User Manual

System Requirements

- Windows 10
- Python 3.7.2
    - pickle
    - pandas 1.0.1
    - os
    - autocorrect 2.0.0
    - spacy 2.3.2
    - random
    - email
    - numpy 1.18.1
    - matplotlib 3.1.3
    - sklearn 0.22.1
    - seaborn 0.10.0
- Jupyter Notebook / Visual Studio Code

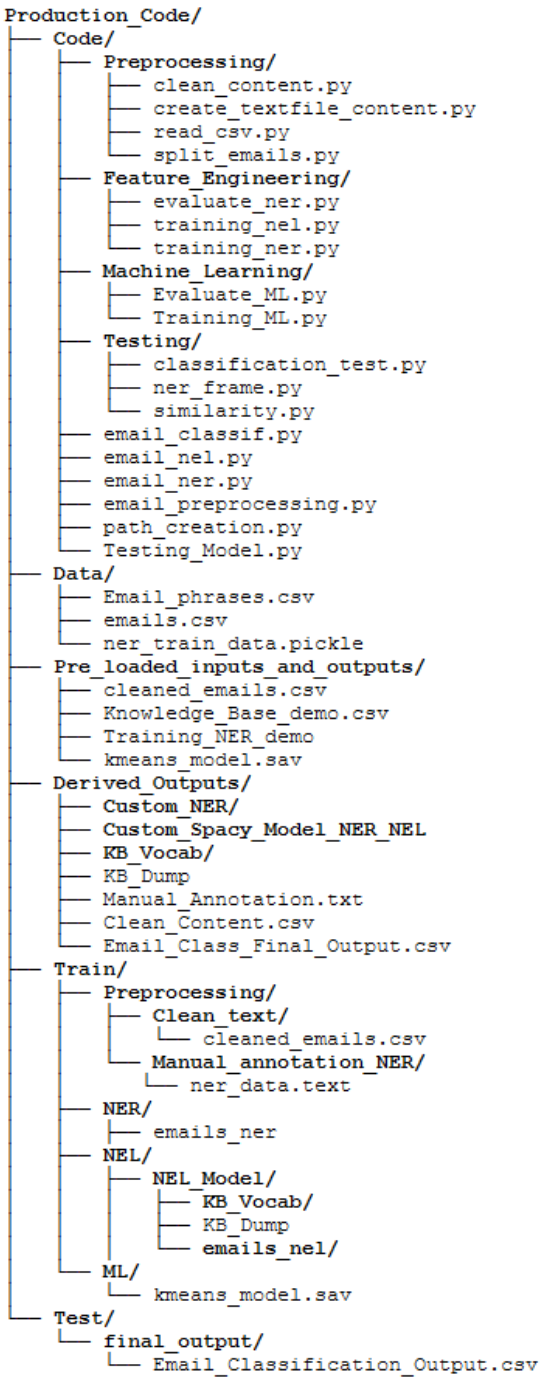
Note: The below user manual is recommended only for windows system

