



Ottomatic Reply Email Concierge

Group 2 – Hackathon 2025

Automating first-line email triage
so agents spend time on
solutions, not sorting.

The screenshot shows a software application window titled "otto group". On the left, a sidebar displays "Please select the Support Team" with a dropdown menu set to "All" and "Number of unanswered Tickets: 50". The main area is titled "Welcome to Ottomatic Reply" with the sub-headline "Here's where great service starts.". It includes a brief description: "This internal support space is designed to help our team assist customers quickly, confidently, and in true Otto style. We're efficient, empathetic, and always on-brand — just like the service our customers expect. Let's make support smarter, together." Below this is a "Load data" button. The bottom half of the screen is a table listing 20 customer tickets with columns for Redacted Email, Support Team, Sentiment, Urgency, and Draft Reply. The "Sentiment" column uses color-coded boxes (Neutral: light blue, Happy: green, Unhappy: orange, Very unhappy: red) to indicate customer satisfaction levels.

Redacted Email	Support Team	Sentiment	Urgency	Draft Reply
... Subject: Compatibility of XDR-5...	Product Consultation	Neutral	2	Subject: Re: Compatibility...
... Subject: Excited to buy ProMax ...	Product Consultation	Happy	2	Subject: Re: Excited to b...
... Subject: Confused between Pro...	Product Consultation	Unhappy	3	Subject: Re: Confused be...
... Subject: Why isn't TX-100 work...	Product Consultation	Very unhappy	4	Subject: Regarding your...
... Subject: Absolute newbie—Hom...	Product Consultation	Neutral	2	Subject: Re: Absolute ne...
... Subject: Unable to select Ameri...	Order Support	Unhappy	3	Subject: Re: Unable to se...
... Subject: Missing express shippi...	Order Support	Very unhappy	4	Subject: Regarding Expr...
... Subject: Bug in cart: quantity not...	Technical Assistance	Very unhappy	4	Subject: Regarding Your...
... Subject: Newsletter promo code...	Loyalty Programs and Discounts	Neutral	1	Subject: Re: Newsletter p...
... Subject: International shipping t...	Product Consultation	Neutral	2	Subject: Re: Internationa...
... Subject: Installation error 0x800...	Technical Assistance	Unhappy	3	Subject: Re: Installation...
... Subject: Firmware update stops ...	Technical Assistance	Very unhappy	4	Subject: Re: Firmware up...
... Subject: TrailCam 200 won't po...	Technical Assistance	Very unhappy	4	Subject: Re: TrailCam 20...
... Subject: Bluetooth pairing succe...	Technical Assistance	Neutral	2	Subject: Re: Bluetooth pa...
... Subject: Sonoma Beta compatib...	Technical Assistance	Unhappy	3	Subject: Regarding Soun...
... Subject: Order #34567 stuck at ...	Shipping and Delivery Updates	Unhappy	4	Subject: Regarding Your...
... Subject: Tracking link 404—fix it...	Shipping and Delivery Updates	Very unhappy	5	Subject: Regarding Your...
... Subject: Partial delivery for Orde...	Shipping and Delivery Updates	Unhappy	4	Subject: Re: Partial deliv...

