Name: Ubaid UL Majied

Student_Id: 862395234

CalcGPT Report

I. Method for encoding strings (1 point)

Baseline Strategy Input string = `"x1 + x2 = ”`
New Strategy Input string `= "2 + 3 = 5, 7 + 8 = 15, 9 + 10 = 19, x1 + x2 =\n"`

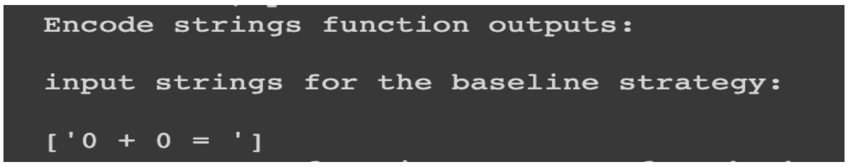
Following are the examples of the outputs encode input function for baseline and new strategy.
For these examples, I have created a dataset with start_id = 0 and end_id = 1

The dataset is small so I can completely display the outputs here.

create_dataset(0, 1)

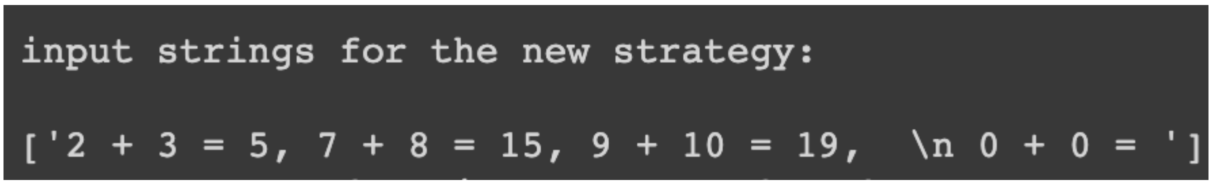
Outputs of the encoding string function and tokenization of these strings :

A. Example of the Baseline input string



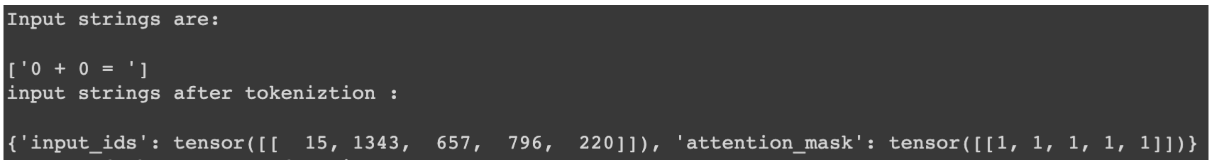
```
Encode strings function outputs:

input strings for the baseline strategy:

['0 + 0 = ']
```

B. Example of the new strategy input string



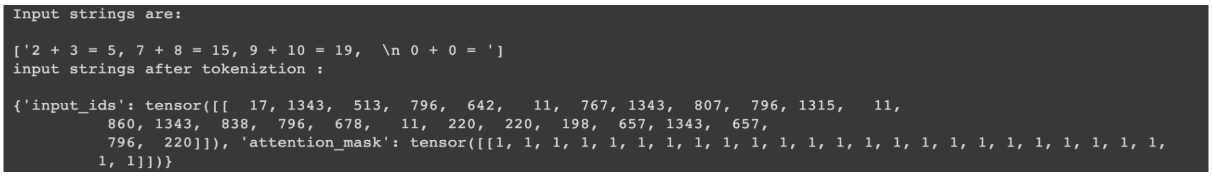
```
input strings for the new strategy:

['2 + 3 = 5, 7 + 8 = 15, 9 + 10 = 19,  \n 0 + 0 = ']
```

C. Example of the Tokenization of the baseline strategy input string



```
Input strings are:

['0 + 0 = ']
input strings after tokeniztion :

{'input_ids': tensor([[  15, 1343,  657,  796,  220]]), 'attention_mask': tensor([[1, 1, 1, 1, 1]])}
```

D. Example of the Tokenization of the new strategy input string



```
Input strings are:

['2 + 3 = 5, 7 + 8 = 15, 9 + 10 = 19,  \n 0 + 0 = ']
input strings after tokeniztion :

{'input_ids': tensor([[  17, 1343,  513,  796,  642,   11,  767, 1343,  807,  796, 1315,   11,
          860, 1343,  838,  796,  678,   11,  220,  220,  198,  657, 1343,  657,
          796,  220]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1]])}
```

In the tokenization output (C, D):

- **Input_ids:** are the indices corresponding to each token in the sentence.
- **Attention_mask:** indicates whether a token should be attended to or not.