Contents

Email Classification

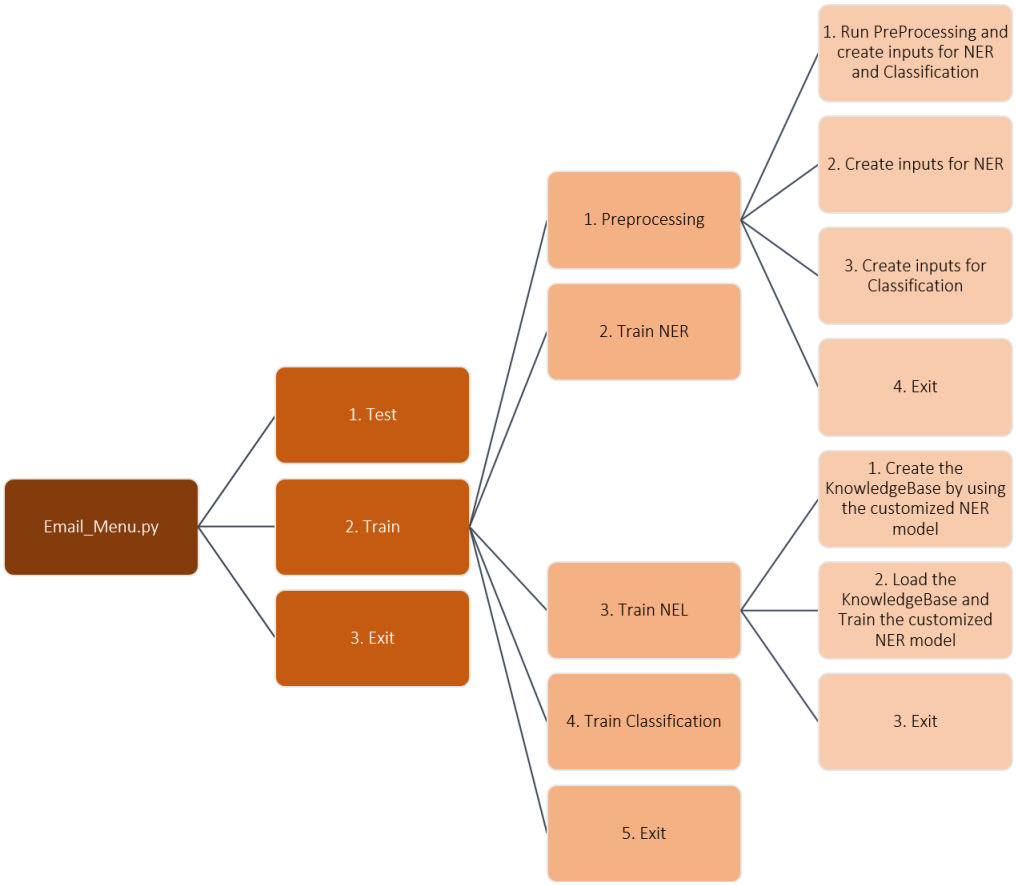
The main goal of this project is to perform information extraction on emails and classify them. This documentation guides through the process involved in email classification and the modules and packages associated with it.

1. Folder Structure



2. Implementation in Theory

Below are the Architecture of the Menu functions used to train and test the model



Initially, we must move our current directory to **Production code** folder and open a terminal based on that directory.

Example:

```
D:\Case Study 1\Production_Code>
```

Since our current directory is now set, in order to start classification of emails we have to execute the **Email_Menu** python file.

Example:

```
D:\Case Study 1\Production_Code>python Email_Menu.py
```

After executing the Email_Menu python file, we will be provided with three options such as

```
1. TEST
2. TRAIN
3. Exit

Enter your choice:
```

If training data is already created, we can proceed with the option **'1. Test'** for classification and information extraction of emails.

If we have to create training data, we have to proceed with the option **'2. Train'** to create the required training data for classification and information extraction of emails.

2.1 Training

The training data required to classify the emails and extract required information is created in this section as follows.

Select '2. Train'

```
1. TEST
2. TRAIN
3. Exit
Enter your choice: 2
```

After entering your choice as **2** the process will enter the training phase and the options available in training will be displayed.

```
1. Preprocessing
2. Train NER
3. Train NEL
4. Train Classification
5. Exit

Enter your choice:
```

1. **Preprocessing:** Preprocessing allows us to create the inputs required for training NER model, NEL model and classification model.
2. **Train NER:** Train NER allows us to train the neural network model provided in SpaCY library for named entity recognition based on our training data created during 'Preprocessing'.
3. **Train NEL:** Train NEL allows us to re-train the neural network model created in 'Train NER' for entity linking based on our training data created during 'Preprocessing'.
4. **Train Classification:** Train Classification allows us to create the machine learning model based on clusters for email classification.

In order to create the training inputs for 'Train NER', 'Train NEL' and 'Train Classification', we have to run 'Preprocessing' first.

Select 1. Preprocessing

```
1. Preprocessing
2. Train NER
3. Train NEL
4. Train Classification
5. Exit
Enter your choice: 1
```

After entering your choice as **1** the process will enter the preprocessing phase for training data and the options available will be displayed.

```
1. Run PreProcessing and create inputs for NER and Classification
2. Create inputs for NER
3. Create inputs for Classification
4. Exit

Enter your choice:
```

1. **Run PreProcessing and create inputs for NER and Classification:** This allows us to preprocess the emails and create the inputs required for both named entity recognition and classification.
2. **Create inputs for NER:** This allows us to create inputs required only for named entity recognition.
3. **Create inputs for Classification:** This allows us to create inputs required only for classification.

