i in range(len(most_occur)):
    vocab.append(most_occur[i][0])

# Creating the Bag of Words model for classification using bayes
    theorem
corpus_piazza_classify=corpus_piazza_stemmed
query_class=query_class_og
print("Creating Bag of words model....")
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(vocabulary=vocab)
corpus_piazza_classify = cv.fit_transform(corpus_piazza_classify
    ).toarray()
```

Now, we manually give weights to the words which represents the category of the query. In our case the words are "battery" and "certificate". This is done by,

Python code snippet:

```
# Giving weights to words which belong to category
category_name=[]
counter_category=Counter(query_class)
most_occur_category=counter_category.most_common(150)
for i in most_occur_category:
    category_name.append(i[0])
all_categories=clean_dataset(category_name)

corpus_piazza_classify_w = corpus_piazza_classify
manual_weight=2
                    # As corpus_piazza_classify is int64
for i in range(len(vocab)):
    if(vocab[i] in all_categories):
        for j in range(len(corpus_piazza_classify)):
            corpus_piazza_classify_w[j][i]=
   corpus_piazza_classify_w[j][i]*manual_weight
```

0.6 Training the Gaussian Naive Bayes model:

Before training the model, it is necessary to encode the dependent variable. This is done by,
Python code snippet:

```
from sklearn.preprocessing import LabelEncoder
# Encoding the Dependent Variable
labelencoder_query_class = LabelEncoder()
query_class = labelencoder_query_class.fit_transform(query_class
    )
```

Now, we fit Gaussian Naive bayes model to our training data. As we have less training data Naive-Bayes is the best classifier, this is the model used for Google spam detection. To fit data in Naive Bayes model we use the following code snippet. Remember is the model only to classify the category of the query. Eg: certificate related, battery related, etc.
Python code snippet:

```
from sklearn.naive_bayes import GaussianNB
classifier_upper_NB = GaussianNB()
classifier_upper_NB.fit(corpus_piazza_classify_w, query_class)
```

0.7 Sentence Similarity Approach:

This approach is based on paper "Sentence Similarity based on Semantics nets" by Y. Li, D. McLean, Z.A. Bandar, J.D. O'Shea, K. Crockett. The github link for the code based on this paper: Sentence Similarity. Refer "Automated_Reply_Similarity.py" for implementation of "Sentence Similarity based on Semantics nets" in Automated Reply for piazza queries: Automated Reply System

0.8 K-NN Approach:

In K-NN approach we find closest similar query considering top words in vocabulary as independent vectors in N-Dimensions (N=150). Now we train K-NN model for all categories separately. This snippet also save the model.

```
for category in category_name:
    train_set_category=[]
    for i in range(len(corpus_piazza_classify)):
        if(query_class_og[i]==category):
            train_set_category.append(corpus_piazza_classify[i])
    train_y_category=range(len(train_set_category))
```

```
    # Fitting K-NN to the Training set
    from sklearn.neighbors import KNeighborsClassifier
    classifier_category = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
    classifier_category.fit(train_set_category, train_y_category)
    filename_category='classifier_KNN_'+category+'.sav'
    joblib.dump(classifier_category, filename_category)
```

0.9 Gmail API:

Gmail API is used to modify email labels, reading mails and sending replies. To access Gmail-API using python project to need to create project on Google API. In order to create project, get credentials and client secrets refer this video. The quick-start documentation and installation instructions are provided here: Google-Documentation. In order to use pip install with anaconda you need to add *"../Anaconda3/Scripts"* path to environment variables.The example of labeling and reply is shown in Figure: Gmail Labeling and Reply.
For the Gmail API python functions refer: Gmail-API Python
All the functions here are modified as per the requirement and well commented for better understanding.

0.10 Reading the latest received message:

Before coming to this section be sure that you have gone through all the links in the Gmail API section. Now before reading the message you need to setup the Gmail API service.Ensure that you have your 'credentials.json', and 'client_secret.json' in the same directory.This is done by,
Python code snippet:

```
# Setup the Gmail API service
SCOPES = 'https://www.googleapis.com/auth/gmail.modify'
store = file.Storage('credentials.json')
creds = store.get()
if not creds or creds.invalid:
    print("here")
    flow = client.flow_from_clientsecrets('client_secret.json', SCOPES)
    creds = tools.run_flow(flow, store)
service = build('gmail', 'v1', http=creds.authorize(Http()))
user_id="me"
```

Now for reading the latest received message,
Python code snippet:

```
# Get the latest Received Message
message_list=ListMessagesMatchingQuery(service, user_id, query='')
latest_received_msg_id = identify_latest_received_message(service, user_id, message_list)
message_read= GetMessage(service, user_id, latest_received_msg_id)
print("Latest received message is: ","\n    Message subject: ", message_read['subject'],"\n    Message body is: ", message_read['body'],"\n    From: ",message_read['From'])
```

0.11 Prediction:

Now we predict the category in which the received message belongs to. The message can be only be passed as an parameter to predict function after it is vectorized as per the vocabulary created. So this is done as follows,
Python code snippet:

```
# Prediction
test_input_clean = clean_dataset([test_input_og])
test_input = cv.transform(test_input_clean).toarray()
test_prediction = classifier_upper_NB.predict(test_input)
test_prediction_proba = classifier_upper_NB.predict_proba(test_input)                    # Get probability for each category for given test input
test_prediction_text=labelencoder_query_class.inverse_transform(test_prediction)
print("The query belong to category --> ",test_prediction_text[0])
```

Most similar query is determined by loading appropriate K-NN model and predict the closet query from the dataset such that corresponding reply can be sent accordingly.
Python code snippet:

