ក្រសួងអប់រំ យុវជន និងកីឡា

វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យព័ត៌មានវិទ្យា និងទំនាក់ទំនង

គម្រោងសញ្ញាបត្រវិស្វករ

ប្រធានបទ : **បំណែងចែកសារអេឡិចត្រូនិចដោយស្វ័យប្រវត្តដោយប្រើប្រព័ន្ធបញ្ញាសិប្បនិមិត្ត**

និស្សិត : **កៅ វិសាល**

ឯកទេស : **វិស្វកម្មព័ត៌មានវិទ្យា និងទំនាក់ទំនង**

គ្រូទទួលបន្ទុក : **លោក សុខ គឹមហេង**

ឆ្នាំសិក្សា : **២០២៤-២០២៥**

MINISTERE DE L'EDUCATION,
DE LA JEUNESSE ET DES SPORTS

INSTITUT DE TECHNOLOGIE DU CAMBODGE

DEPARTEMENT DE GENIE INFORMATIQUE ET COMMUNICATION

MEMOIRE DE FIN D'ETUDES

Titre : **AI-Powered Email Sorting: Leveraging Text and Visual Cues for Intelligent Triage**

Etudiant : **KAO Visal**

Spécialité : **Génie Informatique et Communication**

Tuteur de stage : **Mr. SOK Kimheng**

Année scolaire : **2024-2025**

ក្រសួងអប់រំ យុវជន និងកីឡា

វិទ្យាស្ថានបច្ចេកវិទ្យាកម្ពុជា

ដេប៉ាតឺម៉ង់ ទេពកោសល្យព័ត៌មានវិទ្យា និងទំនាក់ទំនង

គម្រោងសញ្ញាបត្រវិស្វករ

របស់និស្សិត: កៅ វិសាល

កាលបរិច្ឆេទការពារនិក្ខេបបទ: ថ្ងៃទី ៨ ខែ កញ្ញា ឆ្នាំ ២០២៥

អនុញ្ញាតឲ្យការពារគម្រោង

នាយកវិទ្យាស្ថាន:_____

ថ្ងៃទី ខែ ឆ្នាំ ២០២៥

ប្រធានបទ : បំណែងចែកសារអេឡិចត្រូនិចដោយស្វ័យប្រវត្តិ ដោយប្រើប្រព័ន្ធបញ្ជាសិប្បនិមិត្ត

សហគ្រាស : ក្រុម សេ អេស ឌុបលីវ៉ែ

ប្រធានដេប៉ាតឺម៉ង់ : លោក ឡៃ ហេង _____

សាស្ត្រាចារ្យជឹកនាំគម្រោង : លោក សុខ គឹមហេង_____

អ្នកទទួលខុសត្រូវក្នុងសហគ្រាស : លោក ស៊ីកូ ហ្យ៉ូម _____

រាជធានីភ្នំពេញ ឆ្នាំ២០២៥

MINISTERE DE L'EDUCATION,
DE LA JEUNESSE ET DES SPORTS

INSTITUT DE TECHNOLOGIE DU CAMBODGE
DEPARTEMENT DE GENIE INFORMATIQUE ET COMMUNICATION

MEMOIRE DE FIN D'ETUDES

DE M. KAO Visal

Date de soutenance : le 8 septembre 2025

« Autorise la soutenance du mémoire »

Directeur de l'Institut : _____

Phnom Penh, le 8 septembre 2025

Titre : **AI-Powered Email Sorting: Leveraging Text and Visual Cues for Intelligent Triage**

Etablissement du stage : Groupe CSW

Chef du département : **M. LAY Heng** _____

Tuteur de stage : **M. SOK Kimheng** _____

Responsable de l'établissement : M. Jérôme SICO _____

PHNOM PENH, 2025

សេចក្តីសង្ខេប

ការបំលែងកិច្ចការច្រំដែលអោយទៅជាការងារអូតូម៉ាទិក ជាការប្រើប្រាស់បច្ចេកវិទ្យាបញ្ញាសិប្បនិម្មិត (AI) ដែលគេជួបប្រទះច្រើនបំផុតក្នុងសម័យបច្ចុប្បន្ន។ ក្រុមហ៊ុន និងអង្គភាពជាច្រើន ទាំងក្នុងស្រុក និងអន្តរជាតិ កំពុងអនុវត្តន៍និងប្រើប្រាស់បញ្ញាសិប្បនិម្មិតដើម្បីដកស្រង់ព័ត៌មានពីឯកសារ និងធ្វើទំនាក់ទំនងជាមួយអតិថិជនដោយស្វ័យប្រវត្តិ។ វិស័យមួយក្នុងចំណោមវិស័យដែលមានសក្តានុពលខ្ពស់ក្នុងការដាក់បញ្ចូល AI គឺប្រព័ន្ធបញ្ជូនអ៊ីមែលដោយស្វ័យប្រវត្តិ។ នៅក្នុងអង្គភាពមួយចំនួន ប្រអប់សាររួម (shared mailbox) ត្រូវបានប្រើប្រាស់ដោយបុគ្គលិកច្រើននាក់ ដើម្បីធ្វើទំនាក់ទំនងជាមួយភាគីក្រៅក្រុមហ៊ុន ឬក្រៅអង្គភាព ដើម្បីរក្សានូវសណ្តាប់ធ្នាប់ក្នុងក្រុមហ៊ុន។ ប្រអប់សាររួមនេះ ត្រូវបានគ្រប់គ្រងដោយបុគ្គលិកមួយរឺច្រើន ដែលទទួលខុសត្រូវក្នុងការអាន វិភាគ និងចែកអ៊ីមែលដែលបានផ្ញើចូលមក អោយទៅអ្នកដែលត្រូវទទួលរ៉ាប់រងឆ្លើយ និងទទួលសារនោះ។

គោលបំណងនៃនិក្ខេបបទនេះ គឺដើម្បីស្រាវជ្រាវ ពិសោធន៍ និងដាក់អោយដំណើរការ នូវសូហ្វវែ (software) មួយដែលអាច អានយល់ពីអត្ថន័យរបស់អ៊ីមែល ពីវត្ថុបំណងរបស់អ្នកផ្ញើ និងអាចចែកចាយអ៊ីមែលនោះទៅឱ្យបុគ្គលិកណាមួយ ក្នុងក្រុមហ៊ុន ដែលអាចឆ្លើយនូវតម្រូវការរបស់ភ្ញៀវបាន។ ការងារនេះត្រូវបានអនុវត្តនៅក្រុមហ៊ុន Groupe CSW (Cosywee) ដោយប្រើបច្ចេកវិទ្យាប្រកបដោយបញ្ញាសិប្បនិម្មិត ដែលរួមមាន Machine Learning និង ម៉ូឌែល RoBERTa បន្ថែមជាមួយ Computer Vision Methods ដែរ។ ម៉ូដែល RoBERTa (CamemBERT) ត្រូវបានជ្រើសរើសមកប្រើប្រាស់ក្នុងគម្រោងនេះ ព្រោះវាត្រូវបានបណ្តុះបណ្តាល (Train) ដោយទិន្នន័យច្រើនជាភាសាបារាំង រួមទាំងarchitecture របស់វា ប្រើនូវ self-attention រួមទាំងមុខងារ Bidirectional ដែលអាចចាប់យកបាននូវន័យរបស់ឃ្លា ឬអត្ថបទបានទាំងមូល។

ដំណើរការបណ្តុះបណ្តាលម៉ូឌែល បានប្រើប្រាស់បណ្ណាល័យ និងឧបករណ៍ជាច្រើន ដើម្បីធានាឱ្យមានប្រសិទ្ធភាព និងមានភាពងាយស្រួល។ Weights & Biases (wandb) ត្រូវបានប្រើសម្រាប់តាមដានការសាកល្បង កំណត់ និងប្រៀបធៀប hyperparameter របស់ម៉ូឌែលនីមួយៗ។ HuggingFace ត្រូវបានប្រើសម្រាប់ទាញយកម៉ូឌែលដែលបានបណ្តុះបណ្តាលរួច និងធ្វើការប្រៀបធៀបតាម benchmark។ បច្ចេកវិទ្យាទាំងនេះ ត្រូវបានជ្រើសរើសពីព្រោះមានសមត្ថភាពធ្វើមាត្រដ្ឋានបាន (scalability) មានភាពបត់បែនខ្ពស់ និងងាយស្រួលក្នុងការគ្រប់គ្រងម៉ូឌែល។ នៅចុងបញ្ចប់នៃគម្រោង បណ្តាញកម្មវិធីត្រូវបានអនុវត្តន៍ និងសាកល្បងក្នុងបរិបទពិតប្រាកដ ដោយប្រើកុំព្យូទ័រដែលមានកម្លាំងគណនោតិចបំផុត។

លទ្ធផលនៃការស្រាវជ្រាវនេះ បានជួយឱ្យពេលវេលាដែលត្រូវប្រើសម្រាប់ដំណើរការដោយដៃថយចុះជាង 90% ដែលបានជួយកាត់បន្ថយចំណាយ និងបង្កើនប្រសិទ្ធភាពនៅកន្លែងធ្វើការ។

Résumé

L'automatisation des tâches répétitives est l'une des applications les plus courantes de l'intelligence artificielle aujourd'hui. Les entreprises, qu'elles soient locales ou internationales, adoptent et utilisent de plus en plus l'intelligence artificielle pour traiter automatiquement des documents physiques, extraire des informations et interagir avec les clients. L'un des domaines prometteurs de l'intégration de l'IA est le routage automatique des e- mails. Dans certaines organisations, les boîtes mail partagées sont généralement gérées par une ou plusieurs personnes chargées de lire, d'analyser et de classer manuellement les e-mails afin de les transférer aux destinataires appropriés.

Le processus d'entraînement s'est appuyé sur plusieurs bibliothèques et outils afin d'assurer efficacité et reproductibilité. Weights & Biases (wandb) a été utilisé pour suivre les expéri- ences, gérer les hyperparamètres et comparer les performances des différents modèles. Hug- gingFace a permis d'accéder à des modèles pré-entraînés et d'évaluer des benchmarks. Ces technologies ont été choisies pour leur scalabilité, leur flexibilité, et leur facilité d'intégration pour la gestion des modèles. À la fin du projet, la solution a été déployée et testée dans un environnement réel sur une machine disposant de ressources minimales.

L'objectif de ce rapport est de présenter le travail réalisé au sein du Groupe CSW (Cosy- wee) pour automatiser cette tâche, en utilisant une combinaison d'algorithmes tradition- nels d'apprentissage automatique et de modèles basés sur RoBERTa, intégrés à des tech- niques de détection d'indices visuels sur les images. Le modèle CamemBERT (basé sur l'architecture RoBERTa) a été choisi pour son encodeur Transformer avec mécanisme d'attention, qui permet au modèle de comprendre le contexte des mots, quelle que soit leur distance. De plus, CamemBERT a été pré-entraîné sur un large corpus en français, ce qui le rend particulièrement adapté à ce projet.

Cela a permis de réduire de près de 90 % le temps de traitement manuel, réduisant ainsi considérablement les coûts et améliorant l'efficacité au travail.

Abstract

Automating repetitive tasks is one of the most common applications of Artificial Intelligence today. Companies, both local and international, have been increasingly adopting and using Artificial Intelligence to automatically process physical documents, extract information, and interact with clients. One promising area of AI integration is Automatic Email Routing systems. In some organizations, shared mailboxes are usually managed and controlled by one or more people who are responsible for manually reading, analyzing, and classifying those emails to forward them to appropriate recipients.

The objective of this report is to present the work carried out at Groupe CSW (Cosywee) to automate this task, using a combination of traditional machine learning algorithms and RoBERTa-based models, integrated with visual clue detection on images. The CamemBERT model (based on the RoBERTa architecture) was chosen due to its Transformer encoder with self-attention. This enables the model to capture the context of words regardless of their distance. Additionally, CamemBERT was pre-trained on a large French corpus, making it particularly well-suited for this project. The training process leveraged several libraries and tools to ensure efficiency and repro- ducibility. Weights & Biases (wandb) was used to monitor experiments, track hyperparam- eters, and compare the performance of different models. HuggingFace was employed to access pre-trained models and evaluate benchmarks. These technologies were chosen for their scalability, flexibility, and ease of integration for model management. At the end of the project, the solution was deployed and tested in a real-world environment on a machine with minimal resources.

This resulted in a reduction of almost 90 % in manual processing time, significantly cutting costs and improving workplace efficiency.

ACKNOWLEDGEMENTS

The success of this project is not just the result of my own effort, but a shared journey made possible by the support, encouragement, and kindness of many people. Before diving into the work itself, I want to take a moment to express my deep gratitude to those who have helped me and are still helping me along the way.

First, I wish to extend my sincere appreciation to **H.E. Dr. PO Kimtho**, Director of the Institute of Technology of Cambodia. His excellent leadership and the strong cooperation with partner universities both locally and internationally have significantly contributed to improving the quality of education for engineers in our country, and that cooperation is the reason why I get a chance to pursue my double degree in France as well.

I am also deeply appreciative of **Mr. LAY Heng**, head of GIC department, for his valuable support and guidance during my time as a student in his department.

I would like to acknowledge the faculty and administrative staff of the GIC department (Génie Information et Communication) at Institute of Technology of Cambodia for fostering an engaging and intellectually rich environment. I am especially thankful to **Mr. SOK Kimheng**, my academic supervisor, for his kindness and ongoing assistance throughout this internship. I am grateful to **Dr. VALY Donna**, the head of ViLa Research Lab, for introducing me into the field of Artificial Intelligence, providing me with materials and giving me a chance to set my foot in the gate of this domain.

I want to thank **Mr. Jérôme SICO**, my supervisor during this end-of-study project (Projet de Fin d'Études) at Cosywee, for his trust, guidance, and constant support. His sharp insights, thoughtful feedback, and professional approach helped shape this thesis and pushed me to grow both technically and personally.

Above all, I owe the deepest gratitude to my family, including my parents, my aunt, my siblings, and my partner, for their unconditional love, sacrifices, and motivation. Their belief in me has been the foundation of every success I've achieved.

ABBREVIATION LIST

AI : Artificial Intelligence

DICE : Department of Information and Communication Engineering

BDC : Bon de Commande (Purchase Order)

ITC : Institute of Technology Cambodia

CSW : **Groupe CSW (Cosywee)**

BERT : Bidirectional Encoder Representations from Transformers model

ML : Machine Learning

NLP : Nature Language Processing

TF-IDF : Term Frequency - Inverse Document Frequency

RoBERTa : A Robustly Optimized BERT Pretraining Approach

CamemBERT: RoBERTa model pre-trained on French corpus

MP : Marseille-Perpignan (shortcut for a label in classification model)

OCR : Optical Character Recognition

TABLE OF CONTENTS

Résumé 4

Abstract 5

ACKNOWLEDGEMENTS 6

ABBREVIATION LIST 7

TABLE OF CONTENTS 8

LIST OF FIGURES 11

LIST OF TABLES 12

I - INTRODUCTION 13

1.1 General Presentation 13

1.1.1 Company Presentation 13

1.1.2 Company and Supervisor contact information 13

1.2 Project Presentation 13

1.2.1 Problem Statement 13

1.2.2 Mission Objective 14

1.2.3 Types of Incoming Emails 15

1.2.4 Trade-offs 16

1.3 Project Timeline 16

II - RELATED WORK 18

III - TECHNOLOGIES 19

3.1 Tools and Technologies 19

3.1.1 Python 19

3.1.2 Anaconda 19

3.1.3 Sourcetree 19

3.1.4. Bitbucket 20

3.1.5. Jira 20

3.1.6. Visual Studio Code 20

3.1.7. Transformers 21

3.1.8. Spacy 21

3.1.9. OpenCV 21

3.1.10. Pytesseract 21

3.1.11. EasyOCR 21

3.1.12. nltk 22

3.1.13. Microsoft Graph API 22

3.1.14. HuggingFace 22
3.1.15. Weights and Bias (WandB) 22
3.2. Coding Guideline 23
IV - MODEL DEVELOPMENT 24
4.1. Data Collection 24
4.1.1. Problem 24
4.1.2. Solution 26
4.2. Data Cleaning 26
4.2.1. Email body cleaning 26
4.2.2. Attachment cleaning 27
4.2.3. Final Data 30
4.3. Model Experimentation 30
4.4. Evaluation Metrics 34
Confusion Matrix and Terminology 35
Evaluation Methods 35
4.5. Results 36
4.5.1. Intent classification model 36
4.5.2. Alès, Lyon, MP classification 37
4.5.3. Marseille, Perpignan, Roller Shutter *(VR)* classification 39
4.6. Final Pipeline 41
V - IMPLEMENTATION & PROTOTYPE BUILDING 43
5.1. Project Development 43
5.1.1. Environment configuration 43
5.1.2. Project Structure 44
5.2. Heuristics methods 45
5.2.1. Original Sender of the email 45
5.2.2. Document Creator 45
5.2.3. Product extraction from Purchase Order 48
5.2.4. Intent extraction from Purchase Order 50
5.3. Helper Functions with Microsoft Graph API 52
5.3.1. Read Emails 53
5.3.2. Get Folder ID 54
5.3.3. Move Email 54
5.4. Final Classification Pipeline Implementation 55
5.5. Optimization and Improvement 58
5.5.1. Version v1.0 58

