

SymTrain Simulation Intelligence Agent

Cheng Ding, James Lauer, Harrison Rubin

Walkthrough of Design

1. Extracted the .json files from the provided data folders
2. Merged transcript lines into one continuous string for each conversation
3. Extracted the reasons for the person calling, and the steps provided by the agent, using the Microsoft phi-3-mini-4k-instruct transformer and GPT-5
4. Categorized each conversation into one of 8 categories (refer to next slide), using the Facebook bart-large-mnli transformer and GPT-5
5. Built Streamlit app that categorizes test data and provides steps for the agent to take
6. Built Docker file and Docker image

Categories of Conversations

Order Status & Fulfillment

Returns, Cancellations, &
Exchange

Travel & Hospitality
Bookings

Insurance Claims &
Coverage

Account Management &
Billing

Technical Support &
Troubleshooting

Sales & Quotes

Other

SymTrain Customer Assistance Tool

Customer request:

Hi, can you help me file a claim

Run

Generated Response

Predicted Category

Insurance Claims & Coverage

Number of Steps

22

Steps

1. Greet the customer and acknowledge their request to file a claim, expressing willingness to help.
2. Determine what type of claim they want to file (e.g., auto, home, health, life, disability, other) by asking a clarifying question.
3. Ask for the customer's full name and any other basic identifying details needed by your company (e.g., date of birth, phone number, or last four digits of SSN) to locate or create their account.
4. Request the relevant policy or account number associated with the claim so you can access the correct policy in the system.
5. Verify the customer's identity according to your company's authentication requirements (e.g., security questions, mailing address, or email on file).
6. Confirm that the correct policy has been pulled up and restate the purpose: to file a new claim for the specific incident or loss.
7. Ask for the date and approximate time of the incident or event that is leading to the claim.
8. Gather a brief description of what happened, in the customer's own words, including how, where, and why the incident occurred.

SymTrain Customer Assistance Tool

Customer request:

Hi, I recently ordered a book online. Can you give me an update on the order status?

Run

Generated Response

Predicted Category

Order Status & Fulfillment

Number of Steps

15

Steps

1. Greet the customer and acknowledge their request for an order status update.
2. Ask the customer for their order identifier (order number, confirmation number, or account details).
3. Request additional verification details as needed (e.g., email address on the order, full name, and shipping address) to confirm the customer's identity.
4. Access the order management system and locate the order using the provided information.
5. Review the order details, including items purchased, order date, and current status (e.g., processing, shipped, delivered, canceled).
6. If the order has shipped, check the shipment details and carrier tracking information, including ship date and estimated delivery date.
7. Communicate the current status of the order clearly to the customer, including the shipping date, carrier, tracking number, and expected delivery timeframe if available.
8. If tracking shows the order as delivered, ask the customer to check common delivery locations (mailbox, porch, side/rear doors, building office, neighbors).
9. If the customer still cannot locate a delivered package, review internal policies to determine whether to treat the order as lost in transit.

Real-Time Demo

GPT Reasoning and Steps

```
##OpenAI version
client = OpenAI()

results = []

for i, transcript in enumerate(merged_transcripts):
    print(f"Processing transcript {i+1}...")
    full_prompt = create_prompt(transcript)

    response = client.chat.completions.create(
        model="gpt-5-nano-2025-08-07",
        messages=[
            {"role": "user", "content": full_prompt}
        ]
    )

    generated_text = response.choices[0].message.content

    results.append({
        "original_transcript": transcript,
        "extraction": generated_text
    })

# Save results to JSON file
with open('reason_step_openai.json', 'w', encoding='utf-8') as f:
    json.dump(results, f, indent=2, ensure_ascii=False)

print(f"Saved {len(results)} results to reason_step_openai.json")
```

Python

cleaning_and_extraction.ipynb M

{ } reason_step_categories_openai.json ×

{ } reason_step_categories_openai.json > ...

```
1  [
2      {
3          "original_transcript": "Trainee: Thank you for calling Northwestern Mutual",
4          "extraction": {
5              "reason": "Customer is asking why the underwriting case is taking so",
6              "steps": [
7                  "Ask for the policy number to locate the case in the system.",
8                  "Pull up the case (policy T195129) and confirm that the customer",
9                  "Review case details to identify the delay: records from Dr. Smith",
10                 "Check whether the authorization form was sent to the client; de",
11                 "Note that the underwriter requires the records to reach a final",
12                 "Provide the form to the client to help reduce further delay.",
13                 "Close the conversation with courtesy and offer further assistance",
14             ]
15         },
16         "category": "Insurance Claims & Coverage"
17     },
18     {
19         "original_transcript": "SYM: [Phone rings] Trainee: Thank you for calling",
20         "extraction": {
21             "reason": "Customer wants to return the Olivia Metal and Wood Platform",
22             "steps": [
23                 "Answered the call and collected customer name and email; retrieved",
24                 "Validated the order details: bed frame and mattress ordered online",
25                 "Confirmed delivery date of bed frame (Jan 13, 2024) and that it",
26                 "Advised the customer on the return process: return the bed frame",
27                 "Arranged a prepaid return label and provided FedEx pickup instructions",
28                 "Informed the customer that a refund will be issued to the original",
29                 "Asked if there is anything else and closed the call with follow-up",
30             ]
31         },
32         "category": "Returns, Cancellations & Exchange"
33     },
34     {
35         "original_transcript": "SYM: In this simulation, you will assist the customer",
36         "extraction": {
```

