For your ease of reading, I put my discussions here, and my ipynb code below, for reference.

1 BERT classifier

Table 1: Probabilities of Negative and Positive Sentiments from BERT classifier

	Negative	Positive
Strongly positive	0.00670665	0.99329346
Mildly positive	0.00624877	0.99375120
Strongly negative	0.99797410	0.00202591
Mildly negative	0.99895990	0.00104012
Off-topic	0.30161944	0.69838053

The model did a very good job classifying the positivity of the movie review. The probabilities predictions from the model are very close to 1 for review 1-4, indicating a very high confidence in classification. This indicates that our model is performing well on the task to make reliable predictions.

However, the model's classification of the off-topic comment as more negative than expected is worth exploring further. It's likely because of that the model may have been trained on the dataset that did not include enough off-topic comments (i.e, we may put label 0.5 there), which could have affected its ability to accurately classify them.

This aligns well with the previous train/dev accuracy I have seen through out the training process.

2 Causal LM

Table 2: **Normalized** probability of Negative and Positive Sentiments from Causal LM

	Negative	Positive
Strongly positive	0.45442782	0.54557218
Mildly positive	0.15504855	0.84495145
Mildly negative	0.919746	0.080254
Strongly negative	0.87803117	0.12196883
Off-topic	0.59710061	0.40289939

The results of the Causal LM in analyzing the polarity of comments were satisfactory. However, there was a surprising outcome in the case of the second review, which was classified as more positive (with a higher relative probability of being positive) than the first review, which was expected to be classified as "strongly positive". This difference from expectations could be due to the limitations of the model, as it was not specifically designed for the classification tasks (more for generation).

On the other hand, the model performed as expected for reviews 3 to 5, correctly classifying them based on their sentiment. The off-topic comment was slightly classified as more positive, but the difference in the probability was very small.

Overall, while the Causal LM did a reasonably good job in classifying the sentiment of the comments, its performance was not as robust as that of our fine-tuned classifier model, which is specifically trained for the classification task. This highlights the importance of fine-tuning models for specific tasks, as it can lead to more accurate results.

3 GPT-4 (LLM)

Response from GPT-4

a.) Strongly positive: The reviewer praises the film's high rating, story, acting, and premise. They emphasize the long-lasting impact it has on viewers and its themes of friendship, hope, and overcoming adversity. The perfect score (10/10) also indicates a strongly positive review.

b.) Mildly positive: The reviewer finds the film decent for an after-school special, with interesting events happening and some character development. They mention a good feeling and a nice smile left on the face after watching it, but also describe it as plain and good, suggesting a mildly positive review.

c.) Mildly negative: The reviewer criticizes Robert Taylor's performance as dull and unoriginal, and finds fault with the film's storyline and climax. They do, however, give some credit to the other actors and the cinematography. Overall, the review seems to suggest a mildly negative opinion of the film.

d.) Mildly negative: The reviewer calls the film the worst of the series and mentions that it is boring, has forgettable characters, and slow pacing. However, they also acknowledge some aspects of Eastwood's performance as uniquely fitting. Overall, the review suggests a mildly negative opinion on the film.

e.) My apologies, this statement isn't a movie review. Please provide a movie review for me to analyze its positivity.

I used GPT-4 to analyze the positivity the reviews, with the prompt: "Please help me analyze the positivity of the following five movie reviews, please be coherent and short, answer how positive it is (strongly/mildly positive/negative) with a very short explanation."

The model's performance was very impressive, accurately assessing the positivity of all reviews and providing succinct explanations. This even includes the off-topic review. However, the classification of review 4 could have been more negative, which is what we expected.

Remarkably, GPT-4's performance may even surpass that of our fine-tuned model, as it demonstrated an exceptional ability to evaluate the sentiment of the movie reviews, with provided explainations. This outcome suggests that LLMs like GPT-4 can be an effective tool for specific tasks, including sentiment analysis tasks, providing coherent and concise evaluations while maintaining a high degree of accuracy.

```
import numpy as np
import pandas as pd
import torch
from torch.utils.data import DataLoader
from torch.utils.data import Dataset
import torch.nn as nn
from tqdm import tqdm
import matplotlib.pyplot as plt
from types import SimpleNamespace
```