5.5.2. Version v2.0 59
VI - RESULTS AND DISCUSSIONS 61
6.1. Results 61
6.2. Future improvements 63
6.2.1. Optimizing handwriting text recognition model 63
6.2.2. Integration of human feedback loop 63
VII - CONCLUSION 65
References 66
APPENDIX 69
Types of Emails 69
Log files 71

LIST OF FIGURES

Figure 1 - Email Flows Within The Service.Clients Mailbox 15
Figure 2 - Project Timeline 16
Figure 3 – Log Files 23
Figure 4 - Dataset 30
Figure 5 - Overview Of Model Pipeline 31
Figure 6 - Confusion Matrix Of Intents Classification 37
Figure 7 - Fully Integrated Ai Pipeline. 41
Figure 8 - Project Structure 44
Figure 9 - Employee's Name To Company 46
Figure 10 - Employee's Name Extraction From Purchase Order Flowchart 47
Figure 11 - Purchase Order For Mosquito Net 48
Figure 12 - Matching Product's Name/Code To Company 49
Figure 13 - Product Name Extraction From Purchase Order Flowchart 50
Figure 14 - Orb Features Matching 51
Figure 15 - Orientation Corrected Purchase 52
Figure 16 - Works Completed Flowchart 64
Figure 17 - Emails With No Attachments, Containing Only Plain Text Order Or Price Quotation Requests 69
Figure 18 - Emails With Handwritten Attachments Indicating Order Details 69
Figure 19 - Emails With Computer-Generated Attachments (Printed Text) Indicating Order Details 69
Figure 20 - Emails Replying To A Cosywee Employee's Email 70
Figure 21 - Emails Inquiring About Order Progress (With Only A Reference Number And No Product Information) 70
Figure 22 - Order Emails With Cosywee's Purchase Order Form (Bon De Commande) Attached 70
Figure 23 - Emails Containing Cosywee-Generated Documents As References 70
Figure 24 - Time Records Log: Logs The Displacement Or Movement Of Emails, Which Is Useful For Debugging Cases Where Emails Appear To Be Missing. 71
Figure 25 - Prediction Log: Stores Predictions And Confidence Scores From Both The Camembert Model And The Random Forest Model.) 71
Figure 26 - Complete Data Log: Stores All Extracted Data Along With Their Predictions. This Log Is Useful For Fine-Tuning New Models In The Future. 71

LIST OF TABLES

Table 1 - Comparison Of Email Processing Libraries 25
Table 2 - Performance Of Intents Classification Model 37
Table 3 - Confusion Matrix Of Ales, Lyon, Mp Classification 38
Table 4 - Performance Of Alès, Lyon, Mp Classification Model 38
Table 5 - Performance Comparison Of Algorithms - Alès, Lyon, Mp Classification 39
Table 6 - Confusion Matrix Of Marseille, Perpignan, Roller Shutter Classification 40
Table 7 - Performance Of Marseille, Perpignan, Roller Shutter Classification Model 40
Table 8 - Performance Comparison Of Algorithms - Marseille, Perpignan, Roller Shutter Classification 40
Table 10 - Processing Time Vs Computing Power On A High Computing Power Machine 58
Table 11 - Processing Time Vs Computing Power On A Low Computing Power Machine 59
Table 12 - Processing Time Vs Computing Power Of Version V2.0 On A Low Computing Power Machine 60
Table 13 - Ai System Capability (With 500 Emails Sorted) 62

I - INTRODUCTION

1.1 General Presentation

1.1.1 Company Presentation

The CSW Group (Cosywee), specialized in closure and solar protection products, has been building and maintaining a relationship of trust with professionals since 2003. The CSW Group comprises several companies at the south of France including Cosywee Alès, Cosywee Perpignan, Cosywee Nimes, Cosywee Lyon, Cosywee Marseille, and Cosywee Bourg-en-Bresse. Each of these entities contributes to the Group's specialization in closure and solar protection products, ensuring a constant level of expertise and service in the different regions.

With a nationwide presence, the CSW Group has developed over the years on the basis of top-of-the-range products assembled in France. The group's 130 employees work at 6 production sites and 1 design office located near Perpignan, Nîmes, Marseille and Lyon.

1.1.2 Company and Supervisor contact information

COSYWEE Alès (bureau de l'équipe informatique) is at Alès 30100, France :

- Google Map: https://maps.app.goo.gl/Uiary9pKzYfv5bJLA
- Adresse : 22b Avenue du général De Gaulle, 30100 ALES
- Tel : +33 4 48 40 99 89 • Email : jerome.sico@cosywee.com
- Website: https://www.cosywee.com

1.2 Project Presentation

1.2.1 Problem Statement

Within the company, there is a shared mailbox where all employees in Service Clients Department communicate with clients answering product inquiry, customer survey, account openning (New clients registration), price quotation, orders, after-sales services, order confirmation, payments, accounting related problems, etc. As all employees shared the same mailbox, all emails sent and received flow into one email address, "service.clients@cosywee". To make things even more complicated, each product has a unique code, a different Bon de Commande (Purchase order) template, and is produced by different company (factory) of the Group.

Usually, there is a group of employees, at each company of the group (or each factory), who are responsible for reading emails, including attachments, responding to them and manually

forwarding those emails to the corresponding people or factory. As there are more and more mails incoming, it can easily get out of hand and can be one of the problems for the productivity and efficiency of the company.

In order to solve this problem, there was an attempt to use keyword matching to parse the email's body and subject to redirect those emails, but as the patterns are too complicated, with too many categories and purpose involves, and missing information from attachments, this option is out of consideration.

1.2.2 Mission Objective

The main objective of this internship is to implement artificial intelligence to improve productivity in the company. My scope (responsibility) includes problem definition, requirement gathering, data collection, finetuning a pre-trained model or training one from scratch, up to model implementation, pipeline building, and model/program hosting as well.

Basically, this main objective is to build an end-to-end system improving efficiency, reducing and to eliminate manual, repetitive tasks in the company. In this case, the aim of the internship to build an end-to-end program to sort or redirect incoming email from Service Clients mailbox to appropriate recipients, within a reasonable delay and high accuracy to reduce manual tasks. This includes problem formulation, requirement gathering, data collection, data cleaning, model experimentation, complete program pipeline, and hosting. This might include data extraction, data scraping, API integration, intention classification, Optical Character Recognition (OCR), keywords matching, etc. In the diagram below (Figure 1), you can see the flow of emails in the Service.Clients mailbox. When a client sends an email to the Service.Clients address, employee from Client Service department reads the message, analyzes its intent, checks whether any products are 3 mentioned, identifies which company or factory is responsible for the product, and then re-routes the email to the appropriate recipients.

Furthermore, since all emails sent by employees from the Service Clients Department use the "service.clients@cosywee.com" address, any replies to those emails must also be read and re-routed to the original sender.

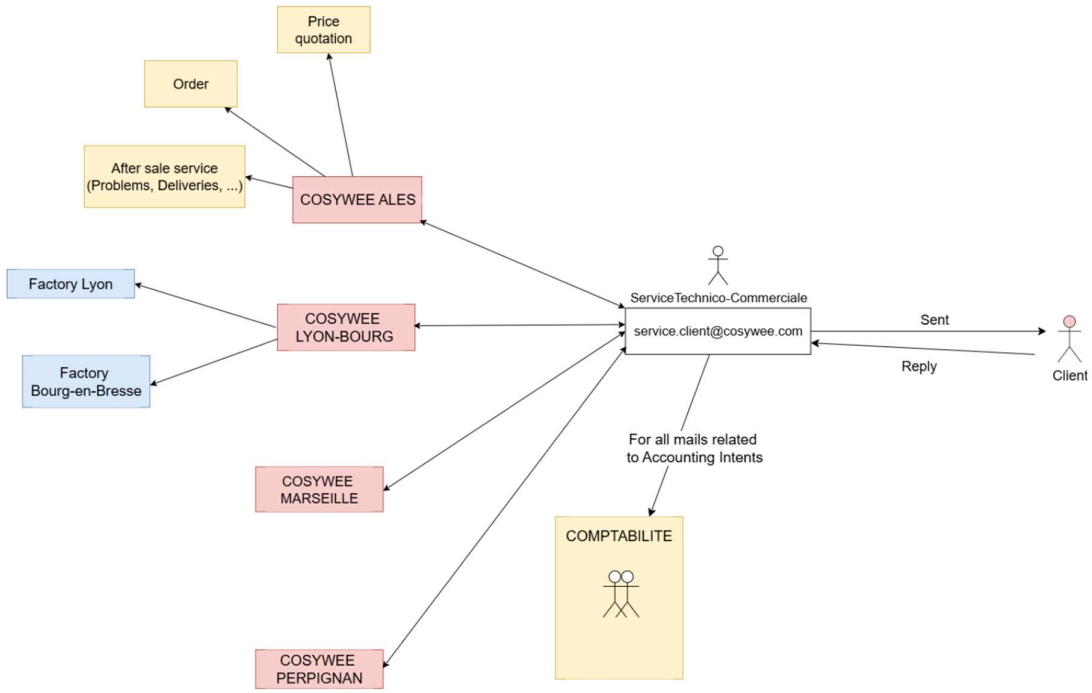


Figure 1 - Email flows within the service.clients mailbox

Each email can contain multiple types of attachments, including images, PDF files, .txt files, .csv files, .xlsb files, .docx files, etc. These files come in different formats, related to different products and serving unique purposes.

1.2.3 Types of Incoming Emails

There are many types of emails that arrive in service.client@cosywee.com mailbox.

Below are some common types of emails that we would get daily:

- Emails with no attachments, containing only plain text order or price quotation requests
- Emails with handwritten attachments indicating order details
- Emails with computer-generated attachments (printed text) indicating order details
- Emails replying to a Cosywee employee's email
- Emails inquiring about order progress (with only a reference number and no product information)
- Order emails with Cosywee's Purchase Order form (Bon de Commande) attached
- Emails containing Cosywee-generated documents as references
- Emails announcing unrelated information (vacation dates, congratulations, etc.)

You can find examples of each type of email in Types of Emails.

1.2.4 Trade-offs

In this section, I will talk about the constraints, problems, and trade-offs considered prior to proposing a solution.

First of all, **COSYWEE MARSEILLE** and **COSYWEE PERPIGNAN** share overlapping product lines. These products include all items related to **Roller shutters (Volet Roulant)** (Traditional Roller shutters, Bloc-Blae roller shutters, Roller Shutter Renovation, Solar Roller Shutters, Special Roller Shutters, etc.). Due to this overlap, all products related to Roller shutters must be classified manually by a human (excluded from the classification pipeline).

Secondly, the processing time of the program must be minimized as it must deal with multiple incoming emails in one second. In this sense, the company is willing to accept a small trade-off in accuracy in favor of the program's processing time. Similarly, the computational resources required by the program must be kept as low as possible, since it will be hosted on a shared company server that also supports other critical applications.

Thirdly, the company prioritizes explainability over small gains in accuracy. This indicates that if two algorithms perform quite similarly in terms of accuracy and precision, we tend to choose the one with more explainability regardless of its performance.

Finally, security is a critical pillar of the system. Since the corporate mailbox contains confidential business information, it represents one of the most sensitive components of the company’s infrastructure. As such, security protocols must be implemented with the highest priority and rigor.

1.3 Project Timeline

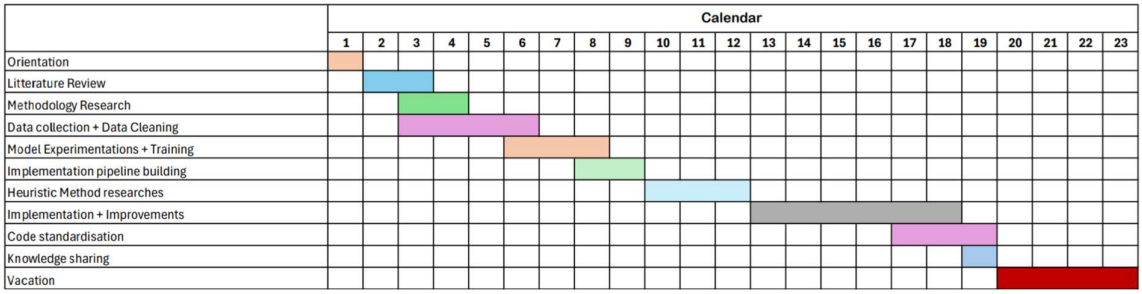


Figure 2 - Project Timeline

The project started in early March (week 1 in Figure 2) when I was introduced to the problem. I then spent some time on literature review and researching potential methodologies. After that, I began collecting data, finding ways to extract and clean it, spending a significant amount of time on the OCR part, as it provides the most critical information from attachments.

Next, I experimented with different machine learning algorithms and deep learning architecture before starting to build the pipeline and API endpoints needed to read and move emails for implementing the AI system. I spent about two weeks setting up a secure API inference using Microsoft Graph API and configuring the API to access emails in a shared mailbox.

Note that, to implement this application, we had to create a new rule, a new user, and a dedicated account for the entire automation system. After that, I focused on developing heuristic methods to improve the system's accuracy and efficiency before deploying it to production in the 13th week. From that point onward, I began releasing new versions of the program every few days, addressing reported bugs and adding new features.

And since the 17th week, I started to standardize my code and write detailed reports on how to fine-tune the model, how to retrain it from scratch, and how the company can maintain and scale the program in the future.

It is worth noting that the project was completed by week 19. Between weeks 20 and 23, Groupe CSW, as well as many other companies in France, observed the annual summer holiday break.

II - RELATED WORK

Email classification and text classification have long been active research areas in the field of Natural Language Processing (NLP). Previous research and published works in these areas have laid a strong foundation for building a good, reliable and robust multi-modal systems.

Historically, email classification, or text classification in general, relied on traditional machine learning methods like Naive Bayes and Random Forest, while leveraging features selection techniques such as TF-IDF or bag-of-words vectors, etc. These ancient approaches are considered efficient and interpretable but they all assume words independence, making them less suitable for real-world tasks like semantic or intent classification.

Newer methods introduced a few years back include transformer-based language models like BERT (Al., 2018) and its variances. One well-known variance for French Natural Language Processing is CamemBERT (Martin, et al., 2020), a RoBERTa-based (Liu, et al., 2019) transformer that was pretrained on a large French corpus by Facebook researchers and researchers from Sorbonne University. It has shown a promising context-awareness ability and is often used as a benchmark, a state-of-the-art model for French language across multiple tasks including classification, named entity recognition, and semantic similarity, etc.

In this project, a combination of Camembert and Random Forest were used to make predictions regarding products and entities, to reduce bias and prevent overconfident outputs. Additionally, the CamemBERT model alone is used to predict the intent of emails as well.

III - TECHNOLOGIES

3.1 Tools and Technologies

In the course of this project, a variety of tools, libraries, and platforms were used to ensure efficient development, collaboration, and deployment. Below is an overview of the key technologies:

3.1.1 Python

Python is the first programming language choice for Artificial Intellgence tasks thanks to its simplicity and feasibility to implement. One of the main advantages of using Python in this project is its diverse ecosystem and libraries available such as torch (Pytorch), Tensorflow and Scikit-learn, etc. These tools have many features and functionalities that can facilitate the process of data processing, cleaning, model experimentation and deployment as well.

3.1.2 Anaconda

Anaconda is a platform of virtual environments and packages management that is mostly used in data science and AI development. In this project, I used Anaconda to manage different environments (Testing environment, development environment, and deployment environment) since I had to test different models with diverse dependency, which would cause a problem if I only used global environment.

3.1.3 Sourcetree

Sourcetree is a Git client that allows simplification of version control process for all software development projects. It provides engineers and developers with a graphic interface that

allows us to effectively review visually of changes we made on code and to manage other git manipulation operations as well.

3.1.4. Bitbucket

Bitbucket is a web platform to host git projects, essentially for version control and collaboration in a software development team. In this project, Bitbucket is used to manage the code repository and version control.

3.1.5. Jira

Jira, developed by Atlassian, is a tool of project management used to follow tasks and workflows, mostly in the software development life cycle (Development, improvement, bug fixing, maintenance, etc). In this project, Jira is used to measure the completion of each step, each task. Jira facilitated task management and progress monitoring, ensuring efficient project coordination and on-time delivery.

3.1.6. Visual Studio Code

Visual Studio Code, developed by Microsoft, is one of the most popular code editors used for software development today. It offers many functionalities such as syntax highlighting, debugging, version control integration, and more.

In this project, Visual Studio code is used to write and test code, to run and to debug the project as well.

3.1.7. Transformers

This library was essential for using and fine-tuning state-of-the-art pre-trained language models for NLP tasks such as text classification and sentiment analysis.

3.1.8. Spacy

Spacy (spaCy French Language Model: fr_core_news_lg, 2024) is utilized for fast and efficient natural language processing, especially for tasks such as noise removal, named entity recognition, and part-of-speech tagging in this project.

3.1.9. OpenCV

OpenCV is an image-processing-focused library, utilized for image pre-processing and manipulation in this project (particularly in document layout analysis tasks).

3.1.10. Pytesseract

Pytesseract (pytesseract: Python wrapper for Google Tesseract-OCR, 2024) is a Python wrapper for Google's Tesseract-OCR Engine, widely used for Optical Character Recognition (OCR) mostly on printed text. In this project, it is leveraged for text detection and extraction from scanned documents.