The Team



Nicola Dale
Stellenbosch University



Alexander Mitte
TU Dresden



Sean Kiley
Stellenbosch University



Kevin Hoang
TU Dresden

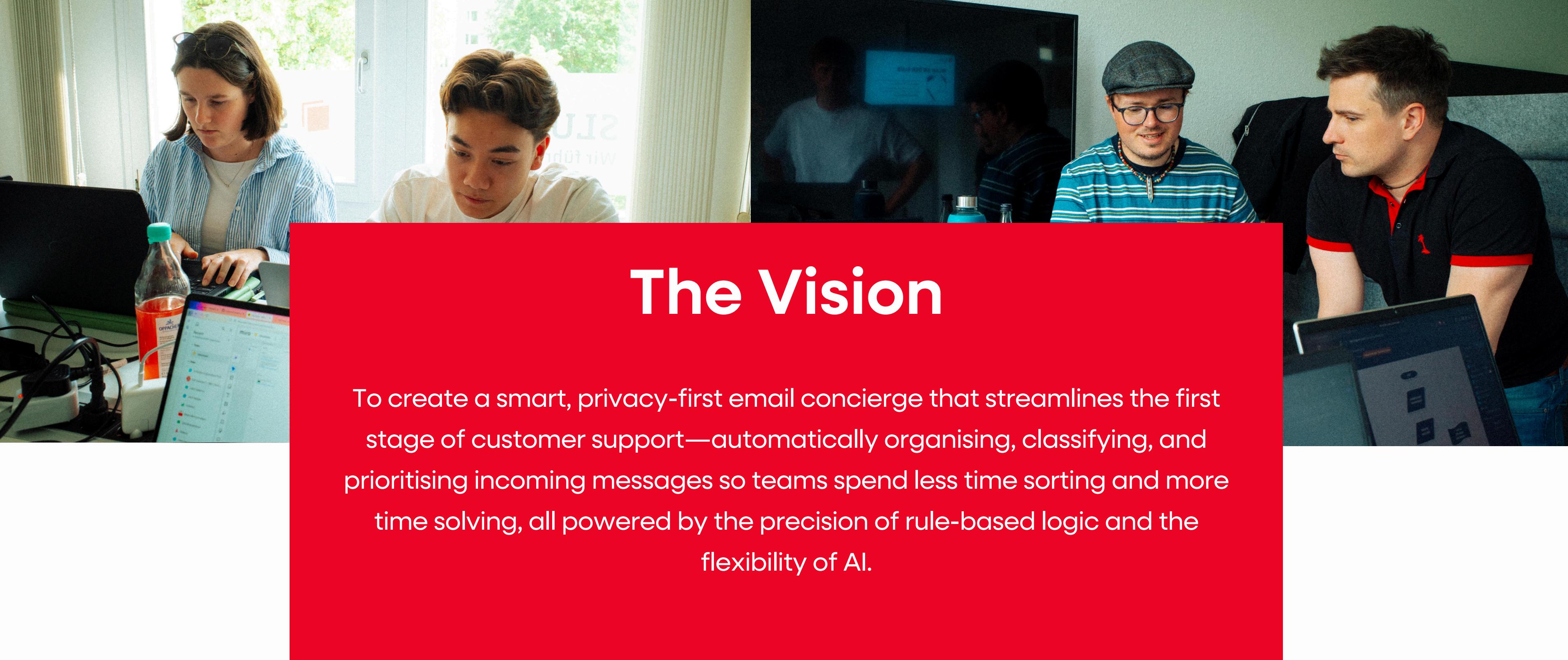


Jordan Luck
Stellenbosch University

Overview

- ▶ Our Vision 01
- ▶ Solution Design 02
- ▶ Project Plan 03
- ▶ Architecture Diagram 04
- ▶ Quality Assurance 05
- ▶ Dashboard 06
- ▶ The Team 07





The Vision

To create a smart, privacy-first email concierge that streamlines the first stage of customer support—automatically organising, classifying, and prioritising incoming messages so teams spend less time sorting and more time solving, all powered by the precision of rule-based logic and the flexibility of AI.

Solution Design

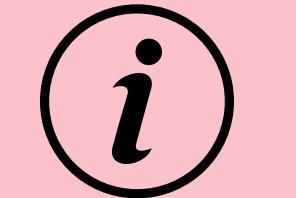
Our solution design leverages natural language processing to automate and enhance email support through five core processes: classifying emails by topic, masking personal information for privacy, detecting sentiment to guide tone, identifying intent to inform response selection, and scoring urgency to prioritise action. Together, these components form an intelligent concierge system that ensures efficient, accurate, and human-aligned responses to customer emails.

Problem 01



Classifying
emails by topic

Problem 02



Masking personal
information

Problem 03



Recognising
sentiment

Problem 04



Recognising
sentiment

Problem 05



Scoring urgency



Project Plan

```
# requirement already satisfied: sniffio<1.1 in /opt/conda/lib/python3.10/site-packages (from aiohttp>=3.0.3->language<2.0.0->langsmith)
# Report libraries
from google import genai
from google.cloud import storage
from google.cloud import language_v2
from google.genai import types
from google.genai.types import HttpOptions
from langsmith import traceable
from langsmith.run_helpers import get_current_run_tree
from typing import List
import re, spacy
import datetime
import os
import json
import uuid

# Sample customer-service e-mails with tags
#Training Data
#[{"Product Consultation / Tone: Neutral | Urgency: low"]
email1 = """Subject: Question about the Nebula-X Smart Projector before I buy
Hi team,
I'm Maria-Luisa Ortega and I'm really excited about the Nebula-X projector! A quick check before I place my order:"""

#["Product Consultation / Tone: Neutral | Urgency: low"]
email2 = """Subject: Missing prepaid label for JoyRide Bike
Key Support, initiated a return for JoyRide E1...nCheers, (alpha
support_team) Returns and Exchanges Management, 'urgency': 'low'
d future request for <coroutine test_ottomatic with_csv_data[email_data2]>
est_mark.asyncio
est_mark.parametrize('email_data', load_emails_from_csv(CSV_PATH))
def test_ottomatic_with_csv_data(email_data, request):
    test_fields = request.config.getoption('--test-fields').split(',')
    result = await ottomatic(email_data['email_text'], "test@example.com")
    if "support_team" in test_fields or "all" in test_fields:
        assert result['support_team'].lower() == email_data['support_team'].lower()
    if "sentiment" in test_fields or "all" in test_fields:
        assert result['sentiment_category'].lower() == email_data['sentiment'].lower()
        #assert result['sentiment'].lower() == 'neutral'
        #AssertionError: assert 'unhappy' == 'neutral'
        # + neutral
        # + unhappy
    #33: AssertionError
    #----- test_ottomatic_with_csv_data[email_data3] -----
email3 = ('email_text': 'Subject: Damaged mugs-demanding immediate exchange!\nWith,\nThe Premium Ceramic Mug Set (Order #2345...)\nMichael Becker', support_team: 'Returns and Exchanges Management', 'urgency': 'high')
d future request for <coroutine test_ottomatic_with_csv_data[email_data4]>
est_mark.asyncio
est_mark.parametrize('email_data', load_emails_from_csv(CSV_PATH))
def test_ottomatic_with_csv_data(email_data, request):
    test_fields = request.config.getoption('--test-fields').split(',')"""