Tokenization of the input strings is done using **GPT2Tokenizer.**

II. Method for generating text (1 point)

A. Language Model Used: GPT2

1. **Size** = Large
2. **Parameters:** 1.5 billion
3. **Dataset Lenght:** 8 million web pages
4. **Objective:** predict the next word given all the words in the text
5. **Available in 5 different sizes:** small, medium, large, xl, and a distilled

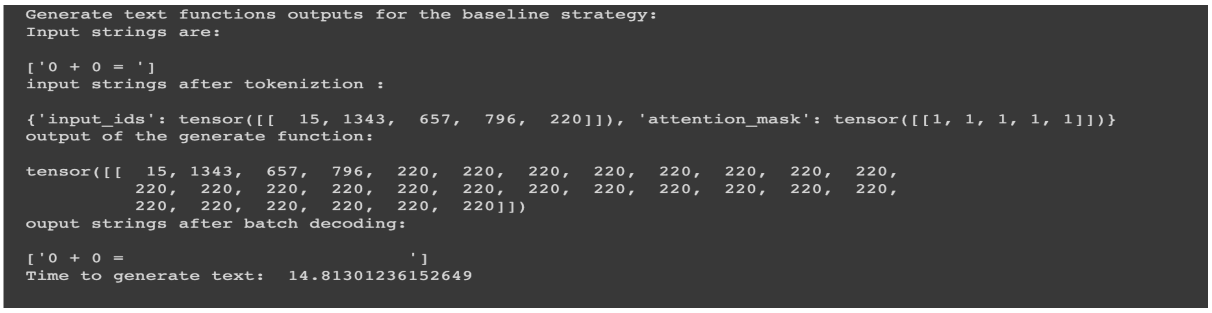
B. Hyperparameters chosen

1. **num_beams: 5,** Shifts the model from greedy search to beam search
2. **Do_sample = true,** Enables top K sampling and top P sampling but haven't done any of them
3. **Temperature = 0.8,** This value modulates the probability of the next token
4. **Pad_token_id = 50256,** Id of the padding token
5. **Max_new_tokens = 5,** Tells the model how many new tokens to generate

III. Method for decoding strings (1 point)

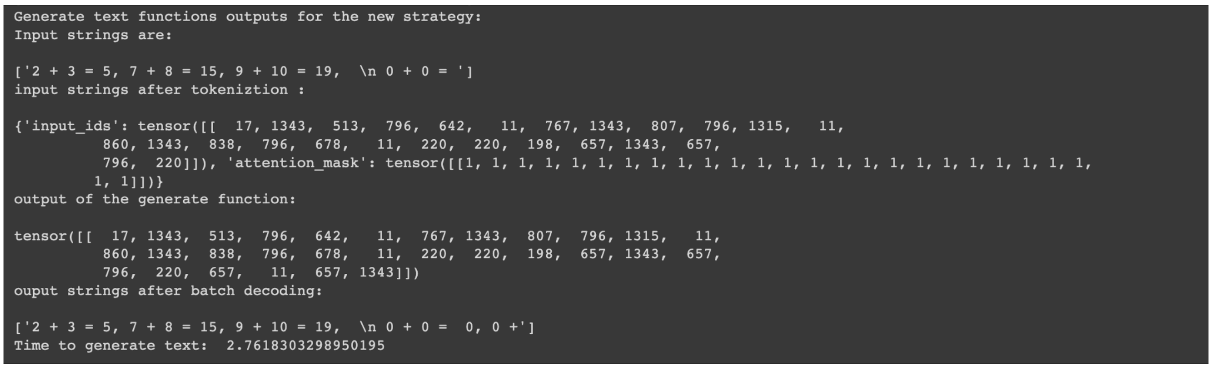
example of the results of the text generate function contains also the output tokens