Based on the requirement, we can select the suitable choice

Select 1. Run PreProcessing and create inputs for NER and Classification

```
1. Run PreProcessing and create inputs for NER and Classification
2. Create inputs for NER
3. Create inputs for Classification
4. Exit
Enter your choice: 1
```

After entering your choice as **1** the files available in the **data** folder will be displayed

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
```

Enter the name of the csv file that needs to be preprocessed. In our case **'emails.csv'** has the enron email data that needs to be preprocessed.

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv
```

After entering the name of the csv file, the number of rows that has to read from csv file is required to preprocess

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']
Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv

Enter the nrows to be considered:
```

Enter the number of rows. In our case, for sample we are reading just 10 rows by entering number **10**

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']
Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv

Enter the nrows to be considered: 10
```

After entering the number of rows, the columns available in the dataset is displayed

```
Provided dataset has below columns
file
message

Please provide the data column to train the model:
```

Here, we have to enter the name of the column that contains email data and needs to be preprocessed. In our case, **message** column contains email data and it is entered as input in the provided field.

```
Provided dataset has below columns
file
message

Please provide the data column to train the model: message
```

After entering the column name, a csv file is generated which contains the inputs for training named entity recognition model and it is saved to (./Train/Preprocessing/Manual_annotation_NER).

```
Entered column name is message
Generating file for the NER manual annotation in the below location...
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Train\Preprocessing\Manual_annotation_NER

Enter the File Name :
```

Enter the file name for the csv generated to save it in the location with the name provided.

```
Enter the File Name : training_ner.txt
```

A csv file is generated for the similarity functionality with the clean text (without forward chain mails). It will be stored in the following location (./Train/Preprocessing/Clean_text)

```
Enter the File Name :

Generating file for the Similarity in the below location...
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Train\Preprocessing\Clean_text
```

Enter a name for that file to store it in the location with the name provided.

```
Enter the File Name : train_similarity.csv
```

Select 2. Create inputs for NER

This option is selected when only training data for named entity recognition is required.

```
1. Run PreProcessing and create inputs for NER and Classification
2. Create inputs for NER
3. Create inputs for Classification
4. Exit

Enter your choice: 2
```

After selecting **2** as your choice the files available in **data** folder will be displayed.

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
```

Enter the name of the csv file that needs to be preprocessed. In our case **'emails.csv'** has the enron email data that needs to be preprocessed.

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv
```

After entering the name of the csv file, the number of rows that has to read from csv file is required to preprocess

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']
Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv

Enter the nrows to be considered:
```

Enter the number of rows. In our case, for sample we are reading just 10 rows by entering number **10**

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']
Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv

Enter the nrows to be considered: 10
```

After entering the number of rows, the columns available in the dataset is displayed

```
Provided dataset has below columns
file
message

Please provide the data column to train the model:
```

Here, we have to enter the name of the column that contains email data and needs to be preprocessed. In our case, **message** column contains email data and it is entered as input in the provided field.

```
Provided dataset has below columns
file
message

Please provide the data column to train the model: message
```

After entering the column name, a csv file is generated which contains the inputs for training named entity recognition model and it is saved to (./Train/Preprocessing/Manual_annotation_NER).

```
Entered column name is message
Generating file for the NER manual annotation in the below location...
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Train\Preprocessing\Manual_annotation_NER

Enter the File Name :
```

Enter the file name for the csv generated to save it in the location with the name provided.

```
Enter the File Name : training_ner.txt
```

Select 3. Create inputs for classification:

This option is selected when only training data foe classification is required. After selecting **3** as your choice the files available in **data** folder will be displayed.

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
```

Enter the name of the csv file that needs to be preprocessed. In our case **'emails.csv'** has the enron email data that needs to be preprocessed.

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv
```

After entering the name of the csv file, the number of rows that has to read from csv file is required to preprocess

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']
Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv

Enter the nrows to be considered:
```

Enter the number of rows. In our case, for sample we are reading just 10 rows by entering number **10**

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']
Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv

Enter the nrows to be considered: 10
```

After entering the number of rows, the columns available in the dataset is displayed

```
Provided dataset has below columns
file
message

Please provide the data column to train the model:
```

Here, we have to enter the name of the column that contains email data and needs to be preprocessed. In our case, **message** column contains email data and it is entered as input in the provided field.

```
Provided dataset has below columns
file
message

Please provide the data column to train the model: message
```

A csv file is generated for the similarity functionality with the clean text (without forward chain mails). It will be stored in the following location (./Train/Preprocessing/Clean_text)

```
Enter the File Name :

Generating file for the Similarity in the below location...
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Train\Preprocessing\Clean_text
```

Enter a name for that file to store it in the location with the name provided.

```
Enter the File Name : train_similarity.csv
```

After preprocessing, we return back to training menu again.