```

Tuesday

Getting emails into Notebook

Creating basic test data

Categorising each email by Support Team

Built simple draft reply based on each email

Sentiment Analysis: Output sentiment category & scores

```
# language_v2.LanguageServiceClient()
client = language_v2.LanguageServiceClient()

content = {
    "content": email_text,
    "type_": language_v2.Document.Type.PLAIN_TEXT,
}

encoding_type = language_v2.EncodingType.UTF8

response = client.analyze_sentiment(
    request={"document": document, "encoding_type": encoding_type}
)

magnitude = response.document.sentiment.magnitude
score = response.document.sentiment.score

internet_score(score: float) -> str:
    if score < -0.35:
        return "Very unhappy"
    elif score < -0.1:
        return "Unhappy"
    elif score < 0.25:
        return "Neutral"
    elif score < 0.75:
        return "Happy"
    else:
        return "Very Happy"

#----- test_ottomatic_with_csv_data[email_data3] -----
email3 = ('email_text': 'Subject: Damaged mugs-demanding immediate exchange!\nWith,\nThe Premium Ceramic Mug Set (Order #2345...)\nMichael Becker', support_team: 'Returns and Exchanges Management', 'urgency': 'high')
d future request for <coroutine test_ottomatic_with_csv_data[email_data4]>
est_mark.asyncio
est_mark.parametrize('email_data', load_emails_from_csv(CSV_PATH))
def test_ottomatic_with_csv_data(email_data, request):
    test_fields = request.config.getoption('--test-fields').split(',')"""

```

Wednesday

Basic PII Reduction

Enhance PII reduction to catch all relevant data

Get urgency from Sentiment Analysis and Support Team Category

Add Tracing with Langsmith to follow

Define output format for different functions

```
#----- test_ottomatic_with_csv_data[email_data3] -----
email3 = ('email_text': 'Subject: Missing prepaid label for JoyRide Bike\nKey Support, initiated a return for JoyRide E1...nCheers, (alpha
support_team) Returns and Exchanges Management, 'urgency': 'low'
d future request for <coroutine test_ottomatic_with_csv_data[email_data2]>
est_mark.asyncio
est_mark.parametrize('email_data', load_emails_from_csv(CSV_PATH))
def test_ottomatic_with_csv_data(email_data, request):
    test_fields = request.config.getoption('--test-fields').split(',')
    result = await ottomatic(email_data['email_text'], "test@example.com")
    if "support_team" in test_fields or "all" in test_fields:
        assert result['support_team'].lower() == email_data['support_team'].lower()
    if "sentiment" in test_fields or "all" in test_fields:
        assert result['sentiment_category'].lower() == email_data['sentiment'].lower()
        #assert result['sentiment'].lower() == 'neutral'
        #AssertionError: assert 'unhappy' == 'neutral'
        # + neutral
        # + unhappy
    #33: AssertionError
    #----- test_ottomatic_with_csv_data[email_data3] -----
email3 = ('email_text': 'Subject: Damaged mugs-demanding immediate exchange!\nWith,\nThe Premium Ceramic Mug Set (Order #2345...)\nMichael Becker', support_team: 'Returns and Exchanges Management', 'urgency': 'high')
d future request for <coroutine test_ottomatic_with_csv_data[email_data4]>
est_mark.asyncio
est_mark.parametrize('email_data', load_emails_from_csv(CSV_PATH))
def test_ottomatic_with_csv_data(email_data, request):
    test_fields = request.config.getoption('--test-fields').split(',')"""

```

Thursday

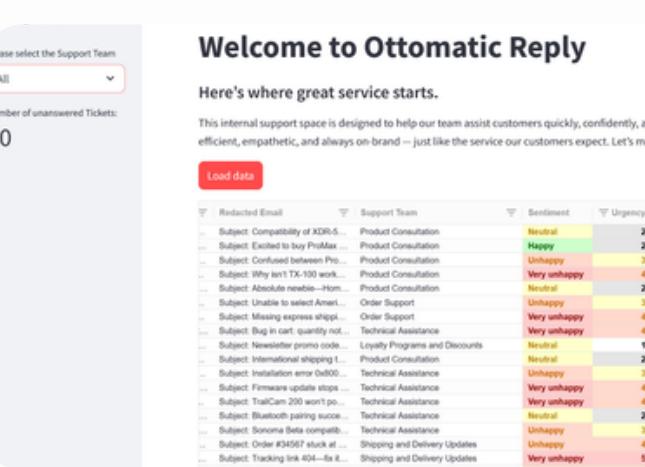
Creating more test data

Creating automatic Unit Testing and scores to understand how good the pipeline is

Enhance dashboard with additional filters, formatting...

