Optimized Demonstration Selection for GPT-4 based NER Model
Project Report
Jayant Sharma (jayantsh)

Part – 1 (Methodology):

LLM Experiment Model:

The task presented is a Named Entity Recognition task which has to **classify** the entities among a set of seven labels. I started with researching on prompt engineering for LLMs, through review scholarly articles and class lecture. Sahoo et al. [1] clearly overviews techniques for prompt engineering for LLMs, but none except ***zero and few shot learning*** are effective for the multinomial classification task that NER demands.

As we have tried that approach already in the LLM Baseline model, results in *Table 1*; this limits us to experimenting with demonstrations largely. Another factor is that our dataset is **limited to one given** (data_splits['train']).

Therefore, I decided to **experiment with the variation in the training of the model,** so that all labels are given an equal weightage, as I could spot a direct correlation between **the number of times a label occurs in the validation dataset** and **the F1 score for that label** *(Figure 1),* but **the labels which occurred first (and consequently more) in the training set ended up getting a higher F1 score, despite having a low count in the validation set.**

Demonstration Selection Experiment:

Numerous articles such as *"Strategic Demonstration Selection for Improved Fairness in LLM In-Context Learning"* by J.Hu et al. [2] have used strategies that select demonstrations for LLMs ***based on the confidence it shows in predicting certain labels***. I have used a simpler metric to assess the model's ability to label an entity correctly i.e. **the F1 score of the label's identification in the validation dataset.**

Observation:
I got the best F1 score in the validation dataset for five shot learning, so I have used that for the further analysis. The following are the results of the labels after the model was run on the validation dataset, with an overall F1 score of 0.263. ***(Table 1):***

Label	**Frequency of occurrence**	**F1 Score**
Deity	115	0.345
Goddess	68	0.269
Aquatic Animal	62	0.217
Cretaceous dinosaur	36	0.346
Aquatic Mammal	35	0.043
Mythological King	14	0.088
Aquatic Plant	0	0

While we can spot a positive correlation in the two variables, **why do Cretaceous dinosaur and Aquatic Mammal have such a stark difference in their F1 scores, while having a similar frequency of occurrence.** *(Refer to Figure 1: Scatter Plot)*

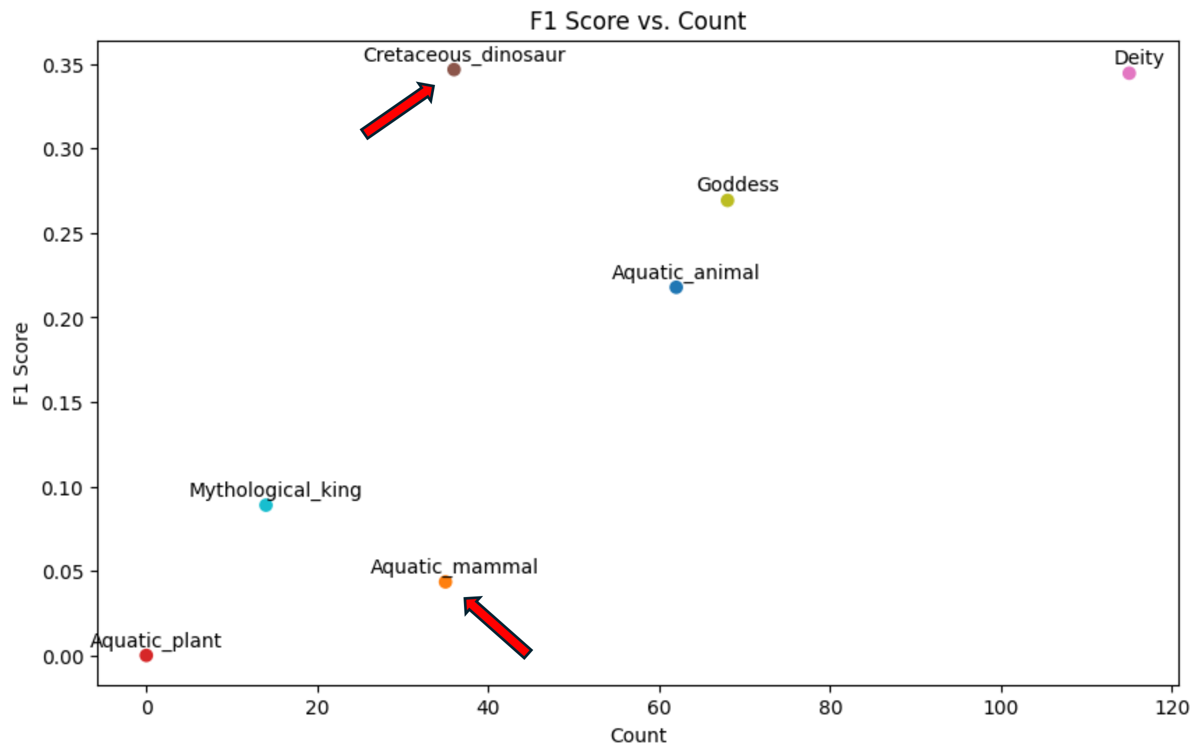


Figure 1: F1 Score for LLM Baseline vs Count of labels in the validation dataset. All labels except Aquatic Mammal and Creataous Dinosaur fall into the x=y line.

