AI Workshop - Lab 2-2: Intent Classification

In this lab, we'll build a system to classify customer text messages into different categories (called **intents**) using a powerful type of AI model called a transformer. Transformers are a key technology behind tools like ChatGPT and other modern language systems, but don't worry if you're new to them—we'll break it down step by step.

Data Overview

We're working with a dataset of customer text messages that has already been labeled with their intent (e.g., "Order Status", "Product Inquiry", "Account Help"). The goal is to teach the model to recognize these patterns so it can classify new messages correctly.

- **Dataset**:
  - Provided as two files: one for training and one for testing.
  - Training data is used to teach the model, and testing data is used to see how well it learned.
- **Number of Categories**: 27 different intents.

What We'll Do in This Lab

1. **Load the Data**:
   - Open and inspect the dataset to understand its structure.
   - Check how many examples we have for each intent.
2. **Prepare the Data**:
   - Use a tool called a **tokenizer** to break down text messages into a format the model can understand.
   - Convert the intent labels into numbers so the model can learn from them.
3. **Use a Pre-Trained Model**:
   - Start with an existing model called `T5-small` that already knows a lot about language.
   - Customize (or fine-tune) it to focus on the intents in our dataset.
4. **Train the Model**:
   - Use the prepared data to train the model step by step.
   - Measure how well it's doing along the way.
5. **Evaluate the Model**:
   - Test the model on new data it hasn't seen before.
   - Check how accurate it is and where it might make mistakes.

What You'll Learn

- **Transformers**: Get an introduction to these models and why they're so powerful for language tasks.
- **Fine-Tuning**: Learn how to take a pre-trained model and adapt it to solve a specific problem.
- **Model Evaluation**: Understand how to measure a model's performance and interpret its predictions.

HuggingFace Libraries

So far we have been working with Keras, a popular library for building neural networks. In this lab, we'll use the HuggingFace libraries, which are designed specifically for working with transformers.

The main HuggingFace library is called `transformers`, and it provides tools for working with pre-trained models, tokenizers, and training pipelines. We'll also use `datasets` to load and process our data. `accelerate` and `evaluate` are additional libraries that help speed up training and evaluate models, respectively. Install them below:

In [1]:
```
!pip install -Uq datasets transformers accelerate evaluate
```

For this lab, it's essential that we have a GPU available to speed up training. On Google Colab, you can enable a GPU by going to **Runtime** > **Change runtime type** > **Hardware accelerator** > **GPU**.

The following line of code will check if a GPU is available:

In [2]:
```
import torch
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
if device.type == 'cuda':
    print('GPU is available!')
else:
    print('GPU is not available. Enable a GPU runtime in Colab under "Runtim
```

```
GPU is available!
```

Loading the Dataset

Now that we've set up our environment and imported the necessary packages, let's begin by loading our dataset.

In this lab, we'll work with a dataset of **customer text messages** that have been labeled with their **intent**. Each sample in the dataset includes a text message and a corresponding label indicating the intent behind the message (e.g., inquiry, complaint, order request). This dataset will allow us to build and evaluate models for intent classification.

Steps:

1. **Load the Dataset**:
   - Use the `load_dataset` function from the `datasets` library to download and load the dataset.
   - The dataset we're using is hosted at `"alexwaolson/customer-intents"`.
2. **Inspect the Dataset**:
   - After loading, examine the training split (`intents['train']`) to understand its structure and the data it contains.

In [3]:
```python
from datasets import load_dataset

# Load the customer intents dataset
intents = load_dataset("alexwaolson/customer-intents")

# Display the training split
intents['train']
```

Out[3]:
```
Dataset({
    features: ['message', 'label'],
    num_rows: 1555
})
```

The dataset consists of two key columns:

- **`message`**: Contains the text of the customer message.
- **`label`**: Contains the intent category for each message.

There are **27 possible intent categories** in this dataset. To understand the distribution of these categories, we can count the number of examples for each intent. This helps us determine whether the dataset is balanced (i.e., whether all categories have similar representation) or imbalanced (some categories have significantly more or fewer samples than others).

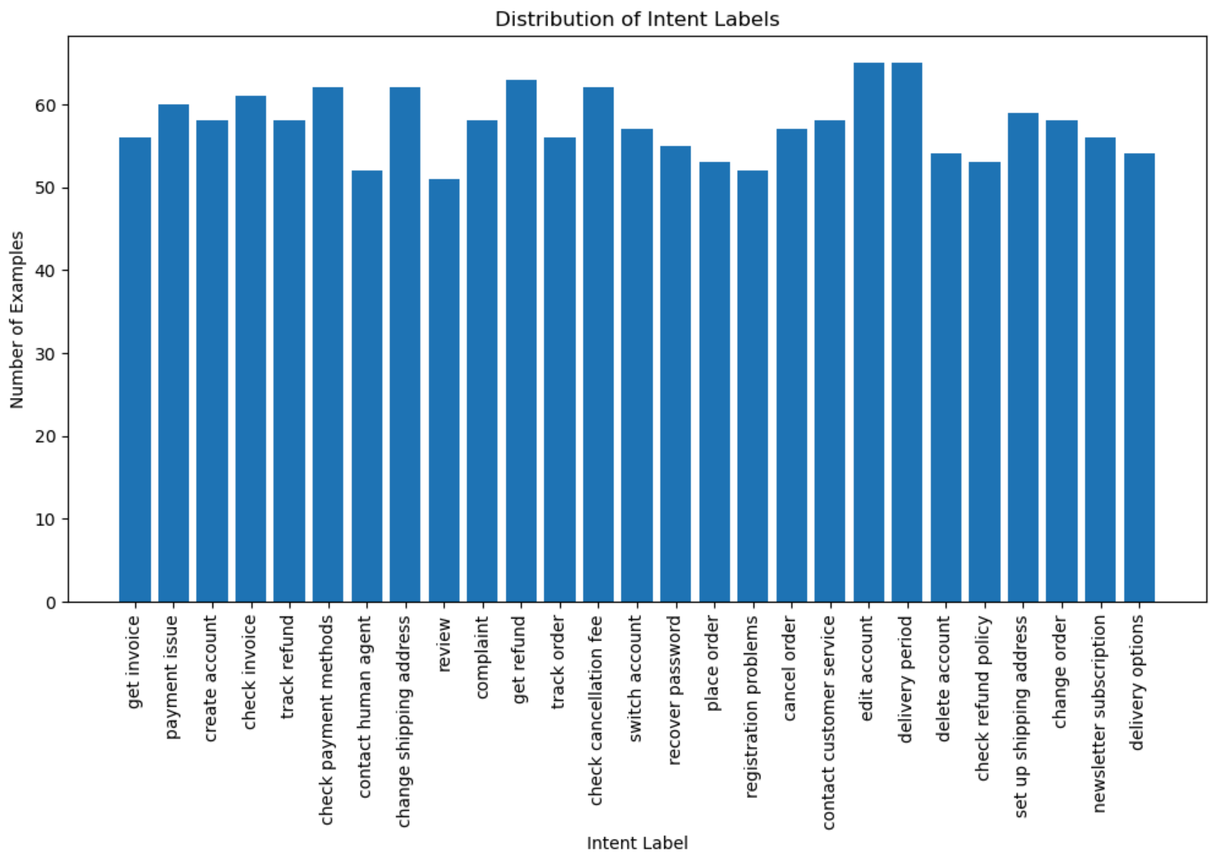
Run the code below to calculate the distribution of intent labels:

In [4]:
```python
from collections import Counter
import matplotlib.pyplot as plt

# Count the occurrences of each intent label in the training data
label_counts = Counter(intents['train']['label'])
print(f'Number of unique intents: {len(label_counts)}')

# Plot the distribution of intent labels
plt.figure(figsize=(12, 6))
plt.bar(label_counts.keys(), label_counts.values())
plt.xlabel('Intent Label')
plt.ylabel('Number of Examples')
plt.title('Distribution of Intent Labels')
plt.xticks(rotation=90)
plt.show()
```

```
Number of unique intents: 27
```



Preprocessing the Dataset

Now that our dataset is loaded, we need to prepare it for training. This involves several steps, including tokenization, padding, and setting up a data collator. These steps ensure that the raw text data is transformed into a format that our model can understand and process efficiently.

Tokenization

Most language models, including T5, don't work directly with raw text but with **tokenized inputs**. Tokenization involves breaking down text into smaller units, called **tokens**, which are then converted into numerical IDs that the model can process.

How Tokenization Works:

1. **Splitting Text**:
   - Text is split into words, subwords, or characters.
   - For example, the word "running" might be split into "run" and "##ning" (where `##` indicates a suffix).

2. **Mapping to IDs**:
   - Each token is mapped to a unique integer ID using a vocabulary associated with the tokenizer.
3. **Flexibility**:
   - This approach allows the model to understand the meaning of individual components (like "run" and "-ning") and combine them to interpret words.

Why Tokenization is Important:

- It converts raw text into numerical data that can be used for deep learning.
- It helps the model generalize to unseen words by breaking them into familiar components.

T5 Model and Tokenizer:

We'll use the `AutoTokenizer` class from the `transformers` library to load a tokenizer that matches the **T5-small** model. T5 (Text-to-Text Transfer Transformer) is a versatile language model designed to handle a wide range of NLP tasks by treating all tasks as text-to-text transformations.