Insert retrieved file from cloud storage into Frontend

Craft basic Frontend to show the data



Welcome to Ottomatic Reply

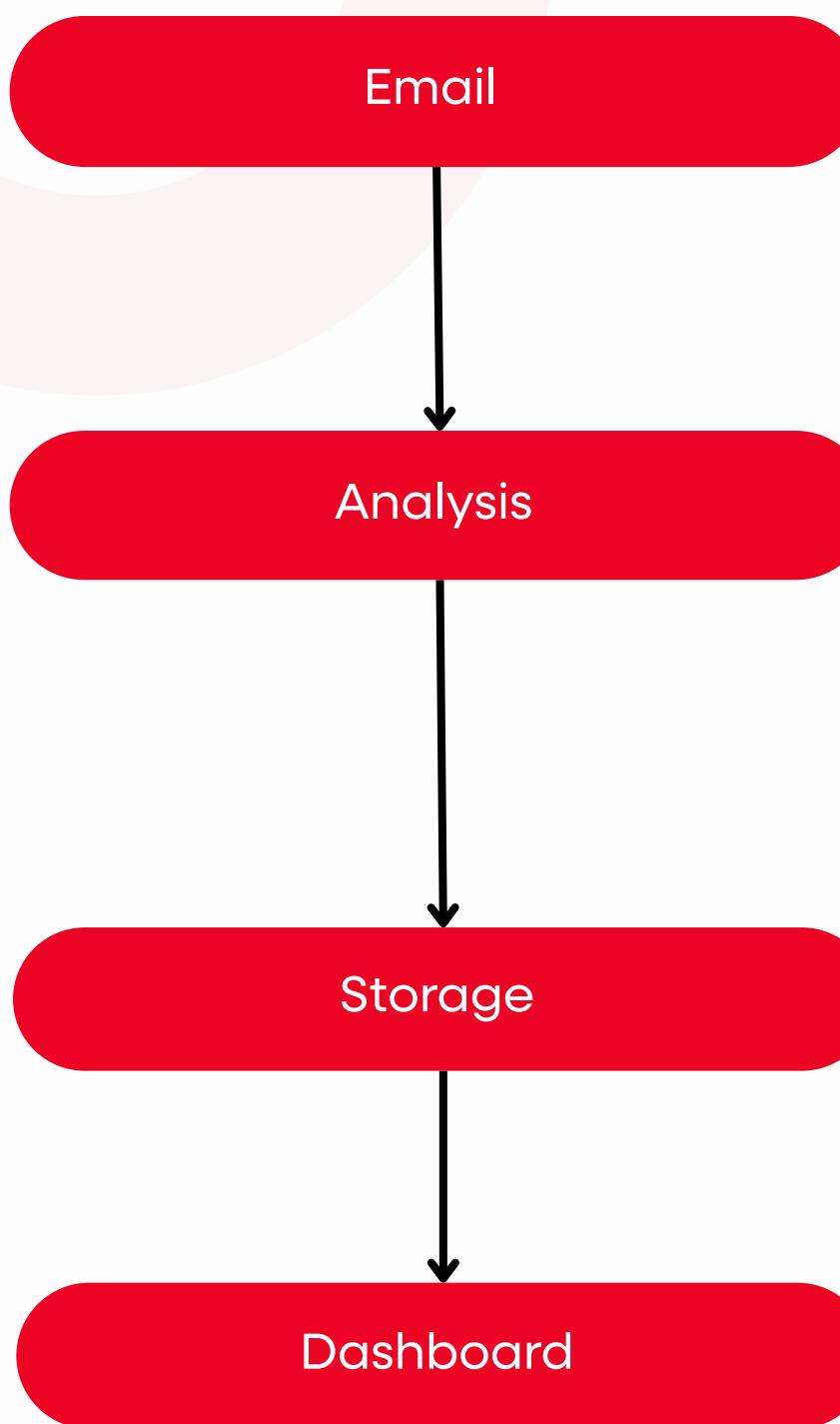
Here's where great service starts.

This internal support space is designed to help our team assist customers quickly, confidently, and in true Otto style. We keep it efficient, empathetic, and always on-brand -- just like the service our customers expect. Let's make support smarter, together.

Friday

Presentation

Architecture Diagram



1. Email (Input Source)

- Source Data: Uses 60 generated emails from ChatGPT for testing.
- Nature: Raw, semi-structured data with varying content, tone, and urgency (focus on unstructured body).

2. Analysis (Intelligent Processing)

- Core intelligence: AI/ML models process email content.
 - PII Redaction: Masks sensitive personal information using Regex AND spaCy's Named Entity Recognition (NER).
 - Classification (Gemini): Categorises emails into 10 specific support groups.
 - Sentiment Analysis (Google Cloud NL API): Determines emotional tone and magnitude.
 - Urgency Scoring: Assigns a priority score from Level 1 (highest) to Level 5 (lowest).
 - This is calculated based on a combination of the email's classified category and its sentiment.
 - Draft Reply Generation (Gemini): Creates brand-aligned, PII-free preliminary response templates.