Inference:
The LLM is learning to identify some labels **better** than the others, or the prompts given to the LLM are **biased** to some labels than the others, which could be the reason for these differences in metrics for labels having the same count.

The Training Data:

In the LLM Baseline code, the number of shots is set to a specific value, which determines the amount of train examples which are prompted to the LLM before asking it to predict on the validation/test dataset. But through the **get_chat_history** function, **the order** in which the training data is **appended** at the end of the **system message** remains constant; **which is infact not diverse in terms of labels.** If the training data is imbalanced (e.g., certain labels dominate), focusing on a diverse mix of examples in your prompt is a deliberate engineering choice to expose the model to underrepresented or challenging labels.

I wrote a function **count_ner_labels** which calculates the occurrences of different labels in the first n examples of the training data, which end up becoming the demonstration for the LLM.

For first **five** training examples: ***(Table 2):***

Label	**Frequency of occurrence**
Goddess	5
Deity	1
Cretaceous dinosaur	1

For first **twenty** training examples: ***(Table 3):***

Label	**Frequency of occurrence**
Aquatic Animal	15
Goddess	9
Deity	5
Cretaceous dinosaur	2
Mythological King	1

For first **hundred** training examples**: *(Table 4):***

Label	**Frequency of occurrence**
Aquatic Animal	52
Deity	32
Goddess	32
Cretaceous dinosaur	14
Aquatic Mammal	11
Mythological King	8

This can serve as a possible explanation for the reason some labels perform better in being identified that others (among other factors). As I used 5 shot training for my LLM results shown in Table 1, **the reason Cretaceous Dinosaur is identified more effectively is because it occurs more in the training examples (Table 2) than Aquatic Mammal, despite the same occurrence in the validation dataset**.

Implementation in the new LLM Model

While a higher few shot learning (shots > 100-150) would be the desired opportune method to get better performance for the LLM, it is **computationally inefficient and cost intensive**, since GPT charges the usage of credits based on the number of tokens in the prompt. Therefore, a diverse training dataset **(or merely changing the order**) could be done to enable a selective demonstration for the LLM. I made changes to the function **get_chat_history** to write a new function: **get_shuffled_chat_history** which **shuffles the training examples (demonstrations) input to the chat history**, but **counting the labels** in the first n shots specified, and shuffles it to achieve a high diversity rating (more diverse labels) for the given set of messages.
Code in the Appendix section.

Basically, the **get_shuffled_chat_history** function starts by adding a system message that sets the context, explaining the task and listing the entity types to label. To ensure diversity and balance in the training examples, the function first counts how often each label occurs in the dataset. It then groups examples by their entity labels and shuffles each group, creating a stratified and representative dataset using **random.shuffle()**

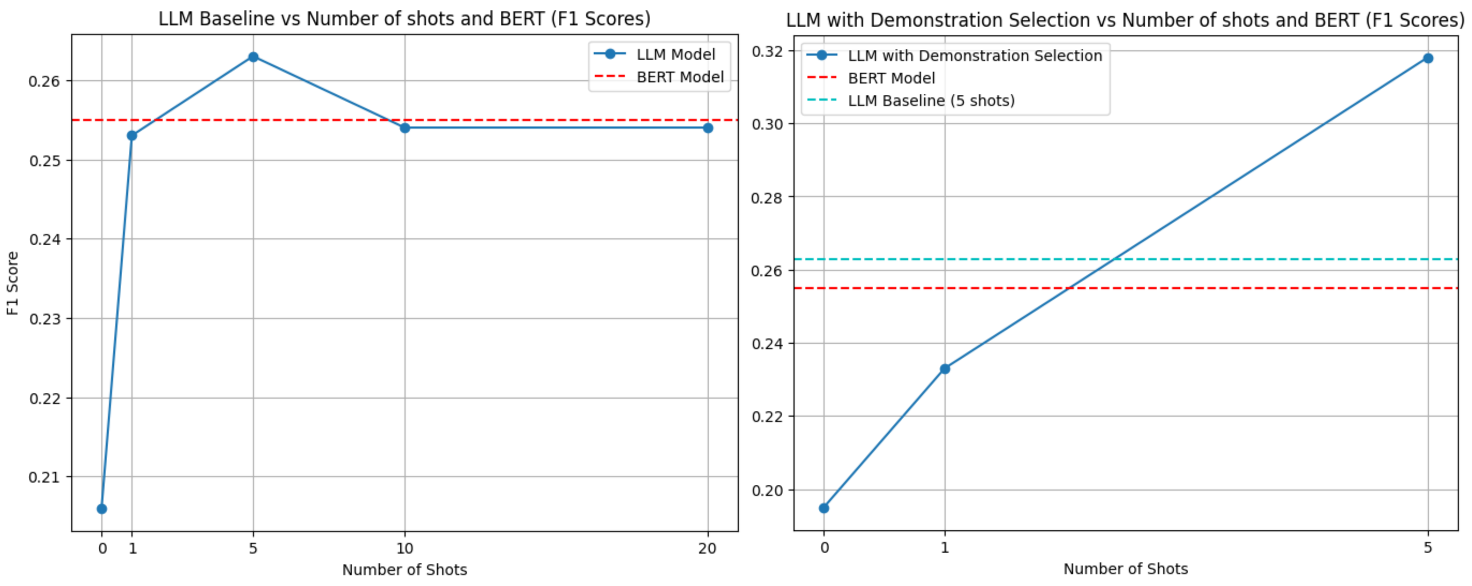
Part - 2 (Experimental Results)