Transformer Reasoning and Steps

```
extractor = pipeline("text-generation", model="microsoft/Phi-3-mini-4k-instruct")

results = []

for i, transcript in enumerate(merged_transcripts[:3]):
    print(f"Processing transcript {i+1}...")
    full_prompt = create_prompt(transcript)

    outputs = extractor(full_prompt, max_new_tokens=500, return_full_text=False, do_sample=False)

    generated_text = outputs[0]['generated_text']

    results.append({"original_transcript": transcript,
                   "extraction": generated_text})

# Save results to JSON file
with open('reason_step.json', 'w', encoding='utf-8') as f:
    json.dump(results, f, indent=2, ensure_ascii=False)

print(f"Saved {len(results)} results to reason_step.json")
```

```
} reason_step.json > ...
```

```
1  [
2      {
3          "original_transcript": "Trainee: Thank you for calling Northwestern Mutual,
4          "extraction": " ````json\n{\n  \"reason\": \"Underwriter delay on Marty Juan
5      },
6      {
7          "original_transcript": "SYM: [Phone rings] Trainee: Thank you for calling Z
8          "extraction": " ````json\n{\n  \"reason\": \"Customer wants to return a bed
9      },
10     {
11         "original_transcript": "SYM: In this simulation, you will assist the custom
12         "extraction": " ````json\n{\n  \"reason\": \"Setting up an automated birthda
13     }
14 ]
```

GPT Categorization

```
def ask_GPT(client, reason, steps):
    full_prompt = f'''
Assign a single category for the reason and steps.

ONLY USE THE 'Other' CATEGORY IF COMPLETELY NECESSARY

REASON:
{reason}

STEPS:
{steps}

PICK A CATEGORY:
"Order Status & Fulfillment",
>Returns, Cancellations & Exchange",
"Travel & Hospitality Bookings",
"Insurance Claims & Coverage",
"Account Management & Billing",
"Technical Support & Troubleshooting",
"Sales & Quotes",
"Other"

ONLY OUTPUT THE CATEGORY:
example: Order Status & Fulfillment

'''

    response = client.chat.completions.create(
        model="gpt-5-nano-2025-08-07",
        messages=[
            {"role": "user", "content": full_prompt}
        ]
    )
    return response.choices[0].message.content
```

```
client = OpenAI()
category_list = []
for item in data:
    extraction = item.get('extraction')
    if extraction and 'reason' in extraction:
        reason_text = extraction.get('reason')
        step_text = extraction.get('steps')
        my_category = ask_GPT(client, reason_text, step_text)
        print(my_category)
        item['category'] = my_category
    category_list.append(my_category)
```

Insurance Claims & Coverage
Returns, Cancellations & Exchange
Account Management & Billing
Sales & Quotes
Technical Support & Troubleshooting
Travel & Hospitality Bookings
Travel & Hospitality Bookings
Insurance Claims & Coverage
Technical Support & Troubleshooting
Insurance Claims & Coverage
Technical Support & Troubleshooting
Order Status & Fulfillment
Insurance Claims & Coverage
Sales & Quotes
Account Management & Billing
Sales & Quotes
Technical Support & Troubleshooting
Order Status & Fulfillment
Travel & Hospitality Bookings
Insurance Claims & Coverage
Insurance Claims & Coverage
Account Management & Billing
Travel & Hospitality Bookings
Technical Support & Troubleshooting
Order Status & Fulfillment
...
Sales & Quotes
Insurance Claims & Coverage
Account Management & Billing
Order Status & Fulfillment

Transformer Categorization

```
from transformers import pipeline

classifier = pipeline(
    "zero-shot-classification",
    model="facebook/bart-large-mnli"
)

def categorize_customer_input(text, candidate_labels):
    result = classifier(
        sequences=text,
        candidate_labels=candidate_labels,
        multi_label=False # Choose the single best label
    )


    # Clean format
    scores = {label: float(score) for label, score in zip(result["labels"], result["scores"])}
    best = result["labels"][0]

    return {
        "input": text,
        "predicted_category": best,
        "scores": scores
    }

CATEGORIES = [
    "Order Status & Fulfillment",
    "Returns, Cancellations & Exchange",
    "Travel & Hospitality Bookings",
    "Insurance Claims & Coverage",
    "Account Management & Billing",
    "Technical Support & Troubleshooting",
    "Sales & Quotes",
    "Other"
]

customer_text = "I'd like to file a new car insurance claim"

print(categorize_customer_input(customer_text, CATEGORIES))
```



```
[Running] python -u "/Users/dingc2/VanderbiltStuff/masters/
DS-Programming/symtrain/categorization_transformer.py"
Device set to use mps:0
{'input': "I'd like to file a new car insurance claim",
 'predicted_category': 'Insurance Claims & Coverage', 'scores':
{'Insurance Claims & Coverage': 0.4719693660736084, 'Returns,
Cancellations & Exchange': 0.24084125459194183, 'Order Status &
Fulfillment': 0.19109278917312622, 'Other': 0.04621239751577377,
'Sales & Quotes': 0.015614424832165241, 'Technical Support &
Troubleshooting': 0.014258733950555325, 'Account Management & Billing':
0.013513119891285896, 'Travel & Hospitality Bookings': 0.
0064980085007846355}}
```

```
[Done] exited with code=0 in 18.371 seconds
```