3.1.11. EasyOCR

EasyOCR (JaidedAI, 2024) is a Python-based OCR library available on Github that supports many languages and is also known for handling both printed and handwritten text. Just like Pytesseract, in this project, EasyOCR is primarily used for text extraction (both handwritten and printed text) from scanned documents and images.

3.1.12. nltk

Natural Language Toolkit (NLTK) is a leading library in Python, mainly used for Natural Language Processing tasks. It provides tools for text processing, stopword removal, and more

3.1.13. Microsoft Graph API

Microsoft Graph API is a unified REST API that allows us, developers, to access Microsoft 365 as well as its services like Outlook, OneDrive in a programmatic way.

In this project, Microsoft Graph API is used to connect from the developed application to the company's Outlook inbox to fetch and read emails to classify.

3.1.14. HuggingFace

Hugging Face is a platform that provides state-of-the-art pretrained models, which can be used in the development of this project via its Transformers library and Model Hub. In this project, Hugging Face is used to load pretrained models such as **almanach/camembert-base** (Martin, et al., 2020) and **Jean-Baptiste/camembert-ner** (CamemBERT NER - HuggingFace model, 2021).

Figure 3.L: HuggingFace Logo

3.1.15. Weights and Bias (WandB)

Weights and Biases (WandB) is an experiment tracking tool widely used in machine learning for logging training runs, visualizing metrics, and managing hyperparameter tuning.

In this project, WandB is used to monitor model training in real-time as well as to compare different experiments to choose the best model for implementation.

3.2. Coding Guideline

In this project, strict coding guideline is implemented to ensure a high-quality and more readable codebase. The rules I followed include:

- Variable names must follow the *snake_case* (like **variable_name**) format (not camelCase).
- There must be no unnecessary spaces between lines of code.
- The code must not contain unnecessary comments.
- Each function must be documented, explaining its purpose, parameters, and outputs.
- Variable names must be meaningful to facilitate code review and debugging by other teammates.
- Four log files are created to record predictions, email routing, processing time, and extracted data for debugging purposes.
- Log files are zipped every day to reduce disk space and zip files that are older than ten days will be deleted.

These guidelines help maintain a clean and organized codebase, facilitating collaboration within the team.

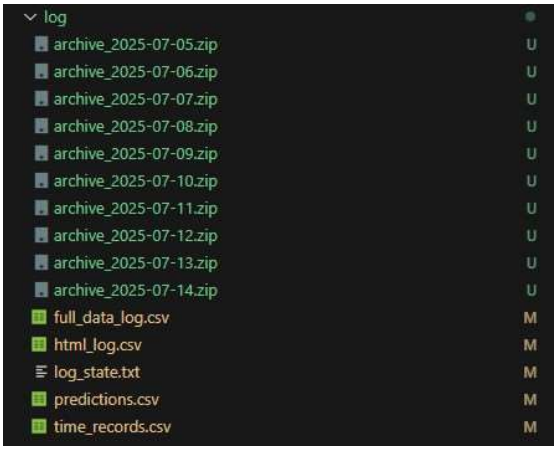


Figure 3 – Log files

IV - MODEL DEVELOPMENT

This chapter will present the process of data collection, data cleaning, literature reviews, model experimentations, evaluation metrics, and model implementation.

4.1. Data Collection

Since we are working with a custom intention classification model, the data that our model will be trained on is fully from the **service.clients@cosywee.com** itself. Due to security reasons, I did not have access to the full dataset from past years, and I only had my hands on the email samples starting from January 2025 onward.

The data is divided into 4 classes for 4 different sociétés/companies:

- Cosywee Alès
- Cosywee Lyon
- Cosywee Marseille
- Cosywee Perpignan

4.1.1. Problem

The first problem that I had was data collection. The data that I first received was in .PST file format. And in order to extract email content and attachments from .PST file, I experimented with some Python libraries including:

- Aspose.Email (Aspose.Email for Python via .NET, 2024)
- extract_msg (extract_msg: Python library to extract emails from MSG files, 2024)
- pypff (libpff/pypff: Library and Python bindings to access Personal Folder File (PST, OST), 2024)
- Outlook COM Automation (win32com)

Each one of these libraries has its own limitations and I evaluate them based on these important categories:

- Format Supported
- Metadata extraction (Subject, sender, recipients, body, attahments)
- Ability to identify inline vs Outline attachments (To remove signatures and email footer images)

- Performance and Setup easibility
- Platform supported
- Pricing and Limitation

Below is a table of performance of each library and its limitations.

Library	Format Support	Metadata Extraction	Inline vs Outline Attachment	Performance & Setup	Platform	Licensing
Aspose.Email	.PST, others	Yes	Yes	Fast & Easy	Crossplatform	Free for 50 mails, subscription required
Extract_msg	.MSG only	Yes	No	Fast & Easy	Crossplatform	Open-source, format limited
pypff	.PST	Yes	No	Slow, Complex Setup	Crossplatform	Open-source, hard to automate
Outlook COM	.MSG, .PST	Yes	Yes	Good but Platform bound	Windows only	Free, not portable

Table 1 - Comparison of Email Processing Libraries

- **Aspose.Email** is capable of extracting attachments, body text, subject lines, and other metadata accurately. It can also distinguish between inline and regular attachments, which is useful for filtering out elements like logos and email signatures. However, it comes with a major limitation: it's subscription-based. The free version only allows extraction from the first 50 emails of each .PST file, after which a paid license is required.
- **Extract_msg** can accurately extract metadata such as sender, recipients, subject, and body text. However, it only supports .MSG files, making it incompatible with our current dataset. Additionally, it cannot differentiate between inline and regular attachments—an essential feature for removing embedded images and signature blocks.
- **pypff** offers similar functionality to extract_msg but is notably slower and more complex to set up. This makes it unsuitable for our goal of building a repeatable and

automated system, especially one that can be easily re-executed via shell scripts when retraining the model.

- **Outlook COM Automation** is another option that provides access to both .MSG and .PST formats and supports inline attachment detection. However, it only works on Windows, which limits its portability and makes it less suitable for a cross-platform or server-based deployment.

4.1.2. Solution

Since each library has its own limitation and there is none of them fits it all, I had to work around them. The solution I implemented was to combine the **Aspose.Email**'s ability to process .PST file and **extract_msg** library to extract information.

The script to collect dataset is divided into 3 parts:

- Converting .PST file to multiple .MSG files (one .MSG for one email) with **Aspose.Email**
- Utilizing extract_email library to extract information and attachment from .MSG files.
- Implementing safeguards and filters to remove signature and footer images and attachments.

4.2. Data Cleaning

4.2.1. Email body cleaning

To extract clean and usable data from customer emails, I developed a multi-step text preprocessing pipeline. The primary goal was to isolate the main message content and remove non-informative elements like email signatures, automatic footers, greetings, and previous replies.

I begin by parsing the HTML content of the email using the BeautifulSoup(8) library. Tags such as
, <p>, and were converted into newline placeholders to ensure the structure is preserved when extracting plain text. Special attention was given to elements with class or ID values containing the word "signature", these were removed entirely to avoid capturing footers or company branding.

After converting the HTML to text, I inserted line breaks before certain key phrases like "Bonjour (*Translation: Hello*)", "Envoyé (*Translation: Sent* - Indicating sent time)", or "De : (*Translation: From*)" to improve the segmentation of email components. I also

eliminated hidden or corrupt Unicode characters and split the text into individual lines for easier analysis.

Then each line was then cleaned further:

Lines containing metadata (e.g., sender, recipient, subject) or unwanted content (URLs, phone numbers, formal closings like "Cordialement" (*Translation: Best Regards*)) were ignored.

If the system detected the beginning of a reply to Cosywee's customer service team, it stopped processing further, as these are often auto-generated or duplicate content.

The remaining lines, assumed to be part of the user's message, were concatenated into a unified body.

Lastly, we performed post-processing to remove repeated symbols (e.g., dashes or underscores), and polite phrases that do not add analytical value (e.g., "Merci beaucoup (*Translation: Thank you*)", "Bonne journée (*Translation: Hope you have a good day*)"). The resulting cleaned message was then stored in a structured format.

4.2.2. Attachment cleaning

Since I was using **Tesseract** (Pytesseract) (pytesseract: Python wrapper for Google Tesseract-OCR, 2024) and **EasyOCR** (JaidedAI, 2024) to extract texts and OCR images and PDF files, I had to clean those extracted text again to remove low-quality texts and noises. Here is a function where I clean text, remove noises and OOV (out of vocabulary) words, except custom words used in the company.

Here is the function to extract texts from images and pdf files:

Algorithm 1 Extract Forwarded Custom Email Data

```
procedure ExtractEmailData(html)
    Parse html with BeautifulSoup into soup
    Replace <br>, <p>, <div> with [[NEWLINE]]
    Remove <div>s with "signature" in class or id
    Convert soup to plain text
    Replace [[NEWLINE]] with real line breaks
    Insert line breaks before keywords (e.g. “Envoyé”, “De :”, “Bonjour”)
    Remove non-UTF characters
    Save result to log file
    Initialize data dictionary with empty fields
    Initialize body_content as empty string
    for each line in the cleaned text do
        Normalize and lowercase line
        if line starts with metadata (e.g., "de :", "cc :", "objet :") then
            continue
        else if line is a known Cosywee reply signature then
            break
        else if line contains contact info or common closing phrases then
            continue
        else
            Append cleaned line to body_content
        end if
    end for
    Clean body_content (remove dashes, signatures, greetings)
    Set data.body and data.body_detached
    return data
end procedure
```

Algorithm 2 Extract Text Word by Word with EasyOCR, Tesseract and Trained TrOCR

```
procedure ExtractText(image, header)
    Correct image orientation
    Initialize empty text and seen words set
    Crop image from header to bottom as img_full
    Convert img_full to grayscale
    Determine orientation (vertical or horizontal)
    Divide image dimension into 3 sections
    for i ∈ {0, 1, 2} do
        Extract section slice from grayscale image
        Try
          Run EasyOCR on section
        Catch
          Set EasyOCR results to empty
        Extract bounding boxes from EasyOCR results with confidence threshold
        Run Tesseract OCR on section, extract bounding boxes with confidence threshold
        Merge and sort all bounding boxes from both OCRs
        for each bounding box do
            Pad and crop bounding box area
            if crop valid then
                Run OCR on cropped image to get word (trained TrOCR)
                if word not empty and unseen then
                    Add word to text and seen set
                    Draw bounding boxes on debug images
                end if
            end if
        end for
    end for
    Save debug images
    clean accumulated text with clean_text() function
    return accumulated text
end procedure
```

Algorithm 2 above shows the process of extracting text from an image. First, the scanned document(image) orientation is corrected (by rotating the document four times with different angles and selecting the angle that results in the most text written from left to right). Then, the scanned document is cropped into three different smaller images (or

sections), and each section is passed to EasyOCR and Pytesseract OCR engines to detect text.

The reason why we use both OCR engines is because EasyOCR tends to work better with handwritten text, while Pytesseract performs well on printed text.

Next, for each detected text, the program extracts their bounding boxes (coordinates of the detected text). Using those coordinates, smaller images are cropped, one image per word or phrase in a line. Each image is then passed through OCR again, and the extracted text is added to a list to build the dataset. This is how the program combines text detection capabilities from EasyOCR and Pytesseract with the text extraction capabilities of TrOCR.

And here is my function to clean the text after performing OCR as described above. In this case, I used two additional resources for the French language, including:

- spacy's **'fr_core_news_lg'** (spaCy French Language Model: fr_core_news_lg, 2024) model
- **Jean-Baptiste/camembert-ner** (CamemBERT NER - HuggingFace model, 2021) model from HuggingFace

Algorithm 3 above shows the process of cleaning text using spaCy and Named Entity Recognition (NER) filtering. First, the input string is cleaned by removing commas and semicolons. Then, spaces are inserted around punctuation that may be attached to words. The cleaned string is then processed with spaCy's NLP pipeline, which converts the text into lowercase and tokenizes it.

For each token, the algorithm checks whether the word is found in the company's custom vocabulary. If so, it is preserved and directly added to the list of cleaned tokens. Otherwise, it continues the filtering steps. Tokens that are punctuation marks, spaces, numbers, or contain any digits are skipped. It also checks whether the token's base form is a color or a month, etc. If it actually is, the token will be removed too. After that, some tokens will be filtered out if they are recognized with high confidence by the NER engine as locations, organizations, or people, etc. Finally, a token is only kept if it is not a stopword, not out-of-vocabulary, has at least two characters, and is not a duplicate.

4.2.3. Final Data

In this subsection, the final data is presented. The training and testing data are shuffled and selected randomly from a dataset that is saved in a .CSV (Comma-separated Values) file. This file consists of multiple columns containing different information including sender, recipients, subject (Object of the email), body, contents extracted from attachments, etc. You may find an example of the dataset in Figure 4 - Dataset.

Figure 4 - Dataset

This data was then split into training, testing, and validation set. Since two types of AI models/architectures were used (to be detailed in the next chapters), the data is split differently as well.

- **CamemBERT**: Mainly 70% for training, 15% for validation and 15% for testing *
- **Traditional Machine Learning Algorithms (Random Forest)**: Mainly 80% for training, and 20% for testing **

**** This is default partition. Some models are trained on different partition depending on availability of data.***

***** This is partition is for training. For model experimentation, only 70% of the data were used to reduce workload.***

4.3. Model Experimentation

After all of the data are cleaned and written into a .CSV file, we finally get dataset with 7 columns: **Label, Sender, Recipients, SentTime, Subject, Body text,** and **Text extracted From Attachments.**

This data were used to trained multiple algorithms and architectures of models including Naive Bayes, Random Forest, Logistic Regression, Support Vector Machine, etc. Moreover, it was also used to finetune existing models like CamemBERT (French Large Language Model) as well.

As a reminder, we have to classify between four different factories, COSYWEE ALES, COSYWEE LYON, COSYWEE MARSEILLE, COSYWEE PERPIGNAN. At the same time, we also have to classify email's intents (whether it is an order, an invoice, a price quotation request, an After-Sale service request, or anything related to Accounting department, etc). Moreover, we have to take into account the fact that two factories produces same type of products, meaning if the mail is related to the shared products, the program have to sort them to a sub-folder for human to manually classify them.

In figure 4.B, you may find the proposed pipeline.

- An AI model for intent classification
- An AI model for classification between **COSYWEE ALES, COSYWEE LYON**, and **MP** (short of Marseille and Perpignan)
- Another AI model for classification between **COSYWEE MARSEILLE, COSYWEE PERPIGNAN** and **Roller Shutters** products.

