Problem 1

```
1 !pip install -q transformers
```

```
1 from transformers import GPT2LMHeadModel, GPT2TokenizerFast
2 import torch
3 import matplotlib.pyplot as plt
4 import numpy as np
5 import pandas as pd
```

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

⇥ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun

```
1 test = pd.read_csv('/content/drive/MyDrive/CS6983_GenAI_Folder/Homework_1/Problem1_dataset.csv')
```

```
1 print('First 5 test samples:')
2
3 for i, example in enumerate(test.text.values[:5]):
4    print(f'{i+1}. {example}')
```

⇥ First 5 test samples:
    1. = Robert Boulter =

    2. Robert Boulter is an English film , television and theatre actor . He had a guest @-@ starring role on the televisio

    3. In 2006 , Boulter starred alongside Whishaw in the play Citizenship written by Mark Ravenhill . He appeared on a 200

    4. = = Career = =

    5. = = = 2000 - 2005 = = =

```
1 #Wiki text samples converted to a list of strings:
2
3 wiki_list_of_strings = []
4 for i, example in enumerate(test.text.values):
5   #print(f'{i+1}. {example}')
6   wiki_list_of_strings.append(example)
7
8 cleaned_wiki_list_of_strings = [string.replace('\n', '') for string in wiki_list_of_strings] #List[str]
9 #print(len(cleaned_wiki_list_of_strings))
10
```

⇥   ' Robert Boulter is an English film , television and theatre actor '
    '. He had a guest @-@ starring role on the television series The '
    'Bill in 2000 . This was followed by a starring role in the play '
    'Herons written by Simon Stephens , which was performed in 2001 at '
    'the Royal Court Theatre . He had a guest role in the television '
    'series Judge John Deed in 2002 . In 2004 Boulter landed a role as '
    '" Craig " in the episode " Teddy \'s Story " of the television '
    'series The Long Firm ; he starred alongside actors Mark Strong and '
    'Derek Jacobi . He was cast in the 2005 theatre productions of the '
    'Philip Ridley play Mercury Fur , which was performed at the Drum '
    'Theatre in Plymouth and the Menier Chocolate Factory in London . '
    'He was directed by John Tiffany and starred alongside Ben Whishaw '
    ', Shane Zaza , Harry Kent , Fraser Ayres , Sophie Stanton and '
    'Dominic Hall . ',
    ' In 2006 , Boulter starred alongside Whishaw in the play '
    'Citizenship written by Mark Ravenhill . He appeared on a 2006 '
    'episode of the television series , Doctors , followed by a role in '
    'the 2007 theatre production of How to Curse directed by Josie '
    'Rourke . How to Curse was performed at Bush Theatre in the London '
    'Borough of Hammersmith and Fulham . Boulter starred in two films '
    'in 2008 , Daylight Robbery by filmmaker Paris Leonti , and Donkey '
    'Punch directed by Olly Blackburn . In May 2008 , Boulter made a '
    'guest appearance on a two @-@ part episode arc of the television '
    'series Waking the Dead , followed by an appearance on the '
    'television series Survivors in November 2008 . He had a recurring '
    'role in ten episodes of the television series Casualty in 2010 , '
    'as " Kieron Fletcher " . Boulter starred in the 2011 film '
    'Mercenaries directed by Paris Leonti . ',
    ' = = Career = = ',
    ' = = = 2000 - 2005 = = = ',
    ' In 2000 Boulter had a guest @-@ starring role on the television '
    'series The Bill ; he portrayed " Scott Parry " in the episode , " '
    'In Safe Hands " . Boulter starred as " Scott " in the play Herons '

```
"Royal Court Theatre . A review of Boulter 's performance in The "
'Independent on Sunday described him as " horribly menacing " in '
'the role , and he received critical reviews in The Herald , and '
'Evening Standard . He appeared in the television series Judge John '
'Deed in 2002 as " Addem Armitage " in the episode " Political '
'Expediency " , and had a role as a different character " Toby '
'Steele " on The Bill . ',
' He had a recurring role in 2003 on two episodes of The Bill , as '
'character " Connor Price " . In 2004 Boulter landed a role as " '
'Craig " in the episode " Teddy \'s Story " of the television '
'series The Long Firm ; he starred alongside actors Mark Strong and '
'Derek Jacobi . Boulter starred as " Darren " , in the 2005 theatre '
'productions of the Philip Ridley play Mercury Fur . It was '
'performed at the Drum Theatre in Plymouth , and the Menier '
'Chocolate Factory in London . He was directed by John Tiffany and '
'starred alongside Ben Whishaw , Shane Zaza , Harry Kent , Fraser '
'Ayres , Sophie Stanton and Dominic Hall . Boulter received a '
'favorable review in The Daily Telegraph : " The acting is '
'shatteringly intense , with wired performances from Ben Whishaw ( '
"now unrecognisable from his performance as Trevor Nunn 's Hamlet ) "
', Robert Boulter , Shane Zaza and Fraser Ayres . " The Guardian '
'noted , " Ben Whishaw and Robert Boulter offer tenderness amid the '
'savagery . " ',
' = = = 2006 – present = = = ',
```

```
1  # Print dataset statistics:
2
3  # The TOTAL number of samples in dataset:
4  count = len(test.text.values)
5  print(f'Total number of samples: {count}')
6
7  # The AVERAGE number of characters per sample:
8  num_char, count = 0, 0
9  for example in (test.text.values):
10     num_char += len(example)
11     count += 1
12 print(f'The average number of characters per sample {(num_char/count)}')
13
```