Let's load the tokenizer and inspect its properties:

In [5]:
```
from transformers import AutoTokenizer

# Load our tokenizer
model_name = 't5-small'
# The AutoTokenizer class will automatically select the correct tokenizer cl
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

Inspecting the Tokenizer

Before we start using the tokenizer to preprocess our dataset, let's take a closer look at its capabilities. Understanding the tokenizer's vocabulary and special tokens will help us use it effectively.

Vocabulary Size

The tokenizer's **vocabulary size** represents the total number of unique tokens it can recognize and map to IDs. This includes:

- Words
- Subwords
- Special tokens

A larger vocabulary allows the model to understand a wider range of text but may also increase the computational requirements.

Special Tokens

Special tokens are reserved tokens used for specific purposes in a model:

- **Start and End Tokens**: Indicate the beginning and end of sequences.
- **Padding Token**: Ensures all sequences in a batch have the same length by filling shorter sequences.
- **Other Tokens**: For tasks like separating sentences or marking different segments in the input.

In [6]:
```
# Inspect the tokenizer's vocabulary and special tokens
print(f'Vocab size:     {tokenizer.vocab_size} tokens')  # Total number of t
print(f'End token:      {tokenizer.eos_token}')          # End of sequence t
print(f'Padding token:  {tokenizer.pad_token}')          # Padding token
print(f'Unknown token:  {tokenizer.unk_token}')          # Unknown token, fo
```

```
Vocab size:     32100 tokens
End token:      </s>
Padding token:  <pad>
Unknown token:  <unk>
```

The tokenizer not only converts text into token IDs but can also convert token IDs back into text. This process involves:

- **Encoding**: Breaking a string into tokens and mapping them to unique IDs.
- **Decoding**: Converting token IDs back into a human-readable string.

To better understand how tokenization works, we'll use a custom function, `show_tokenization`, which:

1. Displays the original text.
2. Encodes the text into token IDs.
3. Decodes each token ID back into its corresponding token.

Your Turn:

- Write a sentence in the `show_tokenization` function to see how it is tokenized by the model.
- Try using:
    - A sentence with common words.
    - A sentence containing made-up words.
    - Words that might be split into multiple tokens (e.g., "unbelievably").

In [7]:
```
def show_tokenization(tokenizer, text):
    print(f'Original text: {text}')
    tokens = tokenizer(text, truncation=True)['input_ids']
    for token in tokens:
        print(f'{tokenizer.decode([token]):10} -> {token}')

# Write a sentence to see how it gets tokenized:
show_tokenization(tokenizer, 'your sentence here')
```

```
Original text: your sentence here
your       -> 39
sentence   -> 7142
here       -> 270
</s>       -> 1
```

Tokenizer Basics: Padding and Attention Masks

Now that you’ve seen how tokenization works, let’s cover a few additional concepts: **attention masks** and **padding**. These concepts are crucial for processing variable-length sequences when training models.

Attention Mask

- The **attention mask** is a vector that tells the model which tokens are actual input and which are padding.
- For example:
  - A token with a mask value of `1` indicates that it is part of the original input.
  - A token with a mask value of `0` indicates that it is a padding token and should be ignored during processing.

You may see this term pop up later when preparing data for the model.

Padding

When training models in batches, all sequences in a batch must have the same length to allow parallel processing. Since sequences vary in length, we:

1. Add **padding tokens** to the shorter sequences to make them the same length as the longest sequence in the batch.
2. Use the attention mask to ensure the model ignores these padding tokens during training.

Example:

If a batch has a maximum length of 10 tokens and a sequence like `"Hello, world!"` (with 2 tokens) is in the batch:

- **Original**: `"Hello, world!"`
- **Padded**: `"<pad>, "Hello", ",", "world", "!", </s>, <pad>, <pad>, <pad>, <pad>"`
- **Attention Mask**: `[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]`

Setting Maximum Length

By default, the T5-small model expects sequences of up to 512 tokens. However, for this task, we don’t need sequences that long. We’ll set the maximum sequence length to **40 tokens**. This ensures:

- Sequences longer than 40 tokens are truncated.
- Shorter sequences are padded to reach 40 tokens.

Run the following code to configure the tokenizer:

```
In [8]: # Define maximum sequence length for inputs
        max_input_length = 40

        # Set the tokenizer's maximum length
        tokenizer.model_max_length = max_input_length
```

Handling Long Sequences: Truncation

When input sequences exceed the specified maximum length, the tokenizer will **truncate** them to fit. Truncation simply means cutting off tokens that exceed the limit, and this can be done either from the start (left truncation) or the end (right truncation) of the sequence.

Truncation Options

- **Right Truncation** (default): Removes tokens from the end of the sequence.
- **Left Truncation**: Removes tokens from the start of the sequence.

The choice between left or right truncation depends on the task:

- For summarization or text generation, keeping the start of the text (left truncation) might not make sense since context at the end is often important.
- For tasks like sentence classification, keeping the start (right truncation) may suffice if the relevant information is typically in the first part of the text.

Your Turn:

1. Use the cell below to input a long sentence and observe how the tokenizer truncates it.
2. Experiment by changing:
   - `truncation_side` to `'left'` or `'right'` to see how truncation behaves.
   - The `max_input_length` to control the maximum allowed tokens for input sequences.

```
In [9]: # Set truncation behavior
        tokenizer.truncation_side = 'right'  # Truncate from the end
        # tokenizer.truncation_side = 'left'  # Uncomment to truncate from the start

        # Test the tokenizer with a long sentence
        show_tokenization(tokenizer, "write a REALLY long sentence in here and see w
Original text: write a REALLY long sentence in here and see what happens
write       -> 1431
            -> 3
a           -> 9
REAL        -> 17833
LY          -> 5121
long        -> 307
sentence    -> 7142
in          -> 16
here        -> 270
and         -> 11
see         -> 217
what        -> 125
happens     -> 2906
</s>        -> 1
```

Preparing the Dataset for Training

Now that we've configured the tokenizer, the final step before training is to **preprocess the data**. This involves tokenizing the text messages in our dataset so they are converted into token IDs and ready for the model to process.

Defining a Preprocessing Function

We'll define a function, `preprocess_function`, to handle this tokenization. The function will:

1. Take a batch of examples from the dataset.
2. Use the tokenizer to tokenize the `message` field, with truncation enabled to handle sequences longer than the maximum allowed length.

Here's the function:

In [10]:
```
def preprocess_function(examples):
    return tokenizer(examples["message"], truncation=True)
```

Tokenizing the Dataset

The map method is used to apply the preprocess_function to each batch of examples in the dataset:

- Setting `batched=True` ensures that the function processes multiple examples at once, making it more efficient.
- The tokenized output will include token IDs and attention masks for each example.

Run the following code to tokenize the dataset:

In [11]:
```
tokenized_intents = intents.map(preprocess_function, batched=True)
```

In [12]:
```
tokenized_intents['train']
```

Out[12]:
```
Dataset({
    features: ['message', 'label', 'input_ids', 'attention_mask'],
    num_rows: 1555
})
```

Data Collation

Before training our model, we need to handle one last preprocessing step: **data collation**. This involves combining individual examples into batches that the model can process efficiently. During collation:

- All sequences in a batch are padded to the same length to ensure uniformity.
- The resulting batch is formatted into a structure compatible with the model.

Setting Up a Data Collator

We'll use the `DataCollatorWithPadding` class from the `transformers` library. This class:

- Uses the tokenizer to handle padding.
- Returns the collated batch as PyTorch tensors (`return_tensors="pt"`).

In [13]:
```
from transformers import DataCollatorWithPadding

# Create a data collator for padding and batching
data_collator = DataCollatorWithPadding(tokenizer=tokenizer, return_tensors=
```

Evaluation Metric

During training, we need a way to evaluate the model's performance. We'll compute metrics like **accuracy** to measure how well the model's predictions match the true labels. This will help us monitor progress and identify areas for improvement.

Setting Up the Metric

We'll use the `evaluate` library to calculate accuracy. To do this, we define a function called `compute_metrics`, which processes the model's predictions and computes the desired metric.

In [14]:
```
import evaluate
import numpy as np

# Load the accuracy metric
accuracy = evaluate.load("accuracy")


def compute_metrics(eval_pred):
    # Unpack predictions and labels
    predictions, labels = eval_pred.predictions, eval_pred.label_ids

    # Handle tuple predictions (e.g., logits)
    if isinstance(predictions, tuple):
        predictions = predictions[0]  # Use the logits

    # Convert predictions to a NumPy array if necessary
    predictions = np.array(predictions)

    # Compute class predictions (argmax for classification)
    predictions = np.argmax(predictions, axis=1)

    # Return the computed accuracy
    return accuracy.compute(predictions=predictions, references=labels)
```

Encoding Labels

To train our model, we need to convert the **intent labels** into numerical values. Machine learning models work with numbers, not text, so we'll create a mapping that translates each text label into a unique integer. This process is known as **label encoding**.

Steps:

1. **Create a Mapping**:
   - `label2id` : Maps each label (text) to a unique integer.
   - `id2label` : Provides the reverse mapping, converting integers back to labels for easy interpretation later.
2. **Apply the Mapping**:
   - Use the `map` function to encode the `label` field in the dataset, replacing text labels with their corresponding integers.

In [15]:
```
# Create mappings from labels to integers and vice versa
label2id = {label: i for i, label in enumerate(intents['train'].unique('labe
id2label = {i: label for label, i in label2id.items()}

# Function to encode labels
def encode_label(example):
    example['label'] = label2id[example['label']]
    return example

# Apply the label encoding to the dataset
tokenized_intents = tokenized_intents.map(encode_label)

# Inspect the encoded training dataset
tokenized_intents['train']
```

Out[15]:
```
Dataset({
    features: ['message', 'label', 'input_ids', 'attention_mask'],
    num_rows: 1555
})
```

Loading the Model

With our data preprocessed, it's time to load the **pre-trained model** that we'll fine-tune for our classification task. Pre-trained models are powerful because they already have a general understanding of language, allowing us to focus on adapting them to our specific problem.

Model Setup

We'll use the `AutoModelForSequenceClassification` class from the `transformers` library to load a model designed for sequence classification tasks. This class:

- Loads a pre-trained model for tasks where the input is a sequence (e.g., text) and the output is a classification label.
- Automatically configures the model for the number of classes (27 intent categories in our case).

Key Parameters:

- **Pre-trained Model**: We're using `t5-small`, a small but capable version of the T5 model, designed for efficient fine-tuning.
- **Number of Labels**: `num_labels=27` specifies the number of intent categories in our dataset.
- **Label Mappings**: `id2label` and `label2id` ensure that the model can interpret numeric labels and translate them back into text labels when needed.

When you run the code to load the model, you'll likely see a warning message like this:

```
Some weights of T5ForSequenceClassification were not
initialized from the model checkpoint at t5-small and are
newly initialized: ['classification_head.dense.bias',
'classification_head.dense.weight',
'classification_head.out_proj.bias',
'classification_head.out_proj.weight'] You should probably
TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

This message is expected because we haven't trained the model yet. It's a reminder that the model's classification head (the part responsible for classifying sequences) is initialized randomly and needs to be trained on our dataset.

```
In [16]: from transformers import AutoModelForSequenceClassification, TrainingArgumer

         # Load the pre-trained model for sequence classification
         model = AutoModelForSequenceClassification.from_pretrained(
             "t5-small",          # Model name
             num_labels=27,       # Number of intent categories
             id2label=id2label,   # Mapping from IDs to labels
             label2id=label2id    # Mapping from labels to IDs
         )
```

```
Some weights of T5ForSequenceClassification were not initialized from the mo
del checkpoint at t5-small and are newly initialized: ['classification_head.
dense.bias', 'classification_head.dense.weight', 'classification_head.out_pr
oj.bias', 'classification_head.out_proj.weight']
You should probably TRAIN this model on a down-stream task to be able to use
it for predictions and inference.
```

Training

We're now ready to train our model! The `Trainer` class from the `transformers` library simplifies the training process by handling key tasks like batching, evaluation, and gradient updates. To customize the training, we define a set of **training arguments** that specify how the training should proceed.