```
send_flag=0
if(test_prediction_text[0]!="others"):
    # Load all replies from predicted category
    all_replies_from_predicted_category=[]
    send_flag=1
    for i in range(len(query_class_og)):
        if(test_prediction_text[0]==query_class_og[i]):
            all_replies_from_predicted_category.append(Dataset['REPLY'][i])
```

```
    pred=classifier_category.predict(test_input)
    reply_prediction_probability = classifier_category.
predict_proba(test_input)

    reply_prediction_probability = np.array(
reply_prediction_probability[0])
    top_3_indices = reply_prediction_probability.argsort()
[-3:][::-1]
    print(top_3_indices)

    generated_reply = all_replies_from_predicted_category[pred
[0]]

    next_2_predictions = []
    for i in top_3_indices:
        if(i != pred[0]):
            next_2_predictions.append(
all_replies_from_predicted_category[i])
```

0.12 Modifying label:

Ensure that you have gone through all the links in the Gmail API section. This is the code snippet to modify the email label (default is "Inbox") with the category name. Before running this code snippet you can either make the category labels manually on gmail or by using Gmail-API. (**Note:** Keep the label name same as the category name, if not then "get_label_id" function will cause error). The code snippet for modifying label is as shown below,
Python code snippet:

```
# Modifying message's label
print("Adding label to the latest received message....")
msg_labels=CreateMsgLabels()
                    # Create object to update labels
msg_labels['addLabelIds']=[get_label_id(test_prediction_text[0],
    service, user_id)]
ModifyMessage(service, user_id, latest_received_msg_id,
   msg_labels)
```

0.13 Sending Reply:

After getting the closest query from K-NN model. The reply corresponding to that query would be sent using following code snippet. The below code is executed if "send_flag" is set.(send_flag is set if query doesn't belong to other

category)

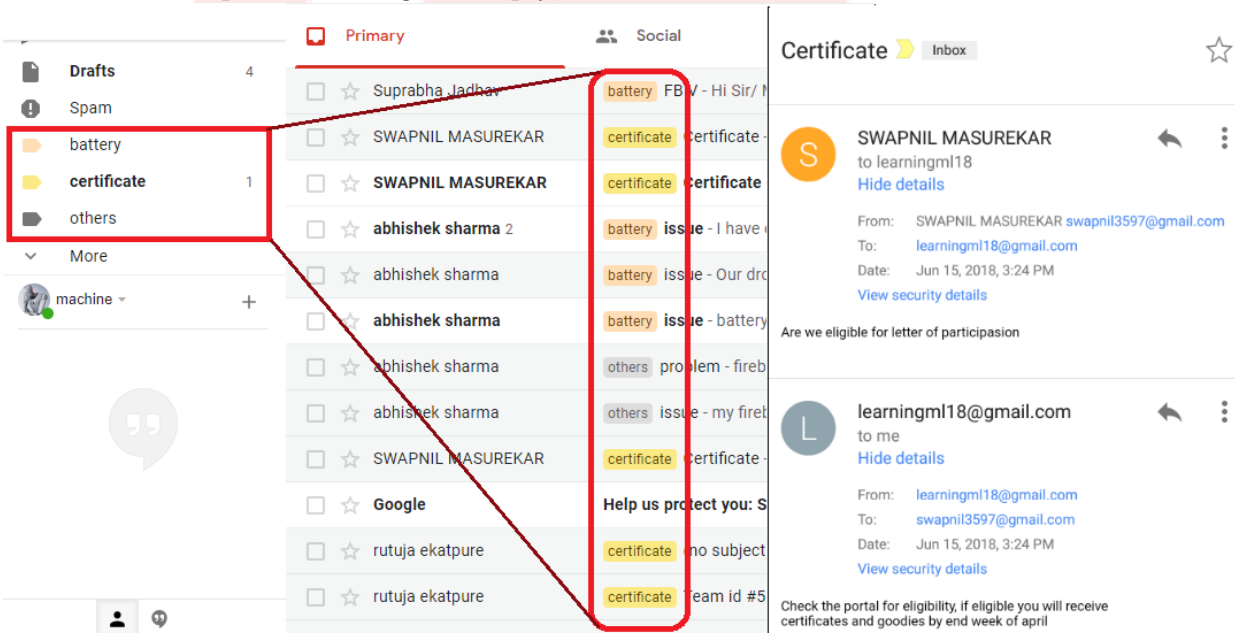
Python code snippet:

```
# Sending reply
if(send_flag==1):
    print("\nGenerated Reply is ==> ",generated_reply,"\n")
    print("Next suggestions are:")
    k=''
    for i in next_2_predictions:
        if(i != generated_reply and i!=k):
            k = i
            print("   ",i) # print next sugestions

    print("Sending Reply....")
    message_text=generated_reply
    message_send = CreateMessage(user_id, message_read['From'],
'Re:'+message_read['subject'], message_text)
      #'Re:'+message_read['subject'] fo replying
    SendMessage(service, user_id, message_send)
```

0.14 OUTPUT

This is an example of labeling and reply,



Useful ML Resources:

- Text Documents
    1. Machine Learning and Algorithms Here you will get the brief idea about ML and Its basic learning algorithm
    2. Basic Terminology related to ML Here you will learn about the basic terminologies and jargons used in ML and Neural Network
    3. Artificial Neural network A Quick introduction to ANN
    4. Convolution Neural network Introduction to Architecture of CNN
- For video tutorial over machine learning refer to following links
    1. Machine learning course by Andrew ng - Click here (enroll for free)
    2. Machine learning A-Z Hands on python in data science - click here(refer tutorials for Python only)
    3. Recurrent Neural Network by Andrew ng - Click here(refer video till LSTM only)
    4. Deep learning by Andrew ng - Click here
    5. Convolution Neural Network by Andrew ng - Click here