Total number of samples: 187
The average number of characters per sample 483.80213903743316

```
1  # Complete function to Calculate Perplexity
2
3  def calculate_perplexity(text, model, tokenizer, window_length=1024, step_size=256):
4    model_inputs = tokenizer.encode(text, return_tensors='pt')
5    perplexities = []
6    DIMENSION = 1
7    for start in range(0, model_inputs.size(DIMENSION), step_size):
8        end = min(start + window_length, model_inputs.size(DIMENSION))
9        window = model_inputs[:, start:end]
10
11       with torch.no_grad():
12           outputs = model(window, labels=window)
13
14       neg_log_likelihood = outputs.loss
15
16       perplexity = torch.exp(neg_log_likelihood)
17       perplexities.append(perplexity.item())
18
19       average_perplexity = sum(perplexities) / len(perplexities) if perplexities else float('inf')
20   return average_perplexity
21
```

```
1  # Complete code for HuggingFace GPT2 model (small, medium, and large) evaluation, according to the perplexity metric
2
3  # HuggingFace GPT2 model – Small
4  # small_tokenizer = GPT2TokenizerFast.from_pretrained('gpt2')
5  # small_model = GPT2LMHeadModel.from_pretrained('gpt2')
6
7  # perplexity_dicitonary_gpt2 = dict()
8  # for i in range(len(cleaned_wiki_list_of_strings)):
9  #   perplexity_dicitonary_gpt2[i]=calculate_perplexity(cleaned_wiki_list_of_strings[i],small_model,small_tokenizer)
10 # print(perplexity_dicitonary_gpt2)
11
12
13 # # HuggingFace GPT2 model – Medium
14
15 # medium_tokenizer = GPT2TokenizerFast.from_pretrained('gpt2–medium')
16 # medium_model = GPT2LMHeadModel.from_pretrained('gpt2–medium')
17 # perplexity_dicitonary_gpt2_medium = dict()
18 # # for i in range(len(cleaned_wiki_list_of_strings)):
19 # #   perplexity_dicitonary_gpt2_medium[i]=calculate_perplexity(cleaned_wiki_list_of_strings[i],medium_model,medium_toker
```

```
20 # print(perplexity_dicitonary_gpt2_medium)
21
22
23 # HuggingFace GPT2 model - Large
24 large_tokenizer = GPT2TokenizerFast.from_pretrained('gpt2-large')
25 large_model = GPT2LMHeadModel.from_pretrained('gpt2-large')
26
27 perplexity_dicitonary_gpt2_large = dict()
28 for i in range(len(cleaned_wiki_list_of_strings)):
29   perplexity_dicitonary_gpt2_large[i]=calculate_perplexity(cleaned_wiki_list_of_strings[i],large_model,large_tokenizer)
30 print(perplexity_dicitonary_gpt2_large)
31
32
33
```

```
 1 items = list(perplexity_dicitonary_gpt2_large.items())
 2 items.sort(key=lambda x: x[1])
 3
 4 items.sort(key=lambda x: x[1], reverse=True)
 5
 6
 7
 8
 9 #Lowest perplexity samples
10 pprint.pp(cleaned_wiki_list_of_strings[127], depth=1, width=90)
11 pprint.pp(cleaned_wiki_list_of_strings[83], depth=1, width=90)
12 pprint.pp(cleaned_wiki_list_of_strings[84], depth=1, width=90)
13
14 #Highest perplexity samples
15 pprint.pp(cleaned_wiki_list_of_strings[93], depth=1, width=90)
16 pprint.pp(cleaned_wiki_list_of_strings[183], depth=1, width=90)
17 pprint.pp(cleaned_wiki_list_of_strings[14], depth=1, width=90)
18
```

```
Sorted in ascending order: [(2, 27.221298217773438), (1, 29.44189453125), (5, 30.019569396972656), (4, 88.4051513671875)
Sorted in descending order: [(0, 1270.773193359375), (3, 360.9991149902344), (4, 88.4051513671875), (5, 30.0195693969726
(" The ships ' secondary armament consisted of twenty 50 @-@ calibre 14 @-@ centimetre "
 'Type 3 . Eighteen of these were mounted in casemates in the forecastle and '