Training Arguments Explained:

- **`per_device_train_batch_size`** : The number of samples processed at once on each device (e.g., GPU). Smaller batch sizes may be necessary if memory is limited.
- **`per_device_eval_batch_size`** : Similar to the training batch size but used for evaluation.
- **`num_train_epochs`** : The number of passes through the entire training dataset.
- **`logging_dir`** : Directory where training logs are saved for monitoring.
- **`logging_steps`** : How often (in steps) training progress is logged.
- **`eval_strategy`** : Specifies when evaluation is performed. Here, it's set to evaluate every few steps.
- **`eval_steps`** : The number of steps between evaluations.
- **`save_strategy`** : Determines when the model is saved. Here, saving is disabled with `"no"` .
- **`output_dir`** : Directory where final model files and outputs will be saved. This is required even if saving is disabled.

Setting Up the Trainer

The `Trainer` class brings everything together:

- **Model**: The model to be trained.
- **Arguments**: The training configuration (`training_args`).
- **Datasets**: The preprocessed training and evaluation datasets.

- **Data Collator**: Handles batching and padding during training.
- **Metrics**: The `compute_metrics` function for evaluating performance.

In [17]:
```
from transformers import TrainingArguments, Trainer

# Define training arguments
training_args = TrainingArguments(
    per_device_train_batch_size=8,    # Batch size for training
    per_device_eval_batch_size=8,    # Batch size for evaluation
    num_train_epochs=3,              # Number of training epochs
    logging_dir='logs',              # Directory for logs
    logging_steps=10,                # Log progress every 10 steps
    eval_strategy="steps",           # Evaluate every X steps
    eval_steps=10,                   # Perform evaluation every 10 steps
    save_strategy="no",              # Disable model saving
    output_dir='output'              # Directory for output files
)

# Set up the Trainer
trainer = Trainer(
    model=model,                     # Model to train
    args=training_args,              # Training arguments
    train_dataset=tokenized_intents['train'],  # Training dataset
    eval_dataset=tokenized_intents['test'],    # Evaluation dataset
    data_collator=data_collator,     # Handles padding and batching
    compute_metrics=compute_metrics  # Evaluates model performance
)

# Start training
trainer.train()
```

[585/585 01:05, Epoch 3/3]

Step	Training Loss	Validation Loss	Accuracy
10	3.335700	3.355077	0.030848
20	3.365600	3.326983	0.041131
30	3.325800	3.300330	0.061697
40	3.320300	3.277448	0.071979
50	3.290100	3.261157	0.087404
60	3.294700	3.249416	0.102828
70	3.298500	3.230739	0.143959
80	3.219800	3.208219	0.179949
90	3.191900	3.190287	0.143959
100	3.214800	3.173454	0.115681
110	3.195100	3.154064	0.107969
120	3.185000	3.132910	0.115681
130	3.161300	3.108371	0.133676
140	3.209500	3.082083	0.141388
150	3.213600	3.056199	0.174807
160	3.140000	3.035336	0.241645
170	3.091900	3.014059	0.257069
180	3.040800	2.991459	0.249357
190	3.037900	2.972116	0.226221
200	3.042900	2.951756	0.210797
210	3.028800	2.935601	0.195373
220	3.021200	2.909835	0.215938
230	2.982600	2.883194	0.267352
240	2.919500	2.862450	0.293059
250	2.982800	2.844471	0.277635
260	2.939500	2.821646	0.305913
270	2.919000	2.797715	0.339332
280	2.938000	2.778098	0.336761
290	2.911000	2.756023	0.380463

Step	Training Loss	Validation Loss	Accuracy
300	2.879000	2.736224	0.390746
310	2.724800	2.718025	0.383033
320	2.797500	2.699818	0.377892
330	2.782200	2.681572	0.388175
340	2.749000	2.662954	0.398458
350	2.766900	2.644386	0.416452
360	2.753800	2.626633	0.437018
370	2.709800	2.609021	0.465296
380	2.666100	2.596332	0.462725
390	2.788400	2.586444	0.465296
400	2.672000	2.574779	0.475578
410	2.630900	2.569066	0.442159
420	2.661900	2.557627	0.470437
430	2.662800	2.550715	0.452442
440	2.644400	2.541655	0.439589
450	2.659900	2.524944	0.462725
460	2.622500	2.512599	0.483290
470	2.676200	2.500810	0.493573
480	2.522800	2.491506	0.493573
490	2.677700	2.484492	0.493573
500	2.563800	2.478365	0.488432
510	2.595100	2.471816	0.501285
520	2.588000	2.466702	0.498715
530	2.674200	2.461253	0.501285
540	2.594000	2.456957	0.503856
550	2.494800	2.453914	0.506427
560	2.624500	2.451606	0.506427
570	2.553500	2.450185	0.506427
580	2.572100	2.449440	0.506427

Out[17]:
```
TrainOutput(global_step=585, training_loss=2.89636112767407, metrics={'trai
n_runtime': 66.2023, 'train_samples_per_second': 70.466, 'train_steps_per_s
econd': 8.837, 'total_flos': 20767034735856.0, 'train_loss': 2.896361127674
07, 'epoch': 3.0})
```

Evaluating the Model

After training, it’s time to evaluate the model's performance on the **test dataset**, which contains unseen examples. This allows us to measure how well the model generalizes to new data.

Current Performance

- After a few epochs of training, the model achieves around **55% accuracy**. While this is a good start, it suggests there’s room for improvement with additional tuning and training.
- Remember, with 27 intent categories, random guessing would yield an accuracy of ~3.7%, so 55% represents a significant improvement.

Evaluation Tools

To better understand the model's strengths and weaknesses, we’ll:

1. **Generate a Classification Report**:
   - The classification report provides metrics like precision, recall, and F1-score for each intent category.
   - This helps identify which categories the model performs well on and which are more challenging.
2. **Produce a Confusion Matrix**:
   - The confusion matrix shows how often each category is confused with others.
   - It helps identify specific pairs of intents that the model struggles to distinguish.

In [18]:
```python
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import numpy as np

# Get predictions on the test dataset
predictions = trainer.predict(tokenized_intents['test']).predictions[0]

# Compute class predictions
class_predictions = np.argmax(predictions, axis=1)

# Get true labels
true_labels = tokenized_intents['test']['label']

# Convert labels to text
true_labels_text = [id2label[label] for label in true_labels]
predicted_labels_text = [id2label[label] for label in class_predictions]
```

```python
# Display classification report
report = pd.DataFrame(classification_report(true_labels_text, predicted_labe
report
```

Out[18]:

	precision	recall	f1-score	support
cancel order	0.666667	0.142857	0.235294	14.000000
change order	0.363636	0.888889	0.516129	18.000000
change shipping address	0.285714	0.250000	0.266667	8.000000
check cancellation fee	0.714286	0.909091	0.800000	11.000000
check invoice	0.571429	0.615385	0.592593	13.000000
check payment methods	0.909091	0.769231	0.833333	13.000000
check refund policy	1.000000	0.062500	0.117647	16.000000
complaint	0.750000	0.923077	0.827586	13.000000
contact customer service	1.000000	0.818182	0.900000	11.000000
contact human agent	0.928571	0.650000	0.764706	20.000000
create account	0.216216	1.000000	0.355556	16.000000
delete account	1.000000	0.055556	0.105263	18.000000
delivery options	1.000000	0.705882	0.827586	17.000000
delivery period	0.160714	0.900000	0.272727	10.000000
edit account	0.636364	0.700000	0.666667	10.000000
get invoice	0.500000	0.071429	0.125000	14.000000
get refund	0.384615	0.500000	0.434783	10.000000
newsletter subscription	1.000000	1.000000	1.000000	11.000000
payment issue	0.500000	1.000000	0.666667	11.000000
place order	1.000000	0.235294	0.380952	17.000000
recover password	1.000000	0.100000	0.181818	20.000000
registration problems	1.000000	0.176471	0.300000	17.000000
review	0.666667	0.210526	0.320000	19.000000
set up shipping address	0.562500	0.818182	0.666667	11.000000
switch account	1.000000	0.052632	0.100000	19.000000
track order	0.800000	0.250000	0.380952	16.000000
track refund	0.823529	0.875000	0.848485	16.000000
accuracy	0.506427	0.506427	0.506427	0.506427
macro avg	0.720000	0.543710	0.499521	389.000000

	precision	recall	f1-score	support
weighted avg	0.751149	0.506427	0.477029	389.000000

Visualizing the Confusion Matrix

The **confusion matrix** provides a detailed view of the model's performance by showing how often each category is predicted correctly or confused with others. It’s a square matrix where:

- Rows represent the **true labels**.
- Columns represent the **predicted labels**.

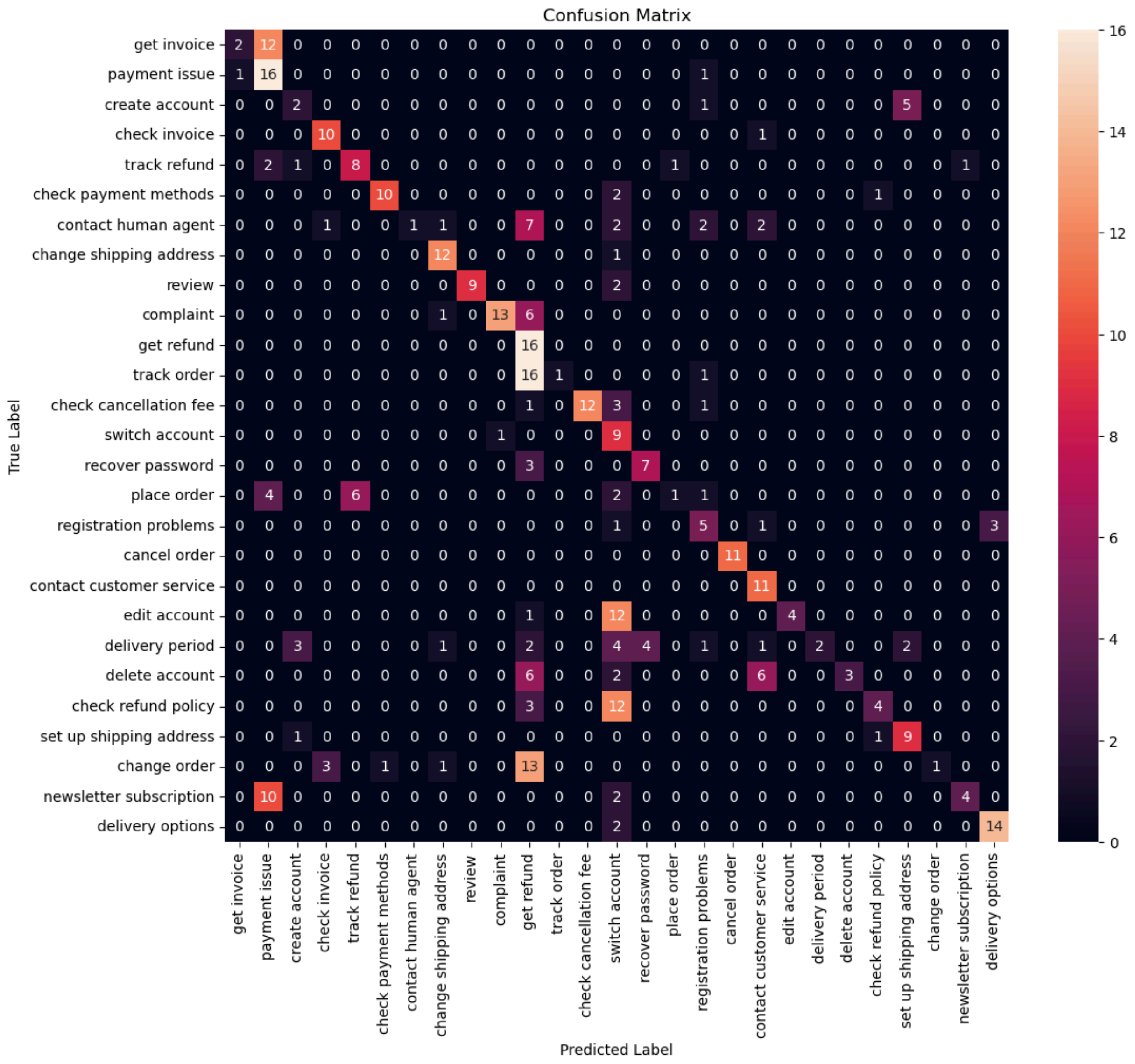
How to Interpret the Confusion Matrix:

- Each cell ((i, j)) shows the number of examples with true label (i) that were predicted as label (j).
- **Diagonal values**: Represent correct predictions (true label matches predicted label).
- **Off-diagonal values**: Represent misclassifications, highlighting categories that are commonly confused.

In [19]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Compute confusion matrix
conf_matrix = confusion_matrix(true_labels_text, predicted_labels_text)

# Plot the confusion matrix
plt.figure(figsize=(12, 10))
sns.heatmap(conf_matrix, annot=True, fmt='d', xticklabels=id2label.values(),
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



Model in Practice

In real-world applications, a model like this can be used to automatically classify customer messages. However, it's often important to account for uncertainty in the model's predictions. For example:

- If the model is **confident**, the predicted intent can be used directly.
- If the model's **confidence is low**, the message can be flagged for manual review, with the top predicted intents provided to assist human agents.

Implementing a Practical Prediction Method

We'll write a method, `predict_intent`, that:

1. **Classifies Messages**:
    - Predicts the most likely intent for each message.
2. **Handles Uncertainty**:

- If the model’s confidence (probability of the top prediction) is below a specified threshold, it returns the top 3 most likely intents instead of a single prediction.

```
In [20]: def predict_intent(model, tokenizer, messages, id2label, threshold=0.5):
             # Tokenize the messages
             tokenized_messages = tokenizer(messages, truncation=True, padding=True,

             model.to(device)

             # Get model predictions
             predictions = model(**tokenized_messages)

             # Get logits as numpy array for processing
             logits = predictions.logits.detach().cpu().numpy()

             # Get predicted class
             predicted_class = np.argmax(logits, axis=1).tolist()

             # Get predicted probabilities
             predicted_probs = np.max(logits, axis=1).tolist()

             # Get predicted labels
             predicted_labels = [id2label[label] for label in predicted_class]

             # Get top intents if confidence is low
             if any(prob < threshold for prob in predicted_probs):
                 top_intents = [
                     [id2label[i] for i in np.argsort(logit)[::-1][:3]]
                     for logit in logits
                 ]
                 return top_intents[0]
             else:
                 return predicted_labels[0]

         for message in intents['test']['message'][0:15]:
             print(f"Message: {message}")
             print(f"Predicted Intent: {predict_intent(model, tokenizer, message, id2
             print()
```

```
Passing a tuple of `past_key_values` is deprecated and will be removed in Tr
ansformers v4.48.0. You should pass an instance of `EncoderDecoderCache` ins
tead, e.g. `past_key_values=EncoderDecoderCache.from_legacy_cache(past_key_v
alues)`.
```

```
Message: I do not know how I can get the bill from Anna Freeman
Predicted Intent: check invoice

Message: help me to check how long it takes for my item to arrive
Predicted Intent: delivery period

Message: I have to see how long it takes for the package to arrive
Predicted Intent: delivery period

Message: how do I file a customer claim against your organization?
Predicted Intent: complaint

Message: I need help to see in what cases can I ask for refunds
Predicted Intent: create account

Message: i try to talk to an agent
Predicted Intent: contact human agent

Message: i dont know what i have to do to buy some of ur product
Predicted Intent: delivery period

Message: help to notify of an payment error
Predicted Intent: payment issue

Message: chat wth somebody
Predicted Intent: contact human agent

Message: i have got to request goddamn erimbursements of my money
Predicted Intent: get refund

Message: how could i talk with an assistant
Predicted Intent: create account

Message: wanna switch an item of order 113542617735902 ohw to do it
Predicted Intent: ['change order', 'check invoice', 'track order']

Message: i do not know how i could cancel order 00123842
Predicted Intent: change order

Message: want help tracking order 113542617735902
Predicted Intent: change order

Message: question about cancelling purchase 732201349959
Predicted Intent: change order
```

Zero-Shot Learning

One of the most powerful features of large language models is their ability to perform **zero-shot learning**. Unlike traditional models that require task-specific training, a zero-shot learning model can classify text based on its general understanding of language, even if it hasn't been explicitly trained on that specific task.

How It Works:

- Instead of fine-tuning the model, you provide it with a **prompt** that describes the task and possible labels (e.g., "What is the intent of this message?").
- The model uses its pre-trained knowledge to predict the most appropriate label.

This approach leverages the model's extensive training on a wide variety of text, making it flexible for many tasks.

Why Use Zero-Shot Learning?

- **Quick Prototyping**: No need to preprocess or fine-tune the model for every new task.
- **Versatility**: Works for tasks the model wasn’t explicitly trained on, as long as the task can be described in a prompt.

Using a Larger Model:

For zero-shot classification, we’ll use the `flan-t5-large` model, which is better suited for this task due to its size and broader understanding of language. Since this model doesn’t require fine-tuning, we can focus on testing its performance directly.

Installation:

The `sentencepiece` library is required to use the `flan-t5-large` model. Install it by running the following command:

In [21]:
```
!pip install sentencepiece
```

```
huggingface/tokenizers: The current process just got forked, after paralleli
sm has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
	- Avoid using `tokenizers` before the fork if possible
	- Explicitly set the environment variable TOKENIZERS_PARALLELISM=(tr
ue | false)
Requirement already satisfied: sentencepiece in /home/alex/mambaforge/envs/l
ab_2_2/lib/python3.13/site-packages (0.2.0)
```

Zero-Shot Intent Classification with Flan-T5

We’ll now use the **Flan-T5 large** model to classify intents via zero-shot learning. This approach involves crafting a **prompt** that describes the task and provides the model with the possible labels. The model then uses its language understanding to predict the intent without task-specific training.

Prompt Construction

The prompt is key to zero-shot learning. For our task:

1. The prompt begins by instructing the model to classify the intent of the message.
2. It lists the available intent categories.

3. Finally, it appends the message to classify.

In [22]:
```
from transformers import T5Tokenizer, T5ForConditionalGeneration

# Load the tokenizer and model
tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-large")
model = T5ForConditionalGeneration.from_pretrained("google/flan-t5-large", c

# Define the prompt
prompt = "Classify the intent of the following message using these categorie
for label in id2label.values():
    prompt += f"- {label}\n"
prompt += "Message: "

print(prompt)

# Function for zero-shot classification
def zero_shot_intent_classification(model, prompt, message):
    # Combine the prompt and the message
    input_text = prompt + message
    # Tokenize the input text
    input_ids = tokenizer(input_text, return_tensors="pt").input_ids.to(devi
    # Generate a prediction
    output = model.generate(input_ids, max_length=50, num_beams=5, early_sto
    # Decode the prediction into text
    return tokenizer.decode(output[0], skip_special_tokens=True)
```

```
You are using the default legacy behaviour of the <class 'transformers.model
s.t5.tokenization_t5.T5Tokenizer'>. This is expected, and simply means that
the `legacy` (previous) behavior will be used so nothing changes for you. If
you want to use the new behaviour, set `legacy=False`. This should only be s
et if you understand what it means, and thoroughly read the reason why this
was added as explained in https://github.com/huggingface/transformers/pull/2
4565
```

```
Classify the intent of the following message using these categories:
- get invoice
- payment issue
- create account
- check invoice
- track refund
- check payment methods
- contact human agent
- change shipping address
- review
- complaint
- get refund
- track order
- check cancellation fee
- switch account
- recover password
- place order
- registration problems
- cancel order
- contact customer service
- edit account
- delivery period
- delete account
- check refund policy
- set up shipping address
- change order
- newsletter subscription
- delivery options
Message:
```

Testing Zero-Shot Intent Classification

You can now test the zero-shot classification capabilities of the `flan-t5-large` model on a subset of messages from the test set. This will provide a sense of how well the model performs without task-specific training.

In [23]:
```python
for message in intents['test']['message'][25:35]:
    print(f"Message: {message}")
    print(f"Predicted Intent: {zero_shot_intent_classification(model, prompt
    print()
```

```
Message: delete Gold account
Predicted Intent: delete account

Message: I am trying to unsubscribe to the newsletter
Predicted Intent: newsletter subscription

Message: what do I need to do to change to the free account?
Predicted Intent: switch account

Message: open anotherstandard account
```

```
Predicted Intent: create account

Message: I'd like to switch to the damn Premium account how to do it
Predicted Intent: switch account

Message: editing standard account
Predicted Intent: edit account

Message: wanna order several items help me
Predicted Intent: place order

Message: problems with standard account terminations
Predicted Intent: contact customer service

Message: want help ti earn some of ur product
Predicted Intent: place order

Message: I need help notifying of a trouble with online payment
Predicted Intent: payment issue
```

Evaluating Zero-Shot Model Performance on the Test Dataset

To evaluate the performance of the `flan-t5-large` zero-shot model on the entire test dataset, we'll:

1. **Generate Predictions**: Use the `zero_shot_intent_classification` function to predict intents for all test messages.
2. **Compare Predictions**: Compare the zero-shot predictions to the true labels in the test set.
3. **Generate a Classification Report**: Use `classification_report` to compute metrics such as precision, recall, and F1-score for each intent category.