baseline strategy.



```
Generate text functions outputs for the baseline strategy:
Input strings are:

['0 + 0 = ']
input strings after tokeniztion :

{'input_ids': tensor([[   15, 1343,   657,   796,   220]]), 'attention_mask': tensor([[1, 1, 1, 1, 1]])}
output of the generate function:

tensor([[   15, 1343,   657,   796,   220,   220,   220,   220,   220,   220,   220,   220,
           220,   220,   220,   220,   220,   220,   220,   220,   220,   220,   220,   220,
           220,   220,   220,   220,   220,   220]])
ouput strings after batch decoding:

['0 + 0 =                          ']
Time to generate text:  14.81301236152649
```

New Strategy



```
Generate text functions outputs for the new strategy:
Input strings are:

['2 + 3 = 5, 7 + 8 = 15, 9 + 10 = 19,  \n 0 + 0 = ']
input strings after tokeniztion :

{'input_ids': tensor([[  17, 1343,  513,  796,  642,   11,  767, 1343,  807,  796, 1315,   11,
          860, 1343,  838,  796,  678,   11,  220,  220,  198,  657, 1343,  657,
          796,  220]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1]])}
output of the generate function:

tensor([[  17, 1343,  513,  796,  642,   11,  767, 1343,  807,  796, 1315,   11,
          860, 1343,  838,  796,  678,   11,  220,  220,  198,  657, 1343,  657,
          796,  220,  657,   11,  657, 1343]])
ouput strings after batch decoding:

['2 + 3 = 5, 7 + 8 = 15, 9 + 10 = 19,  \n 0 + 0 =  0, 0 +']
Time to generate text:  2.7618303298950195
```

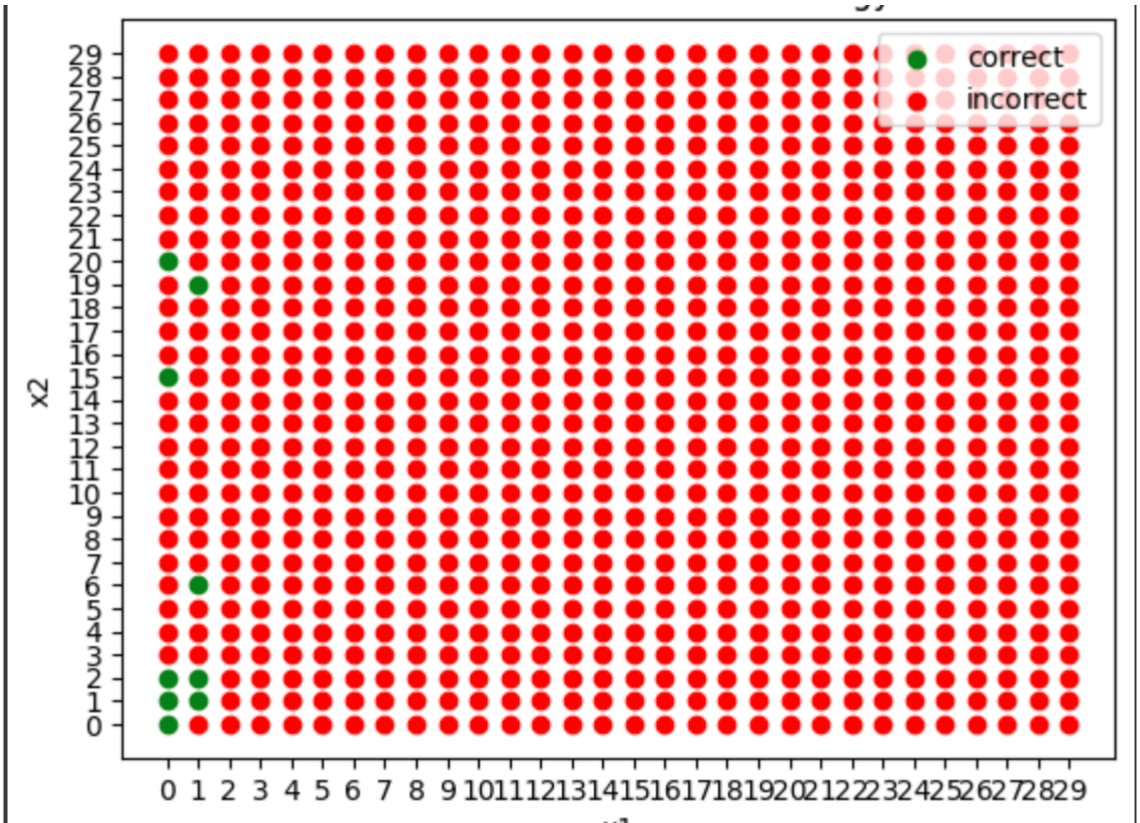
IV. Results (6 points)