3. Storage (Data Persistence)

- Saves all original and analysed email data for persistence and access.
 - Google Cloud Storage (GCS): Comprehensive analysed email data is converted to a JSON file and uploaded to a GCS bucket, serving as the central archive for all processed insights.

4. Dashboard (Streamlit UI)

- Interface: Built using Streamlit, serving as the intuitive front-end for human agents.
- Functionality:
 - Displays a clear table showing each email's: Timestamp, address, original email, redacted email, support team category, sentiment, urgency, and draft reply.
 - Enables quick understanding, prioritisation, and efficient review/editing of responses.



Leveraging LangSmith



We use LangSmith for enhanced development and monitoring of our AI pipeline.

How We Use LangSmith (Our Observability Platform):

- **Tracing:** Visualises the data flow and execution path through our email processing pipeline.
- **Debugging:** Helps identify and resolve issues within AI calls and data transformations.
- **Monitoring:** Tracks the performance of LLM interactions and overall pipeline for quality assurance.

In short, LangSmith enables us to build, debug, and maintain our AI-driven E-Mail Concierge system effectively.

The screenshot displays the LangSmith platform's interface for monitoring AI pipeline runs. On the left, a sidebar titled "Project-Southafrica" shows a list of "Runs" under the "TRACE" tab. One run, "ottomation 2.66s", is highlighted and expanded. The main panel on the right provides detailed information about this run, including its ID, start and end times, status, total tokens, latency, and type. The "Input" section shows the original email text and subject, which discusses a suggestion for sustainable packaging. The "Output" section shows the generated draft reply template. The entire interface is designed for developers to quickly identify and troubleshoot AI model behavior.

Testing & Storage

1. Testing Data

- **Source:** 60 unique customer service emails generated by ChatGPT.
- **Use:** Rigorous testing and refinement of the GEMINI AI models.

2. Storage (Google Cloud Storage - GCS)

- **Method:** All comprehensive analysed email data is converted into a single JSON file.
- **Location:** Uploaded to a dedicated Google Cloud Storage (GCS) bucket.
- **Role:** Serves as the primary, scalable, and durable archival repository for all processed insights.

Masking Personal Information

To safely use customer emails with LLMs, we implemented a robust PII redaction system. This ensures sensitive data is removed before processing, protecting user privacy and meeting company compliance standards.



Objective 01

We use regex patterns to automatically detect and redact emails, phone numbers, card details, and order IDs, replacing them with “[REDACTED]” to protect user privacy.



Objective 02

Using spaCy’s NER model, we identify and redact names, locations, and other sensitive entities that regex alone might miss.



Objective 03

Redaction happens in two steps—regex first, then NER—processed in reverse to ensure clean, accurate masking without breaking the text structure.

- ① What was new?
What was surprising?
- ② What was good?
What should we continue?
- ③ What can be improved?
What can be changed?

After Lunch

plan
data into storage
data from storage
Output
board

is:
- refine PII
- draft email prompt
- documentation, comments
test data & testing score

w:- presentation, architecture,

Classification



Support Teams

We fine-tuned a system prompt to classify emails into one of ten support categories, covering everything from presales inquiries to post-purchase complaints. This ensures accurate routing and faster response times.



Gemini Prediction

The Gemini 2.0 Flash model is used to process each incoming email. It applies the classification logic asynchronously using Google Vertex AI, ensuring scalability and low-latency predictions.



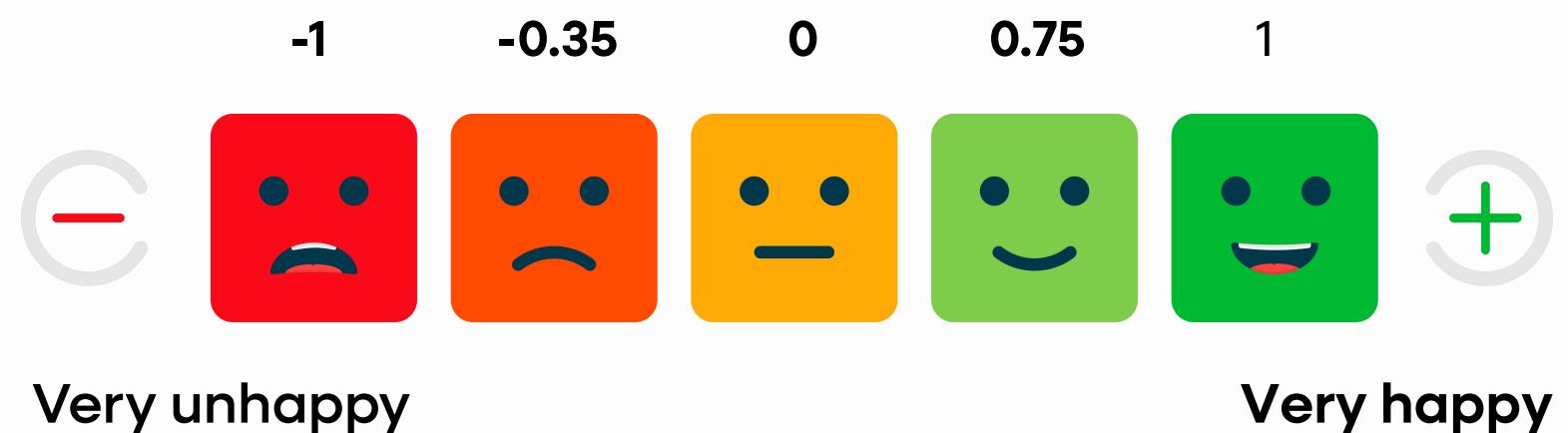
Data Cleaning

Every response is stripped of noise and returned as a clean label, allowing for real-time analytics, prioritisation, and dashboard integration across support teams.

