Email Classifier Implemented Using Xtreme Programming

Team: Wonder Girls

Siddhi Deepak Sawant (23181753), Wutyi Kyi Toe (23194286), Yixuan Wang (23162767)

Task 0: Brainstorm supervised and unsupervised learning

To develop an email classifier for our client, we could consider both supervised and unsupervised learning approaches. We will brainstorm both methods to compare the pros and cons.

Unsupervised learning

For unsupervised learning, training data only contains features without labels, so the training will be solely based on the email body.

Initially, we could consider clustering algorithms, which group emails based on certain similarities. Some popular ones are K-means clustering and Hierarchical clustering. With K-means, the number of classes (denoted as K) is fixed, while Hierarchical clustering has an unknown number of classes. For example, when classifying emails using K-means, if we set the clusters (K) to four, then emails are grouped into four classes. We can then extract the top terms per cluster to summarize the final labels. In our case, each email needs to be classified on multiple levels. Hence, we should train multiple clusters, but the classification outputs may not align with the labels our client provided. The classification results might be hard to interpret and unexpected; however, the algorithm may find some valuable and hidden connections in the emails.

We can also consider TF-IDF or topic modelling for feature extraction, gaining insights into top keywords or themes. Additionally, other unsupervised techniques can be applied, such as Principal Component Analysis (PCA) or autoencoders for dimensionality reduction. We could then cluster the extracted features or lower-dimensional representations achieved through these models to classify emails.

In general, the advantages of using unsupervised algorithms for email classifications can be summarized as follows. Firstly, datasets do not require labels, which are easier to obtain. Secondly, not requiring labelling reduces the workload, costs, and simplifies data preprocessing step. Thirdly, unsupervised learning may discover hidden relationships and unnoticeable but valuable patterns in the data. Fourthly, the algorithm can adapt to new datasets without requiring retraining.

On the other hand, there are several cons to consider. Firstly, the classification results cannot be predefined, which may not exactly meet our client's expectations if they require specific and hierarchical labels. Secondly, results may be hard to interpret and validate. Thirdly, results may not be accurate as there are no specific guidelines (labels) for training. Finally, the model may be hard to scale when the dataset is large.

Supervised learning

Supervised learning uses labeled dataset, where each email is associated with predefined categories. In our case, emails could be classified into different categories across multiple levels. The first level is the general category. The second one is service categories, such as “IT service”, “Financial issues”. The third level indicates specific aspects, which may include cloud autoscaling, or billing issues. Finally, the last level is detailed issues, for example, under-provisioning of HDD, or duplicate charges.

To choose a suitable classifier, we may consider logistic regression, which is a simple and easily understandable approach. Alternatively, we can use Naive Bayes, which is based on probabilities, accurate, fast, and widely adapted in text classifications. Random forests, which are robust, easy to interpret, and excel in handling large datasets, are another option. Support Vector Machines (SVMs), designed to find the best decision boundaries between classes, are effective for multi-class text classification tasks, but require more training time.

We can also choose deep learning models, such as BERT and Transformers for complex data. These models achieve state-of-the-art performance for text classification. They understand words based on context and can classify email content accurately. Many pre-trained models and fine-tuned versions for specific tasks are available. We could take advantage of training results from large corpora using transfer learning. On the other hand, these models are computationally expensive, memory-intensive, not as fast as simpler models, and hard to interpret.

In summary, there are several advantages to applying supervised learning. Firstly, they tend to achieve higher accuracy as there are explicit labels in the training data. Secondly, compared to unsupervised learning, they can create tailored solutions, in our case, classifying them based on predefined categories provided by customers. Thirdly, mature techniques for multi-level text classification algorithms are available, some already developed based on large corpora. Finally, regarding our scenario, data is already labelled, thus supervised learning can make full use of our data.

However, there are limitations as well. Initially, there is a risk of overfitting, if the dataset is not large enough or unrepresentative, especially when we select large models such as BERT; this should be addressed during model design. Secondly, models may become overcomplicated for some deep learning approaches and require significant computational power. Thirdly, to ensure performance, the model needs to integrate new datasets and be updated regularly.

After brainstorming both techniques, we select supervised learning. In our scenario, a high volume of data is available in the cloud. Initial data is available as shown in the sample, which facilitates model design. Established algorithms exist for classification tasks, and the data is already labelled. Moreover, these models achieve higher accuracy and provide task-specific classification outcomes, which suit our customer’s requirements. Although there are some challenges we need to tackle, such as addressing overfitting issues and high maintaining costs, the benefits outweigh drawbacks.