Note: I am sharing the results of the start_id: 0 and end_id: 30 because I am getting good results with this as compared to start_id: 0 and end_id: 50 but I will be submitting i-pynb files for both start and end ids. Results will be displayed in those files aslo.

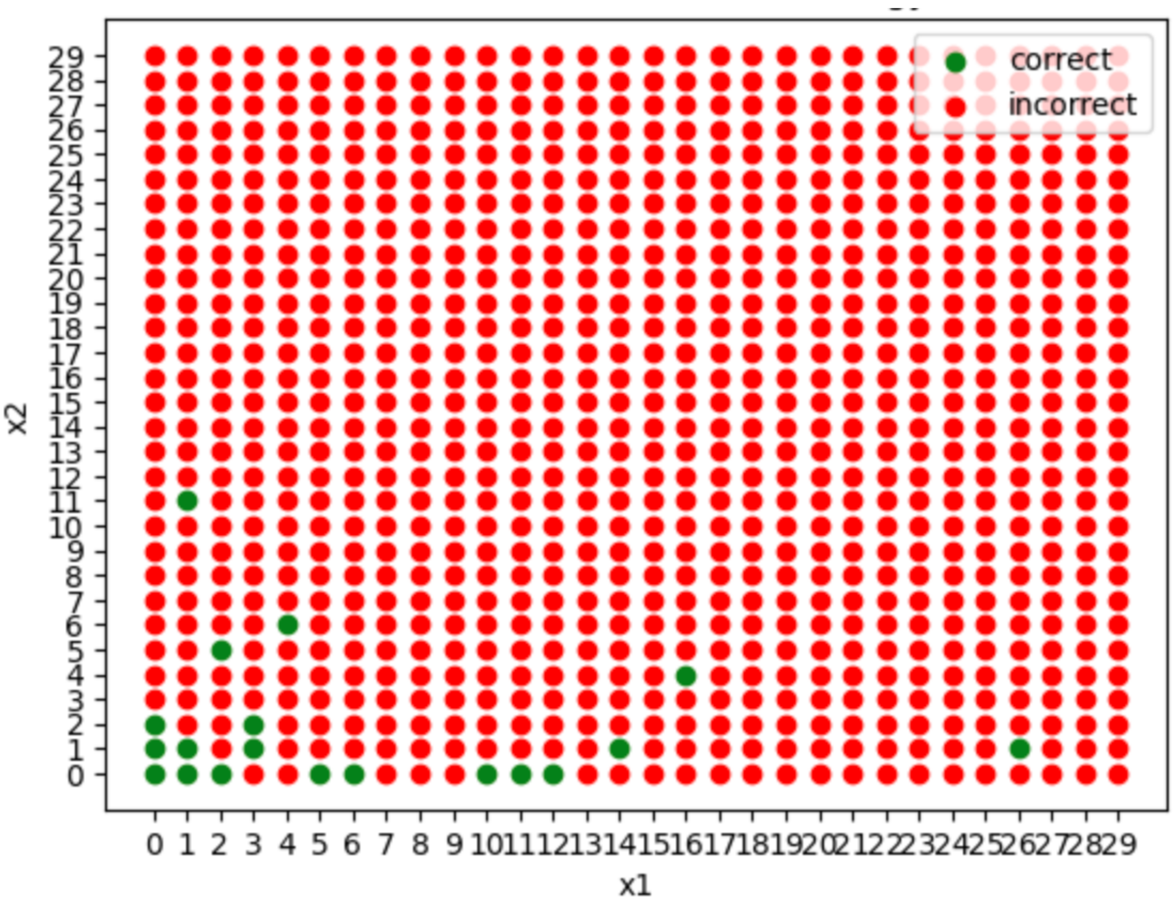
A. LLM Accuracy for the Baseline Strategy = **007777 (0.7%)**