Now we have the training data for 'Train NER' and 'Train NEL' and 'Train Classification' ready created from preprocessing step.

Select 2. Train NER

When **2** is selected the training of named entity recognition phase is entered and provided with the following options. It displays the available pickle file in the **data** folder which is required as input for training named entity recognition.

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of NER train data in Data folder.
['ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
```

Enter the name of the pickle file

```
Please provide the name to choose the file. If you want to choose other path enter 'y':
ner_train_data.pickle
```

Enter the number of iterations to train the model

```
Enter the number of iterations to train the NER model : 20
```

Enter the name for the model that has been trained

```
Enter your Model Name: ner_model
```

Now the trained NER model is saved to (./Train/NER/) location

Select 3. Train NEL

When **3** is selected the training of named entity linking is entered and provided with the following options

```
1. Create the KnowledgeBase by using the customized NER model
2. Load the KnowledgeBase and Train the customized NER model
3. Exit

Enter your choice:
```

Select 1. Create the KnowledgeBase by using the customized NER model

This allows us to create the knowledge base based on the model created during 'Train NER' and enter the name of the model.

```
Enter your choice: 1
We found below models in the existing trained NER folder.
['emails_ner', 'ner_model']

Please provide the folder name to choose for training. If you want to choose in other path enter 'y':
ner_model
```

Enter the name of pickle file which was given as input in 'Train NER' for training the model

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of NER train data in Data folder.
['ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
ner_train_data.pickle
```

Enter the name of the knowledge base file created

```
Enter the File Name of the KB with the extension .csv : my_kb.csv
```

Enter the name of the csv file generated as the input for entity linking training

```
Enter the File Name to save the inputs for Training with the extension .csv :
my_input.csv
```

Select 2. load the KnowledgeBase and Train the customized NER model

This allows us to load the knowledge base based on the model created during 'Train NER' and enter the name of the model.

```
Enter your choice: 1
We found below models in the existing trained NER folder.
['emails_ner', 'ner_model']

Please provide the folder name to choose for training. If you want to choose in other path enter 'y':
ner_model
```

Enter the name of the knowledge base file to be loaded.

```
We found below files/models of Knowledge Base in Pre_input folder.
['cleaned_emails.csv', 'Knowledge_Base_demo.csv', 'Training_NER_demo.csv']

Please provide the name to choose the file. If you want to choose other path enter 'y':
Knowledge_Base_demo.csv
```

Enter the training inputs for training entity linking model

```
Please provide the name to choose the file. If you want to choose other path enter 'y':
Training_NER_demo.csv
```

Select 4. Train Classification

When **4** is selected, the training of classification model is entered and enter the name of the csv file with clean emails

```
We found below files/models of Clean Text in Preprocessing folder.
['cleaned_emails.csv', 'train_similarity.csv']

Please provide the name to choose the file. If you want to choose other path enter 'y':
cleaned_emails.csv
```

Enter the name of the column required to train the classification model.

```
Provided dataset has below columns
message
clean_text

Please provide the data column to train the model: clean_text
```

The model is successfully trained and saved to the disk in (./Train/ML) location

```
Entered column name is clean_text
Clustering using KMeans...
Please wait for model training.........
clusters are 1
Clustering model has been successfully trained
```

2.2 Testing

After completing the creation of training data and required inputs in training phase, in order to test the trained model with test data **Testing** is implemented.

Select **1** in the **Email_Menu** option to enter testing phase.

```
1. TEST
2. TRAIN
3. Exit
Enter your choice: 1
```

After selecting testing phase, the following inputs are required.
Enter the csv file name with raw email data for classification and information extraction

```
D:\Masters\Sem\Sem2\Case Study 1\Final_pipeline\Production_Code\Data
We found below files/models of Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
emails.csv
```

Enter the number of rows that needs to be read from the csv file. In our case, for sample number **10** is given.

```
Enter the nrows to be considered: 10
```

Enter the name of the data column that contains raw email data.

```
CSV imported successfully

Provided dataset has below columns
file
message

Please provide the data column to test the model: message
```

Enter the name of the phrases file for similarity functionality.

```
We found below files/models of Generic Phrases in Data folder.
['emails.csv', 'Email_phrases.csv', 'ner_train_data.pickle']

Please provide the name to choose the file. If you want to choose other path enter 'y':
Email_phrases.csv
```