Task 1: Planning Game

The planning of development and release is implemented in **two primary sprints** each of **seven** and **ten days** with **three AI developers**. Third sprint can be considered for improvements, bug fixing or change request highlighted by the business user in UAT.

Sprint 1: 7days (one week)

An initial sprint (**Sprint 1**) of seven days (one week) with three AI developers, hence considering the estimated duration for implementing the activities. Activities considered for the initial sprint are as follows:

1. **Data selection** – considering only required features.
2. **Data grouping** – grouping tickets based on the conversation.
3. **Dealing with multiple languages** - involving translating into English.
4. **Dealing multilevel data -** the developers would brainstorm on multi-level modelling and user stories would be drafted and finalized and further would be considered for implementation.
5. **Textual data representation change** – converting text data to numeric data.
6. **Dealing imbalanced data** will be developed and unit testing will be performed on the same.

The above activities will be developed, and unit testing will be conducted and further based on the unit testing it would be deployed to UAT environment for user acceptance testing.

Reason for choosing the activities:

As data cleaning and having sound data with text data representation, addressing imbalanced data forms the base for applying machine learning models. With this business can get an essence of the progress made following the agile method and they can also share their feedback which will be implemented in second sprint along with the activities mapped in the second sprint.

Sprint 2: 10 days (about 1 and a half weeks)

The next sprint (**Sprint 2**) we would consider the sprint of ten days with three AI developers. Activities considered for the next sprint are as follows:

1. **Brainstorming of the machine learning model** to be chosen as per the requirement, supervised or unsupervised, among AI architects and developer further drafting and concluding the user stories for the same.
2. **Data preparation and modelling,**
3. **Implementing model such as random forest for classifying emails**
4. **Model training and testing**.

5. Additionally, this sprint would also include development of the **change request or enhancement** mentioned by the business from the previous release during **user acceptance** testing.

Unit and **integration** testing will be performed for all the activities as well as model will undergo load testing.

This sprint will run longer than the previous, including current sprint user stories and improvements for previous release and testing.

Reason for choosing the activities:

This release will provide business with an AI ticket/email classifier using machine learning model grouping the tickets as per their categorization.

S**print 3** can be considered as a small one **addressing the improvements from business.**

Task 2: Small release

We would release **Sprint 1** which was implemented in seven days to the business as an initial release in order that the **business stays informed about the progress** about development phase and can stay confident on the next release. Additionally, stay informed that the **enhancements, like if any suggestion were** given, will also be addressed in the **next release**. The release will cater to data cleaning and grouping of the emails and address the multilingual text converting them to one language. Also addressing the noise and unwanted phrases in the data. This small release will **store clean data which can be further used for analyzing, processing, an idea to decide on the model, training the model and testing the model.**

Task 3: Metaphor

Our client is a multinational company for IT-related services to which diverse customers are contacting for different services. Due to the numerous emails from international users as per daily basic, the customer service cannot engage well within the desirable service level agreement (SLA). As a result, it causes the users to be dissatisfied with the customer service team as well as company, which can impact the company's reputation. So, the company contacted us to develop a chatbot to classify emails in different levels. Our Project Manager (PM), Business Analyst (BA), and Technical Team Lead have had meetings and questionnaire sessions with the client to achieve functional and technical assessments for this project. A functional specification document has been produced by our BA, and there are three AI engineers assigned to develop for this chatbot project. Agile methodology with extreme programming is applied and version control "Git" is used. Three AI engineers are allocated specific tasks like data preprocessing, dealing with multiple languages,

choosing the machine learning algorithm to train the model with given data, evaluating and preparing test cases not only for unit test, load test and security test but also for user acceptance test. To oversee the progression of the project, our team creates a kanban project in Confluence (Atlassian), which is useful for development and also for DevOps, and then in the backlog, there are subtasks (issues) which are assigned to each AI engineer. Moreover, the ticketing cloud system (Jira) will also be used so that our client can raise service request or change request to us and these issues (tickets) from Jira will be reviewed by PM or Technical Team Lead and assigned to respective AI engineers or BA as per requirements. For the developing part, we arrange to make an initial release for those fundamental features and another release will be for the features which are not in initial release or add-ons as well. Before we deploy the fundamental release to the production environment, we will have an interim phase as deploying to environment, in which users will be able to test with different test cases and if any bug is found, it will be created in the Jira system and Jira system, and one of the AI engineers will be assigned to fix cases. Once the test cases are successful, the updated version will be released to the production environment, and our team will be on standby for production support for our client up to a specific period. At the same time, we will develop and prepare for add-on features which we have agreed to release as second phase. Moreover, if our client overlooked some features to explain or share with us at the requirement gathering stage but these features should be in this new chatbot, we will ask them to raise a change request and a quotation will be estimated by PM and sent to our client.