In [24]:
```python
from tqdm import tqdm
from sklearn.metrics import classification_report
import pandas as pd

# Generate predictions for the test dataset using the zero-shot model
zero_shot_predictions = [
    zero_shot_intent_classification(model, prompt, message)
    for message in tqdm(intents['test']['message'])
]

# Create a classification report
zero_shot_report = pd.DataFrame(
    classification_report(true_labels_text, zero_shot_predictions, output_di
).T

# Display the report
zero_shot_report
```

```
  0%|
| 0/389 [00:00<?, ?it/s]
  0%|▌
| 1/389 [00:00<00:42,  9.12it/s]
  1%|█
| 2/389 [00:00<00:41,  9.25it/s]
  1%|█▋
| 3/389 [00:00<00:41,  9.29it/s]
  1%|██
| 4/389 [00:00<00:42,  8.97it/s]
  1%|██▋
| 5/389 [00:00<00:45,  8.43it/s]
  2%|███▎
| 6/389 [00:00<00:45,  8.39it/s]
  2%|███▊
| 7/389 [00:00<00:45,  8.46it/s]
  2%|████▎
| 8/389 [00:00<00:43,  8.70it/s]
  2%|████▉
| 9/389 [00:01<00:43,  8.76it/s]
  3%|█████▉
| 11/389 [00:01<00:40,  9.40it/s]
  3%|██████▍
| 12/389 [00:01<00:40,  9.28it/s]
  3%|███████
| 13/389 [00:01<00:40,  9.28it/s]
  4%|███████▌
| 14/389 [00:01<00:40,  9.31it/s]
  4%|████████
| 15/389 [00:01<00:40,  9.34it/s]
  4%|████████▋
| 16/389 [00:01<00:39,  9.38it/s]
  4%|█████████▏
| 17/389 [00:01<00:39,  9.40it/s]
  5%|█████████▋
| 18/389 [00:01<00:39,  9.33it/s]
  5%|██████████▎
| 19/389 [00:02<00:40,  9.25it/s]
  5%|██████████▊
| 20/389 [00:02<00:39,  9.27it/s]
  6%|███████████▊
| 22/389 [00:02<00:37,  9.67it/s]
  6%|████████████▎
| 23/389 [00:02<00:38,  9.51it/s]
  6%|████████████▉
| 24/389 [00:02<00:38,  9.38it/s]
  6%|█████████████▍
| 25/389 [00:02<00:38,  9.36it/s]
```

```
  7%|██████████████
| 26/389 [00:02<00:38,  9.40it/s]
  7%|███████████████
| 28/389 [00:03<00:36,  9.80it/s]
  7%|███████████████
| 29/389 [00:03<00:41,  8.70it/s]
  8%|████████████████
| 30/389 [00:03<00:53,  6.77it/s]
  8%|████████████████
| 31/389 [00:03<00:49,  7.27it/s]
  8%|█████████████████
| 32/389 [00:03<00:46,  7.69it/s]
  8%|██████████████████
| 33/389 [00:03<00:44,  8.02it/s]
  9%|██████████████████
| 34/389 [00:03<00:49,  7.23it/s]
  9%|███████████████████
| 35/389 [00:04<00:51,  6.87it/s]
  9%|███████████████████
| 36/389 [00:04<00:49,  7.19it/s]
 10%|████████████████████
| 37/389 [00:04<00:46,  7.52it/s]
 10%|████████████████████
| 38/389 [00:04<00:43,  8.01it/s]
 10%|█████████████████████
| 39/389 [00:04<00:42,  8.31it/s]
 10%|█████████████████████
| 40/389 [00:04<00:40,  8.52it/s]
 11%|██████████████████████
| 41/389 [00:04<00:40,  8.56it/s]
 11%|██████████████████████
| 42/389 [00:04<00:39,  8.81it/s]
 11%|███████████████████████
| 43/389 [00:05<00:45,  7.57it/s]
 11%|████████████████████████
| 44/389 [00:05<00:46,  7.45it/s]
 12%|████████████████████████
| 45/389 [00:05<00:44,  7.77it/s]
 12%|█████████████████████████
| 46/389 [00:05<00:41,  8.22it/s]
 12%|█████████████████████████
| 47/389 [00:05<00:42,  8.08it/s]
 12%|██████████████████████████
| 48/389 [00:05<00:42,  8.04it/s]
 13%|██████████████████████████
| 49/389 [00:05<00:41,  8.29it/s]
 13%|███████████████████████████
| 50/389 [00:05<00:39,  8.55it/s]
```

```
 13%|███████████████████████████▌
| 51/389 [00:05<00:38,  8.79it/s]
 13%|████████████████████████████
| 52/389 [00:06<00:37,  8.94it/s]
 14%|████████████████████████████▌
| 53/389 [00:06<00:37,  8.98it/s]
 14%|█████████████████████████████▎
| 54/389 [00:06<00:36,  9.07it/s]
 14%|█████████████████████████████▊
| 55/389 [00:06<00:36,  9.10it/s]
 15%|██████████████████████████████▉
| 57/389 [00:06<00:33, 10.00it/s]
 15%|███████████████████████████████▍
| 58/389 [00:06<00:34,  9.68it/s]
 15%|████████████████████████████████▌
| 60/389 [00:06<00:34,  9.67it/s]
 16%|█████████████████████████████████
| 61/389 [00:07<00:34,  9.49it/s]
 16%|█████████████████████████████████▌
| 62/389 [00:07<00:34,  9.45it/s]
 16%|██████████████████████████████████▏
| 63/389 [00:07<00:34,  9.41it/s]
 16%|██████████████████████████████████▋
| 64/389 [00:07<00:34,  9.42it/s]
 17%|███████████████████████████████████▏
| 65/389 [00:07<00:34,  9.40it/s]
 17%|████████████████████████████████████▎
| 67/389 [00:07<00:32,  9.80it/s]
 17%|████████████████████████████████████▊
| 68/389 [00:07<00:34,  9.34it/s]
 18%|█████████████████████████████████████▎
| 69/389 [00:07<00:35,  9.06it/s]
 18%|█████████████████████████████████████▉
| 70/389 [00:08<00:36,  8.78it/s]
 19%|██████████████████████████████████████▉
| 72/389 [00:08<00:34,  9.29it/s]
 19%|███████████████████████████████████████▌
| 73/389 [00:08<00:37,  8.42it/s]
 19%|████████████████████████████████████████
| 74/389 [00:08<00:38,  8.17it/s]
 19%|████████████████████████████████████████▌
| 75/389 [00:08<00:38,  8.12it/s]
 20%|█████████████████████████████████████████▋
| 77/389 [00:08<00:35,  8.86it/s]
 20%|██████████████████████████████████████████▏
| 78/389 [00:08<00:38,  8.12it/s]
 20%|██████████████████████████████████████████▉
| 79/389 [00:09<00:38,  8.07it/s]
```

```
 21%|███████████████████████████████████████████▌
| 80/389 [00:09<00:37,  8.30it/s]
 21%|████████████████████████████████████████████
| 81/389 [00:09<00:36,  8.54it/s]
 21%|█████████████████████████████████████████████
| 83/389 [00:09<00:33,  9.24it/s]
 22%|█████████████████████████████████████████████▋
| 84/389 [00:09<00:34,  8.91it/s]
 22%|██████████████████████████████████████████████▍
| 85/389 [00:09<00:36,  8.44it/s]
 22%|██████████████████████████████████████████████▉
| 86/389 [00:09<00:37,  8.11it/s]
 22%|███████████████████████████████████████████████▍
| 87/389 [00:10<00:37,  8.02it/s]
 23%|████████████████████████████████████████████████
| 88/389 [00:10<00:36,  8.24it/s]
 23%|████████████████████████████████████████████████▌
| 89/389 [00:10<00:35,  8.41it/s]
 23%|█████████████████████████████████████████████████
| 90/389 [00:10<00:34,  8.67it/s]
 23%|█████████████████████████████████████████████████▋
| 91/389 [00:10<00:33,  8.82it/s]
 24%|██████████████████████████████████████████████████▏
| 92/389 [00:10<00:34,  8.59it/s]
 24%|██████████████████████████████████████████████████▋
| 93/389 [00:10<00:33,  8.89it/s]
 24%|███████████████████████████████████████████████████▏
| 94/389 [00:10<00:33,  8.73it/s]
 24%|███████████████████████████████████████████████████▊
| 95/389 [00:10<00:32,  8.93it/s]
 25%|████████████████████████████████████████████████████▎
| 96/389 [00:11<00:32,  9.04it/s]
 25%|████████████████████████████████████████████████████▊
| 97/389 [00:11<00:32,  9.08it/s]
 25%|█████████████████████████████████████████████████████▉
| 99/389 [00:11<00:30,  9.48it/s]
 26%|██████████████████████████████████████████████████████▎
| 100/389 [00:11<00:30,  9.43it/s]
 26%|██████████████████████████████████████████████████████▊
| 101/389 [00:11<00:30,  9.42it/s]
 26%|███████████████████████████████████████████████████████▊
| 103/389 [00:11<00:29,  9.74it/s]
 27%|████████████████████████████████████████████████████████▍
| 104/389 [00:11<00:29,  9.64it/s]
 27%|████████████████████████████████████████████████████████▉
| 105/389 [00:11<00:30,  9.42it/s]
 27%|█████████████████████████████████████████████████████████▌
| 106/389 [00:12<00:30,  9.38it/s]
```

```
 28%|█████████████████████████████████████████████████████████▍
| 107/389 [00:12<00:30,  9.38it/s]
 28%|██████████████████████████████████████████████████████████
| 108/389 [00:12<00:30,  9.34it/s]
 28%|██████████████████████████████████████████████████████████▌
| 109/389 [00:12<00:29,  9.37it/s]
 28%|███████████████████████████████████████████████████████████
| 110/389 [00:12<00:30,  9.29it/s]
 29%|███████████████████████████████████████████████████████████▋
| 111/389 [00:12<00:29,  9.32it/s]
 29%|████████████████████████████████████████████████████████████▏
| 112/389 [00:12<00:29,  9.30it/s]
 29%|████████████████████████████████████████████████████████████▋
| 113/389 [00:12<00:34,  8.01it/s]
 29%|█████████████████████████████████████████████████████████████▎
| 114/389 [00:13<00:36,  7.55it/s]
 30%|█████████████████████████████████████████████████████████████▊
| 115/389 [00:13<00:34,  7.89it/s]
 30%|██████████████████████████████████████████████████████████████▎
| 116/389 [00:13<00:33,  8.16it/s]
 30%|██████████████████████████████████████████████████████████████▊
| 117/389 [00:13<00:32,  8.43it/s]
 30%|███████████████████████████████████████████████████████████████▍
| 118/389 [00:13<00:31,  8.66it/s]
 31%|███████████████████████████████████████████████████████████████▉
| 119/389 [00:13<00:30,  8.80it/s]
 31%|████████████████████████████████████████████████████████████████▍
| 120/389 [00:13<00:30,  8.87it/s]
 31%|█████████████████████████████████████████████████████████████████
| 121/389 [00:13<00:29,  8.98it/s]
 31%|█████████████████████████████████████████████████████████████████▌
| 122/389 [00:13<00:29,  9.00it/s]
 32%|██████████████████████████████████████████████████████████████████
| 123/389 [00:14<00:29,  9.11it/s]
 32%|██████████████████████████████████████████████████████████████████▋
| 124/389 [00:14<00:29,  9.08it/s]
 32%|███████████████████████████████████████████████████████████████████▏
| 125/389 [00:14<00:29,  9.09it/s]
 32%|███████████████████████████████████████████████████████████████████▋
| 126/389 [00:14<00:28,  9.09it/s]
 33%|████████████████████████████████████████████████████████████████████▏
| 127/389 [00:14<00:28,  9.18it/s]
 33%|████████████████████████████████████████████████████████████████████▊
| 128/389 [00:14<00:28,  9.22it/s]
 33%|█████████████████████████████████████████████████████████████████████▎
| 129/389 [00:14<00:28,  9.19it/s]
 33%|█████████████████████████████████████████████████████████████████████▊
| 130/389 [00:14<00:28,  9.19it/s]
```

```
 34%|███████████████████████████████████████████████████████████████████████▎
| 131/389 [00:14<00:27,  9.25it/s]
 34%|███████████████████████████████████████████████████████████████████████▊
| 132/389 [00:15<00:27,  9.26it/s]
 34%|████████████████████████████████████████████████████████████████████████
▍
| 133/389 [00:15<00:27,  9.20it/s]
 35%|████████████████████████████████████████████████████████████████████████
█▌
| 135/389 [00:15<00:26,  9.53it/s]
 35%|████████████████████████████████████████████████████████████████████████
██
| 136/389 [00:15<00:26,  9.51it/s]
 35%|████████████████████████████████████████████████████████████████████████
██▌
| 137/389 [00:15<00:26,  9.48it/s]
 35%|████████████████████████████████████████████████████████████████████████
███▏
| 138/389 [00:15<00:26,  9.45it/s]
 36%|████████████████████████████████████████████████████████████████████████
███▋
| 139/389 [00:15<00:27,  9.06it/s]
 36%|████████████████████████████████████████████████████████████████████████
████▏
| 140/389 [00:15<00:27,  9.10it/s]
 36%|████████████████████████████████████████████████████████████████████████
████▊
| 141/389 [00:15<00:26,  9.20it/s]
 37%|████████████████████████████████████████████████████████████████████████
█████▎
| 142/389 [00:16<00:29,  8.50it/s]
 37%|████████████████████████████████████████████████████████████████████████
█████▊
| 143/389 [00:16<00:30,  8.06it/s]
 37%|████████████████████████████████████████████████████████████████████████
██████▎
| 144/389 [00:16<00:29,  8.25it/s]
 37%|████████████████████████████████████████████████████████████████████████
██████▉
| 145/389 [00:16<00:28,  8.42it/s]
 38%|████████████████████████████████████████████████████████████████████████
███████▍
| 146/389 [00:16<00:28,  8.65it/s]
 38%|████████████████████████████████████████████████████████████████████████
████████
| 147/389 [00:16<00:27,  8.83it/s]
 38%|████████████████████████████████████████████████████████████████████████
████████▌
| 148/389 [00:16<00:26,  8.94it/s]
```

```
 38%|███████████████████████████████████████████████████████████████████████
█████████
| 149/389 [00:16<00:26,  9.04it/s]
 39%|███████████████████████████████████████████████████████████████████████
██████████
| 151/389 [00:17<00:24,  9.55it/s]
 39%|███████████████████████████████████████████████████████████████████████
██████████▌
| 152/389 [00:17<00:24,  9.49it/s]
 39%|███████████████████████████████████████████████████████████████████████
███████████
| 153/389 [00:17<00:25,  9.40it/s]
 40%|███████████████████████████████████████████████████████████████████████
███████████▌
| 154/389 [00:17<00:25,  9.33it/s]
 40%|███████████████████████████████████████████████████████████████████████
████████████▏
| 155/389 [00:17<00:25,  9.23it/s]
 40%|███████████████████████████████████████████████████████████████████████
████████████▋
| 156/389 [00:17<00:25,  9.16it/s]
 40%|███████████████████████████████████████████████████████████████████████
█████████████▏
| 157/389 [00:17<00:25,  9.19it/s]
 41%|███████████████████████████████████████████████████████████████████████
█████████████▊
| 158/389 [00:17<00:25,  9.11it/s]
 41%|███████████████████████████████████████████████████████████████████████
██████████████▎
| 159/389 [00:18<00:27,  8.24it/s]
 41%|███████████████████████████████████████████████████████████████████████
██████████████▉
| 160/389 [00:18<00:30,  7.55it/s]
 41%|███████████████████████████████████████████████████████████████████████
███████████████▌
| 161/389 [00:18<00:29,  7.76it/s]
 42%|███████████████████████████████████████████████████████████████████████
████████████████
| 162/389 [00:18<00:27,  8.14it/s]
 42%|███████████████████████████████████████████████████████████████████████
████████████████▌
| 163/389 [00:18<00:29,  7.64it/s]
 42%|███████████████████████████████████████████████████████████████████████
█████████████████
| 164/389 [00:18<00:28,  7.92it/s]
 42%|███████████████████████████████████████████████████████████████████████
█████████████████▌
| 165/389 [00:18<00:27,  8.26it/s]
 43%|███████████████████████████████████████████████████████████████████████
██████████████████
| 166/389 [00:18<00:26,  8.52it/s]
```

```
 43%|███████████████████████████████████████████████████████████████████████
███████████████████
| 167/389 [00:19<00:25,  8.72it/s]
 43%|███████████████████████████████████████████████████████████████████████
███████████████████
| 168/389 [00:19<00:24,  8.85it/s]
 43%|███████████████████████████████████████████████████████████████████████
████████████████████
| 169/389 [00:19<00:24,  8.91it/s]
 44%|███████████████████████████████████████████████████████████████████████
████████████████████
| 170/389 [00:19<00:24,  9.00it/s]
 44%|███████████████████████████████████████████████████████████████████████
█████████████████████
| 171/389 [00:19<00:24,  8.83it/s]
 44%|███████████████████████████████████████████████████████████████████████
█████████████████████
| 172/389 [00:19<00:24,  8.96it/s]
 44%|███████████████████████████████████████████████████████████████████████
██████████████████████
| 173/389 [00:19<00:23,  9.06it/s]
 45%|███████████████████████████████████████████████████████████████████████
██████████████████████
| 174/389 [00:19<00:25,  8.45it/s]
 45%|███████████████████████████████████████████████████████████████████████
███████████████████████
| 175/389 [00:19<00:27,  7.78it/s]
 45%|███████████████████████████████████████████████████████████████████████
████████████████████████
| 176/389 [00:20<00:26,  8.15it/s]
 46%|███████████████████████████████████████████████████████████████████████
█████████████████████████
| 178/389 [00:20<00:23,  8.95it/s]
 46%|███████████████████████████████████████████████████████████████████████
█████████████████████████
| 179/389 [00:20<00:23,  8.80it/s]
 46%|███████████████████████████████████████████████████████████████████████
██████████████████████████
| 180/389 [00:20<00:23,  8.87it/s]
 47%|███████████████████████████████████████████████████████████████████████
██████████████████████████
| 181/389 [00:20<00:24,  8.56it/s]
 47%|███████████████████████████████████████████████████████████████████████
███████████████████████████
| 182/389 [00:20<00:26,  7.96it/s]
 47%|███████████████████████████████████████████████████████████████████████
███████████████████████████
| 183/389 [00:20<00:26,  7.90it/s]
 47%|███████████████████████████████████████████████████████████████████████
████████████████████████████
| 184/389 [00:21<00:26,  7.80it/s]
```

```
 48%|███████████████████████████████████████████████████████████████████████
████████████████████████████▍
| 185/389 [00:21<00:28,  7.18it/s]
 48%|███████████████████████████████████████████████████████████████████████
████████████████████████████▉
| 186/389 [00:21<00:27,  7.47it/s]
 48%|███████████████████████████████████████████████████████████████████████
█████████████████████████████▍
| 187/389 [00:21<00:26,  7.70it/s]
 48%|███████████████████████████████████████████████████████████████████████
██████████████████████████████
| 188/389 [00:21<00:27,  7.28it/s]
 49%|███████████████████████████████████████████████████████████████████████
██████████████████████████████▌
| 189/389 [00:21<00:31,  6.29it/s]
 49%|███████████████████████████████████████████████████████████████████████
███████████████████████████████
| 190/389 [00:21<00:29,  6.76it/s]
 49%|███████████████████████████████████████████████████████████████████████
███████████████████████████████▌
| 191/389 [00:22<00:26,  7.36it/s]
 49%|███████████████████████████████████████████████████████████████████████
████████████████████████████████▏
| 192/389 [00:22<00:25,  7.84it/s]
 50%|███████████████████████████████████████████████████████████████████████
████████████████████████████████▋
| 193/389 [00:22<00:23,  8.20it/s]
 50%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████▏
| 194/389 [00:22<00:22,  8.51it/s]
 50%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████▊
| 195/389 [00:22<00:22,  8.70it/s]
 50%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████▎
| 196/389 [00:22<00:21,  8.79it/s]
 51%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████▊
| 197/389 [00:22<00:21,  8.92it/s]
 51%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████▍
| 198/389 [00:22<00:21,  8.89it/s]
 51%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████▉
| 199/389 [00:22<00:21,  8.99it/s]
 51%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████▍
| 200/389 [00:23<00:20,  9.02it/s]
 52%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████▉
| 201/389 [00:23<00:20,  9.00it/s]
```

```
 52%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████
| 202/389 [00:23<00:20,  8.99it/s]
 52%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████
| 203/389 [00:23<00:20,  8.99it/s]
 52%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████
| 204/389 [00:23<00:20,  9.07it/s]
 53%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████
| 205/389 [00:23<00:20,  9.13it/s]
 53%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████
| 206/389 [00:23<00:19,  9.16it/s]
 53%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████
| 207/389 [00:23<00:21,  8.46it/s]
 53%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████
| 208/389 [00:23<00:21,  8.37it/s]
 54%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████
| 209/389 [00:24<00:21,  8.34it/s]
 54%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████
| 210/389 [00:24<00:20,  8.55it/s]
 54%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████
| 212/389 [00:24<00:19,  9.22it/s]
 55%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████
| 213/389 [00:24<00:19,  9.23it/s]
 55%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████
