Objective: Classify insurance claims as claims or not claims based on claims notes. Derive summaries and titles from claim notes"

In [18]:
```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import torch
from torch.utils.data import Dataset, DataLoader
import torch.nn.functional as F
from transformers import BertTokenizer, BertForSequenceClassification
import pandas as pd
from sklearn.model_selection import train_test_split
from datasets import Dataset
from transformers import BartForConditionalGeneration, BartTokenizer, Trainer, Trai
```

Dataset from huggingface

Import from the API or read the csv directly

In [ ]:
```
#download insurance claims dataset from hugging face

import requests

#API endpoints for rows

url = "https://datasets-server.huggingface.co/rows"


#specify params

params = {
    "dataset": "infinite-dataset-hub/TextClaimsDataset",
    "config": "default",
    "split": "train",
        "offset": 0,       # Starting row
    "length": 100,     # Number of rows to fetch
}

#make GET request
response = requests.get(url, params=params)

#check response status

if response.status_code == 200:
    #parse json and concert to dataframe

    data=response.json()
    rows=pd.DataFrame(data["rows"])
    print(rows.head())
```

```
else:
    print(f"Failed to fetch rows: {response.status_code} - {response.text}")
```

In [2]:
```
import pandas as pd

df = pd.read_csv("hf://datasets/infinite-dataset-hub/TextClaimsDataset/data.csv")
```

In [3]:
```
print(df)
```

```
   idx                                               Text        Label
0    0  The policyholder reported a burglary at their ...        Claim
1    1  I purchased a new phone and am not satisfied w...  Not a Claim
2    2  An individual claimed to have been involved in...        Claim
3    3  The user is asking for assistance with underst...  Not a Claim
4    4  A homeowner filed a claim after their property...        Claim
..  ...                                                ...          ...
92  92  The insured party is seeking information on ho...  Not a Claim
93  93  A claim for lost income has been filed by an e...        Claim
94  94  An individual is asking about the claims proce...  Not a Claim
95  96  There's a community event at the park this Sat...  Not a Claim
96  99  The pet owner submitted a claim for their lost...        Claim

[97 rows x 3 columns]
```

In [4]:
```
def label_claims(text):
    if text=="Claim":
        return 1
    else:
        return 0
df["labels_binary"] = df["Label"].apply(label_claims).astype(int)

print(df)

#print(df[["Text", "labels"]].head())
```

```
    idx                                                Text        Label  \
0     0  The policyholder reported a burglary at their ...        Claim
1     1  I purchased a new phone and am not satisfied w...  Not a Claim
2     2  An individual claimed to have been involved in...        Claim
3     3  The user is asking for assistance with underst...  Not a Claim
4     4  A homeowner filed a claim after their property...        Claim
..  ...                                                ...          ...
92   92  The insured party is seeking information on ho...  Not a Claim
93   93  A claim for lost income has been filed by an e...        Claim
94   94  An individual is asking about the claims proce...  Not a Claim
95   96  There's a community event at the park this Sat...  Not a Claim
96   99  The pet owner submitted a claim for their lost...        Claim

    labels_binary
0               1
1               0
2               1
3               0
4               1
..            ...
92              0
93              1
94              0
95              0
96              1

[97 rows x 4 columns]
```

In [5]:
```
# Ensure all text entries are strings
df["Text"] = df["Text"].astype(str).fillna("")
```

In [6]:
```
print(type(df['Text']))
```

```
<class 'pandas.core.series.Series'>
```

In [7]:
```
df = df.drop(columns=['idx'])
```

In [8]:
```
# Create a new column with the index numbers
df['idx'] = df.index
```

In [9]:
```
import pandas as pd
import torch
from torch.utils.data import Dataset, DataLoader
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
from transformers import BertTokenizer, BertForSequenceClassification, AdamW

df = df[["Text", "labels_binary"]]

# Split the data
train_texts, val_texts, train_labels, val_labels = train_test_split(
    df["Text"], df["labels_binary"], test_size=0.2, random_state=42
)
```