Task 4.1, BERT model

```
!pip install tokenizers
!pip install transformers
!pip install openai
```

```
from google.colab import drive
drive.mount('/content/drive')
folder = '/content/drive/MyDrive/CSC401/'
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import sys
sys.path.append(folder)
from classifier import *
from tokenizer import BertTokenizer
```

```
config = {'hidden_dropout_prob': 0.3,
          'num_labels': 2,
          'hidden_size': 768,
          'option': 'flexible'}
config = SimpleNamespace(**config)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
model = BertSentClassifier(config)
model.to(device)
```

```
BertSentClassifier(
  (bert): BertModel(
    (word_embedding): Embedding(30522, 768, padding_idx=0)
    (pos_embedding): Embedding(512, 768)
    (tk_type_embedding): Embedding(2, 768)
    (embed_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (embed_dropout): Dropout(p=0.1, inplace=False)
    (bert_layers): ModuleList(
      (0-11): 12 x BertLayer(
        (self_attention): BertSelfAttention(
          (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)
        )
        (attention_dense): Linear(in_features=768, out_features=768, bias=True)
        (attention_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (attention_dropout): Dropout(p=0.1, inplace=False)
        (interm_dense): Linear(in_features=768, out_features=3072, bias=True)
        (out_dense): Linear(in_features=3072, out_features=768, bias=True)
        (out_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (out_dropout): Dropout(p=0.1, inplace=False)
      )
    )
    (pooler_dense): Linear(in_features=768, out_features=768, bias=True)
    (pooler_af): Tanh()
  )
  (classifier): Sequential(
    (0): Linear(in_features=768, out_features=64, bias=True)
    (1): ReLU()
    (2): Linear(in_features=64, out_features=2, bias=True)
  )
)
```

```
model_state_dict = torch.load(folder+'flexible-10-1e-05.pt', map_location=device)
```

```
model.load_state_dict(model_state_dict['model'], strict=False)
```

```
<All keys matched successfully>
```

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

Double-click (or enter) to edit

```
data = create_data(f'{folder}Sentiment.txt', 'dev')
dataset = BertDataset(data, None)
dataloader = DataLoader(dataset, shuffle=False, batch_size=5,
                                  collate_fn=dataset.collate_fn)
```

```
load 5 data from /content/drive/MyDrive/CSC401/Sentiment.txt
```

```
model.eval()
with torch.no_grad():
  for step, batch in enumerate(dataloader):
    b_ids, b_type_ids, b_mask, b_labels, b_sents = batch['token_ids'], batch['token_type_ids'], batch[
                'attention_mask'], batch['labels'], batch['sents']

    b_ids = b_ids.to(device)
    b_mask = b_mask.to(device)
    b_labels = b_labels.to(device)
    logits = model(b_ids, b_mask)
```

```
rslt = np.exp(logits.cpu().numpy())
print(rslt)
```

```
[[0.00670665 0.99329346]
 [0.00624877 0.9937512 ]
 [0.9979741  0.00202591]
 [0.9989599  0.00104012]
 [0.30161944 0.69838053]]
```

Discussion

The model did a very good job classifying the positivity of the movie review. The probabilities predictions from the model are very close to 1 for review 1-4, indicating a very high confidence in classification. This indicates that our model is performing well on the task to make reliable predictions.

However, the model's classification of the off-topic comment as more negative than expected is worth exploring further. It's likely because of that the model may have been trained on the dataset that did not include enough off-topic comments (i.e, we may put label 0.5 there), which could have affected its ability to accurately classify them.

This aligns well with the previous train/dev accuracy I have seen through out the training process.