Enter the name of the machine learning model for classification of the emails.

```
We found below files/models of email classification in Machine learning folder.
['kmeans_model.sav']

Please provide the name to choose the file. If you want to choose other path enter 'y':
kmeans_model.sav
```

Enter the name of the neural network model trained for named entity recognition and named entity linking

```
We found below models in the existing trained NER folder.
['emails_nel', 'KB_Vocab']

Please provide the folder name to choose for training. If you want to choose in other path enter 'y':
emails_nel
```

Enter the name for the generated csv output file to save it to (./Test/final_output) location

```
Enter the File name for the final output with extension .csv:
test_output.csv
```

3. Implementation in Jupyter Notebook

3.1 Training

The steps involved in training email classification are as follows,

1. Preprocessing
2. Feature Engineering
3. Machine Learning

Importing the file

Initially, the file containing the emails and the number of rows from that file is declared using the below variables.

- "file_path" variable denotes the path of the emails csv file that needs to be processed for email classification
- "rows" variable denotes the number of rows that must be read from the csv file

```
file = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Data/emails.csv"
rows = 10
```

1. Preprocessing

In preprocessing we need to perform following steps,

1.1 Read the CSV File
1.2 Split the Emails according to the email headers
1.3 Export the email body column to "txt" for the Manual Annotation
1.4 Elimination of forwarded content from email body

1.1 Read the CSV File

Here, the imported file is read using the ***read_file*** function available in ***read_csv*** module under ***Preprocessing*** package.

read_file function

Description:
Reads the csv file and checks with the extension whether it is csv else throws an exception to upload the file with csv extension.
Parameters:
Accepts two parameters such as path of the file and the number of rows.
Returns:
returns the csv file read in a dataframe.

```
from Code.Preprocessing.read_csv import read_file
emails = read_file(file,rows)
emails.head(5)
```

```
CSV imported successfully
```

	file	message
0	allen-p/_sent_mail/1.	Message-ID: <18782981.1075855378110.JavaMail.e...
1	allen-p/_sent_mail/10.	Message-ID: <15464986.1075855378456.JavaMail.e...
2	allen-p/_sent_mail/100.	Message-ID: <24216240.1075855687451.JavaMail.e...
3	allen-p/_sent_mail/1000.	Message-ID: <13505866.1075863688222.JavaMail.e...
4	allen-p/_sent_mail/1001.	Message-ID: <30922949.1075863688243.JavaMail.e...

1.2 Splitting the emails

The ***Preprocess*** class in ***split_emails*** module under ***Preprocessing*** package is developed to split the emails based on headers.

class Preprocess

preprocessing_emails function

Description:

splits the emails based on the headers present within the email data.

Parameters:

Accepts one parameter as a dataframe.

Returns:

returns emails with respective headers in each column of the dataframe.

```
from Code.Preprocessing.split_emails import Preprocess
information = Preprocess()
analysis = information.preprocessing_emails(emails, 'message')
analysis.head(5)
```

1.3 Export the email body column to text file for the Manual Annotations

The email column in the dataframe is exported as a text file using ***create_textfile*** function available in ***create_textfile_content*** module under ***Preprocessing*** package for manual annotations.

create_textfile function

Description:

writes the content of a dataframe to a text file.

Parameters:

accepts one parameter as a column of the dataframe.

Returns:

returns a text file with the content of the dataframe which is saved to the local directory.

```
from Code.Preprocessing.create_textfile_content import create_textfile
create_textfile(analysis['email_body'])
```

1.4 Elimination of forwarded content from email body

The original body of the email is segregated from the forwarded content using ***clean_emails*** function in ***clean_content*** package under ***Preprocessing*** package.

clean_emails function

Description:
Eliminates the forwarded content and retains the original body of the email and checks the spelling of the content
Parameters:
Accepts two parameters as a dataframe and a data column name.
Returns:
Returns only the original body of email with corrected spelling in a column of the dataframe.