In [10]:
```
# Initialize tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Tokenize the data
train_encodings = tokenizer(list(train_texts), truncation=True, padding=True, max_l
val_encodings = tokenizer(list(val_texts), truncation=True, padding=True, max_lengt

# Convert labels to tensors (ensure labels are of type LongTensor)
train_labels = torch.tensor(list(train_labels.values), dtype=torch.long)
val_labels = torch.tensor(list(val_labels.values), dtype=torch.long)
```

In [11]:
```
# Create Dataset class
class ClaimsDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        item = {key: val[idx] for key, val in self.encodings.items()}
        item["labels"] = self.labels[idx]
        return item

train_dataset = ClaimsDataset(train_encodings, train_labels)
val_dataset = ClaimsDataset(val_encodings, val_labels)
```

In [12]:
```
# Initialize DataLoader
train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=8)

# Load pre-trained BERT model
model = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labe

# Define optimizer
optimizer = AdamW(model.parameters(), lr=5e-5)

# Define training loop
device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
model.to(device)
```

```
Some weights of BertForSequenceClassification were not initialized from the model ch
eckpoint at bert-base-uncased and are newly initialized: ['classifier.bias', 'classi
fier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for
predictions and inference.
C:\Users\sukri\anaconda3\envs\env\Lib\site-packages\transformers\optimization.py:59
1: FutureWarning: This implementation of AdamW is deprecated and will be removed in
a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `
no_deprecation_warning=True` to disable this warning
  warnings.warn(
```

Out[12]:
```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

In [13]:
```python
# Training and Evaluation
epochs = 3
for epoch in range(epochs):
    model.train()
    for batch in train_loader:
        batch = {key: val.to(device) for key, val in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
```

```
        optimizer.zero_grad()

    # Evaluate the model
    model.eval()
    val_preds = []
    val_labels = []
    with torch.no_grad():
        for batch in val_loader:
            batch = {key: val.to(device) for key, val in batch.items()}
            outputs = model(**batch)
            preds = torch.argmax(outputs.logits, dim=1)
            val_preds.extend(preds.cpu().numpy())
            val_labels.extend(batch["labels"].cpu().numpy())

    accuracy = accuracy_score(val_labels, val_preds)
    precision, recall, f1, _ = precision_recall_fscore_support(val_labels, val_pred
    print(f"Epoch {epoch + 1}: Accuracy: {accuracy:.4f}, Precision: {precision:.4f}
```

```
Epoch 1: Accuracy: 0.9000, Precision: 1.0000, Recall: 0.8000, F1: 0.8889
Epoch 2: Accuracy: 0.9500, Precision: 1.0000, Recall: 0.9000, F1: 0.9474
Epoch 3: Accuracy: 0.9000, Precision: 0.9000, Recall: 0.9000, F1: 0.9000
```

In [42]:
```
# Save the fine-tuned model
model.save_pretrained("./fine_tuned_bert")
tokenizer.save_pretrained("./fine_tuned_bert")
```

Out[42]:
```
('./fine_tuned_bert\\tokenizer_config.json',
 './fine_tuned_bert\\special_tokens_map.json',
 './fine_tuned_bert\\vocab.txt',
 './fine_tuned_bert\\added_tokens.json')
```

In [43]:
```
from sklearn.metrics import classification_report

# Put the model in evaluation mode
model.eval()

# Initialize lists to store predictions and true labels
val_preds = []
val_labels_list = []

with torch.no_grad():
    for batch in val_loader:
        # Move data to device (GPU or CPU)
        batch = {key: val.to(device) for key, val in batch.items()}

        # Forward pass to get predictions
        outputs = model(**batch)

        # Get the predicted class (0 or 1)
        preds = torch.argmax(outputs.logits, dim=1)

        # Store predictions and true labels
        val_preds.extend(preds.cpu().numpy())
        val_labels_list.extend(batch["labels"].cpu().numpy())