 'superstructure and the remaining pair were mounted on the deck above them and '
 'protected by gun shields . They had a maximum elevation of + 20 degrees which gave '
 'them ranges of 16 @,@ 300 metres ( 17 @,@ 800 yd ) . Each gun had a rate of fire of up '
 'to 10 rounds per minute . Anti @-@ aircraft defence was provided by four 40 @-@ '
 'calibre 3rd Year Type 8 @-@ centimetre AA guns in single mounts . The 7 @.@ 62 @-@ '
 'centimetre ( 3 in ) high @-@ angle guns had a maximum elevation of + 75 degrees , and '
 'had a rate of fire of 13 to 20 rounds per minute . They fired a 6 kg ( 13 lb ) '
 'projectile with a muzzle velocity of 680 m / s ( 2 @,@ 200 ft / s ) to a maximum '
 'height of 7 @,@ 500 metres ( 24 @,@ 600 ft ) . The ships were also fitted with six '
 'submerged 53 @.@ 3 @-@ centimetre ( 21 @.@ 0 in ) torpedo tubes , three on each '
 'broadside . They carried twelve to eighteen 6th Year Type torpedoes which had a 200 '
 '@-@ kilogram ( 440 lb ) warhead . They had three settings for range and speed : 15 @,@ '
 '000 metres ( 16 @,@ 000 yd ) at 26 knots ( 48 km / h ; 30 mph ) , 10 @,@ 000 metres ( '
 '11 @,@ 000 yd ) at 32 knots ( 59 km / h ; 37 mph ) , or 7 @,@ 000 metres ( 7 @,@ 700 '
 'yd . ) at 37 knots ( 69 km / h ; 43 mph ) . ')
(' The single made its Irish Singles Chart debut at number 24 on the week ending 13 '
 'December 2012 . It peaked at number seven on the week ending 17 January 2013 , marking '
 'their sixth top ten appearance in Ireland . " Kiss You " entered at number 152 in the '
 'UK Singles Chart on 24 November 2012 . It peaked at number nine on the UK Singles '
 "Chart on 26 January 2013 , becoming One Direction 's sixth top ten hit in the United "
 'Kingdom . On the week ending 18 November 2012 , " Kiss You " debuted at number 90 on '
 'the United States Billboard Hot 100 due to digital download sales from its parent '
 'album . As a result of an " end @-@ of @-@ year download rush " on the week ending 30 '
 'December 2012 , the track re @-@ entered the Hot 100 at number 83 . After the '
 'accompanying music video was released , the song re @-@ entered the Hot 100 at number '
 '65 . " Kiss You " had sold 207 @,@ 000 digital downloads in the US by 18 January 2013 '
 '. The single ultimately peaked at number 46 on the Hot 100 and was certified gold by '
 'the Recording Industry Association of America ( RIAA ) on 25 April 2013 , denoting '
 'shipments of 500 @,@ 000 copies . ')
(" The song became One Direction 's fourth top @-@ forty hit on the Canadian Hot 100 , "
 'peaking at number 30 . The single bowed at number 13 on the Australian Singles Chart '
 "on 27 January 2013 , marking its peak position and the group 's fourth top twenty hit "
 'in Australia . The song has been certified platinum by the Australian Recording '
 'Industry Association ( ARIA ) for shipments of 70 @,@ 000 copies . The track entered '
 'the New Zealand Singles Chart at number 17 on 11 January 2013 . It peaked at number 13 '
 "in its third and fourth charting weeks , becominh the group 's sixth top @-@ forty "
 'appearance in New Zealand . " Kiss You " has received a gold certification from the '
 'Recording Industry Association of New Zealand ( RIANZ ) , indicating sales of 7 @,@ '
 '500 copies . The track also reached the top 40 in both Belgian territories ( Flanders '
 'and Wallonia ) , as well as in the Czech Republic , Denmark , France , the Netherlands '