B. LLM Accuracy for the New Strategy = **0.02 (2%)**

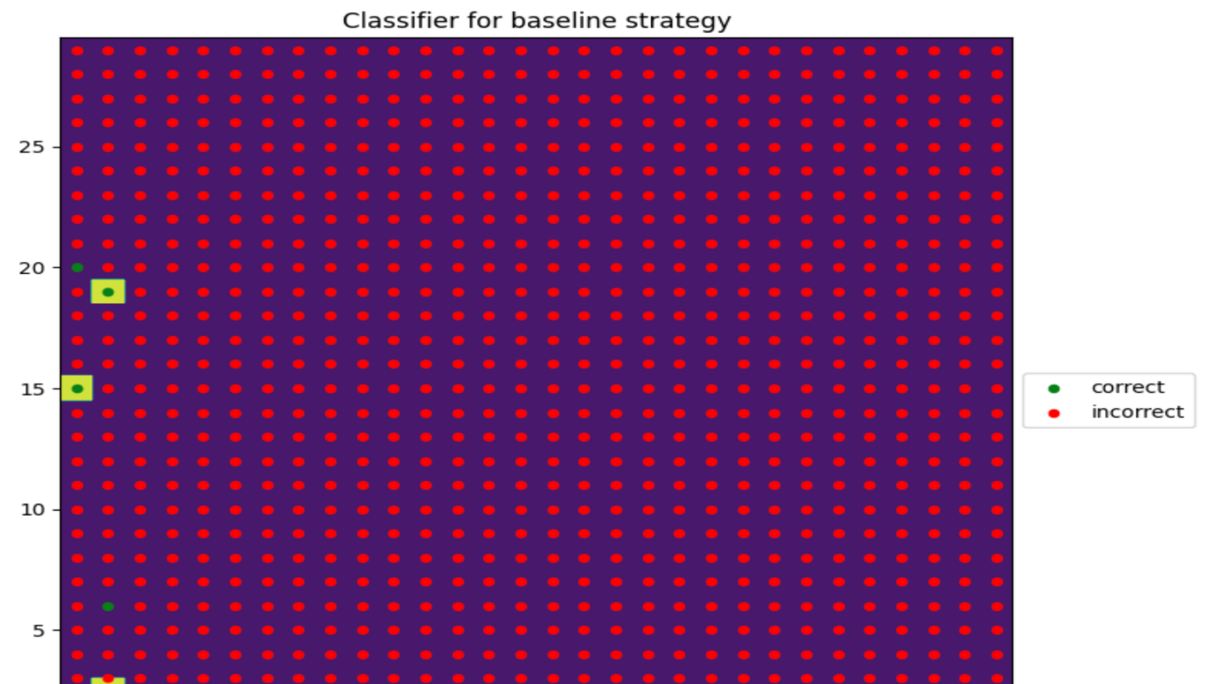
C. Scatter Plot for the **BaseLine Strategy.**