Table 5: Shows the metrics of the BERT model, LLM Baseline and LLM with Demonstration Selection with varying shots.

Approach	Shots	Accuracy	Precision	Recall	F1
Finetune BERT	-	93.6%	0.317	0.213	0.255
Zero-shot LLM Baseline	0	94.00%	0.175	0.251	0.206
Few-shot LLM Baseline	1	94.89%	0.261	0.245	0.253
Few-shot LLM Baseline	5	95.94%	0.301	0.233	0.263
Few-shot LLM Baseline	10	96.23%	0.310	0.215	0.254
Few-shot LLM Baseline	20	95.71%	0.293	0.224	0.254
Zero-shot LLM with Demonstration selection	0	94.26%	0.164	0.242	0.195
Zero-shot LLM with Demonstration selection	1	95.02%	0.216	0.254	0.233
Zero-shot LLM with Demonstration selection	5	95.88%	0.302	0.336	0.318

Figure 2: Graphically represents the performance of the BERT Model and the LLM Baseline as the number of shots is increased. An increase in F1 score can be seen for the LLM Baseline, as the number of shots are increased. This is supported by the fact that the LLM can make accurate identification of the entities, if it is shown more demonstrations of the task.

Figure 2: LLM Baseline F1 scores w.r.t increasing number of shots in few shot learning. BERT F1 score is also represented as a reference (left). LLM experiment (with demonstration selection vs number of shorts)



Part – 3 (Analysis and Discussion)

Referring to Figure 2, it is encouraging to see the LLM with demonstration experiment get a better overall performance for the same number of shots (shots = 5) than the LLM Baseline model.

After using **get_shuffled_chat_history** function, it is clear that the demonstration examples shown to the LLM model are more diverse than the previous function. Upon further analysis, I modified the earlier **count_ner_labels** to get the frequency of labels occurring in the first few training examples after demonstration selection. **Table 6** shows the results:

For first **five** training examples: ***(Table 6):***

Label	**Frequency of occurrence in LLM Baseline**	**Frequency of occurrence in LLM with Demonstration Selection**
Goddess	5	2
Deity	1	1
Cretaceous dinosaur	1	4
Aquatic Animal	0	1

Similarly for first **twenty** training examples: ***(Table 7):***

Label	**Frequency of occurrence in LLM Baseline**	**Frequency of occurrence in LLM with Demonstration Selection**
Aquatic Animal	15	9
Goddess	9	11
Deity	5	6
Cretaceous dinosaur	2	7
Mythological King	1	4

Addition of certain labels such as Aquatic Animal in the five shot demonstration is an encouraging and positive result, cause now the model has a better idea of the example. Furthermore we can see the check this though the **standard deviation** of the frequency of the labels, both for LLM baseline and LLM demonstration selection.

1. Standard deviation of label frequency for LLM Baseline: 5.122
2. Standard deviation of label frequency for LLM with Demonstration Selection: 2.416

So an overall decrease in the standard deviation for the label frequency in the train dataset is a really positive thing for the dataset, making it more diverse and varied.

Since I have used random.shuffle(), these results are hard to replicate but I have used random.seed to curb that issue. Also, the code takes a LOT more time to run to get the metrics after calling the API, since it has to preprocess the training data every time. The five shot run took more than an hour so I couldn't go higher than 5 shots for the LLM with Demonstration selection.

Affect on label specific metrics (Aquatic Mammal vs Cretaceous Dinosaur)

Does this really help the model predict labels in the validation dataset accurately? I took a final deep dive into the label specific metrics of certain labels. *The results of all three runs (zero, one and five shot on the validation dataset are given in the appendix)*