# Evaluation metrics
```

```
print("Classification Report:")
print(classification_report(val_labels_list, val_preds, target_names=["Not a Claim
```

```
# Optionally, print predictions alongside the actual texts
val_results = pd.DataFrame({
    "Text": val_texts.reset_index(drop=True),
    "True_Label": val_labels_list,
    "Predicted_Label": val_preds
})
print(val_results.head())
```

```
Classification Report:
                precision    recall  f1-score   support

Not a Claim (0)       0.90      0.90      0.90        10
      Claim (1)       0.90      0.90      0.90        10

       accuracy                           0.90        20
      macro avg       0.90      0.90      0.90        20
   weighted avg       0.90      0.90      0.90        20

                                                 Text  True_Label  \
0  A customer called to dispute a charge on their...           0
1          An individual wants to report a lost pet.           0
2  A claim for lost income has been filed by an e...           1
3  My friend recommended a new restaurant that se...           0
4  The insured reports that their home security s...           1

   Predicted_Label
0                1
1                0
2                1
3                0
4                1
```

Using GPT for Claim Summarization *Objective: Summarize the insurance claim for customer communication.*

In [14]:
```
df['Text_Length'] = df['Text'].apply(lambda x: len(x))
print(df[['Text', 'Text_Length']].head())
```

```
                                                 Text  Text_Length
0  The policyholder reported a burglary at their ...           94
1  I purchased a new phone and am not satisfied w...           66
2  An individual claimed to have been involved in...          100
3  The user is asking for assistance with underst...           85
4  A homeowner filed a claim after their property...           81
```

In [15]:
```
df['Text_Word_Count'] = df['Text'].apply(lambda x: len(x.split()))
print(df[['Text', 'Text_Word_Count']].head())
```

	Text	Text_Word_Count
0	The policyholder reported a burglary at their ...	16
1	I purchased a new phone and am not satisfied w...	12
2	An individual claimed to have been involved in...	17
3	The user is asking for assistance with underst...	12
4	A homeowner filed a claim after their property...	14

In [30]:
```
from transformers import GPT2LMHeadModel, GPT2Tokenizer
import torch

# Load Pretrained GPT2 Model and Tokenizer

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

# Define pad_token_id explicitly since GPT2 doesn't have a pad token
tokenizer.pad_token = tokenizer.eos_token  # Use eos_token as pad token
model = GPT2LMHeadModel.from_pretrained('gpt2')

# Ensure the model is in evaluation mode
model.eval()

# Function to generate summaries for a given claim
def generate_summary(claim, max_length=50):
    # Encode the input claim (the text to be summarized)
    inputs = tokenizer.encode(claim, return_tensors='pt', max_length=512, truncatio

    # Create attention mask (1 for real tokens, 0 for padding)
    attention_mask = torch.ones(inputs.shape, dtype=torch.long)

    # Ensure attention mask is set correctly
    attention_mask[inputs == tokenizer.pad_token_id] = 0

    # Generate summary using the GPT-2 model
    outputs = model.generate(
        inputs,
        attention_mask=attention_mask,  # Pass attention mask to the model
        pad_token_id=tokenizer.pad_token_id,  # Explicitly set pad_token_id
        max_length=max_length,
        num_return_sequences=1,  # Generate only one summary
        no_repeat_ngram_size=2,  # Avoid repeating phrases
        temperature=0.7,  # Controls randomness of predictions
        top_k=50,  # Top-k sampling for diversity
        top_p=0.95,  # Top-p (nucleus) sampling for diversity
        do_sample=True,  # Enable sampling (as opposed to greedy search)
    )

    # Decode the output summary
    summary = tokenizer.decode(outputs[0], skip_special_tokens=True)

    # Ensure the summary is shorter than the input (if needed)
    #if len(summary.split()) > len(claim.split()):
        #summary = "Summary too long, trimming..."

    return summary.strip()

# Apply summarization to each row in the 'Text' column
```

```
df['Summary'] = df['Text'].apply(lambda x: generate_summary(x))

# Show the resulting dataframe with summaries
print(df[['Text', 'Summary']].head())
```

```
                                                Text  \
0  The policyholder reported a burglary at their ...
1  I purchased a new phone and am not satisfied w...
2  An individual claimed to have been involved in...
3  The user is asking for assistance with underst...
4  A homeowner filed a claim after their property...