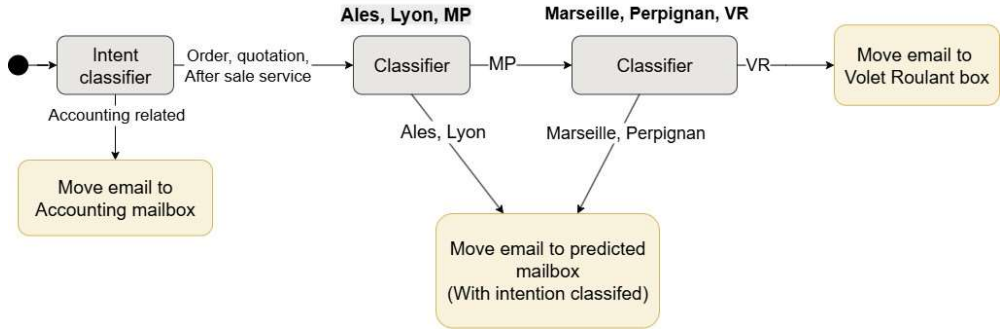


Figure 5 - Overview of model pipeline

As for the model experimentation and development, **Optuna** (Optuna: A Next-generation Hyperparameter Optimization Framework, 2019), and Scikit-learn (Scikit-learn: Machine Learning in Python, 2011) were used to find the best performing model to implement into our pipeline. Each algorithm (Traditional Machine Learning algorithms only) was experimented with hundreds of combinations of parameters with number of trials = 250.

```
nltk.download('stopwords')
french_stopwords = stopwords.words('french')

def get_common_preprocessing(trial):
    max_features = trial.suggest_categorical('vectorizer__max_features', [200,
        400, 500, 750, 1000, 2000, 3000, 4000, 5000])
    ngram_str = trial.suggest_categorical('vectorizer__ngram_range', ['1,1', '1,2
        ', '1,3', '2,2'])
    ngram_range = tuple(map(int, ngram_str.split(',')))
    stop_words_option = trial.suggest_categorical('vectorizer__stop_words', ['
        none', 'french'])
    stop_words = None if stop_words_option == 'none' else french_stopwords
    binary = trial.suggest_categorical('vectorizer__binary', [True, False])
    k = trial.suggest_categorical('selector__k', [70, 80, 100, 150, 200, 300,
        500, 700])
    k = min(k, max_features)
    score_func_name = trial.suggest_categorical('selector__score_func', ['chi2',
        'f_classif'])
    score_func = chi2 if score_func_name == 'chi2' else f_classif
    vectorizer = CountVectorizer(max_features=max_features, ngram_range=
        ngram_range, stop_words=stop_words, binary=binary, preprocessor=
        clean_text_for_company_words)
    selector = SelectKBest(score_func=score_func, k=k)

    return [('vectorizer', vectorizer), ('selector', selector)]
```

The algorithms that I experimented on include:

- Logistic Regression (LR)

```
def lr_objective(trial):
    common_steps = get_common_preprocessing(trial)
    C = trial.suggest_float('C', 0.1, 5.0, log=True)
    penalty = trial.suggest_categorical('penalty', ['l1', 'l2'])
    solver = 'liblinear'
    classifier = LogisticRegression(
        C=C,
        penalty=penalty,
        solver=solver,
        max_iter=1000
    )
    pipeline = Pipeline(common_steps + [('classifier', classifier)])
    scores = cross_val_score(pipeline, data["full_with_att"], data["
        label_encoded"], cv=5, scoring='accuracy')

    return scores.mean()
```

- Support Vector Classifier (SVC)

```
def svm_objective(trial):
    common_steps = get_common_preprocessing(trial)
    C = trial.suggest_float('C', 0.1, 10.0, log=True)
    kernel = trial.suggest_categorical('kernel', ['linear', 'rbf'])
    classifier = SVC(C=C, kernel=kernel)
    pipeline = Pipeline(common_steps + [('classifier', classifier)])
    scores = cross_val_score(pipeline, data["full_with_att"], data["label_encoded
        "], cv=5, scoring='accuracy')

    return scores.mean()
```

- eXtreme Gradient Boosting (XGB) (14)

```python
def xgb_objective(trial):
    steps = get_common_preprocessing(trial)
    n_estimators = trial.suggest_categorical('classifier__n_estimators',
        [100, 300, 400, 500])
    max_depth = trial.suggest_int('classifier__max_depth', 3, 7)
    learning_rate = trial.suggest_float('classifier__learning_rate', 0.01,
        0.2, log=True)
    subsample = trial.suggest_categorical('classifier__subsample', [0.7,
        1.0])
    model = XGBClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        learning_rate=learning_rate,
        subsample=subsample,
        use_label_encoder=False,
        eval_metric='logloss'
    )
    pipeline = Pipeline(steps + [('classifier', model)])
    scores = cross_val_score(pipeline, data["full_with_att"], data["
        label_encoded"].astype(int), cv=5, scoring='accuracy')
    return scores.mean()
```

- Random Forest (RF)

```python
def rf_objective(trial):
    steps = get_common_preprocessing(trial)
    n_estimators = trial.suggest_categorical('classifier__n_estimators',
        [20, 30, 50, 70, 85, 90, 100, 150, 200, 250, 300, 400, 500])
    max_depth = trial.suggest_categorical('classifier__max_depth', [None, 2,
         3, 5, 7, 10, 20, 30])
    min_samples_split = trial.suggest_categorical('
        classifier__min_samples_split', [2, 3, 4, 5, 7, 8, 10])
    model = RandomForestClassifier(
        n_estimators=n_estimators,
        max_depth=max_depth,
        min_samples_split=min_samples_split
    )
    pipeline = Pipeline(steps + [('classifier', model)])
    scores = cross_val_score(pipeline, data["full_with_att"], data["
        label_encoded"], cv=5, scoring='accuracy')

    return scores.mean()
```

- K-Nearest Neighbors (KNN)

```python
def knn_objective(trial):
    steps = get_common_preprocessing(trial)
    n_neighbors = trial.suggest_categorical('classifier__n_neighbors', [3, 5, 7, 10])
    weights = trial.suggest_categorical('classifier__weights', ['uniform', 'distance'])
    algorithm = trial.suggest_categorical('classifier__algorithm', ['auto', 'ball_tree', 'kd_tree', 'brute'])
    p = trial.suggest_categorical('classifier__p', [1, 2])  # 1 = Manhattan, 2 = Euclidean
    model = KNeighborsClassifier(
        n_neighbors=n_neighbors,
        weights=weights,
        algorithm=algorithm,
        p=p
    )
    pipeline = Pipeline(steps + [('classifier', model)])
    scores = cross_val_score(pipeline, data["full_with_att"], data["label_encoded"], cv=5, scoring='accuracy')

    return scores.mean()
```

- Naive-Bayes (NB)

```python
def nb_objective(trial):
    steps = get_common_preprocessing(trial)
    nb_type = trial.suggest_categorical("classifier_type", ["multinomial", "bernoulli"])
    if nb_type == "multinomial":
        alpha = trial.suggest_float("classifier__alpha", 1e-3, 1.0, log=True)
        classifier = MultinomialNB(alpha=alpha)
    else:
        alpha = trial.suggest_float("classifier__alpha", 1e-3, 1.0, log=True)
        binarize = trial.suggest_float("classifier__binarize", 0.0, 1.0)
        classifier = BernoulliNB(alpha=alpha, binarize=binarize)
    pipeline = Pipeline(steps + [('classifier', classifier)])
    scores = cross_val_score(pipeline, data["full_with_att"], data["label_encoded"], cv=5, scoring='accuracy')
    return scores.mean()
```

- CamemBERT (French Large Language Model)

As for the CamemBERT model, which is a transformer-based model, I experimented on multiple hyperparameters including : **Learning rates**, **train and test batch sizes**, **Train epoches**, **Regularization techniques**, etc.

4.4. Evaluation Metrics

In this section, we discuss the performance of our models by evaluating them on the three metrics including Accuracy, Precision and F1-score.

Confusion Matrix and Terminology

These metrics are based on four possible outcomes in a binary classification context:

- **TP (True Positives)**: Instances correctly predicted as positive.
- **TN (True Negatives)**: Instances correctly predicted as negative.
- **FP (False Positives)**: Instances incorrectly predicted as positive (also called Type I error).
- **FN (False Negatives)**: Instances incorrectly predicted as negative (also called Type II error).

These values are typically organized in a confusion matrix as follows:

Actual / Predicted	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Evaluation Methods

- **Accuracy:**

  Accuracy measures the proportion of correct predictions out of the total number of predictions made.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

- **Precision:**

  Precision measures the proportion of positive predictions that were actually correct.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

- **F1 Score:**

The F1 Score is the harmonic meaning of Precision and Recall. It balances both metrics, which are especially useful when dealing with imbalanced datasets.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4.3)$$

4.5. Results

As a reminder, in our pipeline, we need 5 distinct trained models to better capture the context, intent, and products mentioned in each email, while also helping to reduce model complexity. Please check on figure 4.B for the pipeline architecture.

4.5.1. Intent classification model

Because this model focuses on sentiment analysis instead of relying on keywords (product codes or names), I chose to implement BERT architecture model instead of traditional machine learning methods or other architectures.

After multiple experimentations and hyper-parameters testing, these following hyperparameters yield the best results:

```
learning_rate=3e-6
per_device_train_batch_size=8,
per_device_eval_batch_size=6,
num_train_epochs=70,
weight_decay=0.01,
metric_for_best_model="eval_loss",
greater_is_better=False,
early_stopping_patience=10
```

With these parameters, the statistics were:

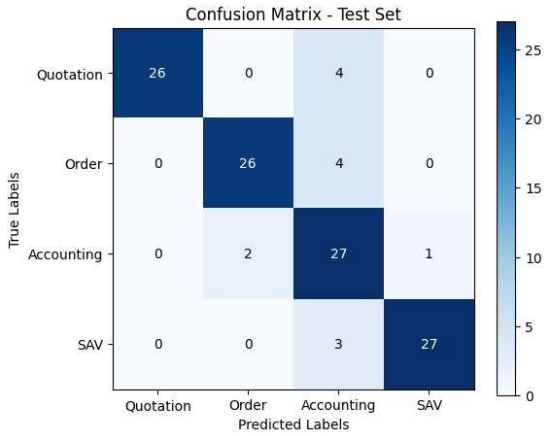


Figure 6 - Confusion matrix of intents classification

As shown in Table 2 and Figure 6, approximately 9 out of 10 emails are classified correctly with this model. Some emails were wrongly classified in Class 2 (Accounting) because of the nature of the emails themselves. Some emails contain multiple intents, which led to confusion. For example, an email is sent to place an order and also to request bank account information (in the same email), etc.

Data	Accuracy	Precision	F1
Validation Set	92.30%	92.41%	93.07%
Test Set	88.33%	90.08%	88.76%

Table 2 - Performance of intents classification model

4.5.2. Alès, Lyon, MP classification

4.5.2.1. CamemBERT Model

After extensive experimentation and hyperparameter tuning, the following settings yielded the best results:

```
learning_rate=6e-6
per_device_train_batch_size=8,
per_device_eval_batch_size=6,
num_train_epochs=70,
weight_decay=0.01,
metric_for_best_model="eval_loss",
greater_is_better=False,
early_stopping_patience=12
```

With these settings, the performance metrics were:

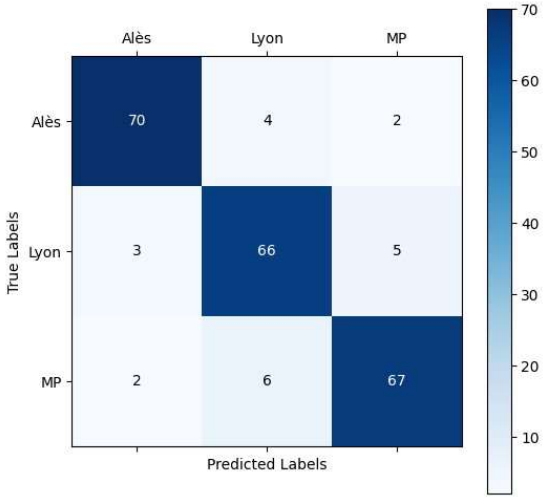


Table 3 - Confusion Matrix of Alès, Lyon, MP classification

As shown in Figure 4.E and Table4.3, around 90% of emails are correctly classified (in the test set). The reason why Class 0 (Alès) has the most correctly classified samples is that the majority of the products in this class are different from the other two classes.

Data	Accuracy	Precision	Recall
Validation Set	92.48%	92.77%	92.51%
Test Set	90.22%	90.24%	90.22%

Table 4 - Performance of Alès, Lyon, MP classification model

4.5.2.2 Traditional Machine Learning Algorithms

Utilizing Optuna and Scikit-learn, I have experimented with seven different models such as XGB, Random Forest,...

I used only 70% of the data for this experimentation, and below are each model's accuracy on the balanced dataset:

Type of Model	Hyperparameters	Accuracy
Logistic Regression	C = 4.2676, penalty = L2	90.28%
Support Vector Classifier	k = 200, Kernel = rbf, C = 8.9056	90.48%
XGB	learning_rate = 0.0469, sub_sample = 1	**91.01%**
Random Forest	n_estimators = 85, max_depth = None = 1, min_samples_split = 2	90.79%
K-Nearest Neighbors	n_neighbors = 10, algorithm = Brute, p = 1	84.46%
Naive-Bayes	type = multinomial, alpha = 0.0090	87.69%

Table 5 - Performance Comparison of Algorithms - Alès, Lyon, MP classification

Based on the results above, the XGBoost algorithm achieves the highest performance (91.01% accuracy). However, considering the importance of model interpretability in operational environments, for traceability, auditing, and maintainability, the Random Forest algorithm (90.79% accuracy) was selected. This model offers a strong balance between performance and transparency, making it more suitable for practical deployment in this context.

After selecting Random Forest as our base model, I split the dataset into training and testing sets using an 80/20 split. Then, I retrained the model, achieving an accuracy of **91.08%** on test set.

4.5.3. Marseille, Perpignan, Roller Shutter *(VR)* classification

4.5.3.1 CamemBERT Model

After extensive experimentation and hyperparameter tuning, the following settings yielded the best results:

```
learning_rate=5e-6
per_device_train_batch_size=8,
per_device_eval_batch_size=6,
num_train_epochs=70,
weight_decay=0.01,
metric_for_best_model="eval_loss",
greater_is_better=False,
early_stopping_patience=7
```

With these settings, the performance metrics were:

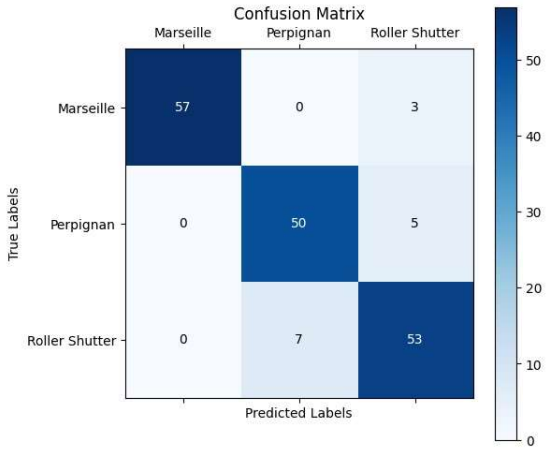


Table 6 - Confusion Matrix of Marseille, Perpignan, Roller Shutter classification

In Table 7 and Table 6, the statistics of the results of the Marseille, Perpignan, Roller Shutter classification model are shown. There is a small confusion between Class 1 and Class 2 (Perpignan and Roller Shutter), because sometimes, the characteristics used to describe the product lines of the 2 factories are identical, making it harder for the model to predict without a definite keyword.

Data	Accuracy	Precision	F1
Validation Set	92.03%	92.29%	92.13%
Test Set	91.44%	91.53%	91.44%

Table 7 - Performance of Marseille, Perpignan, Roller Shutter classification model

4.5.3.2 Traditional Machine Learning Algorithms

Utilizing **Optuna** and Scikit-learn, I have experimented with seven different models such as XGB, Random Forest, ...

I used only 70% of the data for this experimentation and below is each model's accuracy on balanced dataset:

Type of Model	Hyperparameters	Accuracy
Logistic Regression	C = 1.5059, penalty = L1	89.55%
Support Vector Classifier	k = 200, Kernel = rbf, C = 8.26	89.55%
XGB	Learning_rate = 0.0373, sub_sample = 0.7	90.17%
Random Forest	n_estimators = 150, max_depth = None = 1, min_samples_split = 3	**90.29%**
K-Nearest Neighbors	n_neighbors = 3, algorithm = kd_tree, weights = distance, p = 1	83.01%
Naive-Bayes	type = multinomial, alpha = 0.05657	80.91%

Table 8 - Performance Comparison of Algorithms - Marseille, Perpignan, Roller Shutter classification

Based on the results above, the Random Forest algorithms achieve the highest performance metrics (90.29%).

With an 80/20 split, I retrained the model, achieving an accuracy of approximately **92.63%**. This might seem low but considering the circumstances and the nature of the class (overlapping with no clear boundary), these results satisfied the company's expectations.

4.6. Final Pipeline

After the experimentation described above and multiple iterations of trial and error, I implemented a pipeline that combines several models, including CamemBERT-based models and a Random Forest classifier.

As shown in 4.F, the CamemBERT model is used for intent classification, while a combination of CamemBERT and Random Forest is employed to classify between COSYWEE ALES, COSYWEE LYON, COSYWEE MARSEILLE, COSYWEE PERPIGNAN, and Roller shutter *(Volet Roulant - VR)*.

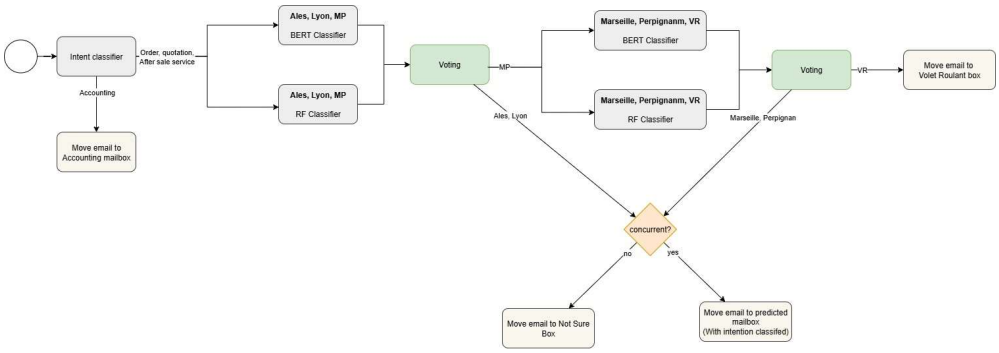


Figure 7 - Fully integrated AI pipeline

As illustrated in the flowchart above, I employed a combination of CamemBERT-based models and Random Forest models to classify its class using a hard voting strategy. This generally means that if both models agree on a specific class, the email would be classified automatically. But if their votes contradict, we will move them to a folder called **"A CLASSER (Not Sure)"**, which will be sorted manually later.

This approach has proven effective as it captures spam emails, and emails with insufficient information to the **"A CLASSER (Not Sure)"** folder for further inspection, ensuring all emails are correctly classified.

V - IMPLEMENTATION & PROTOTYPE BUILDING

After a detailed experimentation and literature review have been conducted, I began to implement the pipeline into production.

Although until now this project may seem like a normal text-classification problem, in order to maximize accuracy, taking advantage of existing clues, and keys I found on project analysis, I decided to implement the trained text classification AI pipeline with other heuristics as well as other methods as well.

Other methods and heuristics include Image classification, OCR, Regular Expression, etc.