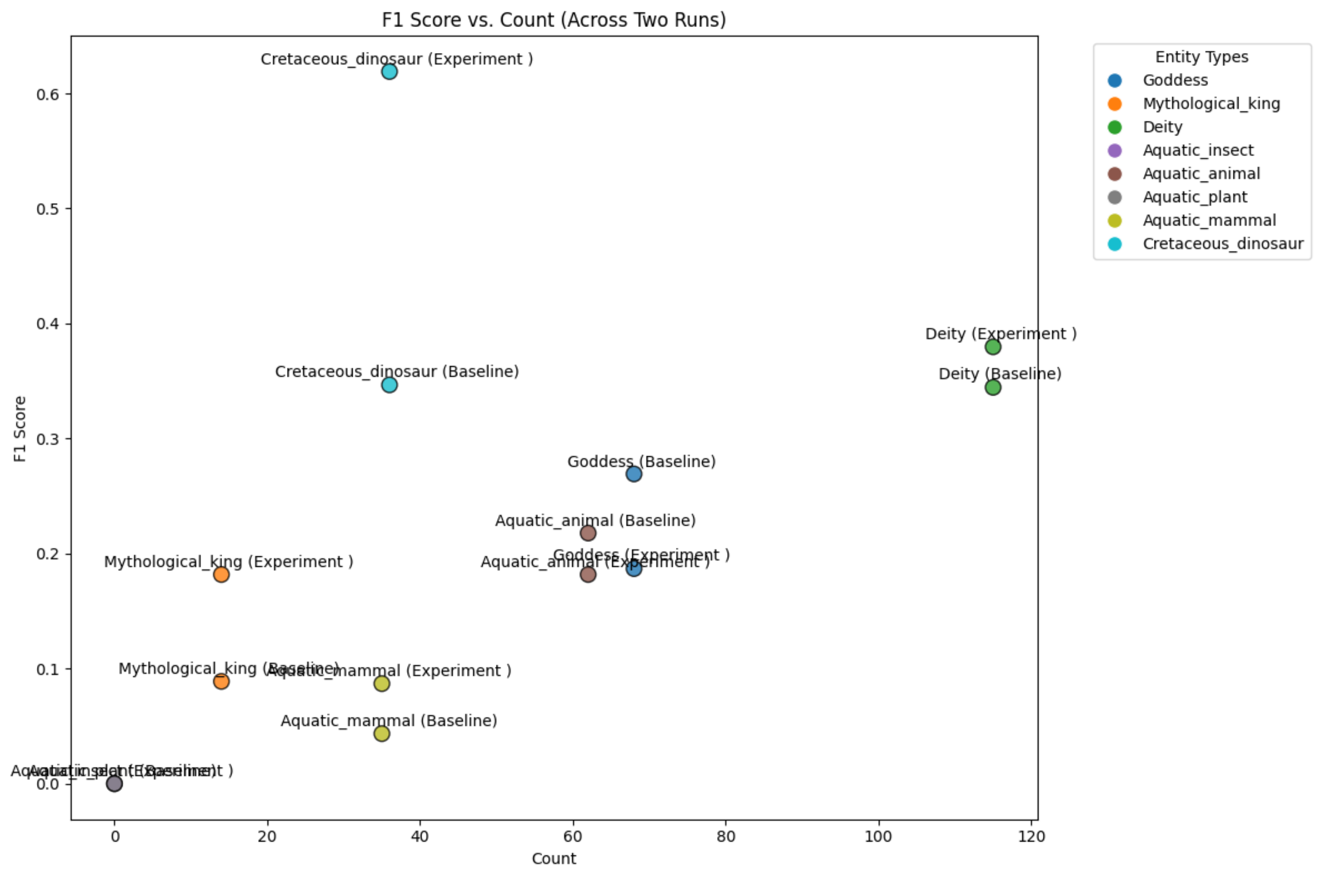


Figure 3: F1 Score vs Counts for both LLM Baseline and LLM with Demonstration Selection

Figure 3 shows an increase in the F1 score of most of the labels, showing the the demonstration selection experiment runs well, and provides an increase in performance. More increase in the F1 Score or other metrics can be made by increasing the number of shots, with demonstration selection.