Task 4.2, CausualLM

```
from transformers import AutoModelForCausalLM, AutoConfig, AutoTokenizer
```

```
causal_tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
Causal_LM = AutoModelForCausalLM.from_pretrained('bert-base-uncased').to(device)
```

```
If you want to use `BertLMHeadModel` as a standalone, add `is_decoder=True.`
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertLMHeadModel: ['cls.seq_relationship.b
- This IS expected if you are initializing BertLMHeadModel from the checkpoint of a model trained on another task or with another a
- This IS NOT expected if you are initializing BertLMHeadModel from the checkpoint of a model that you expect to be exactly identic
```

```
def evaluate_probabilities(causal_LM, causal_tokenizer, sentences, suffix):
    for sentence in sentences:
        # print(sentence)
        input_text = sentence + suffix
        input_ids = causal_tokenizer.encode(input_text, return_tensors='pt').to(device)
        # print(input_ids)

        with torch.no_grad():
            outputs = causal_LM(input_ids)
            logits = outputs.logits

        probabilities = torch.softmax(logits, dim=-1)
        # print(probabilities)
        pos_prob = probabilities[0, -1, causal_tokenizer.encode("positive")[1]].item()
        neg_prob = probabilities[0, -1, causal_tokenizer.encode("negative")[1]].item()
```

```
        print(f"probabilities -> Positive: {pos_prob}, Negative: {neg_prob}")


strongly_positive = "It is no wonder that the film has such a high rating, it is quite literally breathtaking. What can I say that hasn't
mildly_positive = "This film , for an after school special , is n't that bad , and that 's okay . Interesting things happen . You feel as
mildly_negative = "The arrival of vast waves of white settlers in the 1800s and their conflict with the Native American residents of the
strongly_negative = "This was without a doubt the worst of the "" "" Dirty Harry "" "" series . From the opening credits , you 're bored
off_topic = "Can you tell me how much the shirt is? -Yes, it's nine fifteen."


sentences = [strongly_positive, mildly_positive, mildly_negative, strongly_negative, off_topic]


suf = "This sentence is "
print("Results for prefix1:")
evaluate_probabilities(Causal_LM, causal_tokenizer, sentences, suf)
```

```
Results for prefix1:
probabilities -> Positive: 4.163256903666479e-07, Negative: 3.506977463985095e-07
probabilities -> Positive: 7.766830094624311e-07, Negative: 1.425206761496156e-07
probabilities -> Positive: 9.600316275282239e-08, Negative: 1.1032501134877748e-07
probabilities -> Positive: 1.9036541232253512e-07, Negative: 1.3726297538596555e-06
probabilities -> Positive: 3.544085164230992e-11, Negative: 2.3920955494194374e-11
```

Discussion

The results of the Causal LM in analyzing the polarity of comments were satisfactory. However, there was a surprising outcome in the case of the second review, which was classified as more positive (with a higher relative probability of being positive) than the first review, which was expected to be classified as "strongly positive". This difference from expectations could be due to the limitations of the model, as it was not specifically designed for the classification tasks (more for generation).

On the other hand, the model performed as expected for reviews 3 to 5, correctly classifying them based on their sentiment. The off-topic comment was slightly classified as more positive, but the difference in the probability was very small.

Overall, while the Causal LM did a reasonably good job in classifying the sentiment of the comments, its performance was not as robust as that of our fine-tuned classifier model, which is specifically trained for the classification task. This highlights the importance of fine-tuning models for specific tasks, as it can lead to more accurate results.

Task 4.3, ChatGPT LLM

I used GPT4 API to process the result.

```
import openai


with open(folder+'api.txt', 'r') as file:
    API_KEY = file.read()



openai.api_key = API_KEY
message_history = []

def predict(input):
    message_history.append({"role": "user", "content": f"{input}"})

    completion = openai.ChatCompletion.create(
      model="gpt-4",
      messages=message_history
    )

    reply_content = completion.choices[0].message.content

    message_history.append({"role": "assistant", "content": f"{reply_content}"})

    return reply_content


message_history = []
sent = '\n'.join(sentences)
message = "Please help me analyze the positivity of the following five movie reviews, please be coherent and short, answer how positive i
message_history.append({"role": "user", "content": f"{message}"})
message_history.append({"role": "assistant", "content": "ok"})
print(message)
print(message_history)
```

```
Please help me analyze the positivity of the following five movie reviews, please be coherent and short, answer how positive it is
[{'role': 'user', 'content': 'Please help me analyze the positivity of the following five movie reviews, please be coherent and sho
```

```
for sentence in sentences:
  response = predict(sentence)
  print(response)
```

```
Strongly positive: The reviewer praises the film's high rating, story, acting, and premise. They emphasize the long-lasting impact
Mildly positive: The reviewer finds the film decent for an after-school special, with interesting events happening and some charact
Mildly negative: The reviewer criticizes Robert Taylor's performance as dull and unoriginal, and finds fault with the film's storyl
Mildly negative: The reviewer calls the film the worst of the series and mentions that it is boring, has forgettable characters, an
My apologies, this statement isn't a movie review. Please provide a movie review for me to analyze its positivity.
```

Discussion

I used **GPT-4** to analyze the positivity the reviews, with the prompt: "Please help me analyze the positivity of the following five movie reviews, please be coherent and short, answer how positive it is (strongly/mildly positive/negative) with a very short explanation."

The model's performance was very impressive, accurately assessing the positivity of all reviews and providing succinct explanations. This even includes the off-topic review. However, the classification of review 4 could have been more negative, which is what we expected.

Remarkably, GPT-4's performance may even surpass that of our fine-tuned model, as it demonstrated an exceptional ability to evaluate the sentiment of the movie reviews, with provided explainations. This outcome suggests that LLMs like GPT-4 can be an effective tool for specific tasks, including sentiment analysis tasks, providing coherent and concise evaluations while maintaining a high degree of accuracy.