 ', and South Korea . In addition , " Kiss You " received gold certifications from the '
 'IFPI Norway and Denmark associations , signifying collective shipments of 20 @,@ 000 '
 'units . ')
' CD single '
' = 1933 Treasure Coast hurricane = '
' = Du Fu = '
```

```
1 # Print results:
2
3 #GPT-2 Model Results
4 gpt2_values = list(perplexity_dicitonary_gpt2.values())
5
6 #25TH AND 75TH PERCENTILES
7 percentile_25 = np.percentile(gpt2_values, 25)
8 print("25th percentiles of the perplexity: ", percentile_25)
9 percentile_75 = np.percentile(gpt2_values, 75)
10 print("75th percentiles of the perplexity: ", percentile_75)
11
12 gpt2_values.sort()
13 mid = len(gpt2_values) // 2
14 res = (gpt2_values[mid] + gpt2_values[~mid]) / 2
15 print("median perplexity: " + str(res))
16
17 midhinge = (percentile_25 + percentile_75) / 2
18 print("midhinge perplexity:",midhinge)
19
```

```
1 # Comments:
2 # The larger the HuggingFace GPT2 model size, the smaller the perplexity.
3 # The larger the HuggingFace GPT2 model size, the higher the upperbound of the sentence perplexity perplexity calculated
```

```
1 # #Create GPT2-Histogram
2 gpt2_values = list(perplexity_dicitonary_gpt2.values())
3 plt.hist(gpt2_values, edgecolor="red", bins=30)
4
5 # Adding labels and title
6 plt.xlabel('Perplexity GPT-2 Small Values')
7 plt.ylabel('Frequency')
8 plt.title('GPT-2 Histogram (Small)')
9
10 plt.show()
11
```

```
1 #Create a GPT2-Medium Model Histogram
2 gpt2_medium_values = list(perplexity_dicitonary_gpt2_medium.values())
3 plt.hist(gpt2_medium_values, edgecolor="red", bins=30)
4
5 # Adding labels and title
6 plt.xlabel('Perplexity GPT-2 Medium Values')
7 plt.ylabel('Frequency')
8 plt.title('GPT-2 Histogram (Medium)')
9
10 plt.show()
```

```
1 #Create a GPT2-Large Model Histogram
2 gpt2_large_values = list(perplexity_dicitonary_gpt2_large.values())
3 plt.hist(gpt2_large_values, edgecolor="red", bins=30)
4
5 # Adding labels and title
6 plt.xlabel('Perplexity GPT-2 Large Values')
7 plt.ylabel('Frequency')
8 plt.title('GPT-2 Histogram (Large)')
9
10 plt.show()
```

```
1 #GPT-2 Medium Model Results
2 gpt2_medium_values = list(perplexity_dicitonary_gpt2_medium.values())
3
4 #25TH AND 75TH PERCENTILES
5
6 #Percentile Calculation for gpt2
7 percentile_25 = np.percentile(gpt2_medium_values, 25)
8 print("25th percentiles of the perplexity: ", percentile_25)
9 percentile_75 = np.percentile(gpt2_large_values, 75)
10 print("75th percentiles of the perplexity: ", percentile_75)
11
12 gpt2_medium_values.sort()
13 mid = len(gpt2_medium_values) // 2
14 res = (gpt2_medium_values[mid] + gpt2_medium_values[~mid]) / 2
15 print("median perplexity: " + str(res))
16
17 midhinge = (percentile_25 + percentile_75) / 2
18 print("midhinge perplexity:",midhinge)
```

```
 1 #GPT-2 Large Model Results
 2
 3 #25TH AND 75TH PERCENTILES
 4 gpt2_large_values = list(perplexity_dicitonary_gpt2_large.values())
 5 #Percentile Calculation for gpt2
 6 percentile_25 = np.percentile(gpt2_large_values, 25)
 7 print("25th percentiles of the perplexity: ", percentile_25)
 8 percentile_75 = np.percentile(gpt2_large_values, 75)
 9 print("75th percentiles of the perplexity: ", percentile_75)
10
11 gpt2_large_values.sort()
12 mid = len(gpt2_large_values) // 2
13 res = (gpt2_large_values[mid] + gpt2_large_values[~mid]) / 2
14 print("median perplexity: " + str(res))
15
16 midhinge = (percentile_25 + percentile_75) / 2
17 print("midhinge perplexity:",midhinge)
```

