Email classifier user guide

User guide for the Email classifier Cli. Created for the project in the course CS4125 at University of Limerick.

Christoffer Näs - 24271322

Selin Taskin - 24284378

Patrick Vorreiter - 24284335

2024-12-30

Table of Contents

1 Pre requirements

To be able to run the application make sure to install python 3.12.8 and install all the required dependencies mentioned in the requirements.txt file.

2 Starting cli

The application we have created can be started by running the main.py file using python in the terminal from the */src* folder. Once the program starts running, the user gets a confirmation message in the terminal as follows:

```
Startup complete

Welcome to Email Classifier application.
```

3 Test the system and its design patterns

The following lines are commands that can be run one after another to test the application, to test the used design patterns and to get an understanding of the system.

3.1 Create first email classifier

The user can interact with the cli using commands. When using the cli, the user will be provided with suggestions for arguments. To create and train an email classifier in the system, simply use the create_email_classifier command followed by all the needed arguments for the training data path, embedding type, model type and a name for the email classifier. The following is a simple command to create an email classifier:

```
>create_email_classifier AppGallery.csv tfidf randomforest EmailClassifier1
```

This line trains a randomforest model with the AppGallery dataset and the TF-IDF embeddings and gives this email classifier the name EmailClassifier1. The classification metrics are printed.

3.2 Add new input

The user can add new emails to classify with the following command including the path for the emails. Add_emails command utilises the facade pattern as this comment is ran in high-level interface.

```
> add_emails TestEmails.csv
```

3.3 Classify

The user can classify the emails previously added from the data source with the following command.

```
> classify_emails
```

3.4 Add preprocessing

The user can add different preprocessing steps. The Data preprocessor utilizes the decorator pattern. Add_preprocessing command utilizes the decorator and singleton patterns.

```
> add_preprocessing unicode_conversion
```

3.5 Change model

The user can change the model that will be used to classify the emails with the following command. Change_strategy command utilizes the strategy pattern.

```
> change_strategy bayes
```

3.6 Train the new model

After changing the model type or adding preprocessing, the user must retrain the model to utilize the new model.

```
> train_model AppGallery.csv
```

3.7 Create a new classifier

The user can create a new classifier by choosing <path>, <embedding>, <model> and a name.

```
> create_email_classifier Purchasing.csv sentence_transformer svc EmailClassifier2
```

3.8 List new emailclassifier

After creating a new classifier, one can see all the classifiers they have created.

```
> list_email_classifiers
```

3.9 Switch between emailClassifiers

The user can change between different email classifer object they have created.

```
> choose_email_classifier EmailClassifier1
```

3.10 Display the classification report for a classifier

The user can display the classification report of the classifier they have chosen. Display_evaluation command utilizes the observer pattern as when a new classifier is trained observer is notified.

```
> display_evaluation
```

3.11 Remove a classifier

The user can delete an email classifier object. After removing the classifier EmailClassifier1, when the user runs the display_evaluation command, the classification reports for EmailClassifier 2 will be printed.

```
> remove_email_classifier EmailClassifier1
```

3.12 Exit the application

The user can exit the application with the following command.

```
> exit
```

4 Commands

4.1 Help

```
>> -help
```

This will display information and help for the commands you can use for the cli

4.2 Create email classifier

```
>> create_email_classifier <path> <embedding> <model> <name>
```

This will initialize your email classifier with your given parameters and train the type of model you choose. It will also display the classification report.

4.3 Change strategy

```
>> change_strategy <model>
```

This command is used for changing the model of the email classifier. The model names are suggested by the command line. Options are: bayes, randomforest , svc. Once this command is run, the application returns a confirmation message: “Model changed to”. This command does not re-train the model, therefore the user needs to train the model by using another command.

4.4 Train model

```
>> train_model <path>
```

This command can be used the train a model based on the csv file specified as an argument. Once the model is trained, the classification report is displayed. The command trains a machine learning model based on what model was set previously.

4.5 Add emails

```
>> add_emails <path>
```

This command adds input data to be classified from the given path. The purpose of the command is for the user to add the emails they wish to classify to the system. This input data is different than the data that was used for training the model and creating classification report. Whenever this command is ran again, it replaces the previous emails that were added.

4.6 Classify emails

```
>> classify_emails
```

This command can only be used after adding new emails in order to classify these emails into different categories. This command returns the prediction for which class an email fits into and some part of the email that is classified.

4.7 Add preprocessing

```
>> add_preprocessing <feature>
```

With this command the user can add different preprocessing steps, that should be applied to the data before training. Available options are unicode_conversion, noise_removal, translation. The user can only add one preprocessing step at a time. After they have been added the train_model command has to be run again to retrain the model.

4.8 List email classifiers

```
>> list_email_classifiers
```

This command lists all available email classifiers that have been created.

4.9 Choose email classifier

```
>> choose_email_classifier <name>
```

```
This command lets the user choose the email classifier which the other commands will
perform actions upon.
```

4.10 Remove email classifier

```
>> remove_email_classifier <name>
```

This command lets the user remove an email classifier from the system by specifying its name.

4.11 Display evaluation

```
>> display_evaluation
```

This command displays the model evaluation of the currently selected email classifier. That are the metrics, that the model achieved, when training on the given data.

4.12 Undo

```
>> undo
```

This command revokes the last run command.

4.13 Exit

```
>> exit
```

This command is used for closing the application.

5 Arguments

5.1 Embeddings arguments

Table 1. <embeddings> argument that are used to choose embeddings.

Argument: `<embeddings>`	Description
tfidf	Use TF_IDF(term frequency–inverse document frequency) embeddings
sentence_transformer	Use Sentence transformer embeddings from the library sentenct-transformers
tfidf-sentence_transformer	Use the TF-IDF embeddings together with the Sentence Transformer embeddings

5.2 Model arguments

Table 2. <model> arguments that are used to change models.

Argument: `<model>`	Description
Randomforest	Use the Randomforest machine learning model from the scikit-learn library
Bayes	Use the Bayes machine learning model from the scikit-learn library
Svc	Use the SVC machine learning model from the scikit-learn library

5.3 Preprocessing arguments

Table 3. <feature> arguments that can be used to add preprocessing.

Argument: `<feature>`	Description
noise_removal	Preprocessing to remove noise in the data before creating embeddings.
Translation	Preprocess data to translate data before creating embeddings.
unicode_conversion	Preprocess data to convert data to Unicode characters before creating embeddings.

6 Add new training data and emails to classify

New training data or emails to classify can be dropped into the data folder of the application. The system will now recognize the new data and make it available to the user as an option in the terminal.

7 Extra configurations

In the event that extra configuration of the classification process is needed such as changing which columns are chosen for classification labels, the attributes of the *ClassifierSingletonConfig* can be changed directly in the code.