                                             Summary
0  The policyholder reported a burglary at their ...
1  I purchased a new phone and am not satisfied w...
2  An individual claimed to have been involved in...
3  The user is asking for assistance with underst...
4  A homeowner filed a claim after their property...
```

In [31]:
```
df['Summary_Length'] = df['Summary'].apply(lambda x: len(x))
print(df[['Text', 'Summary', 'Summary_Length']].head())
```

```
                                                Text  \
0  The policyholder reported a burglary at their ...
1  I purchased a new phone and am not satisfied w...
2  An individual claimed to have been involved in...
3  The user is asking for assistance with underst...
4  A homeowner filed a claim after their property...

                                             Summary  Summary_Length
0  The policyholder reported a burglary at their ...             149
1  I purchased a new phone and am not satisfied w...             221
2  An individual claimed to have been involved in...             249
3  The user is asking for assistance with underst...             256
4  A homeowner filed a claim after their property...             251
```

Train BART

In [19]:
```
dataset = Dataset.from_pandas(df)
dataset = dataset.train_test_split(test_size=0.1)  # Split into train and test
```

In [20]:
```
# Load BART Model and Tokenizer
model_name = "facebook/bart-large-cnn"
tokenizer = BartTokenizer.from_pretrained(model_name)
model = BartForConditionalGeneration.from_pretrained(model_name)
```

In [21]:
```
# Summarize Each Row in the 'text' Column
def summarize_text(text):
    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length
    summary_ids = model.generate(inputs, max_length=10, min_length=5, length_penalt
    return tokenizer.decode(summary_ids[0], skip_special_tokens=True)
```

In [22]:
```
df["summary"] = df["Text"].apply(summarize_text)
```

```
# Print Summaries
print(df)
```

```
                                                 Text  labels_binary  \
0   The policyholder reported a burglary at their ...              1
1   I purchased a new phone and am not satisfied w...              0
2   An individual claimed to have been involved in...              1
3   The user is asking for assistance with underst...              0
4   A homeowner filed a claim after their property...              1
..                                                ...            ...
92  The insured party is seeking information on ho...              0
93  A claim for lost income has been filed by an e...              1
94  An individual is asking about the claims proce...              0
95  There's a community event at the park this Sat...              0
96  The pet owner submitted a claim for their lost...              1

    Text_Length  Text_Word_Count                                    summary
0            94               16      The policyholder reported a burglary at
1            66               12                    summarize: I purchased a
2           100               17  An individual claimed to have been involved
3            85               12                      summarize: The user is
4            81               14        A homeowner filed a claim after their
..          ...              ...                                         ...
92           80               13  The insured party is seeking information on
93          114               24             A claim for lost income has been
94           87               13     An individual is asking about the claims
95           52                9             There's a community event at the
96          110               19     The dog was stolen during a neighborhood

[97 rows x 5 columns]
```

In [23]:
```
df['Summary_Word_Count'] = df['summary'].apply(lambda x: len(x.split()))
print(df[['Text','summary', 'Summary_Word_Count']].head())
```

```
                                                Text  \
0  The policyholder reported a burglary at their ...
1  I purchased a new phone and am not satisfied w...
2  An individual claimed to have been involved in...
3  The user is asking for assistance with underst...
4  A homeowner filed a claim after their property...