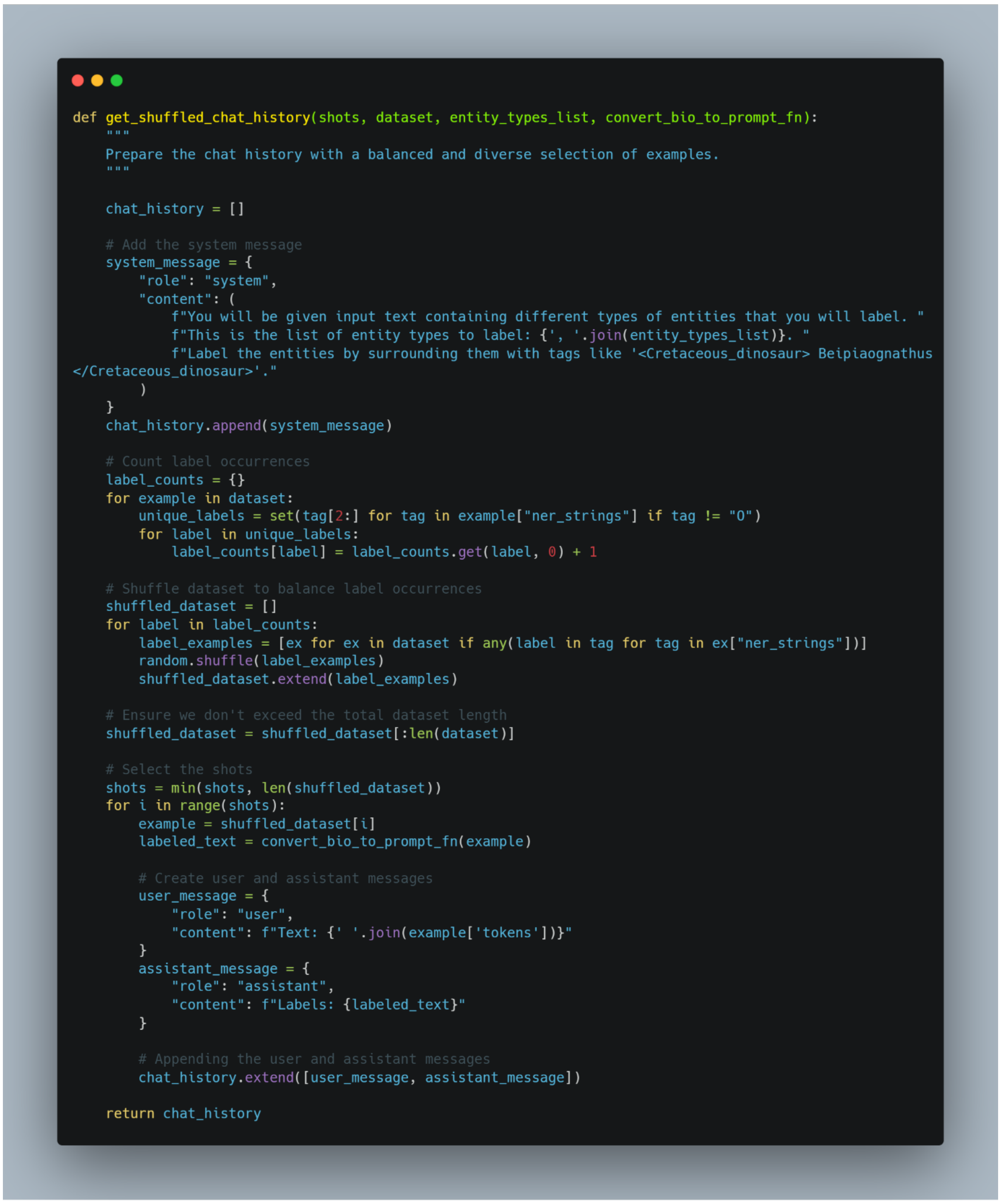
A significant increase in Cretatoues Dinosaur might be due to the sheer number of labels in the training dataset, which would subside if more shots are incorporated/ a more efficient code can be written.

Conclusion

The LLM performance increased significantly, as compared to the LLM Baseline model for the same number of shots (initially tested with 5). Additionally, the identification of labels that did not have a high count in the **validation set** (Mythological King and Aquatic Mammal) was much better than the LLM baseline model, shown by an increase in F1 score.

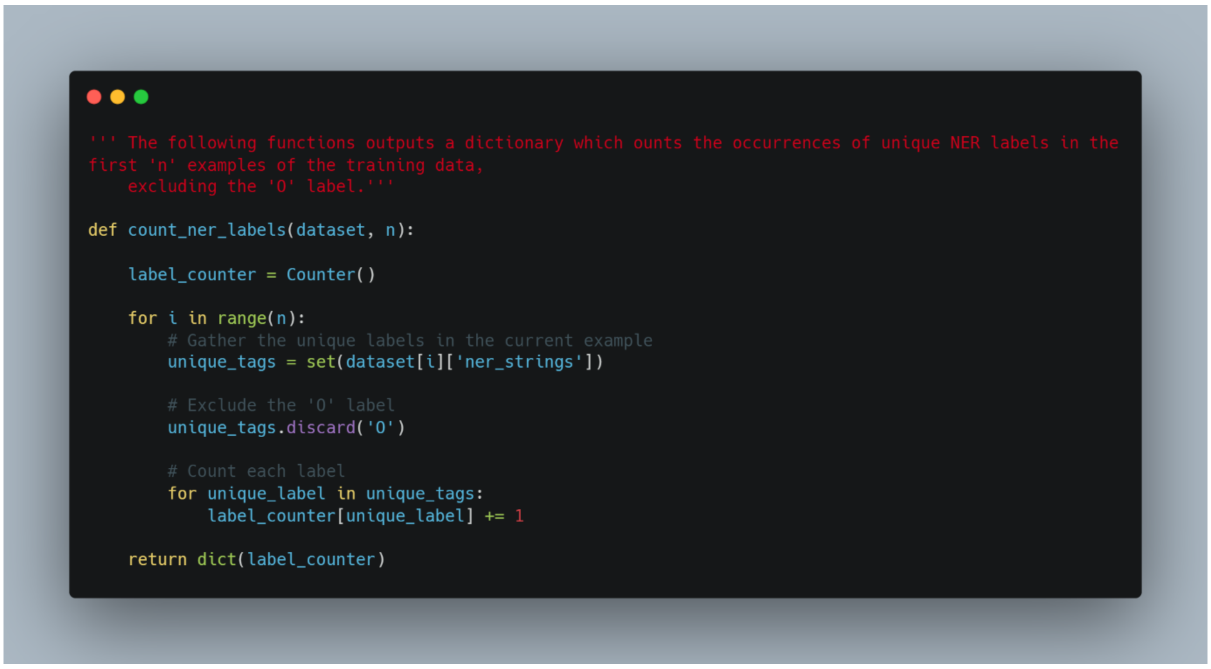
Appendix:

get_shuffled_chat_history function: This was the main function that I changed to run my experiment, and the **call_openai_api** function called this function instead of **get_chat_history** function on the validation dataset.