⌄ Problem 2

```
 1 !pip install mauve-text
```

> Requirement already satisfied: mauve-text in /usr/local/lib/python3.10/dist-packages (0.4.0)
> Requirement already satisfied: numpy>=1.18.1 in /usr/local/lib/python3.10/dist-packages (from mauve-text) (1.26.4)
> Requirement already satisfied: scikit-learn>=0.22.1 in /usr/local/lib/python3.10/dist-packages (from mauve-text) (1.5.2)
> Requirement already satisfied: faiss-cpu>=1.7.0 in /usr/local/lib/python3.10/dist-packages (from mauve-text) (1.8.0.post
> Requirement already satisfied: tqdm>=4.40.0 in /usr/local/lib/python3.10/dist-packages (from mauve-text) (4.66.5)
> Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from mauve-text) (2.32.3)
> Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from faiss-cpu>=1.7.0->mauve-text)
> Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22.1->mauve
> Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22.1->mauv
> Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22.
> Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->mauve
> Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->mauve-text) (3.10
> Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->mauve-text)
> Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->mauve-text)

```
 1 #Imports
 2 from transformers import GPT2Tokenizer, GPT2LMHeadModel
 3
 4 from tqdm import tqdm
 5 import torch
 6 import mauve
 7 import json
 8 import os
 9 import sys
10 import requests
11 import pickle
12
13 seed = 123
14 torch.manual_seed(seed)
15 np.random.seed(seed)
```

```
 1 from google.colab import drive
 2 drive.mount('/content/drive')
```

> Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun

```
 1 # Load test set
 2 path = '/content/drive/MyDrive/CS6983_GenAI_Folder/Homework_1/Problem2_dataset.pkl'
 3
 4 with open(path, 'rb') as file:
 5     data = pickle.load(file)
 6 print(data)
 7
 8 samples = ""
 9 for i in range(len(data)):
10   text_data = data[i]['text']
11   samples += text_data
12 #print(samples)
13 #print(type(samples))
14 #print(len(samples))
15
16 print(len(samples.split("\n\n")))
17 text_input = samples.split("\n\n")
18 print(len(text_input))
```

> signing him for Aston Villa.\n\nHe stated on Twitter that he should move to Monaco.\n\nGarde admitted he is keen on signi

```
1 device = torch.device("cuda")
2 large_tokenizer = GPT2TokenizerFast.from_pretrained('gpt2-large')
3 large_model = GPT2LMHeadModel.from_pretrained('gpt2-large')
```

```
1 large_tokenizer.eos_token_id
```