Recognising Sentiment

We used a ML model to assign a sentiment score and magnitude ranging from very unhappy to very happy.

We then categorize these scores into five distinct intervals: very unhappy, unhappy, neutral, happy, or very happy based on generated test data.



Learnings:

Hard to define intervals without properly validated test data, risk of overfitting

Customers often only contact customer Service when they are unhappy (motivation)



Scoring Urgency

Each email is scored out of 5 based on a combination of its category and sentiment:

- Category weighting:
 - Aftersales (e.g., Shipping, Returns) → +3
 - Presales/General Support → +2
 - Feedback/Promotions → +1
- Sentiment modifier:
 - "Very unhappy" → +2
 - "Unhappy" → +1
 - Neutral/Happy → +0

→ Final Score = Category + Sentiment

Scores range from 1 (low urgency) to 5 (critical priority), helping the system route emails efficiently and fairly.

01

Not urgent

02

We first explored using an LLM to grade urgency, but found the results to be inconsistent and somewhat random. We transitioned to a more reliable rule-based system.

03

We discovered that not all problems within a given support category share the same level of urgency. This suggests a need for more detailed categorization.

04

Our current model inherently prioritizes existing customers through the assignment of urgency based on the support team.

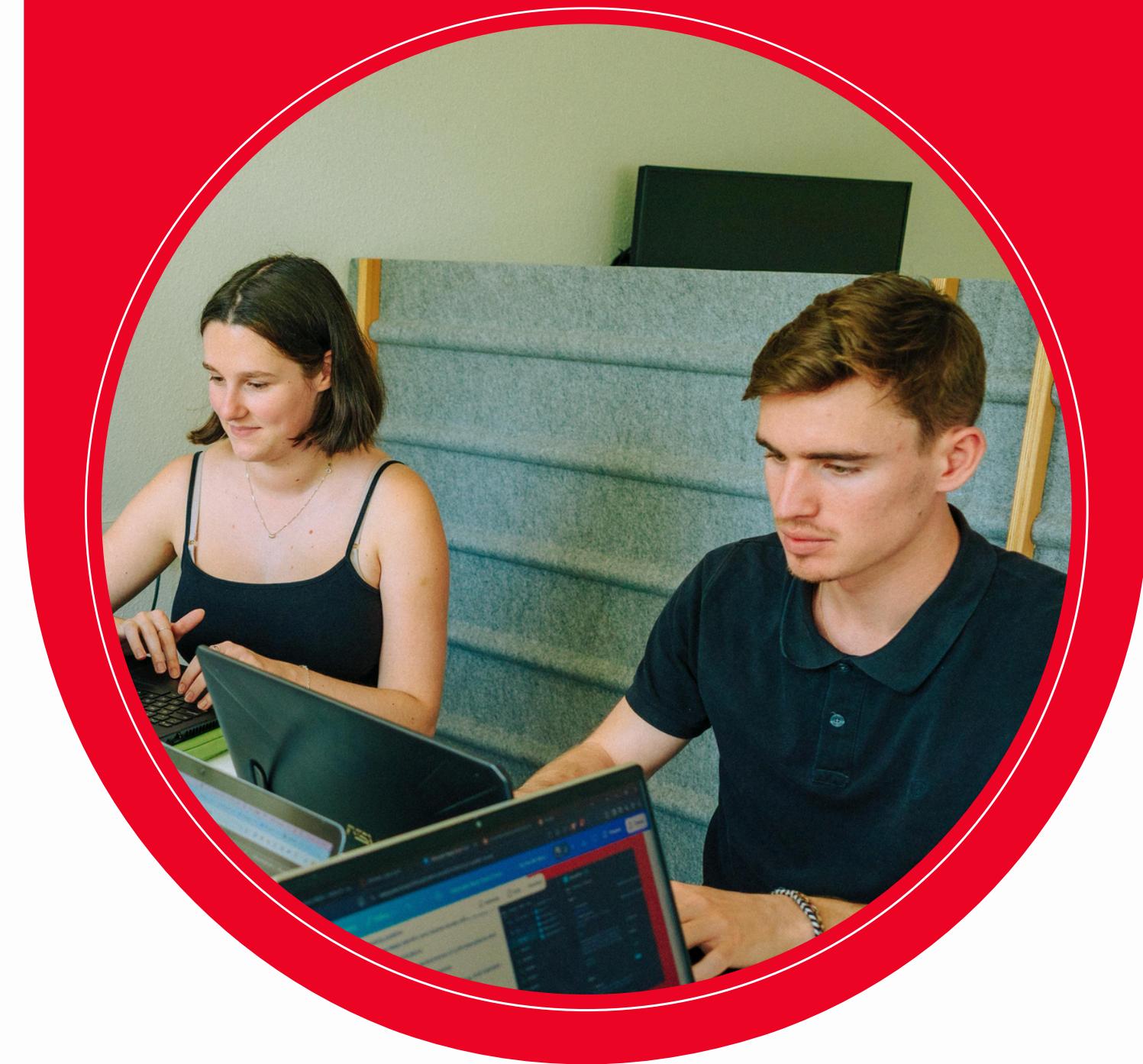
05

Very urgent

Draft Reply Generation

This function uses Google's Gemini 2.0 Flash model to generate a brand-aligned, privacy-safe response to customer emails. After receiving the original email, it builds a prompt instructing the AI to act like a support agent from the Otto Group—explicitly excluding any personal data. The Gemini model then returns a helpful draft reply, enabling faster, consistent customer communication while keeping human interaction in the loop for final review.

```
draft_prompt = f"""You are a customer support agent.  
You are generating a brand-aligned draft reply template from the Otto Group in response to a customer message.  
Do not include any personal information about the customer in the template.  
  
Here is the customer's original email:  
---  
{email_text}  
---  
  
Based on all this information, please write a draft reply.  
"""
```