The output of clean_emails function is exported as a text file using the create_textfile function.

```
from Code.Preprocessing.clean_content import CleanEmails
clean_class = CleanEmails(analysis,'message')
clean_nlp_content = clean_class.clean_emails()
clean_nlp_content.head(5)
```

2. Preprocessing

The following steps are performed in feature engineering,

2.1 Named Entity Recognition
2.2 Named Entity Linking

2.1 Named Entity Recognition

train_spacy function

Description:
With the manually annotated data as training data, a blank neural model in spacy library is trained for extracting named entities from email data.
Parameters:
Accepts two parameters such as training data (manually annotated pickle file) and number of iterations.
Returns:
A neural network model trained specific to the training data and training data for evaluation purpose.

```
from Code.Feature_Engineering.training_ner import train_spacy
import pickle
ner_pickle_input = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14
/Data/ner_train_data.pickle"
ner_iterations = 20
_, nlp = train_spacy(ner_pickle_input,ner_iterations)
ner_model = input("Provide the name for NER model")
nlp.to_disk(ner_model)
```

2.2 Named Entity Linking

A neural model specific to the training data is developed to extract named entity linking for the exising named entities in the data using ***NelTraining*** class under ***Train_NEL*** package.

NelTraining class

Description:

Class uses trained NER model to generate the knowledge base and trains the names entity linking.

Parameters:

Accepts one parameter such as location of the trained NER model.

```
from Code.Feature_Engineering.training_nel import NelTraining
import pandas as pd

cus_ner = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Derived_Outputs/Custom_NER"
cus_ner_path = NelTraining(cus_ner)
```

2.2.1 creating_knowledge

creating_knowledge function

Description:

With NER training data and by converting the NER data with custom NER model, knowledge base is generated.

Parameters:

Accepts one parameter such as location of the NER training data pickle file.

Returns:

A .csv file with entities and frequencies.

```
cus_ner_path.creating_knowledge(ner_pickle_input)
```

2.2.2 settingup_knowledgebase

settingup_knowledgebase function

Description:

Function imports knowledge base data to set up entities and aliases using the vocabulary from trained NER model, and trains the named entity linking on top of the trained NER model.

Parameters:

Accepts two parameters such as location of the knowledge base file and training NER data with QID's

Returns:

Two pickle files with knowledge base dump and knowledge base vocab, and a neural model trained specific to the training data

```
kb_input = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Pre_loaded_inputs_and_outputs
/Knowledge_Base_demo.csv"
names =  pd.read_csv(kb_input, sep=';')
training_nel_data = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14
/Pre_loaded_inputs_and_outputs/Training_NER_demo.csv"
train_data_2 =  pd.read_csv(training_nel_data, sep=';')
cus_ner_path.settingup_knowledgebase(names,train_data_2)
```

3. Machine learning

Machine learning unsupervised model specific to the trained data would be generated to automatically clustering the data ***TrainML*** class under ***Training_ML*** package

TrainML Class

Description:

Class will train Machine learning model to generate the clusters on preprocessed training data.

Parameters:

Accepts two parameters such as a dataframe and a data column.

```
from Code.Machine_Learning.Training_ML import TrainML
from Code.Preprocessing.read_csv import read_file

ml_input = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Pre_loaded_inputs_and_outputs
/cleaned_emails.csv"
ml_data = read_file(ml_input)
print(ml_data.head(5))
cust_ml_model = TrainML(ml_data,'clean_text')
```

3.1 Training Clustering model

train_ml function

Description:

Function takes the dataframe from the TrainML class to generate the model with clusters based on the Elbow strength of the Inertia scores

Parameters:

Accepts three parameters while training the model such as random state, number of iterations and top words for display purpose.

Default parameters:

Random state = 42, iterations = 100, top words = 20

Returns:

One pickle file with trained clustering model specific to the training data

```
cust_ml_model.train_ml()
```

3.2 Testing

The steps involved in testing email classification are as follows,

1. Preprocessing
2. Feature Engineering
3. Machine Learning

Importing the file

Initially, the file containing the emails and the number of rows from that file is declared using the below variables.

- "file_path" variable denotes the path of the emails csv file that needs to be processed for email classification
- "rows" variable denotes the number of rows that must be read from the csv file

```
file = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Data/emails.csv"
rows = 10
```

1. Preprocessing

In preprocessing we need to perform following steps,

1.1 Read the CSV File
1.2 Split the Emails according to the email headers

1.3 Elimination of forwarded content from email body

1.1 Read the CSV File

Here, the imported file is read using the ***read_file*** function available in ***read_csv*** module under ***Preprocessing*** package.

read_file function

Description:

Reads the csv file and checks with the extension whether it is csv else throws an exception to upload the file with csv extension.

Parameters:

Accepts two parameters such as path of the file and the number of rows.