                                    summary  Summary_Word_Count
0      The policyholder reported a burglary at                  6
1                    summarize: I purchased a                  4
2  An individual claimed to have been involved                  7
3                      summarize: The user is                  4
4        A homeowner filed a claim after their                  7
```

Train T5

In [ ]:
```
#!pip install SentencePiece
```

In [24]:
```
# Load Pretrained T5 Model and Tokenizer
from transformers import T5ForConditionalGeneration, T5Tokenizer
model_name = "t5-small"
```

```
tokenizer = T5Tokenizer.from_pretrained(model_name)
model = T5ForConditionalGeneration.from_pretrained(model_name)
```

```
You are using the default legacy behaviour of the <class 'transformers.models.t5.tok
enization_t5.T5Tokenizer'>. This is expected, and simply means that the `legacy` (pr
evious) behavior will be used so nothing changes for you. If you want to use the new
behaviour, set `legacy=False`. This should only be set if you understand what it mea
ns, and thoroughly read the reason why this was added as explained in https://githu
b.com/huggingface/transformers/pull/24565
```

In [25]:
```
# Summarize Each Row in the 'text' Column
def summarize_text(text):
    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length
    summary_ids = model.generate(inputs, max_length=10, min_length=5, length_penalt
    return tokenizer.decode(summary_ids[0], skip_special_tokens=True)

df["summary"] = df["Text"].apply(summarize_text)

# Print Summaries
print(df)
```

```
                                                 Text  labels_binary  \
0   The policyholder reported a burglary at their ...              1
1   I purchased a new phone and am not satisfied w...              0
2   An individual claimed to have been involved in...              1
3   The user is asking for assistance with underst...              0
4   A homeowner filed a claim after their property...              1
..                                                ...            ...
92  The insured party is seeking information on ho...              0
93  A claim for lost income has been filed by an e...              1
94  An individual is asking about the claims proce...              0
95  There's a community event at the park this Sat...              0
96  The pet owner submitted a claim for their lost...              1

    Text_Length  Text_Word_Count  \
0            94               16
1            66               12
2           100               17
3            85               12
4            81               14
..          ...              ...
92           80               13
93          114               24
94           87               13
95           52                9
96          110               19

                                              summary  Summary_Word_Count
0          policyholder reported a burglary at their                   6
1                      i purchased a new phone and am                   4
2      individual claimed to have been involved in a                    7
3   the user is asking for assistance with underst...                   4
4              a homeowner filed a claim after their                    7
..                                                ...                 ...
92  the insured party is seeking information on ho...                   7
93            an employee was injured on the job and                    7
94  an individual is asking about the claims proce...                   7
95                  there's a community event at the                    6
96         the pet owner submitted a claim for their                    7

[97 rows x 6 columns]
```

Extracting title

In [26]:
```
# Summarize Each Row in the 'text' Column
def generate_title(text):
    inputs = tokenizer.encode("generate a concise claim title: ", text, return_tens
    summary_ids = model.generate(inputs,
                                 max_length=8,
                                 min_length=5,
                                 length_penalty=2.0, # Encourage concise output
                                 num_beams=6,   ## More beams for better exploratio
                                 early_stopping=True)
    return tokenizer.decode(summary_ids[0], skip_special_tokens=True)

df["Title"] = df["Text"].apply(generate_title)
```

```
# Print Titles
print(df[["Text", "Title"]])
```

```
                                                 Text  \
0   The policyholder reported a burglary at their ...
1   I purchased a new phone and am not satisfied w...
2   An individual claimed to have been involved in...
3   The user is asking for assistance with underst...
4   A homeowner filed a claim after their property...
..                                                ...
92  The insured party is seeking information on ho...
93  A claim for lost income has been filed by an e...
94  An individual is asking about the claims proce...
95  There's a community event at the park this Sat...
96  The pet owner submitted a claim for their lost...

                                    Title
0               home and submitted a claim
1                       a new phone and am
2                  have been involved in a
3   The user is asking for assistance with
4      after their property was damaged by
..                                     ...
92        is seeking information on how to
93           lost income has been filed by
94              the process for a business
95                a community event at the
96             owner submitted a claim for

[97 rows x 2 columns]
```