Statistics

Quality Assurance and Performance

- measured using automatic testing

88%

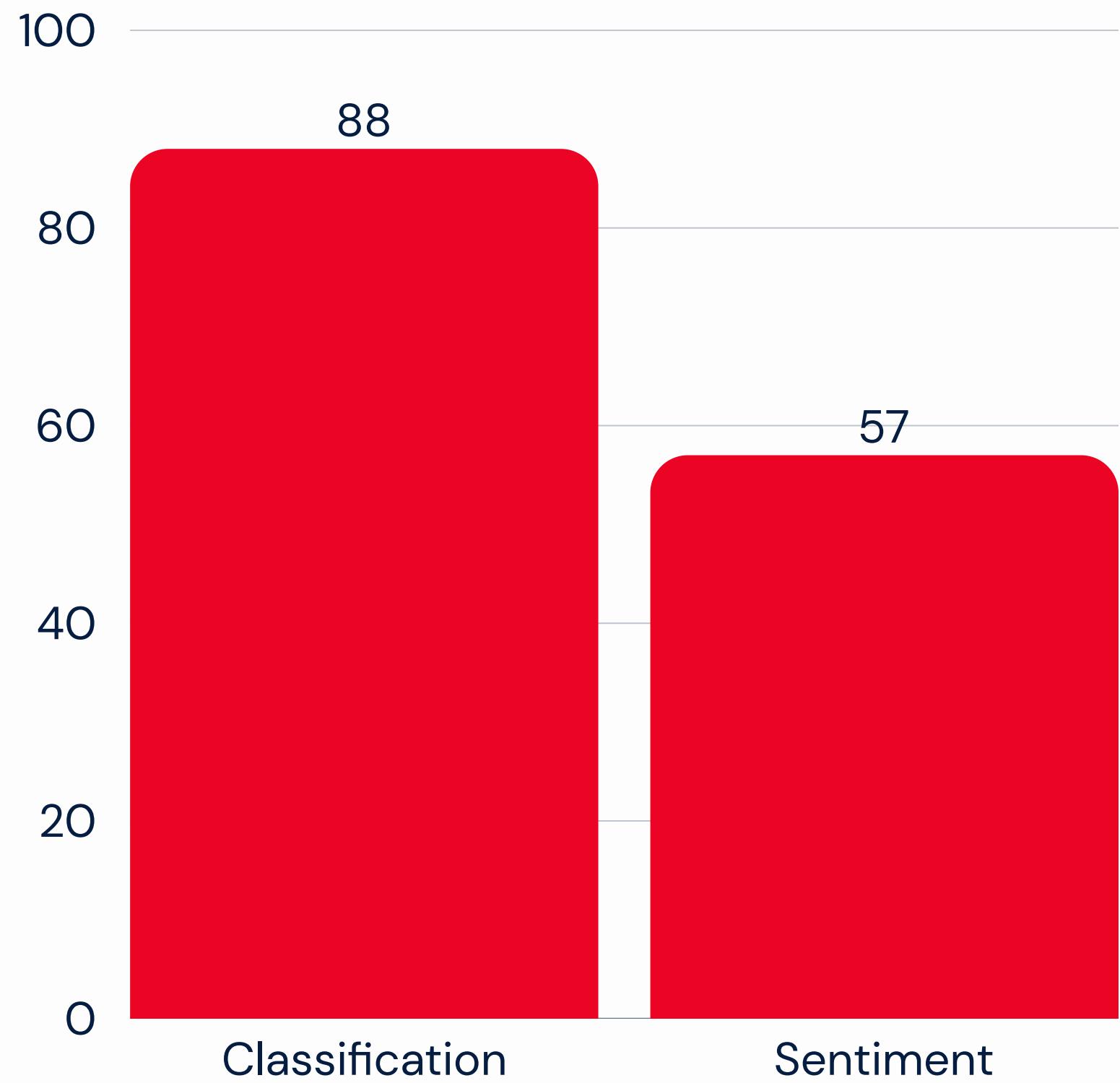
Of support team
classification
accuracy

57%

Sentiment
accuracy

60

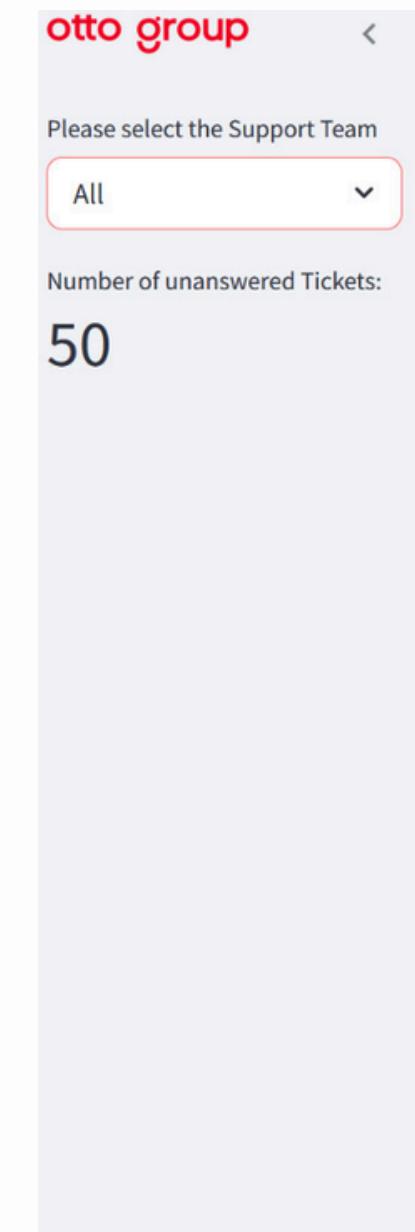
Test emails



Dashboard (Streamlit UI)

Our Streamlit UI provides an intuitive interface for customer service agents.

- **Purpose:** At-a-glance view of processed email insights.
- **Displays:** Tabular data including:
Timestamp, address, original email, redacted email, support team category, sentiment, urgency, and draft reply.
- **Benefit:** Enables rapid review, prioritization, and efficient response generation.



Welcome to Ottomatic Reply

Here's where great service starts.

This internal support space is designed to help our team assist customers quickly, confidently, and in true Otto style. We keep it efficient, empathetic, and always on-brand — just like the service our customers expect. Let's make support smarter, together.

Redacted Email	Support Team	Sentiment	Urgency	Draft Reply
... Subject: Compatibility of XDR-5...	Product Consultation	Neutral	2	Subject: Re: Compatibility of XD...
... Subject: Excited to buy ProMax ...	Product Consultation	Happy	2	Subject: Re: Excited to buy Pro...
... Subject: Confused between Pro...	Product Consultation	Unhappy	3	Subject: Re: Confused between ...
... Subject: Why isn't TX-100 work...	Product Consultation	Very unhappy	4	Subject: Regarding your inquiry ...