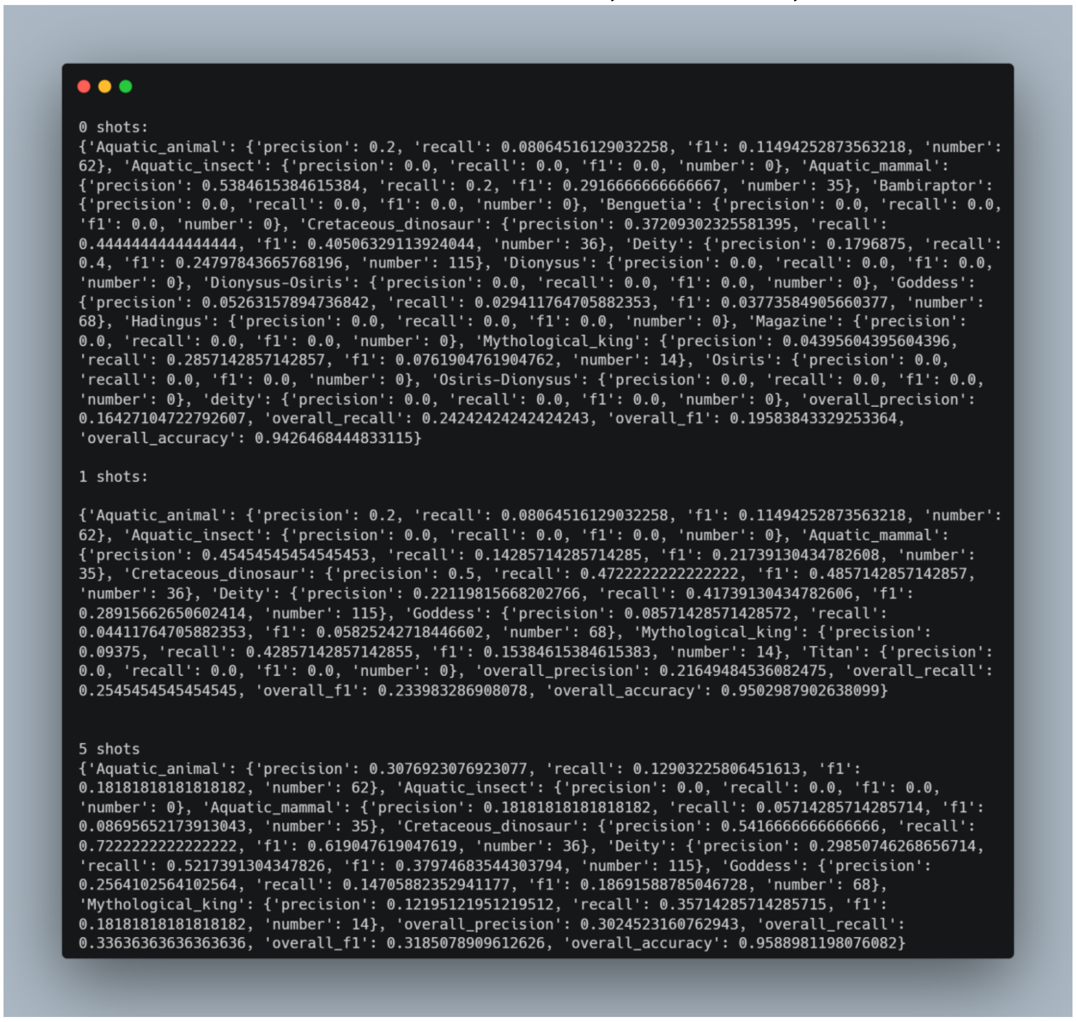
```python
def get_shuffled_chat_history(shots, dataset, entity_types_list, convert_bio_to_prompt_fn):
    """
    Prepare the chat history with a balanced and diverse selection of examples.
    """

    chat_history = []

    # Add the system message
    system_message = {
        "role": "system",
        "content": (
            f"You will be given input text containing different types of entities that you will label. "
            f"This is the list of entity types to label: {', '.join(entity_types_list)}. "
            f"Label the entities by surrounding them with tags like '<Cretaceous_dinosaur> Beipiaognathus
</Cretaceous_dinosaur>'."
        )
    }
    chat_history.append(system_message)

    # Count label occurrences
    label_counts = {}
    for example in dataset:
        unique_labels = set(tag[2:] for tag in example["ner_strings"] if tag != "O")
        for label in unique_labels:
            label_counts[label] = label_counts.get(label, 0) + 1

    # Shuffle dataset to balance label occurrences
    shuffled_dataset = []
    for label in label_counts:
        label_examples = [ex for ex in dataset if any(label in tag for tag in ex["ner_strings"])]
        random.shuffle(label_examples)
        shuffled_dataset.extend(label_examples)

    # Ensure we don't exceed the total dataset length
    shuffled_dataset = shuffled_dataset[:len(dataset)]

    # Select the shots
    shots = min(shots, len(shuffled_dataset))
    for i in range(shots):
        example = shuffled_dataset[i]
        labeled_text = convert_bio_to_prompt_fn(example)

        # Create user and assistant messages
        user_message = {
            "role": "user",
            "content": f"Text: {' '.join(example['tokens'])}"
        }
        assistant_message = {
            "role": "assistant",
            "content": f"Labels: {labeled_text}"
        }

        # Appending the user and assistant messages
        chat_history.extend([user_message, assistant_message])

    return chat_history
```