5.1. Project Development

5.1.1. Environment configuration

Before diving into the implementation, it is crucial to configure the development and testing environment correctly first. This project needs multiple libraries and tools to function efficiently and correct without error like Pytorch(15), Pytesseract(9), OpenCV(16), beautifulsoup4(8), EasyOCR(10), Pandas(17), Transformers, PyMuPDF(18), etc. for the text classification process.

Those libraries include:

opencv-python	==	4.11.0.86
pytorch	==	2.7.0
pytesseract	==	0.3.13
beautifulsoup4	==	4.13.3
PyMuPDF	==	0.0.1.dev2
EasyOCR	==	1.7.2
transformers	==	4.50.1
docx2txt==0.9	==	0.9
nltk	==	3.9.1
paddleocr	==	2.4
extract_msg	==	0.54.1
spacy	==	3.7.5
msal	==	1.32.0
pandas	==	1.5.3
pillow	==	10.4.0

ftfy == 6.3.1

These specific library versions are required to ensure the program runs correctly. In particular, compatibility issues exist between newer versions of opencv-python and paddleocr. Starting from version 4.12.0.0 of opencv-python, certain internal dependencies and APIs used by paddleocr are no longer supported or behave differently, leading to runtime errors such as segmentation faults, unexpected crashes, or broken image preprocessing pipelines. So we have to install the libraries version as shown above to ensure the compatibility between those libraries.

5.1.2. Project Structure

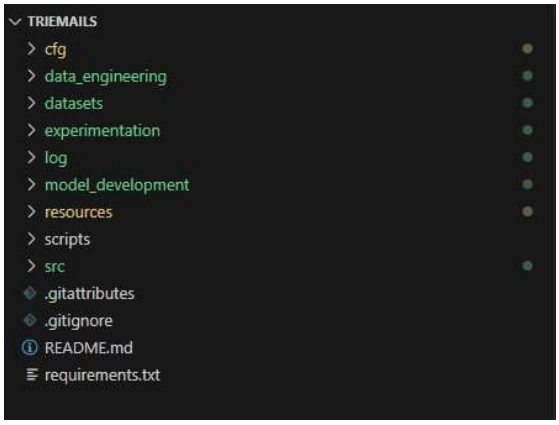


Figure 8 - Project Structure

Figure 8 shows the structure of the entire project:

- **cfg**: Contains configuration files such as access tokens and path settings.
- **data_engineering**: Contains scripts to scrape and clean data from .PST and .MSG files.
- **datasets**: Contains all scraped and cleaned datasets.
- **experimentation**: Contains all .ipynb and .py scripts used to experiment with different libraries and machine learning model architectures.
- **log**: Contains log files (mainly for debugging).
- **model_development**: Contains trained models and code (with documentation) to retrain or fine-tune them if necessary.
- **resources**: Contains important files such as mappings between products and companies, or employees and companies.
- **scripts**: Contains shell scripts (.sh) used to automate parts of the project.

- **src**: Contains all script files needed to deploy the project.
- **requirements.txt**: A text file listing all packages and libraries required to run the program.

5.2. Heuristics methods

To quote one of my professors in class during his first lecture, Monsieur Sébastien Harispe, "**Avoid Machine Learning as much as possible**", as the system built with it would need maintenance over time once deployed.

With that said, I tried to integrate other heuristic approaches to achieve the same tasks as much as possible before using our trained model as the last resort.

That being said, I analyzed the flow of the emails, the attachments, and the senders themselves and found some things that I could take advantage of using some heuristic methods.

Those characteristics include:

5.2.1. Original Sender of the email

As previously mentioned above, the email flows into **service.client@cosywee.com** is bidirectional. Some emails are initiated by clients (Inquiry about new products, orders, or price quotation etc). Others are replies from client responding to the company employees. Upon careful analysis, a consistent pattern is identified. All emails originally sent by company employee starts with this pattern in the subject (*objet*) :

"(Cosywee-XXX) Commande nº" or **"(Cosywee-XXX) Devis nº"**

The program checks if the email was originally sent by an employee by parsing its subject with Regular Expression to find the specific patterns above. If the Regular Expression is matched, it filters out all lines of the email except the signature section to find the name of employee who sent the email . This then matches this name to the company associated with it, classifying emails correctly.

5.2.2. Document Creator

Another heuristic that can be leveraged involves identifying the document creator. Cosywee uses its own in-house software platform, developed by the company's IT team,

called Nexus. This platform is used to create and manage business documents such as Quotation, Purchase Order, Order Confirmation, among others.

Particularly, each document generated by an employee using this internal software includes the creator's name embedded in the document itself. As I did some research and analysis on customer behavior, I found that customers sometimes attach these documents to their emails as a reference for their inquiries.

Based on the information above, this feature of **Nexus** actually presents an opportunity for another heuristic approach with the help of OCR and the pdf_extraction library.

Figure 9 - Employee's name to Company

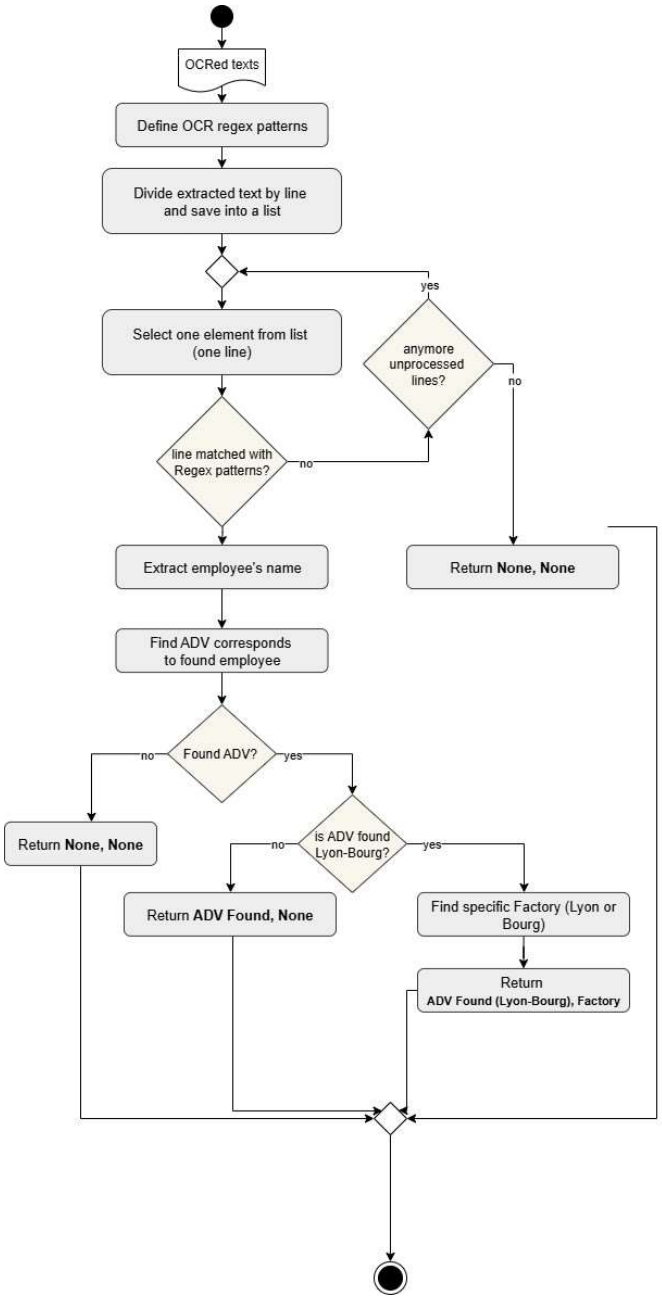


Figure 10 - Employee's name extraction from Purchase Order Flowchart

Therefore, by parsing extracted/OCRed text with a regular expression that searches for the pattern '**Document établi par XXXXXXX**' (English translation: Document created by XXXXXXX), we can then extract the name of the employee. This name can then be mapped and matched to the associated company, allowing a more reliable and accurate email classification feature Figure 9. Figure 10 shows the flow of the process to find the employee's name from the document generated.

5.2.3. Product extraction from Purchase Order

For each product produced by Cosywee, there is a Purchase Order that allows clients to complete the relevant product details such as dimensions, color, etc. This purchase order is created by Marketing team and follows the format as shown in Figure 11.

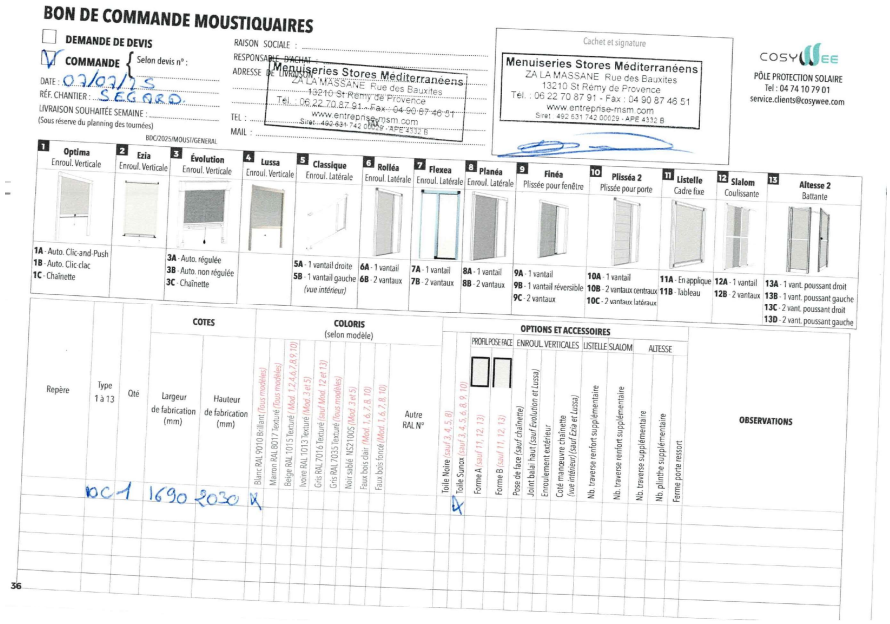


Figure 11 - Purchase order for mosquito net

As you can see in Purchase order Figure 12, you can find the name of the product on the top left of the document (**BON DE COMMANDE MOUSTIQUAIRES**). By using the algorithm presented in Figure 12, we can extract the name of the product from the Purchase order.

The flow of this program starts by calculating the dimension of the input image. Then it crops the top-left section of the image (assuming the name of the product is in that section). Then it performs OCR with Pytesseract to extract texts from the image. If it is unsuccessful, this mostly means the image is too small, meaning it's likely that the image is not a Purchase Order. Else, we split the extracted text line by line, comparing them to find the pattern (seeing if the line starts with **"BON DE COMMANDE"** (Translation: *PURCHASE ORDER* ...). If it does, we extract the texts after the matched words, which is the name of the product in the purchase order. Then we search the name of the product in our lists to match them with its corresponding manufacturer Figure 13Figure 12; then we can simply forward that message to the correct manufacturer.

Figure 12 - Matching Product's name/code to Company

Sometimes the rotated scan does crop out the header of the Purchase Order, which leads to missing product names as well. To counter this scenario, a few lines below the header are also OCRed to find the identification code as well. In Figure 11 the code is in this format **"BDC/2025/MOUST/GENERAL"**. With this code extracted, we can map it to the name of the product Figure 12, as well as to the company/factory that produces those products as well.

To sum up, with the name of the product on attachments, we can map it to the producer, leading to direct email classification without using a trained AI model.

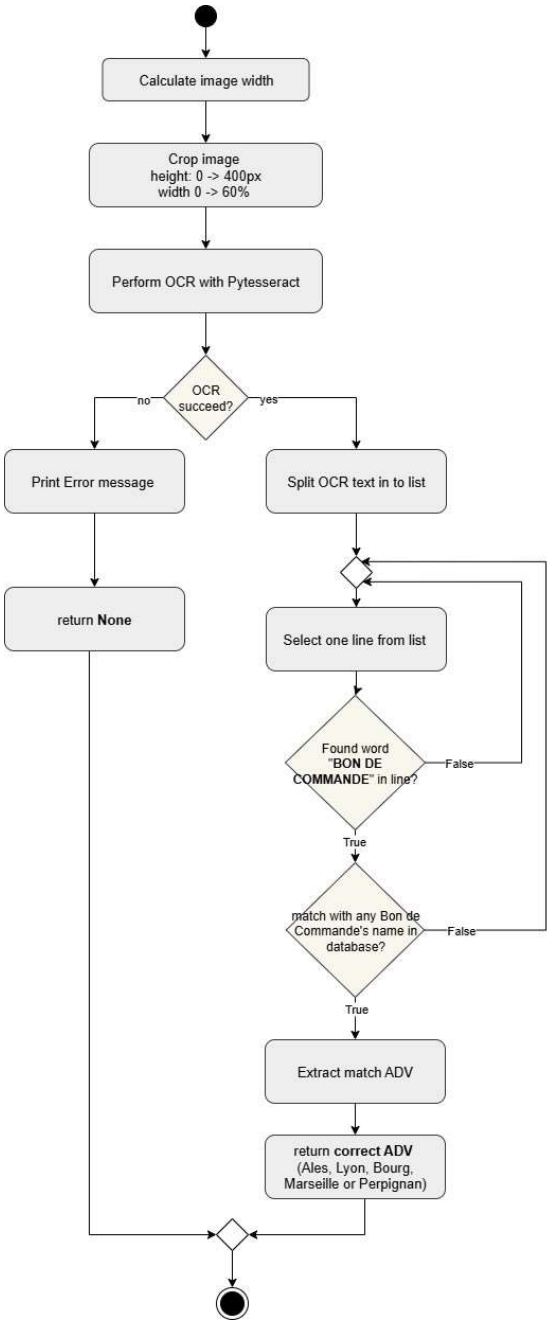


Figure 13 - Product name extraction from Purchase Order Flowchart

5.2.4. Intent extraction from Purchase Order

One more thing we can heuristically extract in the Purchase order is the intent of emails. You can see below the product name (Title of the document), there are two checkboxes, one for *Devis* (Quotation) and another one for *Commande* (Order). Clients may express their intent by selecting one of these checkboxes, and I took this into account by developing a function that uses Image Processing techniques to correct orientation and detect the marked checkbox.

Taking into account the nature of the scanned document as an input, the orientation, placement, scale, and the quality of the scanned *Purchase Order* are not always perfect. For example, as shown in Figure 15, the scanned document is at an angle. With these issues, it becomes almost impossible to detect the checkbox marks directly without applying preprocessing and cleaning steps first.

To solve this problem, a Feature Matching algorithm called Oriented FAST and Rotated BRIEF (ORB) was implemented. This algorithm was introduced in the paper **ORB: an efficient alternative to SIFT or SURF** by Rublee et al (ORB: An efficient alternative to SIFT or SURF, 2011). in 2011, proposing a new and more efficient way to extract interesting features from images compared to other previously introduced methods such as Scale-Invariant Feature Transform (SIFT) (Distinctive Image Features from Scale-Invariant Keypoints, 2004), and Speeded Up Robust Feature (SURF) (Machine learning for high-speed corner detection, 2006).

```python
orb = cv2.ORB_create(8500)
keypoints_scanned, descriptors_scanned = orb.detectAndCompute(scanned_image, None)
keypoints_model, descriptors_model =orb.detectAndCompute(model_image, None)
# Use BFMatcher to match descriptors
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(descriptors_scanned, descriptors_model)
# Sort matches by distance (best matches first)
matches = sorted(matches, key=lambda x: x.distance)
# Filter out low quality points
points_scanned = np.zeros((len(matches), 2),dtype=np.float32)
points_model = np.zeros((len(matches), 2),dtype=np.float32)
for i, match in enumerate(matches):
    points_scanned[i, :] = keypoints_scanned[match.queryIdx].pt
    points_model[i, :] = keypoints_model[match.trainIdx].pt
# Find the homography matrix
homography_matrix, mask =cv2.findHomography(points_scanned, points_model, cv2.RANSAC)
# Warp the scanned image to align with the reference model image
height, width = scanned_image.shape
corrected_scanned_image = cv2.warpPerspective(
    scanned_image,
    homography_matrix,
    (width, height),
    borderValue=(255,255, 255)
)
```

My method is to implement ORB feature matching between the scanned document (scanned Purchase Order), and a reference model image (a saved image of perfectly scanned document) matching those descriptors using Brute Force Hamming to correct the orientation and scale of the scanned Purchase Order.

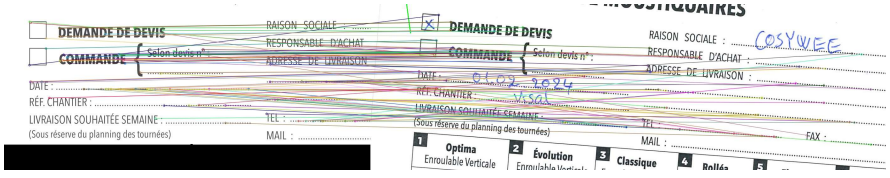


Figure 14 - ORB Features matching

EXAMPLE: Figure 15 (left side) show a rotated scan of a Purchase Order. If we were to crop out the checkbox for Demande de devis (Price Quotation request) and the checkbox for Commande (Order) with pre-defined coordinates, we wouldn't get the checkbox itself.