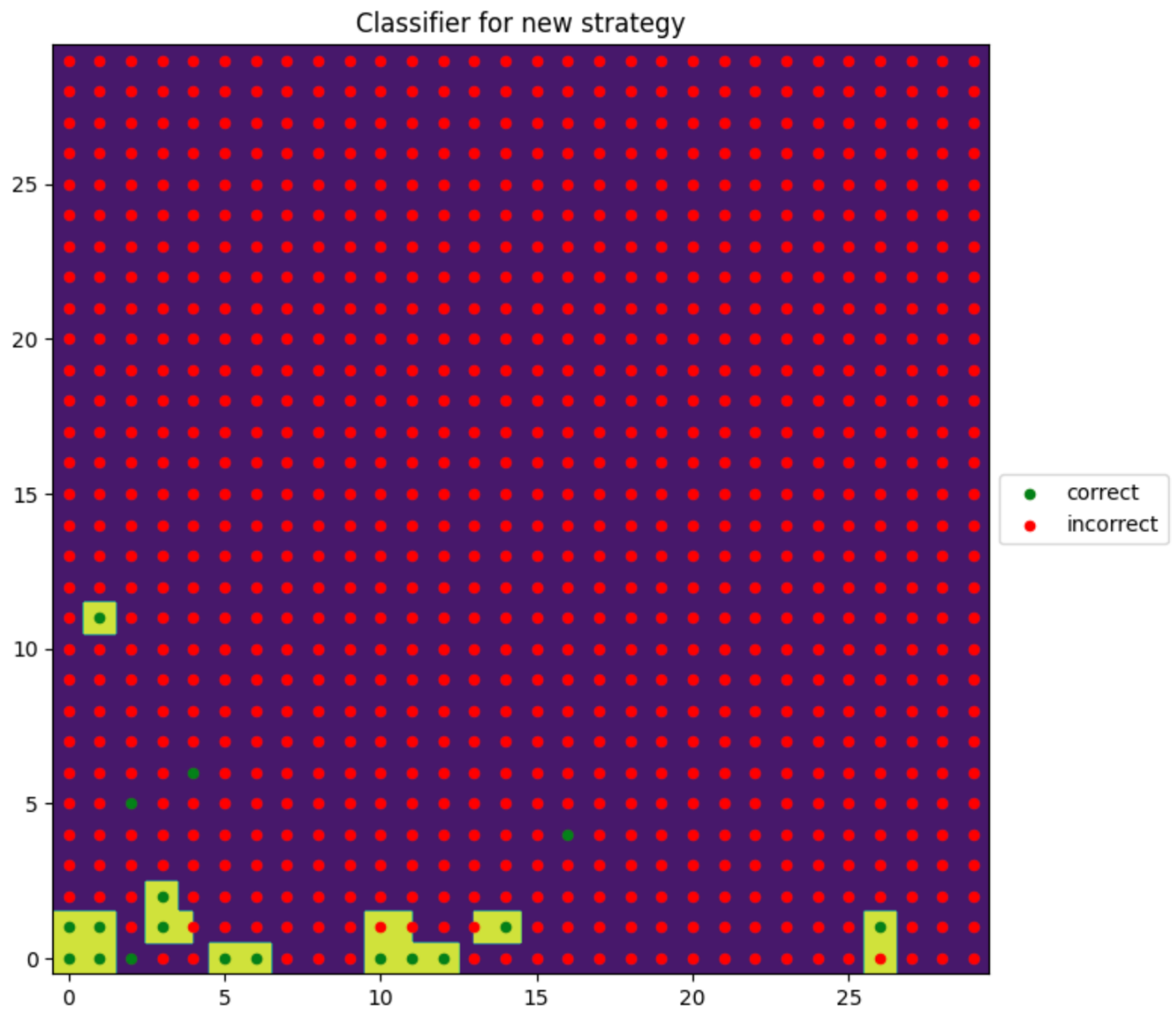
D. Scatter Plot for the **New Strategy**



E. The Classifier Accuracy score for the baseline strategy is: 0.988 (98%)



F. The Classifier Accuracy score for the new strategy is : 0.97 (97%)


Classifier for new strategy
correct
incorrect