The **count_ner_labels** was a preprocessing step, I used it to do data analysis on the training data to show the frequency of labels occurring in the first ‘n’ examples of the training data.



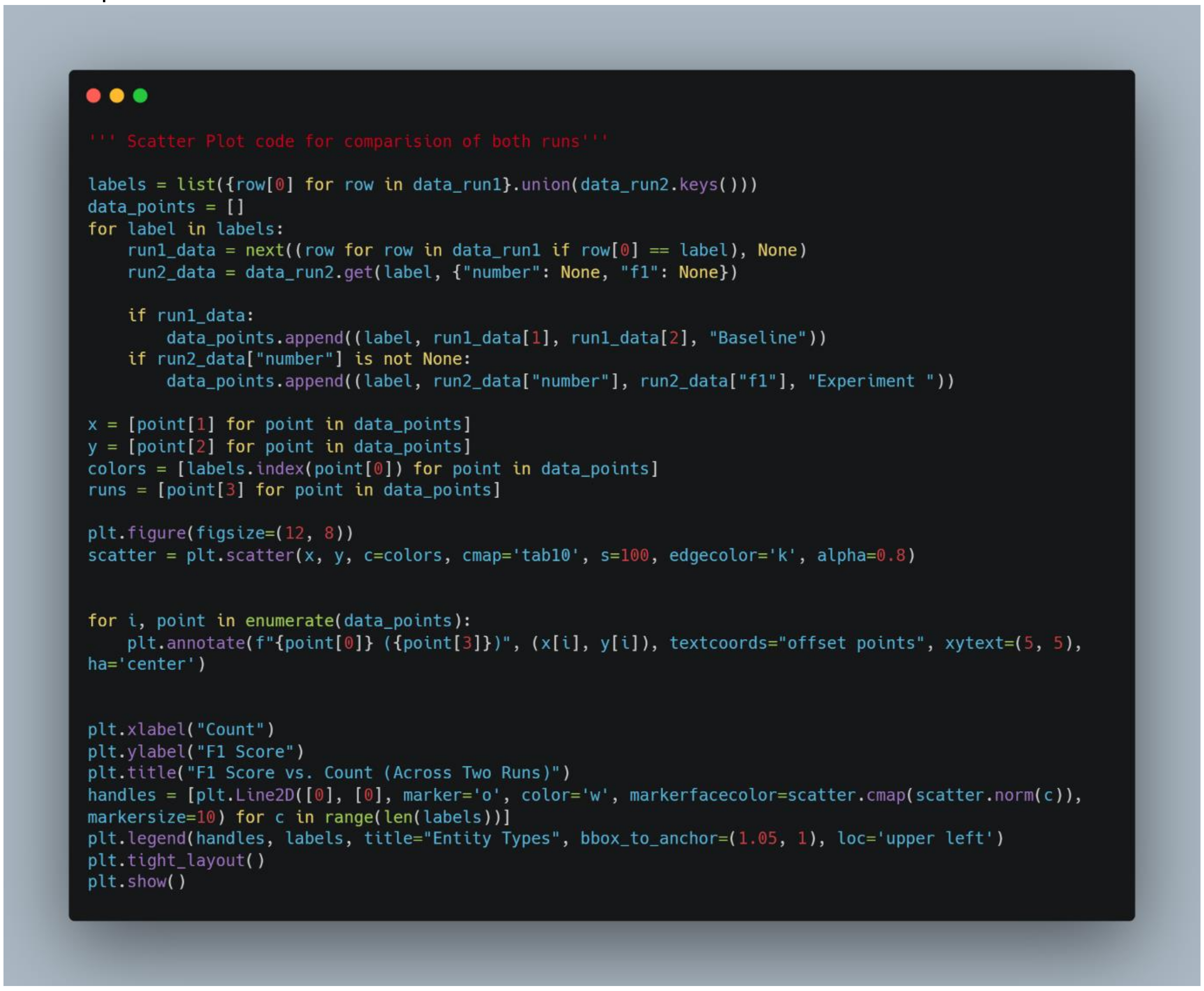
```python
''' The following functions outputs a dictionary which ounts the occurrences of unique NER labels in the
first 'n' examples of the training data,
    excluding the 'O' label.'''

def count_ner_labels(dataset, n):

    label_counter = Counter()

    for i in range(n):
        # Gather the unique labels in the current example
        unique_tags = set(dataset[i]['ner_strings'])

        # Exclude the 'O' label
        unique_tags.discard('O')

        # Count each label
        for unique_label in unique_tags:
            label_counter[unique_label] += 1

    return dict(label_counter)
```