Figure 14, you can find the matched descriptors between the scanned image and the model image (while we are using ORB detection). Figure 15presents the result of the scanned image after rotation and translation correction.

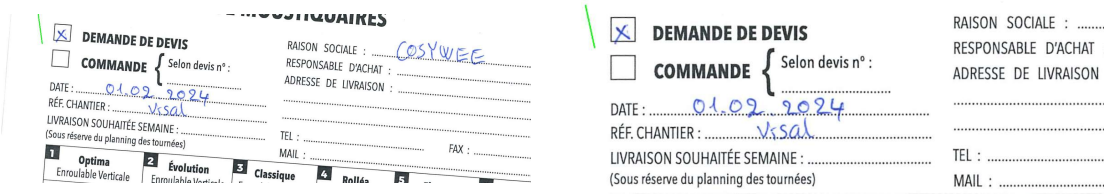


Figure 15 - Orientation corrected Purchase

After this, we can detect objects in each checkbox of the Purchase Order with the Contours detection feature from the OpenCV Library (OpenCV: Open Source Computer Vision Library, 2000).

5.3. Helper Functions with Microsoft Graph API

The main objective of this project is to build an automated system utilizing Artificial Intelligence to sort incoming emails to designated recipients. One main task of the program is to be able to read and move emails within the mailbox itself.

This subsection presents the methodology and steps involved in setting up the API to access company shared mailbox emails securely. This task requires a lot of research and careful attention due to strict security protocols and risk assessment aimed at preventing attacks in the company mailbox.

Email inboxes are among most sensitive components of a company's infrastructure. They usually contain confidential and secret client information, personal data (of employee and clients), internal conversations, all of which are highly susceptible to cyberattacks and data breaches.

To integrate this application on Microsoft Azure, first we had to choose an appropriate login method. Considering security protocols and importance, I opted for Access Token method which expires and is renewable every two hours using a refresh token and client secret (Client secret expires and requires manual reset every 18 months). This method offers a good balance between security and accessibility, ensuring a secure connection and program efficiency. The API permissions are **Mail.Send**, **Mail.ReadWrite** and **Mail.ReadWrite.Shared**. This allows the application to read and move emails in a shared mailbox.

5.3.1. Read Emails

In this subsection, I will present the flow to extract the metadata of the email such as the sender, subject, body, and any attachments. This function sends a request to **f"https://graph.microsoft.com/v1.0/users/shared_mailbox/mailFolders/folderID"**toread all emails within the mailbox.

```
def read_emails():
    global access_token, headers
    if inbox_folder_id is None or results_id is None:
        print("Error while fetching folder IDs")
        return
    messages_endpoint = f"{MAILFOLDER}/{inbox_folder_id}/messages?$top=100"
    start_time_program = datetime.datetime.now()
    while True:
        messages_response = requests.get(messages_endpoint, headers=headers)
        if messages_response.status_code != 200:
            print(f"Error retrieving messages: {messages_response.status_code}, {
                messages_response.json()}")

            access_token = authentication.get_access_token()  # Assume
                authentication module exists
            if not access_token:
                print("Access token not available")
                return
            print("Acquired New Token")
            headers = {AUTHORIZATION: f"Bearer {access_token}"}
            messages_response = requests.get(messages_endpoint, headers=headers)
            if messages_response.status_code != 200:
                return

        for index_message, message in enumerate(messages_response.json().get("
            value", [])):
            message_id = message["id"]
            message_body_html = message.get("body", {}).get("content", "")
            sender = (
                f"{message['sender']['emailAddress']['name']} <{message['sender
                    ']['emailAddress']['address']}>"
            )
            subject = message.get("subject") or ""
            # [.... Other processing code ...]
```

5.3.2. Get Folder ID

This function returns folder ID, based on the folder path. In Microsoft graph API, we cannot move email with the folder name explicitly. In order to move email, we have to pass three parameters:

- **Header** : the authentication token, containing access token to access the mailbox
- **MessageID**: the identification of the message to move
- **DestinationFolderID**: the identification of the folder we want to move message to

The get_folder_id function above allows us to get identification of the destination folder, with the display name and path of the folder itself.

```
def get_folder_id(headers, folder_name, parent_folder_id=None):
    if parent_folder_id:
        endpoint = f"{MAILFOLDER}/{parent_folder_id}/childFolders"
    else:
        endpoint = MAILFOLDER
    response = requests.get(endpoint, headers=headers)
    if response.status_code == 200:
        folders = response.json().get('value', [])
        for folder in folders:
            if folder['displayName'] == folder_name:
                return folder['id']
    print(f"{folder_name} folder not found.")
    return None
```

5.3.3. Move Email

To move email from one folder to another folder, we must send a request to **graph.microsoft.com** with a /**move** request.

The function below shows how a message is moved from one folder to another. It takes header (which is basically access_token), ID of the message to move, and the destination folder ID as parameters.

```python
def move_email(headers, message_id, destination_folder_id):
    move_endpoint = f"https://graph.microsoft.com/v1.0/me/messages/{message_id}/
        move"
    payload = json.dumps({"destinationId": destination_folder_id})
    move_headers = {
        AUTHORIZATION: headers[AUTHORIZATION],
        CONTENT_TYPE: APPLICATION_JSON
    }
    move_response = requests.post(move_endpoint, headers=move_headers, data=
        payload)
    if move_response.status_code == 201:
        print(f"Moved message {message_id} to folder {destination_folder_id}.")
        new_message_id = move_response.json().get('id')
        mark_as_unread(headers, new_message_id)
        return True
    else:
        print(f"Error moving message {message_id}: {move_response.status_code}, {
            move_response.json()}")
        return False
```

5.4. Final Classification Pipeline Implementation

The section presents the whole pipeline of the fully developed solution.

1. When a new email arrives in the inbox, our system will paste its subject and body text to our trained intent classification model.

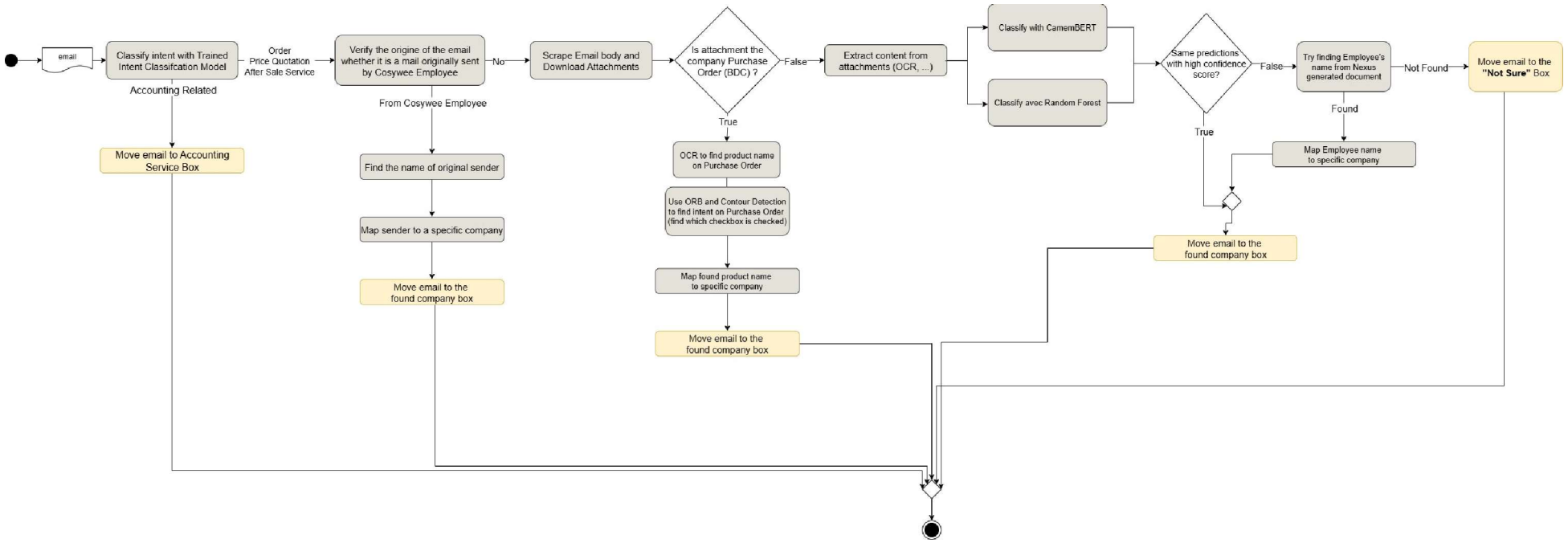
2. If it predicts that the intent of email is related to **Accounting purpose** (Service Comptabilité), the email will be sorted directly to Accounting mailbox and the classification pipeline ends there. Other than that, it extracts other metadata like sender, recipients, attachments, etc before checking if the email is a reply to a message originally sent by a Cosywee employee. If so, the system will automatically forward the message to the original sender.

3. If the email is not a reply, the system will parse the HTML body text and combine it with information extracted from any attachments. This includes text from .docx, .csv, .txt, and .pdf files, as well as OCR-extracted text from images or scanned PDFs.

4. In this stage, we apply another heuristic, which is to determine whether the attachment is actually a Purchase Order or not. If it is a Purchase order, we simply extract product name, extract intent from the checkbox, before mapping the product name to a specific company that produces it. Once the company responsible for the product is found, the system forwards the email directly to the corresponding department or recipient responsible for that specific Purchase Order type.

5. In the event the attachment is not a Purchase Order, all the gathered content above will be cleaned and passed through our trained models, which will predict and vote on the most likely recipient. The email will then be forwarded to the predicted recipient.

6. In cases where the predictions between the models are nonconcurrent, we pass them through another heuristic method, which is finding employee name from Nexus generated document. Just like in step 2 if the employee's name is found, it is mapped to a specific company the employee works at, then we can simply move email to their mailbox. If none of conditions above are matched, we simply move that email to **A CLASSER** (Not sure) box for manual inspection.


email
Classify intent with Trained Intent Classifcation Model
Order
Price Quotation
After Sale Service
Accounting Related
Move email to Accounting Service Box
Verify the origine of the email whether it is a mail originally sent by Cosywee Employee
No
From Cosywee Employee
Find the name of original sender
Map sender to a specific company
Move email to the found company box
Scrape Email body and Download Attachments
Is attachment the company Purchase Order (BDC) ?
False
True
OCR to find product name on Purchase Order
Use ORB and Contour Detection to find intent on Purchase Order (find which checkbox is checked)
Map found product name to specific company
Move email to the found company box
Extract content from attachments (OCR, ...)
Classify with CamemBERT
Classify avec Random Forest
Same predictions with high confidence score?
True
False
Move email to the found company box
Try finding Employee's name from Nexus generated document
Not Found
Found
Map Employee name to specific company
Move email to the "Not Sure" Box

5.5. Optimization and Improvement

As at Cosywee, projects are in **Agile** environment, we implemented the first prototype since early June to test our program. The goal is to develop the first program, receive feedback, improve, release a new version, receive more feedback, so on and so forth.

This environment really did help me in terms of agility, productivity and efficiency as I repair crucial bugs as I go instead of focusing on aspect that didn't matter as much.

5.5.1. Version v1.0

5.5.1.1 Pros

The first prototype of the AI system employs a custom-built OCR system and PaddleOCR (PaddleOCR: Optical Character Recognition tools based on PaddlePaddle, 2024) to extract texts from images and PDF files as we want to capture as much context as possible (PaddleOCR is used as a text detector, and the text images were then cropped and passed to a finetuned TrOCR model for text extraction). The goal was to OCR all texts, printed or handwritten, structured or non-structured data.

Figure below shows the capability of the system to extract non-structure and handwritten text from scanned document despite poor quality of document itself.

5.5.1.2 Cons

Although the program works well in all attachment conditions, it has its own perks as well.

As the AI system employs PaddleOCR and custom-trained OCR model (finetuned model from TrOCR), it takes a lot of computational power and time compared to other OCR engines like Pytesseract and EasyOCR.

The table below shows the differences between computational power, processing time and system capabilities between different setups.

	GPU Available?	RAM	CPU cores	Processing Time
With no attachment	Yes	16 GB	32	3 seconds per email
With attachment	Yes	16 GB	32	45 seconds per email

Table 9 - Processing Time vs Computing Power on a high computing power machine

	GPU Available?	RAM	CPU cores	Processing Time
With no attachment	No	8 GB	16	15 seconds per email
With attachment	No	8 GB	16	55 minutes per email

Table 10 - Processing Time vs Computing Power on a low computing power machine

As shown in Table 10, the processing time for emails with attachments on a low-resource computer takes too long (at least 55 minutes per email), which doesn't meet the company's criteria for low computing power and fast processing time.

After analyzing the attachment statistics, it was discovered that less than 6% of the attachments were handwritten. Considering the small proportion of handwritten attachments and the significant processing time they require, and after careful consideration, experimentation, and discussion with my project supervisor, we decided to release a new version of the AI email routing system with reduced OCR capability but with much more efficient processing time and lower computational cost.

5.5.1.3 Improvement aspects

This program performs quite well in terms of accuracy, but if we were to consider the processing time, it is not optimized yet. It requires a lot of processing power and takes too much time to process one email, sometimes even longer than manual classification. In this reason, this version is not deployed in production and was later improved in Version v2.0.

5.5.2. Version v2.0

This version is an improvement from last version (v1.0) as it drastically improves processing time, reduces computational costs and is more suitable for a server-based sub-process that runs on the background.

5.5.2.1 Pros

In this newer version, Pytesseract OCR engine is used to replace the combination of PaddleOCR and custom finetuned TrOCR to improve processing time as well as to reduce computational cost.

Table below shows the statistical comparison between our AI system with (PaddleOCR + TrOCR) and the AI system that employs the Pytesseract engine on a small and less powerful machine.

	GPU required?	RAM	CPU cores	Processing Time

With no attachment	No	4 GB	4	2 seconds per email
With attachment	No	4 GB	4	15 seconds per email

Table 11 - Processing Time vs Computing Power of version v2.0 on a low computing power machine

5.5.2.2 Cons

The newer version of the AI system performs significantly better in terms of computing power and processing time compared to the older version. However, there is one downside: it cannot extract handwritten text from attachments, as Pytesseract is mainly built for printed and uniform text.

Although this limitation exists, if we were to consider the differences in processing time and computational efficiency, it becomes clear that the newer version is better suited for production deployment compared to version v1.0.

VI - RESULTS AND DISCUSSIONS

For the last few months, I have been making my way up the ladder to develop an AI system that is capable of classifying and routing emails automatically to designated recipients from a shared mailbox.

I am pleased to say I developed a reliable and production-ready system for email classification, taking into account the body content, subject, and attachments' content.

In Fig 6.A, the process diagram of my work during this project is presented. In the first week, I was presented with the project, then I performed research and literature reviews while starting to collect and clean data to train the models. After all data were collected, multiple experiments were performed, and a few final models were trained to be used in production. I set up the API, project routes, project structures, and mail pipeline, then did some research on available heuristics to improve the accuracy of the AI system before implementing them and deploying it in production. Since then, I have been receiving feedback and reports from people in each factory and released multiple newer versions until a reliable and sustainable system is in place. At the end, I standardized my code (commenting, renaming variables to improve coherence, etc). In the last few weeks, I created a script to finetune models and clean data for the company to make changes when new problems/errors occur in the future. I also gave a presentation to my team's developers to share knowledge on topics such as model fine-tuning, data cleaning, and evaluating trained models.

Compared to rule-based email classification features proposed in Microsoft Outlook and other email provider services, this method can analyze the context, intent, and product mentioned in emails, making the classification more reliable and sentiment-based instead of keyword matching. The essence of this method is to implement as many heuristics as possible to ensure the reliability of the program before passing it to our trained classification model.

6.1. Results

Moreover, this project is run on a lower resource and computing power computer:

- **CPU:** 8 Cores, 12 threads
- **RAM:** 4GB
- No Graphics Processing Unit (GPU)

Other than this project, my role at the company also involves other tasks as well, such as:

- Setting up, organizing and lead weekly meetings every other week to talk about strength, weaknesses and interesting things every IT team member has faced in the prior week, as well as to evoke problems that everyone faces so that the team can help with their own experience.
- Reviewing, testing code, documentation written by other teammates
- Documenting processes and ensuring project scalability.

The final implementation is described as follows:

- Models Used: A hard voting ensemble composed of 2 Random Forest classifiers and 2 CamemBERT-based transformer models.
- Performance Optimization: In alignment with company requirements, handwriting recognition was deprioritized in favor of faster processing. As a result, the system uses Tesseract OCR (PyTesseract), which improves processing speed by over 80% compared to handwriting-capable OCR models. This approach also significantly reduces resource requirements, operating efficiently without GPU acceleration and with less than 8GB of RAM.
- Processing Time: The system processes emails without attachments in under 2 seconds. For emails with attachments requiring OCR and text extraction, the average processing time is approximately 15 seconds per attachment.
- Error Rate: The overall error rate remains below 2%. Approximately 12% of attachments require manual review due to factors such as handwritten content or insufficient information for confident classification.

Table6.1indicateshowmanyemails(inpercentage)aresortedwitheachheuristicmethod and with our trained model. It also shows the amount of email that the AI system failed to classify due to lack of information and/or other problems that will be discussed below.