| 214/389 [00:24<00:18,  9.22it/s]
 55%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████
| 215/389 [00:24<00:18,  9.24it/s]
 56%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████
| 216/389 [00:24<00:18,  9.25it/s]
 56%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████
| 217/389 [00:24<00:18,  9.31it/s]
 56%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████
| 218/389 [00:25<00:18,  9.27it/s]
 56%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████
| 219/389 [00:25<00:18,  9.31it/s]
```

```
 57%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████
| 220/389 [00:25<00:18,  9.30it/s]
 57%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████
| 221/389 [00:25<00:18,  9.32it/s]
 57%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████
| 222/389 [00:25<00:17,  9.30it/s]
 57%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████
| 223/389 [00:25<00:17,  9.35it/s]
 58%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████
| 224/389 [00:25<00:17,  9.36it/s]
 58%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████
| 225/389 [00:25<00:17,  9.33it/s]
 58%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████
| 226/389 [00:25<00:17,  9.29it/s]
 58%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████
| 227/389 [00:25<00:17,  9.27it/s]
 59%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████
| 229/389 [00:26<00:16,  9.71it/s]
 59%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████
| 230/389 [00:26<00:16,  9.57it/s]
 59%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████
| 231/389 [00:26<00:16,  9.49it/s]
 60%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████
| 232/389 [00:26<00:16,  9.48it/s]
 60%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████
| 233/389 [00:26<00:16,  9.42it/s]
 60%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████
| 234/389 [00:26<00:16,  9.38it/s]
 60%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████
| 235/389 [00:26<00:16,  9.38it/s]
 61%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████
| 236/389 [00:26<00:16,  9.36it/s]
 61%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████
| 238/389 [00:27<00:15,  9.76it/s]
```

```
 61%|████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████
| 239/389 [00:27<00:15,  9.68it/s]
 62%|████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████
| 240/389 [00:27<00:15,  9.57it/s]
 62%|████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████
| 241/389 [00:27<00:15,  9.49it/s]
 62%|████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████
| 242/389 [00:27<00:15,  9.44it/s]
 62%|████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████
| 243/389 [00:27<00:15,  9.45it/s]
 63%|████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████
| 244/389 [00:27<00:15,  9.44it/s]
 63%|████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████
| 245/389 [00:27<00:15,  9.43it/s]
 63%|████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████
| 246/389 [00:27<00:15,  9.42it/s]
 63%|████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████
| 247/389 [00:28<00:15,  9.37it/s]
 64%|████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████
| 248/389 [00:28<00:15,  9.34it/s]
 64%|████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████
| 249/389 [00:28<00:14,  9.34it/s]
 64%|████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████
| 250/389 [00:28<00:14,  9.32it/s]
 65%|████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████
| 251/389 [00:28<00:14,  9.34it/s]
 65%|████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████████
| 252/389 [00:28<00:15,  8.88it/s]
 65%|████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████████
| 254/389 [00:28<00:14,  9.39it/s]
 66%|████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████
| 255/389 [00:28<00:14,  9.37it/s]
 66%|████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████
| 256/389 [00:29<00:14,  9.20it/s]
```

```
 66%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████
| 257/389 [00:29<00:16,  7.78it/s]
 66%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████
| 258/389 [00:29<00:16,  8.10it/s]
 67%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████
| 259/389 [00:29<00:15,  8.36it/s]
 67%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████████████
| 260/389 [00:29<00:15,  8.50it/s]
 67%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████████████
| 262/389 [00:29<00:13,  9.09it/s]
 68%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████████████
| 263/389 [00:29<00:15,  8.38it/s]
 68%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████████
| 264/389 [00:30<00:17,  7.03it/s]
 68%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████
| 265/389 [00:30<00:17,  7.25it/s]
 68%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████
| 266/389 [00:30<00:15,  7.73it/s]
 69%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████████████████
| 267/389 [00:30<00:14,  8.15it/s]
 69%|███████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████████████████
| 268/389 [00:30<00:15,  7.70it/s]
 69%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████████████████
| 269/389 [00:30<00:16,  7.47it/s]
 69%|███████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████████████████
| 270/389 [00:30<00:17,  7.00it/s]
 70%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████████████
| 271/389 [00:31<00:15,  7.38it/s]
 70%|███████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████████████
| 272/389 [00:31<00:15,  7.79it/s]
 70%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
| 273/389 [00:31<00:14,  8.08it/s]
```

```
 70%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
▏                                                             | 274/389 [00:
31<00:13,  8.29it/s]
 71%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█▎                                                            | 276/389 [00:
31<00:12,  9.03it/s]
 71%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█▊                                                            | 277/389 [00:
31<00:12,  9.03it/s]
 71%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██▎                                                           | 278/389 [00:
31<00:12,  9.06it/s]
 72%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██▉                                                           | 279/389 [00:
31<00:12,  9.04it/s]
 72%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███▍                                                          | 280/389 [00:
32<00:12,  9.03it/s]
 72%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███▉                                                          | 281/389 [00:
32<00:11,  9.07it/s]
 72%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████▌                                                         | 282/389 [00:
32<00:11,  9.15it/s]
 73%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████                                                         | 283/389 [00:
32<00:11,  9.17it/s]
 73%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████▌                                                        | 284/389 [00:
32<00:13,  7.52it/s]
 73%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████                                                        | 285/389 [00:
32<00:14,  7.23it/s]
 74%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████▋                                                       | 286/389 [00:
32<00:13,  7.61it/s]
 74%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████▏                                                      | 287/389 [00:
32<00:13,  7.69it/s]
```

```
 74%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████▋                                                      | 288/389 [00:
33<00:12,  8.12it/s]
 74%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████▎                                                     | 289/389 [00:
33<00:11,  8.36it/s]
 75%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████▊                                                     | 290/389 [00:
33<00:11,  8.54it/s]
 75%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████▎                                                    | 291/389 [00:
33<00:11,  8.71it/s]
 75%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████▉                                                    | 292/389 [00:
33<00:10,  8.85it/s]
 75%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████▍                                                   | 293/389 [00:
33<00:10,  8.92it/s]
 76%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████▉                                                   | 294/389 [00:
33<00:10,  8.99it/s]
 76%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████▍                                                  | 295/389 [00:
33<00:10,  8.97it/s]
 76%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████                                                  | 296/389 [00:
33<00:10,  9.02it/s]
 76%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████▌                                                 | 297/389 [00:
34<00:10,  9.05it/s]
 77%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████                                                 | 298/389 [00:
34<00:10,  9.07it/s]
 77%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████▋                                                | 299/389 [00:
34<00:09,  9.15it/s]
 77%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████▏                                               | 300/389 [00:
34<00:09,  9.18it/s]
```

```
 77%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████▋                                               | 301/389 [00:
34<00:09,  9.17it/s]
 78%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████▊                                              | 303/389 [00:
34<00:09,  9.33it/s]
 78%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████▎                                             | 304/389 [00:
34<00:09,  9.08it/s]
 78%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████▊                                             | 305/389 [00:
34<00:09,  8.69it/s]
 79%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████▍                                            | 306/389 [00:
35<00:09,  8.70it/s]
 79%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████▉                                            | 307/389 [00:
35<00:09,  8.83it/s]
 79%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████▍                                           | 308/389 [00:
35<00:09,  8.66it/s]
 79%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████                                           | 309/389 [00:
35<00:10,  7.82it/s]
 80%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████▌                                          | 310/389 [00:
35<00:09,  8.26it/s]
 80%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████                                          | 311/389 [00:
35<00:09,  8.44it/s]
 80%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████▏                                        | 313/389 [00:
35<00:08,  9.13it/s]
 81%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████▋                                        | 314/389 [00:
35<00:08,  9.15it/s]
 81%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████▏                                       | 315/389 [00:
36<00:08,  9.21it/s]
```

```
 81%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████▊                                       | 316/389 [00:
36<00:07,  9.23it/s]
 81%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████▎                                      | 317/389 [00:
36<00:07,  9.29it/s]
 82%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████▊                                      | 318/389 [00:
36<00:07,  9.24it/s]
 82%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████▍                                     | 319/389 [00:
36<00:07,  9.20it/s]
 82%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████▉                                     | 320/389 [00:
36<00:07,  9.14it/s]
 83%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████▍                                    | 321/389 [00:
36<00:07,  9.16it/s]
 83%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████                                    | 322/389 [00:
36<00:07,  9.22it/s]
 83%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████                                   | 324/389 [00:
37<00:06, 10.02it/s]
 84%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████▌                                  | 325/389 [00:
37<00:06,  9.81it/s]
 84%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████▏                                 | 326/389 [00:
37<00:06,  9.21it/s]
 84%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████▏                                | 328/389 [00:
37<00:06,  9.43it/s]
 85%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████▊                                | 329/389 [00:
37<00:07,  8.52it/s]
 85%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████▎                               | 330/389 [00:
37<00:07,  7.82it/s]
```

```
 85%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████                               | 331/389 [00:
37<00:07,  7.67it/s]
 85%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████                              | 332/389 [00:
38<00:07,  8.03it/s]
 86%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████                              | 333/389 [00:
38<00:06,  8.13it/s]
 86%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████                             | 334/389 [00:
38<00:06,  8.18it/s]
 86%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████                             | 335/389 [00:
38<00:06,  8.41it/s]
 86%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████                            | 336/389 [00:
38<00:06,  8.62it/s]
 87%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████                           | 338/389 [00:
38<00:05,  9.25it/s]
 87%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████                           | 339/389 [00:
38<00:05,  9.29it/s]
 87%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████                          | 340/389 [00:
38<00:05,  9.31it/s]
 88%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████                          | 341/389 [00:
38<00:05,  9.31it/s]
 88%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████                         | 342/389 [00:
39<00:05,  9.29it/s]
 88%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████                        | 343/389 [00:
39<00:05,  8.48it/s]
 88%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████                        | 344/389 [00:
39<00:05,  8.43it/s]
```

```
 89%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████▎                       | 345/389 [00:
39<00:05,  8.62it/s]
 89%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████▉                       | 346/389 [00:
39<00:04,  8.83it/s]
 89%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████▍                      | 347/389 [00:
39<00:05,  7.86it/s]
 89%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████▉                      | 348/389 [00:
39<00:05,  8.19it/s]
 90%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████▌                     | 349/389 [00:
39<00:04,  8.45it/s]
 90%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████                     | 350/389 [00:
40<00:04,  8.68it/s]
 90%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████▌                    | 351/389 [00:
40<00:04,  8.82it/s]
 90%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████                    | 352/389 [00:
40<00:04,  8.90it/s]
 91%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████▋                   | 353/389 [00:
40<00:04,  8.97it/s]
 91%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████▏                  | 354/389 [00:
40<00:03,  9.03it/s]
 91%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████▋                  | 355/389 [00:
40<00:03,  9.05it/s]
 92%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████▎                 | 356/389 [00:
40<00:03,  9.08it/s]
 92%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████▊                 | 357/389 [00:
40<00:03,  9.12it/s]
```

```
 92%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████▎                | 358/389 [00:
40<00:03,  9.14it/s]
 92%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████▉                | 359/389 [00:
41<00:03,  9.17it/s]
 93%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████▍               | 360/389 [00:
41<00:03,  9.16it/s]
 93%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████▉               | 361/389 [00:
41<00:03,  9.18it/s]
 93%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████▍              | 362/389 [00:
41<00:02,  9.13it/s]
 93%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████              | 363/389 [00:
41<00:02,  9.18it/s]
 94%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████             | 365/389 [00:
41<00:02,  9.62it/s]
 94%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████▏           | 367/389 [00:
41<00:02, 10.21it/s]
 95%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████▎          | 369/389 [00:
42<00:02,  9.56it/s]
 95%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████▊          | 370/389 [00:
42<00:02,  9.44it/s]
 95%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████▎         | 371/389 [00:
42<00:02,  8.27it/s]
 96%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████▊         | 372/389 [00:
42<00:02,  7.74it/s]
 96%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████▉        | 374/389 [00:
42<00:01,  8.38it/s]
```

```
 96%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████▍       | 375/389 [00:
42<00:01,  8.48it/s]
 97%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████       | 376/389 [00:
42<00:01,  8.66it/s]
 97%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████▌      | 377/389 [00:
43<00:01,  8.81it/s]
 97%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████      | 378/389 [00:
43<00:01,  8.88it/s]
 97%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████▋     | 379/389 [00:
43<00:01,  8.79it/s]
 98%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████▏    | 380/389 [00:
43<00:01,  8.90it/s]
 98%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████▋    | 381/389 [00:
43<00:00,  8.63it/s]
 98%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████▏   | 382/389 [00:
43<00:00,  7.70it/s]
 98%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████▊   | 383/389 [00:
43<00:00,  8.01it/s]
 99%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████▎  | 384/389 [00:
43<00:00,  8.32it/s]
 99%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████▊  | 385/389 [00:
44<00:00,  8.50it/s]
 99%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████▍ | 386/389 [00:
44<00:00,  8.68it/s]
 99%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████▉ | 387/389 [00:
44<00:00,  8.79it/s]
```

```
100%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████▉| 388/389 [00:
44<00:00,  8.95it/s]
100%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████| 389/389 [00:
44<00:00,  8.64it/s]
100%|███████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████| 389/389 [00:
44<00:00,  8.75it/s]
```