Returns:

returns the csv file read in a dataframe.

```
from Code.Preprocessing.read_csv import read_file
emails = read_file(file,rows)
emails.head(5)
```

```
CSV imported successfully
```

	file	message
0	allen-p/_sent_mail/1.	Message-ID: <18782981.1075855378110.JavaMail.e...
1	allen-p/_sent_mail/10.	Message-ID: <15464986.1075855378456.JavaMail.e...
2	allen-p/_sent_mail/100.	Message-ID: <24216240.1075855687451.JavaMail.e...
3	allen-p/_sent_mail/1000.	Message-ID: <13505866.1075863688222.JavaMail.e...
4	allen-p/_sent_mail/1001.	Message-ID: <30922949.1075863688243.JavaMail.e...

1.2 Splitting the emails

The ***Preprocess*** class in ***split_emails*** module under ***Preprocessing*** package is developed to split the emails based on headers.

class Preprocess

preprocessing_emails function

Description:

splits the emails based on the headers present within the email data.

Parameters:

Accepts one parameter as a dataframe.

Returns:

returns emails with respective headers in each column of the dataframe.

```
from Code.Preprocessing.split_emails import Preprocess
information = Preprocess()
analysis = information.preprocessing_emails(emails, 'message')
analysis.head(5)
```

1.3 Elimination of forwarded content from email body

The original body of the email is segregated from the forwarded content using ***clean_emails*** function in ***clean_content*** package under ***Preprocessing*** package.

clean_emails function

Description:
Eliminates the forwarded content and retains the original body of the email and checks the spelling of the content
Parameters:
Accepts two parameters as a dataframe and a data column name.
Returns:
Returns only the original body of email with corrected spelling in a column of the dataframe.

The output of clean_emails function is exported as a text file using the create_textfile function.

```
from Code.Preprocessing.clean_content import CleanEmails
clean_class = CleanEmails(analysis,'message')
clean_nlp_content = clean_class.clean_emails()
clean_nlp_content.head(5)
```

2. Feature Engineering

In this step the Named Entity Recognitions and their respective Named Entity Linking would be extracted based on the trained model of NER and NEL uses ***ner_df*** function in ***ner_frame*** package under ***Testing*** package

ner_df function

Description:
Functions loads the trained NER & NEL model and retrieved the entities and entity linking from the testing emails.
Parameters:
Accepts two parameters as a existing trained NER & NEL model, and testing emails.
Returns:
Returns the entities available in the testing emails along with entity linking

```
model = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Derived_Outputs
/Custom_Spacy_Model_NER_NEL"
```

```
from Code.Testing.ner_frame import ner_df
ner_df(model, emails)
```

3. Machine Learning

The following steps are performed in Machine learning to label and clustering as part of testing,

3.1 Activity Labelling
3.2 Clustering

3.1. Activity Labelling

Document similarity model, that labels emails based on the generic email phrases uses ***Activity*** class under ***similarity*** package under ***Testing*** package

Activity Class

Description:

Class will compare the generic email phrases with input testing emails and label them with Activity.

Parameters:

Accepts one parameter such as a generic input email phrase

activity_entity Function

Description:

Function creates the vectors for both generic phrases and input testing emails by splitting into words and characters and checks the cosine similarity between them.

Parameters:

Accepts one parameter such as a input testing emails

Returns:

Returns the activity labels for the input testing emails

```
from Code.Testing.similarity import Activity
inp_phrases = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Data/Email_phrases.csv"
extraction_2 = Activity(inp_phrases)
extraction_2.activity_entity(emails)
```

3.2. Clustering

This step categorise the input testing emails into clusters based on the trained clustering model uses ***test_cluster_model*** Function in ***classification_test*** package under ***Testing*** package

test_cluster_model Function

Description:

Function creates transforms the text into vectors and fits into the trained clustering model

Parameters:

Accepts two parameters such as a trained clustering model and input testing emails

Returns:

Returns the clusters for the input testing emails

```
from Code.Testing.classification_test import test_cluster_model
model = "/Users/Sangireddy Siva/Dropbox/CaseStudy1/New Production code/Scripts_V14/Pre_loaded_inputs_and_outputs
/kmeans_model.sav"
test_cluster_model(model,emails)
```

```
emails_out = emails[['From','To','Subject','message','Date','activity','cluster','NER_Info']]
```

```
emails_out
```

```
emails_out.to_csv("Email_Class_Final_Output.csv")
```