⇥ 50256

```
1 #The Greedy Algorithm for Sentence Generation
2 torch_device = "cuda"
3 MAX_LEN = 50
4 TOKEN_ID = large_tokenizer.eos_token_id
5 greedy_output_sentences = []
6 for i in range(len(text_input)):
7   prefix=text_input[i].split()[:10]
8   generation_input = ' '.join(prefix)
9   #print(generation_input)
10  input_ids = large_tokenizer.encode(generation_input,return_tensors="pt")
11  attention_mask = torch.ones(input_ids.shape)
12  greedy_output=large_model.generate(input_ids, max_length=MAX_LEN,attention_mask=attention_mask,pad_token_id=TOKEN_ID).to
13  greedy_output_sentences.append(large_tokenizer.decode(greedy_output[0], remove_special_tokens = True))
14
15
16 #Top-k:
17 for i in range(len(text_input)):
18   prefix=text_input[i].split()[:10]
19   generation_input = ' '.join(prefix)
20   input_ids = large_tokenizer.encode(generation_input, return_tensors='pt')
21   output_generated = large_model.generate(input_ids, do_sample = True, max_length = MAX_LEN, top_k = 10,attention_mask=Non
22   print(large_tokenizer.decode(output_generated[0],remove_special_tokens=True))
23
24
25 #Beam Search:
26 beam_generated = []
27 for i in range(len(text_input)):
28   prefix=text_input[i].split()[:10]
29   generation_input = ' '.join(prefix)
30   input_ids = large_tokenizer.encode(generation_input, return_tensors='pt')
31   beam_outputs = large_model.generate(input_ids, max_length=MAX_LEN, num_beams = 5,attention_mask=None,pad_token_id=TOKEN_
32     beam_generated.append(beam_output)
33     print(large_tokenizer.decode(beam_output, remove_special_tokens=True))
34
35
36
37
```

```
1 # print(greedy_output_sentences)
```

⇥ ['Is this restaurant family-friendly? Yes No Unsure\n\nCan a gluten free person get a good meal at this restaurant? Yes

```
1 !pip install evaluate
```

```
1 # Compute the MAUVE score
2
3 #list of original text sentences
4 original_data = []
5 for i in range(len(text_input)):
6   original_data.append(text_input[i])
7 print(len(original_data))
8
9 # print(len(greedy_output_sentences))
10 # print()
```

⇥ 2414

```
1 #Print results:
2 print(len(original_data))
3 print(len(text_input))
4 # print(type(text_input))
5 # print(text_input[0])
6 # print(text_input[60])
7 # print(text_input[2413])
8
9 #Greedy Algorithm Mauve score
10 def tensor_version_of_sentences(list_str):
11     inputs = tokenizer(list_str, return_tensors='pt', padding=True, truncation=True, max_length=50)
```

```
12     return inputs['input_ids']
13
14 generated_tensors = tensor_version_sentences(text_generation_output)
15 original_tensors = tensor_version_sentences(original)
16 mauve_results = mauve.compute_mauve(generated_tensors, original_tensors)
17 print("the mauve score", mauve_results.mauve)
```

```
2414
2414
the mauve score 0.0040720962619612555
```

```
 1 # Select the sentence of the highest and lowest MAUVE score
 2 #for each generation strategy and print it. Discuss your observations.
 3
 4 # The sentence with the highest mauve score was:
 5 #The ships ' secondary armament consisted of twenty 50 @-@ calibre 14 @-@ centimetre "
 6  #'Type 3 . Eighteen of these were mounted in casemates in the forecastle and '
 7 #  'superstructure and the remaining pair were mounted on the deck above them and '
 8 #  'protected by gun shields . They had a maximum elevation of + 20 degrees which gave '
 9 #  'them ranges of 16 @,@ 300 metres ( 17 @,@ 800 yd ) . Each gun had a rate of fire of up '
10 #  'to 10 rounds per minute . Anti @-@ aircraft defence was provided by four 40 @-@ '
11 #  'calibre 3rd Year Type 8 @-@ centimetre AA guns in single mounts . The 7 @.@ 62 @-@ '
12 #  'centimetre ( 3 in ) high @-@ angle guns had a maximum elevation of + 75 degrees , and '
13 #  'had a rate of fire of 13 to 20 rounds per minute . They fired a 6 kg ( 13 lb ) '
14 #  'projectile with a muzzle velocity of 680 m / s ( 2 @,@ 200 ft / s ) to a maximum '
15 #  'height of 7 @,@ 500 metres ( 24 @,@ 600 ft ) .
16
17 # The sentnce with the lowest mauve score was:
18 # CD single
19
```

```
1
2
```