Out[24]:

	precision	recall	f1-score	support
cancel order	0.750000	0.642857	0.692308	14.000000
change order	1.000000	0.500000	0.666667	18.000000
change shipping address	0.275862	1.000000	0.432432	8.000000
check cancellation fee	1.000000	0.727273	0.842105	11.000000
check invoice	1.000000	0.461538	0.631579	13.000000
check payment methods	1.000000	1.000000	1.000000	13.000000
check refund policy	0.789474	0.937500	0.857143	16.000000
complaint	1.000000	0.538462	0.700000	13.000000
contact customer service	0.160000	0.727273	0.262295	11.000000
contact human agent	0.740741	1.000000	0.851064	20.000000
create account	0.727273	1.000000	0.842105	16.000000
delete account	1.000000	0.777778	0.875000	18.000000
delivery options	1.000000	0.176471	0.300000	17.000000
delivery period	1.000000	0.600000	0.750000	10.000000
edit account	0.833333	1.000000	0.909091	10.000000
get invoice	0.928571	0.928571	0.928571	14.000000
get refund	0.416667	1.000000	0.588235	10.000000
newsletter subscription	1.000000	1.000000	1.000000	11.000000
payment issue	0.733333	1.000000	0.846154	11.000000
place order	0.761905	0.941176	0.842105	17.000000
recover password	1.000000	0.650000	0.787879	20.000000
registration problems	1.000000	0.647059	0.785714	17.000000
review	0.947368	0.947368	0.947368	19.000000
set up shipping address	0.000000	0.000000	0.000000	11.000000
switch account	1.000000	0.368421	0.538462	19.000000
track order	0.866667	0.812500	0.838710	16.000000
track refund	1.000000	0.125000	0.222222	16.000000
accuracy	0.712082	0.712082	0.712082	0.712082
macro avg	0.812266	0.722565	0.701378	389.000000