... Subject: Absolute newbie—Hom...	Product Consultation	Neutral	2	Subject: Re: Absolute newbie—...
... Subject: Unable to select Ameri...	Order Support	Unhappy	3	Subject: Re: Unable to select A...
... Subject: Missing express shippi...	Order Support	Very unhappy	4	Subject: Regarding Express Shi...
... Subject: Bug in cart: quantity not...	Technical Assistance	Very unhappy	4	Subject: Regarding Your Cart Is...
... Subject: Newsletter promo code...	Loyalty Programs and Discounts	Neutral	1	Subject: Re: Newsletter promo c...
... Subject: International shipping t...	Product Consultation	Neutral	2	Subject: Re: International Shippi...
... Subject: Installation error 0x800...	Technical Assistance	Unhappy	3	Subject: Re: Installation error 0x...
... Subject: Firmware update stops ...	Technical Assistance	Very unhappy	4	Subject: Re: Firmware update st...
... Subject: TrailCam 200 won't po...	Technical Assistance	Very unhappy	4	Subject: Re: TrailCam 200 won't...
... Subject: Bluetooth pairing succe...	Technical Assistance	Neutral	2	Subject: Re: Bluetooth pairing s...
... Subject: Sonoma Beta compatib...	Technical Assistance	Unhappy	3	Subject: Regarding SoundSpher...
... Subject: Order #34567 stuck at ...	Shipping and Delivery Updates	Unhappy	4	Subject: Regarding Your EcoGa...
... Subject: Tracking link 404—fix it...	Shipping and Delivery Updates	Very unhappy	5	Subject: Regarding Your Smart...
... Subject: Partial delivery for Orde...	Shipping and Delivery Updates	Unhappy	4	Subject: Re: Partial delivery for ...