	Average Processing Time	**Error rate**	**Email to classify manually***
With 500 emails	15 s / email	< 2%	< 12%

Table 12 - AI System capability (With 500 emails sorted)

***Note:** Emails that require manual classification don't necessarily indicate a failure of the AI pipeline. Rather, it's usually due to the nature of those emails being inherently difficult to classify automatically. These cases include:

- Emails with handwritten and unstructured texts
- Emails that don't contain enough information (e.g., only a reference number without mentioning a product or service)
- Spam emails
- Emails describing very generic information (like dimensions or price) that could apply to multiple products from different companies
- Emails with unusual purposes (e.g., registering new clients, announcements about vacation days from partner companies, etc.)

6.2. Future improvements

While current system archives a reliable and satisfactory performance in classify and routing emails, several improvements can be made to enhance accuracy, and robustness of the program.

6.2.1. Optimizing handwriting text recognition model

Currently, **Cosywee** has its own finetuned OCR model for handwriting text recognition. The base model of this OCR model is TrOCR, which is only capable of text recognition but lacks the strength of text detection. Moreover, it can only process text line by line, word by word and it takes a lot of time to predict/recognize each input. As the document has a lot of words/phrases, it would take even longer to process them, that is why in this system, we prioritize the speed of the program over accuracy of handwritten text. If I were to have more time, my priority would be to improve processing time of Cosywee's custom OCR model, which leads to reliable and automated handwriting-based attachment classification as well.

6.2.2. Integration of human feedback loop

Incorporating a real time feedback loop from human validation can help document any error the program has made along the way, which would be a great help for finetuning, and this problem can also be framed as a Reinforcement Learning project as well.

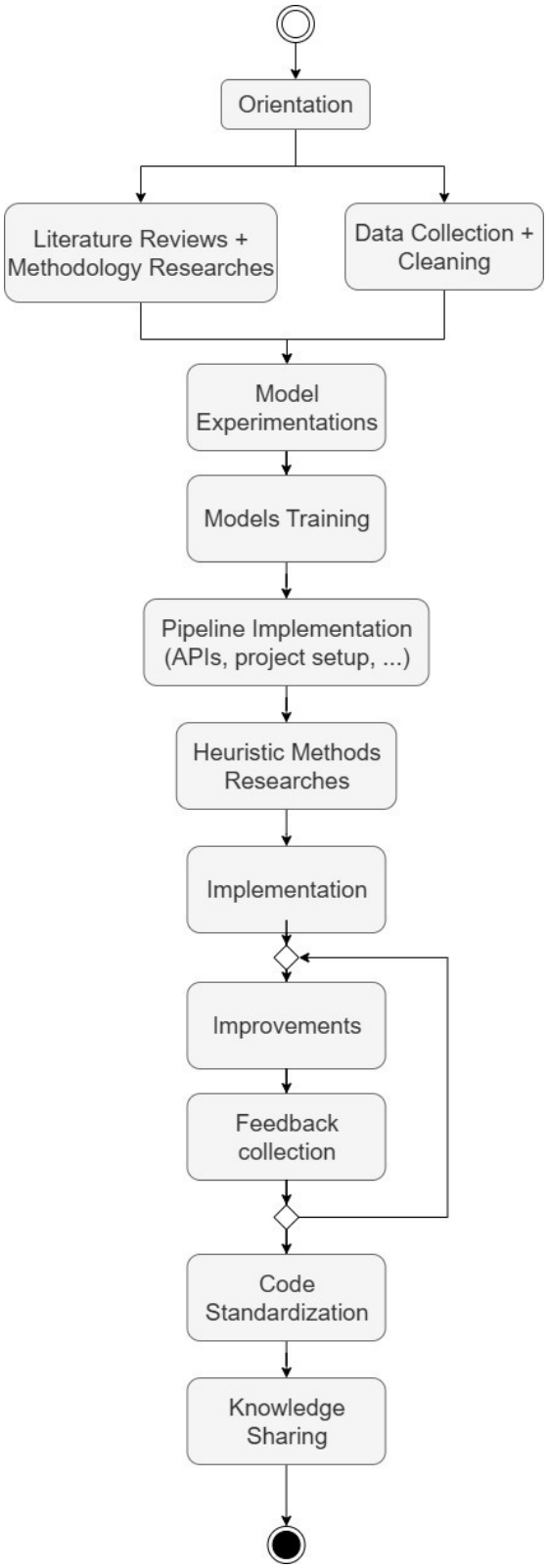


Figure 16 - Works completed flowchart

VII - CONCLUSION

To summarize, this report presented an end-to-end process of building an automatic email routing system from requirement scoping to data collect, to model experimentation, up to finally, deployment. By developing a custom AI models pipeline, the system is now fully capable of reading and understanding email content, analyzing its intent then forwarding them to appropriate recipients. This significantly reduces manual workload, eliminates repetitive tasks, and most importantly, boosts workplace efficiency.

Despite certain limitations, such as occasional misclassification among some emails, the system itself demonstrates robust and reliable performance across different types of emails.

Overall, this project marks a first step in Cosywee's milestone as it is the first homemade (in-house built) AI system that is implemented in the company. With the improvements in the previous section, this system can evolve and be improved into an even more dependable and intelligent solution in the future.

References

Al., D. e. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*

Aspose.Email for Python via .NET. (2024). Retrieved from https://products.aspose.com/email/python-net/

Bay, H., Tuytelaars, T., & Gool, L. V. (2006). SURF: Speeded Up Robust Features.

Beautiful Soup: HTML and XML parser for Python. (2024). Retrieved from https://www.crummy.com/software/BeautifulSoup/

CamemBERT NER - HuggingFace model. (2021). Retrieved from https://huggingface.co/Jean-Baptiste/camembert-ner

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.

Distinctive Image Features from Scale-Invariant Keypoints. (2004). *International Journal of Computer Vision, 60*, 91-110.

Experimentation in Software Engineering - An Introduction. (2012). Kluwer Academic Publishers.

extract_msg: Python library to extract emails from MSG files. (2024). Retrieved from https://github.com/mattgwwalker/msg-extractor

JaidedAI. (2024). *EasyOCR*. Retrieved from https://github.com/JaidedAI/EasyOCR

Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Dinei, F., . . . Wei, F. (2021). TrOCR: Transformer-based OCR with pre-trained Vision and Language Models. Retrieved from https://arxiv.org/abs/2109.10282

libpff/pypff: Library and Python bindings to access Personal Folder File (PST, OST). (2024). Retrieved from https://github.com/libyal/libpff

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach.*

Machine learning for high-speed corner detection. (2006)., (pp. 430-443).

Martin, L., Muller, B., Suàrez, P. J., Dupont, Y., Laurent, R., Clergerie, É. V., . . . Sagot, B. (2020). CamemBERT: a Tasty French Language Model. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

OpenCV: Open Source Computer Vision Library. (2000). *Dr. Dobb's Journal of Software Tools*.

Optuna: A Next-generation Hyperparameter Optimization Framework. (2019)., (pp. 2623-2631).

ORB: An efficient alternative to SIFT or SURF. (2011)., (pp. 2564-2571).

PaddleOCR: Optical Character Recognition tools based on PaddlePaddle. (2024). Retrieved from https://github.com/PaddlePaddle/PaddleOCR

pandas-dev/pandas: Pandas. (2020). *Zenodo*. doi:10.5281/zenodo.3509134

PyMuPDF (fitz): A Python binding for MuPDF PDF and document rendering library. (2024). Retrieved from https://github.com/pymupdf/PyMuPDF

pytesseract: Python wrapper for Google Tesseract-OCR. (2024). Retrieved from https://github.com/madmaze/pytesseract

PyTorch: An Imperative Style, High-Performance Deep Learning Library. (2019). *Advances in Neural Information Processing Systems (NeurIPS), 32*, 8026-8037.

Scikit-learn: Machine Learning in Python. (2011). *Journal of Machine Learning Research, 12*, 2825-2830.

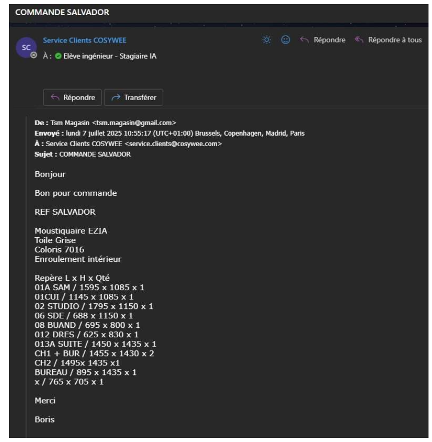
spaCy French Language Model: fr_core_news_lg. (2024). Retrieved from https://spacy.io/models/fr#fr_core_news_lg

Term-weighting approaches in automatic text retrieval. (1988). *Information Processing & Management, 24*, 513-523.

APPENDIX

Types of Emails

Figure 17 - Emails with no attachments, containing only plain text order or price quotation requests



COMMANDE SALVADOR

Service Clients COSYWEE
À : Elève ingénieur - Stagiaire IA

Répondre | Répondre à tous

Répondre | Transférer

De : Tsm Magasin <tsm.magasin@gmail.com>
Envoyé : lundi 7 juillet 2025 10:55:17 (UTC+01:00) Brussels, Copenhagen, Madrid, Paris
À : Service Clients COSYWEE <service.clients@cosywee.com>
Sujet : COMMANDE SALVADOR

Bonjour

Bon pour commande

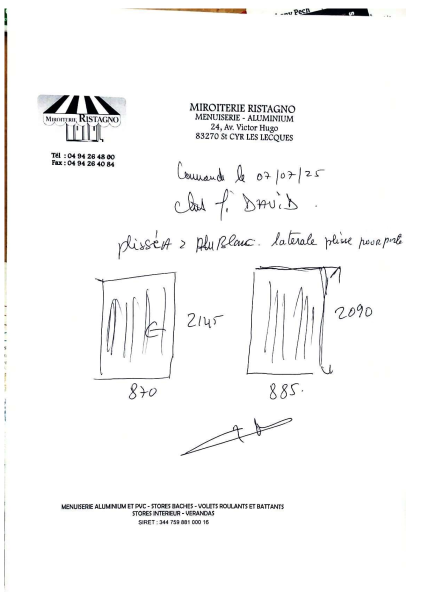
REF SALVADOR

Moustiquaire EZIA
Toile Grise
Coloris 7016
Enroulement intérieur

Repère L x H x Qté
01A SAM / 1595 x 1085 x 1
01CUI / 1145 x 1085 x 1
02 STUDIO / 1795 x 1150 x 1
06 SDE / 688 x 1150 x 1
08 BUAND / 695 x 800 x 1
012 DRES / 625 x 830 x 1
013A SUITE / 1450 x 1435 x 1
CH1 + BUR / 1455 x 1430 x 2
CH2 / 1495x 1435 x1
BUREAU / 895 x 1435 x 1
x / 765 x 705 x 1

Merci

Boris



MIROITERIE RISTAGNO
MENUISERIE - ALUMINIUM
24, Av. Victor Hugo
83270 St CYR LES LECQUES

Tél : 04 94 26 48 00
Fax : 04 94 26 40 84

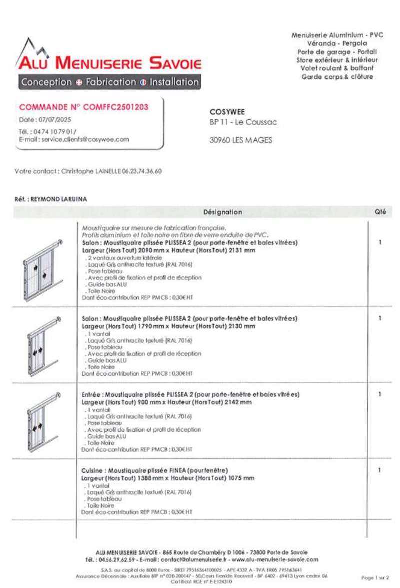
Commande le 07/07/25
plissée 2 Alu Blanc. laterale plisse pour porte

2145 | 2090
870 | 885.

MENUISERIE ALUMINIUM ET PVC - STORES BACHES - VOLETS ROULANTS ET BATTANTS
STORES INTERIEUR - VERANDAS
SIRET : 344 759 881 000 16

Figure 18 - Emails with handwritten attachments indicating order details

Figure 19 - Emails with computer-generated attachments (printed text) indicating order details



ALU MENUISERIE SAVOIE
Conception · Fabrication · Installation

Menuiserie Aluminium - PVC
Véranda - Pergola
Porte de garage - Portail
Store extérieur & intérieur
Volet roulant & battant
Garde corps & clôture

COMMANDE N° COMFFC2501203
Date : 07/07/2025
Tél. : 04 74 10 79 01/
E-mail : service.clients@cosywee.com

COSYWEE
BP 11 - Le Coussac

30960 LES MAGES

Votre contact : Christophe LAINELLE 06.23.74.36.60

Réf. : REYMOND LARUINA

Désignation	Qté
Moustiquaire sur mesure de fabrication française. *Profils aluminium et toile noire en fibre de verre enduite de PVC.* **Salon : Moustiquaire plissée PLISSEA 2 (pour porte-fenêtre et baies vitrées)** **Largeur (Hors Tout) 2090 mm x Hauteur (Hors Tout) 2131 mm** . 2 vantaux ouverture latérale . Laqué Gris anthracite texturé (RAL 7016) . Pose tableau . Avec profil de fixation et profil de réception . Guide bas ALU . Toile Noire Dont éco-contribution REP PMCB : 0,30€ HT	1
Salon : Moustiquaire plissée PLISSEA 2 (pour porte-fenêtre et baies vitrées) **Largeur (Hors Tout) 1790 mm x Hauteur (Hors Tout) 2130 mm** . 1 vantail . Laqué Gris anthracite texturé (RAL 7016) . Pose tableau . Avec profil de fixation et profil de réception . Guide bas ALU . Toile Noire Dont éco-contribution REP PMCB : 0,30€ HT	1
Entrée : Moustiquaire plissée PLISSEA 2 (pour porte-fenêtre et baies vitrées) **Largeur (Hors Tout) 900 mm x Hauteur (Hors Tout) 2142 mm** . 1 vantail . Laqué Gris anthracite texturé (RAL 7016) . Pose tableau . Avec profil de fixation et profil de réception . Guide bas ALU . Toile Noire Dont éco-contribution REP PMCB : 0,30€ HT	1
Cuisine : Moustiquaire plissée FINEA (pour fenêtre) **Largeur (Hors Tout) 1388 mm x Hauteur (Hors Tout) 1075 mm** . 1 vantail . Laqué Gris anthracite texturé (RAL 7016) . Pose tableau . Toile Noire Dont éco-contribution REP PMCB : 0,30€ HT	1

ALU MENUISERIE SAVOIE - 865 Route de Chambéry D 1006 - 73800 Porte de Savoie
Tél. : 04.56.29.62.59 - E-mail : contact@alumenuiserie.fr - www.alu-menuiserie-savoie.com

Page 1 sur 2

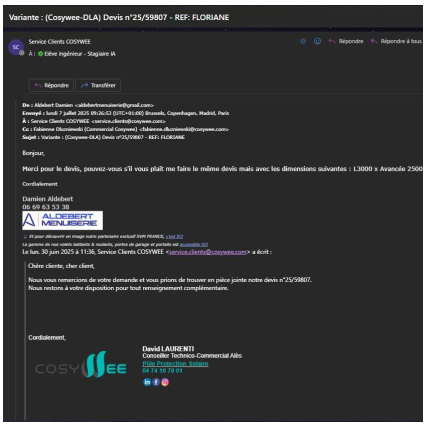


Figure 20 - Emails replying to a Cosywee employee's email

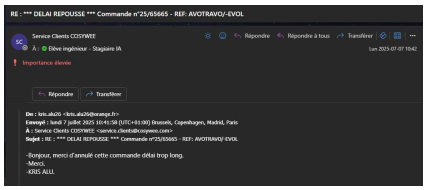


Figure 21 - Emails inquiring about order progress (with only a reference number and no product information)

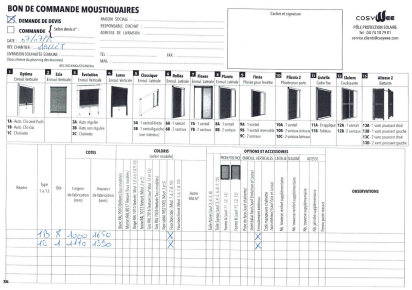
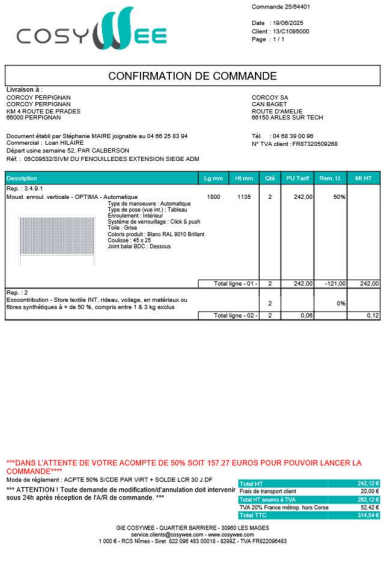


Figure 22 - Order emails with Cosywee's Purchase Order form (Bon de Commande) attached