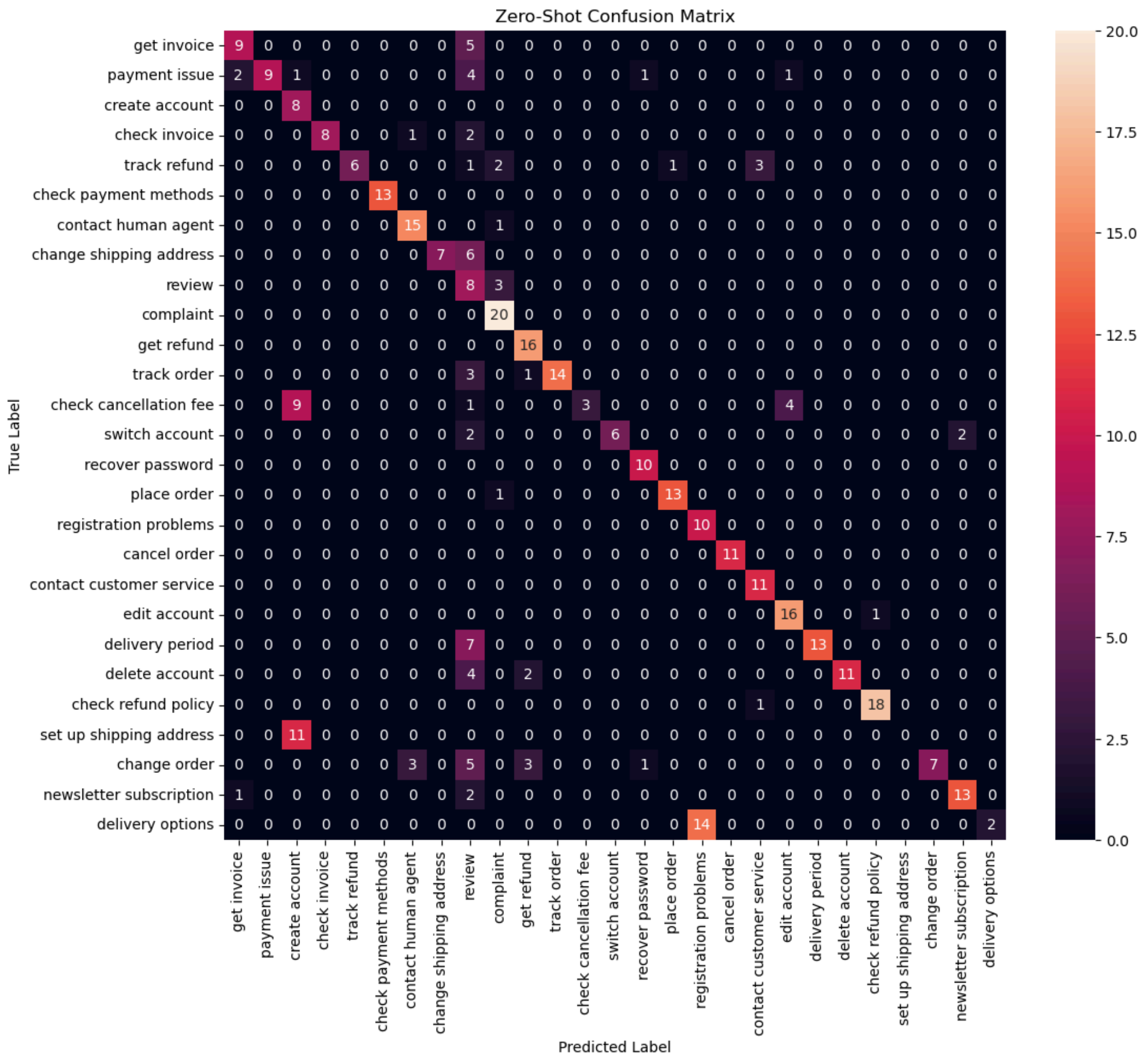
	precision	recall	f1-score	support
weighted avg	0.843022	0.712082	0.712306	389.000000

Incredibly, our accuracy using zero-shot learning is around **70%**, which is significantly higher than our fine-tuned model! This demonstrates the power of large language models and their ability to generalize to new tasks without explicit training.

```
In [25]: # Confusion Matrix

         # Compute confusion matrix
         zero_shot_conf_matrix = confusion_matrix(true_labels_text, zero_shot_predict

         # Plot the confusion matrix
         plt.figure(figsize=(12, 10))
         sns.heatmap(zero_shot_conf_matrix, annot=True, fmt='d', xticklabels=id2label
         plt.xlabel('Predicted Label')
         plt.ylabel('True Label')
         plt.title('Zero-Shot Confusion Matrix')
         plt.show()
```



Conclusion

In this lab, you explored how to classify customer text messages into different intent categories using both a **fine-tuned model** and a **zero-shot learning approach**. Along the way, you learned about key concepts in NLP and machine learning, such as tokenization, data preprocessing, and evaluation metrics.

Key Takeaways:

1. **Fine-Tuning with Transformers**:
   - You fine-tuned the `T5-small` model for intent classification.
   - By adapting a pre-trained model, you achieved an accuracy of around **55%** after training, which is a significant improvement over random guessing (~3.7% accuracy for 27 classes).
2. **Understanding Model Predictions**:
   - You evaluated the model's predictions using metrics like precision, recall, and F1-score, and visualized the results with a confusion matrix.
   - You implemented a method to handle uncertainty, ensuring the model flagged low-confidence predictions for manual review.
3. **Zero-Shot Learning**:
   - Using the `flan-t5-large` model, you explored the power of zero-shot learning, achieving an impressive accuracy of around **70%** without any task-specific training.
   - This highlighted the flexibility and capability of large language models to generalize across tasks.

Real-World Implications:

- The methods you practiced in this lab can be applied to automate tasks like customer service classification, intent detection in chatbots, or filtering messages for human review.
- Zero-shot learning offers a quick way to prototype solutions when labeled data is scarce, while fine-tuning allows for highly customized and accurate models with enough training data.