Problem 3

```
1 !pip install datasets scikit-learn
```

```
1 !pip install pyarrow==15.0.2
```

```
Requirement already satisfied: pyarrow==15.0.2 in /usr/local/lib/python3.10/dist-packages (15.0.2)
Requirement already satisfied: numpy<2,>=1.16.6 in /usr/local/lib/python3.10/dist-packages (from pyarrow==15.0.2) (1.26.
```

```
 1 import torch
 2 from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, AdamW
 3 from transformers import get_scheduler
 4 from transformers import TrainingArguments, Trainer
 5 from datasets import load_dataset
 6 from sklearn.metrics import accuracy_score, precision_recall_fscore_support
 7 from torch.utils.data import DataLoader
 8 from tqdm.auto import tqdm
 9 from sklearn.metrics import roc_curve, auc
10
```

```
1 # Load IMDb dataset and limit to 1000 samples
2 dataset = load_dataset('imdb')
3 train_dataset = dataset['train'].shuffle(seed=42).select(range(1000))
4 test_dataset = dataset['test'].shuffle(seed=42).select(range(100)) #Select 100 Samples
```

```
1 print('5 Training examples')
2 for i, example in enumerate(train_dataset['text'][:5]):
3   print(f'{i+1}. {example}')
4
```

```
5 Training examples
1. There is no relation at all between Fortier and Profiler but the fact that both are police series about violent crime
2. This movie is a great. The plot is very true to the book which is a classic written by Mark Twain. The movie starts o
3. George P. Cosmatos' "Rambo: First Blood Part II" is pure wish-fulfillment. The United States clearly didn't win the w
4. In the process of trying to establish the audiences' empathy with Jake Roedel (Tobey Maguire) the filmmakers slander
5. Yeh, I know -- you're quivering with excitement. Well, *The Secret Lives of Dentists* will not upset your expectation
```

```
1 print('5 Testing examples')
2 for i, example in enumerate(test_dataset['text'][:5]):
3   print(f'{i+1}. {example}')
4
```

    1.

When I unsuspectedly rented A Thousand Acres, I thought I was in for an entertaining King Lear story and
    2. This is the latest entry in the long series of films with the French agent, O.S.S. 117 (the French answer to James Bo
    3. This movie was so frustrating. Everything seemed energetic and I was totally prepared to have a good time. I at least
    4. I was truly and wonderfully surprised at "O' Brother, Where Art Thou?" The video store was out of all the movies I wa
    5. This movie spends most of its time preaching that it is the script that makes the movie, but apparently there was no

```
1 # Complete code for Fine-tuning BERT for sentiment classification
2 tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
3
4 DistilBertmodel = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=2)
5
6 # Freeze all layers
7 for layer in DistilBertmodel.parameters():
8     layer.requires_grad = False
9
10 # Do not freeze last classification layer
11 for layer in DistilBertmodel.classifier.parameters():
12     layer.requires_grad = True
13
```

⇥ /usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning: `clean_up_tokenizat
    warnings.warn(
    Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-un
    You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
1 # Token Creation
2 def tokenize_function(examples):
3     return tokenizer(examples['text'], padding='max_length', truncation=True)
4
5 # Creating the Dataset
6 tokenized_train = train_dataset.map(tokenize_function, batched=True)
7 tokenized_eval = test_dataset.map(tokenize_function, batched=True)
```

```
1 training_args = TrainingArguments(
2     output_dir="my_model_directory",
3     learning_rate=1e-5,
4     per_device_train_batch_size=64,
5     per_device_eval_batch_size=16,
6     num_train_epochs=5,
7 )
8
9 # Create a Trainer instance
10 trainer = Trainer(
11     model=DistilBertmodel,
12     args=training_args,
13     train_dataset= tokenized_train,
14     eval_dataset=tokenized_eval,
15 )
16
17 # Start the training process
18 trainer.train()
19
```

⇥ ████████████████████████████ [80/80 01:37, Epoch 5/5]

Step Training Loss

TrainOutput(global_step=80, training_loss=0.6880020141601563, metrics={'train_runtime': 98.9664,
'train_samples_per_second': 50.522, 'train_steps_per_second': 0.808, 'total_flos': 662336993280000.0, 'train_loss':
0.6880020141601563, 'epoch': 5.0})