Figure 23 - Emails containing Cosywee-generated documents as references



Log files

Dossier	Expéditeur	Heure d'envoi	Sujet	Nombre PJ	Temps Trait.Mail	Temps Moyen Trait.PJ	Taille Globale PJs
PERPIGNAN	M	2025-07-16 06:28:02 CEST+0200	SAV RAVENET	0	0:00:09.024931	0:00:09.024931	1
LYON-BOURG	A	2025-07-16 07:02:26 CEST+0200	REF LIAUTARD	2	0:00:20.185670	0:00:10.092835	1633
ALES_Commande	A	2025-07-16 07:05:21 CEST+0200	REF FONTANES	1	0:00:08.884978	0:00:08.884978	659
COMPTA	G	2025-07-16 07:22:47 CEST+0200	RE: (Cosywee-DLA) Devis n°25/60587 - REF: REF/110/MOUSTIQUAIRES		0	0	
ALES	Ju	2025-07-16 07:45:22 CEST+0200	CLAIRBAIE	1	0:00:18.881854	0:00:18.881854	1281
VOLET ROULANT	A	2025-07-16 07:52:38 CEST+0200	SUD FERMETURE / COMMANDE / SOLER	1	0:00:26.511751	0:00:26.511751	771
ALES	a	2025-07-16 07:41:16 CEST+0200	Message from KM_C3350i	1	0:00:15.402368	0:00:15.402368	873
BOURG	A	2025-07-16 08:05:19 CEST+0200	Commande COSYWEE référence LED ET CO	1	0:00:17.382784	0:00:17.382784	666
A CLASSER	C	2025-07-16 08:07:57 CEST+0200	devis ref pontnier	0	0:00:01.106776	0:00:01.106776	0
ALES_Commande	Ju	2025-07-16 08:11:03 CEST+0200	Commande BOREAL VERANDAS /Ref : BARNES	1	0:00:17.776743	0:00:17.776743	1245
ALES_Commande	Ju	2025-07-16 08:12:34 CEST+0200	Commande BOREAL VERANDAS /Ref : BARNES	1	0:00:15.409414	0:00:15.409414	919
A CLASSER	P	2025-07-16 08:17:57 CEST+0200	Re: Commande "KOZSLOW"	0	0:00:01.430997	0:00:01.430997	0
ALES_Commande	a	2025-07-16 08:19:10 CEST+0200	Commande ALPLAST LA VERANDA DE PRESTIGE /	1	0:00:13.307193	0:00:13.307193	783
LYON	S	2025-07-16 08:30:01 CEST+0200	URGENT commande ref BRUSQUET	1	0:00:30.239344	0:00:30.239344	1188
ALES_Devis	S	2025-07-16 08:34:28 CEST+0200	demande de devis	1	0:00:08.351573	0:00:08.351573	76
ALES_Devis	C	2025-07-16 08:35:07 CEST+0200	RE: (Cosywee-DLA) Devis n°25/60298 - REF: POLICE/BANNETTES		0	0	
A CLASSER	b	2025-07-16 08:39:33 CEST+0200	Demande de prix ref MSPDV	1	0:00:12.915505	0:00:12.915505	2
ALES_Devis	S	2025-07-16 08:41:17 CEST+0200	demande de devis	1	0:00:06.579822	0:00:06.579822	76
A CLASSER	b	2025-07-16 08:46:13 CEST+0200	Demande de prix spdv	1	0:00:13.824423	0:00:13.824423	2
A CLASSER	E	2025-07-16 08:48:05 CEST+0200	RE: Message from "COPIEUR-BLACKSUN-PERONNAS"	0	0:00:11.379544	0:00:11.379544	0
A CLASSER	Ju	2025-07-16 08:49:40 CEST+0200	Commande - Cevenn' Menuiserie - Ref : WIEREPANT PASCAL	1	0:00:21.079606	0:00:21.079606	1548
A CLASSER	b	2025-07-16 08:49:36 CEST+0200	Demande magasin spdv	0	0:00:01.678308	0:00:01.678308	0
LYON-BOURG	P	2025-07-16 08:49:49 CEST+0200	RE: Demande de renseignement ref : ICARDI // paire de bras de store MATEST	0	0:00:04.257399	0:00:04.257399	0
ALES_Sav	C	2025-07-16 08:50:24 CEST+0200	Ref Gascon	4	0:00:16.252827	0:00:04.063207	5
ALES_Commande	Jy	2025-07-16 08:51:34 CEST+0200	Réclamation – suite (Commande réf. CLEMENCEAU)	1	0:00:22.564950	0:00:22.564950	1665
A CLASSER	E	2025-07-16 09:01:12 CEST+0200	Nouvelle commande	1	0:00:24.261203	0:00:24.261203	1020
LYON	Ju	2025-07-16 09:06:28 CEST+0200	TR: Commande ref DESROCHES	1	0:00:27.895424	0:00:27.895424	1363
ALES_Commande	s	2025-07-16 09:10:37 CEST+0200	MENUISERIE FEBVRE - COMMANDE REF.LIEUDENOT	1	0:00:21.953856	0:00:21.953856	948
LYON-BOURG	jy	2025-07-16 09:16:16 CEST+0200	TR: Commande IPM MENUISERIE /Ref : JACOB	1	0:00:16.020409	0:00:16.020409	1176
A CLASSER	A	2025-07-16 09:19:04 CEST+0200	RE: (Cosywee-LSU) Devis n°25/60582 - REF: GRAZIANI	0	0:00:01.848142	0:00:01.848142	0
ALES_Commande	d	2025-07-16 09:20:10 CEST+0200	Commande Mireille.	0	0:00:01.596177	0:00:01.596177	0
PERPIGNAN	N	2025-07-16 09:26:44 CEST+0200	Re: (Cosywee-JPU) Devis n°25/59493 - REF: TRON		0	0	
PERPIGNAN	JC	2025-07-16 09:33:10 CEST+0200	devis ref chanteperdrix	0	0:00:03.229527	0:00:03.229527	0

Figure 24 - Time Records Log: Logs the displacement or movement of emails, which is useful for debugging cases where emails appear to be missing.

Objet	prediction CAMEMBERT	Score de confiance CAMEMBERT	prediction Random Forest	Score de confiance Random Forest	Recherche de mots-clÃ©s VR ?
REF LIAUTARD	1	0.947748601436615	1	0.8941176470588236	False
REF FONTANES	0	0.9848595261573792	0	1.0	False
CLAIRBAIE	0	0.9849482178688049	0	1.0	False
SUD FERMETURE / COMMANDE / SOLER	4	0.9588184952735901	4	0.8746957676195767	True
devis ref pontnier	0	0.6168900728225708	1	0.6497152520681931	False
Commande BOREAL VERANDAS /Ref : BARNES	0	0.9850524067878723	0	0.9764705882352941	False
Commande BOREAL VERANDAS /Ref : BARNES	0	0.9849891662597656	0	0.9647058823529412	False
Re: Commande "KOZSLOW"	1	0.9459795355796814	2	0.6406262921258328	False
Commande ALPLAST LA VERANDA DE PRESTIGE /	0	0.9849154949188232	0	1.0	False
demande de devis	0	0.9830151796340942	0	0.6767787114845938	False
Demande de prix ref MSPDV	4	0.9818257093429965	4	0.6406262921258328	False
demande de devis	0	0.9833956099664917	0	0.6767787114845938	False
Demande de prix spdv	2	0.9841018319129944	0	0.6367647058823529	False
RE: Message from "COPIEUR-BLACKSUN-PERONNAS"	1	0.9852700825950592	2	0.4641119440149236	False
Commande - Cevenn' Menuiserie - Ref : WIEREPANT PASCAL	0	0.9779177904129028	2	0.7529411764705882	False
Demande magasin spdv	2	0.8942098021507263	0	0.4940680320711284	False
RE: Demande de renseignement ref : ICARDI // paire de bras de store MATEST	1	0.9859731197357178	1	0.6497152520681931	False
RÃ©clamation â€“ suite (Commande réf. CLEMENCEAU)	0	0.9834582805633545	0	0.5191176470588236	False
Nouvelle commande	0	0.6911259293556213	1	0.8470588235294118	False
TR: Commande IPM MENUISERIE /Ref : JACOB	1	0.9872497916221619	1	0.8235294117647058	False
RE: (Cosywee-LSU) Devis nÂ°25/60582 - REF: GRAZIANI	4	0.9274955539188385	4	0.6406262921258328	False
Commande Mireille.	0	0.9817107915878296	1	0.5273724958174609	False
devis ref chanteperdrix	3	0.9267887215614319	3	0.7522971781305113	False
COMMANDE 25/63943 REF FERRAZZI	1	0.9819104075431824	1	0.5176470588235295	False
Commande PVI/LANGUEDOC FERMETURES /	0	0.9850181937217712	0	0.9764705882352941	False
Re: (Cosywee-DLA) Devis nÂ°25/60646 - REF: 16/07/25/bras banne	0	0.9837084125574768	0	0.7803729046009706	False
Commande rÃ©fÃ©rence "FELTZINGER"	1	0.9873298406600952	1	0.8352941176470589	False
Profil de cariste expÃ©rimentÃ©e	4	0.9535524845123291	4	0.5869474461534155	False
Demande de prix : Gymnase FrÃ©dÃ©ric Mistral Â St Maurice l'Exil	1	0.986003577709198	1	0.9294117647058824	False
GOTTSCHALK	2	0.5039052367210388	2	0.4647058823529412	False
commande mairie	0	0.9842018485069275	0	0.9411764705882353	False
RE: OUVERTURE DE COMPTE	2	0.8602700233459473	2	0.7064377815358208	False

Figure 25 - Prediction Log: Stores predictions and confidence scores from both the CamemBERT model and the Random Forest model.)

label	senders	subjects	body	attachments
A CLASSER	En	Demande de devis REF : ALVAREZ		wod he aa po yetp
ALES_Commande	ms	Commande RÃ©f. LIEB	ci joint commande rÃ©f. lieb dans l'attente de votre a ccusÃ© de rÃ©ception, c	prix achat mages bp zac madeleine rÃ©sidence clos mail fr commande affaire
MARSEILLE	co	commande rÃ©fÃ©rence calmettes.	vous trouverez en piÃ¨ce jointe notre commande rÃ©fÃ©rence calmettes.	fabricant installateur vÃ©randas volets moustiquaires stores bis vn stores extÃ
ALES_Commande	Jos	a Commande - Cevenn' Menuiserie - Ref : VITA NOVA	veuillez trouver ci-joint notre commande rÃ©fÃ©rence : vita nova livraison sem	contact cm tÃ©l fixe mail commandes menuiserie fr bp mages route st ambroix
MARSEILLE	co	demande de prix rÃ©fÃ©rence Campain.	vous trouverez en piÃ¨ce jointe notre demande de prix rÃ©fÃ©rence campain.	fabricant installateur ot ts cosy vÃ©randas volets moustiquaires stores stores e
A CLASSER	GA	RECLAMATION	au chef de production merci de vÃ©rifier la prÃ©paration des commandes : l'e	
A CLASSER	ser	devis	veuillez trouver ci-joint notre demande de prix ; ref parvaud 1 bÃ¢che pour fermeture de garage - ral 1015 de 3300 mm de large x 1900 mm de haut qui puisse Ã
COMPTA	Co	RE: FW: 251571004791 - LECA - SAV Sud		L'objectif de cet email est RE: FW: 251571004791 - LECA - SAV Sud, suite au m
ALES_Devis	std	demande de prix Ref Buzi	merci de me chiffrer : moustiquaire listelle blanche toile grise - 1 en l 1375 h 1220 - 1 en l1430 h 1230 - 1 l 1325 h 1230 - 1 l 1030 h 1195 - 1 l 1405 h 1205-est il	
ALES_Commande	Co	Commande rÃ©f DELMAS	merci de prendre en compte la commande ci-jointe et de m'adresser un arc.	bon commande moustiquaires raison sociale responsable achat demande dev
A CLASSER	Me	taise@orange.fr>		menuiserie route nationale fax fr expediteur albert blanchard destinataire ms c
PERPIGNAN	Mi	Re: Ouverture de compte	vous venez de me contacter par email pour la premiÃ¨re fois. ma messagerie est protÃ©gÃ©e par la solution protect de mailinblack pour faire face aux cyberatt	
ALES_Commande	Da	Commande DROME OUVERTURES /	veuillez trouver ci-joint ma commande, merci de m'envoyer votre a ccusÃ© de	prix achat mages bp rte fax mail administratif fr commande nÂ° date ouvertures
MARSEILLE	jm	du devis volets battants ref PASTOR JMH	votre message est prÃªt Ã Ãªtre envoyÃ© avec les fichiers ou liens joints suivant	concept rue courriel com concepteur siret intracommunautaire demande be p
LYON-BOURG	S.f	relance commande ref SAV-DUARTE-04109	sauf erreur nous sommes sans retour d'arc suite Ã notre commande ci-dessou	go poke br adv com envoyÃ© vendredi objet cre suivi dossier sav get indicateur
LYON	PH	Commande "KOZSLOW"	ci-joint une nouvelle commande.	bon commande toile acrylique demande devis devis responsable achat ee hÃ©
PERPIGNAN	Gu	Re: (Cosywee-CPO) Commande nÂ°25/64233 - REF: PARKS	merci de trouver ci joint l'arc rÃ©fÃ©rence parks modifiÃ© et validÃ©.	of sou zo she fel sou sine commande date client page confirmation jam ouver
ALES_Commande	De	CDE 10 BSOROSEL 5862 / MSTQ	ci-joint nouvelle commande.	zac rue tÃ©l bs fax email com internet www commande achat nÂ° dte ven ee m
ALES_Commande	De	CDE 10 MARTINEZ / RUBAN	bon pour a ccord ci-joint, merci de mettre pour rÃ©fÃ©rence : 10 martinez.	devis date client page zac rue document Ã©tabli joignable tÃ©l commercial ph
ALES_Commande	rou	commande	veuillez trouver en piÃ¨ce jointe notre commande rÃ©fÃ©rence : carrasco meill	bon commande moustiquaires cachet signature cosy raison sociale dard dema
ALES_Sav	DL	TR: SAV Commande 25/57185	veuillez trouver ci-joint une demande de sav concernant deux moustiquaires su	commande date client page cosy bois activite rue document Ã©tabli stephanie
ALES_Commande	Na	BON DE COMMANDE - STORE BANNE - FONTANIL	trouvez ci-joint notre commande pour suite Ã donner.	nÂ° u. total ht designation adresse livraison rue fontaine bon accord devis com
LYON-BOURG	Mi	Commande MENUISERIE SIMON /Ref : GUYON	veuillez trouver ci-joint ma commande. merci de m'envoyer votre a ccusÃ© de	contact m. tÃ©l mail menuiserie fr bp mages za chemin bois date commande n
ALES_Commande	de	PEREZ	merci de trouver ci-joint une commande rÃ©fÃ©rence : number of images: 1 att	Ã©tres ro altesse coulissante battante vantail jia vantaux ouv vantaux vantail ga
PERPIGNAN	Fra	COMMANDE LACROIX		commande fermetures plasti fermetures rue joseph nÂ° date fermetures fax m
ALES_Commande	PH	Commande REF BRICOLA - V	veuillez trouver ci-joint une nouvelle commande.	bon commande stores bannes raison sociale responsable achat adresse livrai
LYON-BOURG	Arr	Devis Toile Store	pouvez vous me faire un devis pour : toile de store banne modÃ¨le orchestra ar	cp zz oo ym ca cafe co
COMPTA	Co	RE: (Cosywee-MCE) Commande nÂ°25/63831 - REF: NÂ° 2799		L'objectif de cet email est RE: (Cosywee-MCE) Commande nÂ°25/63831 - REF:
LYON	Mi	TR: PIECE JOINTE Ref Mullier/20	, bon pour commande . merci	demande devis commande sen devis raison bon toile acrylique responsable ac
ALES_Devis	Ld	Demande de prix	nous ne trouvons pas le prix des stores micro-perforÃ© dans votre tarif. merci de me communiquer le prix des stores suivants : - 1 x l 1 440 x 1 615 ht vÃ©	
A CLASSER	Nic	RÃ©ponse automatiqueÂ : (Cosywee-SMA) Commande nÂ°25/64625 - REF:	je vous remercie de communiquer vos demandes Ã l'email suivant :	
A CLASSER	Jul	Livraison commandes en cours	nous attendons plusieurs commandes et avions planifiÃ© les pose suivant les arc fournis. pourriez vous faire un point et nous indiquer les dates des livraisons d	
ALES_Commande	Co	Commande	vous trouverez en piÃ¨ce jointe la commande pour la rÃ©fÃ©rence lamalle mul	demande raison sociale aud protection responsable achat commande devis ac
A CLASSER	co	RE: (Cosywee-CPO) Commande nÂ°25/62909 - REF: SOLNAIS	je n'ai toujours pas de nouvelles pour cette commande, le client me demande le délai. merci à vous par avance pour votre retour.	
ALES_Commande	Contact commande Pro	Commande	vous trouverez en piÃ¨ce jointe la commande pour la rÃ©fÃ©rence dubard o.	raison sociale co protection devis responsable ie cite sett aa commande adresse livrais mu

Figure 26 - Complete Data Log: Stores all extracted data along with their predictions. This log is useful for fine-tuning new models in the future.