Task 4: Simple design

Use Case 1: Cleaning data

Description: Once the emails or ticket are received from the end user the system should remove the unwanted phrases such as “thank you”, “please”,” as soon as possible", etc. Additionally, null values from the tickets or emails.

Actors: AI Developer

Steps:

1. AI systems receive an email/ticket logged by the end users.

2. On receiving email/ticket system will trigger the algorithm developed for email classifier which will further trigger the data cleaning source code which will handle noise in the data and removal of unwanted phrases.

Use Case 2: **Handling Multiple Languages**

Description: The system receives emails in multiple languages, which needs to be translated into English for further processing.

Actors: AI Developer

Steps:

1. Receive emails: the system receives emails from multiple languages.
2. Language detection: the language in each email is detected.
3. Language standardization**:** If the language is in English, it proceeds to further processing; If other languages are detected, translation algorithms or services will be used to convert the content into English.
4. Analysis: the emails are then passed to classification algorithms.

Use Case 3: Processing large dataset

Description: As an AI system, it should be able to process and classify large email/tickets dataset, depending on the business requirement if the emails/tickets are to be classified on the scheduled basis. *(The use case is applicable in two scenarios 1. If the business needs to implement the email classifier on a scheduled basis.2. If the business users need to classify the already existing tickets.)*

Actors: AI Developer

Steps:

1. On periodic time intervals, the email/ticket classifier would be scheduled to run every day.
2. The scheduler would fetch the records from the local database and run the email classifier algorithm to classify the emails as 'complaint' for level 1, 'IT service' for level 2, 'cloud autoscaling' for level 3, and 'under-provisioning of HDD' for level 4.
3. Updating the levels in the database.

Use Case 4: User Acceptance Testing

Description: User acceptance tests will be performed by end-users of our client who will test with different scenarios as a customer support team member or as customers.

Actor: End-users

Steps:

1. The customer will login to the system to raise a support case or make enquiry/feedback. In the submission form, the validation will be tested out for limiting word counts, checking the uploading document or image size, or whether the mandatory information is entered or not.
2. The email replied from AI chatbot has been received or not will be validated as a customer.
3. After logging in as support team members, they will check if the logged cases are correctly categorized into different types, the translated languages are correct or not. For service

level agreement (SLA) is setup in each case, support members will check if the SLA duration counter is working correctly or not after the support cases are solved.

Task 5: Testing

There are three unit-testing cases and one load testing. The unitest library and locust (the testing framework) are utilized.

Unit testing - There are three test cases. Each test case will be performed separately during development. But for coding collaboration, I created a class and consolidated all unit test functions.

1. For making sure the scaling of the data is done properly - this test case will be performed after the provided data is preprocessed.
2. For modeling function - this test case will be carried out after the model has been trained to ensure the model is behaving as expected.
3. For model integration - this test will be performed to validate if the output (multiple classification) of the prediction of the trained model is in the range of possible class labels or not.

Load testing – This test will be performed for the trained model with newly generated synthetic data by using locust testing framework. After running it, the log will be generated.

The source codes are consolidated in the Jupyter notebook.

Task 6: Refactoring & Task 7. Pair-programming

Consolidated code and test cases are presented in GitHub.

Task 8: Collective ownership

Siddhi Deepak Sawant (23181753) – Task 1 (Planning Game), Task 2 (Small release), Task 4 (Use Case 1 and 3)

Yixuan Wang (23162767) – Task 0 (Brainstorm supervised and unsupervised learning), Task 4 (Use Case 2), Task 6 (Refactoring)

Wutyi Kyi Toe (23194286) – Task 3 (Metaphor), Task 4 (Use Case 4), Task 5 (Testing)

Task 9: Integration – Github link for consolidated source codes