```
1 # Print results:
2 eval_results = trainer.predict(tokenized_eval)
3
4 preds = np.argmax(eval_results.predictions, axis=1)
5 labels = eval_results.label_ids
6
7 # Accuracy
8 accuracy = accuracy_score(labels, preds)
9
10 # Precision, Recall, F1 Score
11 precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='weighted')
12
13 # Error rate (1 - Accuracy)
14 error_rate = 1 - accuracy
15
16
```

⇥

```
1 #Printing all metrics
2 print("Accuracy metric", accuracy)
3 print("Precision, Recall, F1 Score", precision, recall, f1)
4 print("Error rate", 1 - accuracy)
```

Accuracy metric 0.59
Precision, Recall, F1 Score 0.7108152173913044 0.59 0.49977429467084633
Error rate 0.41000000000000003

```
1 modl_eval_results = trainer.predict(tokenized_eval)
2 labels = eval_results.label_ids
3 model_y_scores = modl_eval_results.predictions[:, 1]
4
5 fpr, tpr, thresholds = roc_curve(labels, model_y_scores)
6 roc_auc = auc(fpr, tpr)
7
8 # Plot
9 plt.figure()
10 plt.plot([0, 1], [0, 1], color='brown', lw=2)
11 plt.plot(fpr, tpr, color='green', lw=2)
12
13 plt.xlabel('False Positive Rate')
14 plt.ylabel('True Positive Rate')
15 plt.title('AUC ROC Curve')
16 plt.show()
17
```



```
1 for layer in DistilBertmodel.parameters():
2     layer.requires_grad = True
```

```
1 training_args = TrainingArguments(
2     output_dir="my_model_directory",
3     learning_rate=1e-5,
4     per_device_train_batch_size=64,
5     per_device_eval_batch_size=16,
6     num_train_epochs=5,
7 )
8
9 # Create a Trainer instance
10 trainer = Trainer(
11     model=DistilBertmodel,
12     args=training_args,
13     train_dataset= tokenized_train,
14     eval_dataset=tokenized_eval,
15 )
16
17 # Start the training process
18 trainer.train()
```

Step	Training Loss
10	0.676600
20	0.609200
30	0.488300
40	0.455800
50	0.321100
60	0.368800
70	0.438300
80	0.214400
90	0.232200
100	0.255800
110	0.206000
120	0.232500
130	0.231100
140	0.150000
150	0.102500
160	0.126000
170	0.066800
180	0.146000
190	0.120100
200	0.070200
210	0.007700
220	0.007100
230	0.037300
240	0.007200
250	0.110700
260	0.027000
270	0.004600
280	0.003200
290	0.018800
300	0.039000
310	0.037200

```
TrainOutput(global_step=315, training_loss=0.18454010839618387, metrics={'train_runtime': 246.1843,
'train_samples_per_second': 20.31, 'train_steps_per_second': 1.28, 'total_flos': 679292820307968.0, 'train_loss':
0.18454010839618387, 'epoch': 5.0})
```

```
1 eval_results = trainer.evaluate()
2 print("The cross-entropy loss",eval_results['eval_loss'])
```

[2/2 07:42]
```
The test cross-entropy loss 0.8423845767974854
```

```
1 print("Accuracy metric", accuracy)
2 print("Precision, Recall, F1 Score", precision, recall, f1)
3 print("Error rate", 1 - accuracy)
```

```
Accuracy metric 0.82
Precision, Recall, F1 Score 0.8300202020202019 0.82 0.8197118847539017
Error rate 0.18000000000000005
```

```
1 #Print results:
2 eval_results = trainer.predict(tokenized_eval)
3
4 preds = np.argmax(eval_results.predictions, axis=1)
5 labels = eval_results.label_ids
6
7 # Accuracy
8 accuracy = accuracy_score(labels, preds)
9
```

```
10 # Precision, Recall, F1 Score
11 precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='weighted')
12
13 # Error rate (1 — Accuracy)
14 error_rate = 1 — accuracy
```

```
1 # Comments:
2 # It is evident that the models that have been fine tuned on the last layer
3 # are much more accurate at classification compared to the models that have only
4 # been fine tuned on the final layer.
```