LLM with demonstration selection results for 0, 1 and 5 shots, both overall and label specific.



```
0 shots:
{'Aquatic_animal': {'precision': 0.2, 'recall': 0.08064516129032258, 'f1': 0.11494252873563218, 'number':
62}, 'Aquatic_insect': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'Aquatic_mammal':
{'precision': 0.5384615384615384, 'recall': 0.2, 'f1': 0.2916666666666667, 'number': 35}, 'Bambiraptor':
{'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'Benguetia': {'precision': 0.0, 'recall': 0.0,
'f1': 0.0, 'number': 0}, 'Cretaceous_dinosaur': {'precision': 0.37209302325581395, 'recall':
0.4444444444444444, 'f1': 0.40506329113924044, 'number': 36}, 'Deity': {'precision': 0.1796875, 'recall':
0.4, 'f1': 0.24797843665768196, 'number': 115}, 'Dionysus': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0,
'number': 0}, 'Dionysus-Osiris': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'Goddess':
{'precision': 0.05263157894736842, 'recall': 0.029411764705882353, 'f1': 0.03773584905660377, 'number':
68}, 'Hadingus': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'Magazine': {'precision':
0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'Mythological_king': {'precision': 0.04395604395604396,
'recall': 0.2857142857142857, 'f1': 0.07619047619047620, 'number': 14}, 'Osiris': {'precision': 0.0,
'recall': 0.0, 'f1': 0.0, 'number': 0}, 'Osiris-Dionysus': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0,
'number': 0}, 'deity': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'overall_precision':
0.16427104722792607, 'overall_recall': 0.24242424242424243, 'overall_f1': 0.19583843329253364,
'overall_accuracy': 0.9426468444833115}

1 shots:

{'Aquatic_animal': {'precision': 0.2, 'recall': 0.08064516129032258, 'f1': 0.11494252873563218, 'number':
62}, 'Aquatic_insect': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'Aquatic_mammal':
{'precision': 0.45454545454545453, 'recall': 0.14285714285714285, 'f1': 0.21739130434782608, 'number':
35}, 'Cretaceous_dinosaur': {'precision': 0.5, 'recall': 0.4722222222222222, 'f1': 0.4857142857142857,
'number': 36}, 'Deity': {'precision': 0.22119815668202766, 'recall': 0.41739130434782606, 'f1':
0.28915662650602414, 'number': 115}, 'Goddess': {'precision': 0.08571428571428572, 'recall':
0.04411764705882353, 'f1': 0.05825242718446602, 'number': 68}, 'Mythological_king': {'precision':
0.09375, 'recall': 0.42857142857142855, 'f1': 0.15384615384615383, 'number': 14}, 'Titan': {'precision':
0.0, 'recall': 0.0, 'f1': 0.0, 'number': 0}, 'overall_precision': 0.21649484536082475, 'overall_recall':
0.2545454545454545, 'overall_f1': 0.233983286908078, 'overall_accuracy': 0.9502987902638099}


5 shots
{'Aquatic_animal': {'precision': 0.3076923076923077, 'recall': 0.12903225806451613, 'f1':
0.18181818181818182, 'number': 62}, 'Aquatic_insect': {'precision': 0.0, 'recall': 0.0, 'f1': 0.0,
'number': 0}, 'Aquatic_mammal': {'precision': 0.18181818181818182, 'recall': 0.05714285714285714, 'f1':
0.08695652173913043, 'number': 35}, 'Cretaceous_dinosaur': {'precision': 0.5416666666666666, 'recall':
0.7222222222222222, 'f1': 0.619047619047619, 'number': 36}, 'Deity': {'precision': 0.29850746268656714,
'recall': 0.5217391304347826, 'f1': 0.37974683544303794, 'number': 115}, 'Goddess': {'precision':
0.2564102564102564, 'recall': 0.14705882352941177, 'f1': 0.18691588785046728, 'number': 68},
'Mythological_king': {'precision': 0.12195121951219512, 'recall': 0.35714285714285715, 'f1':
0.18181818181818182, 'number': 14}, 'overall_precision': 0.3024523160762943, 'overall_recall':
0.33636363636363636, 'overall_f1': 0.3185078909612626, 'overall_accuracy': 0.9588981198076082}
```

Scatter plot code:



```
''' Scatter Plot code for comparision of both runs'''

labels = list({row[0] for row in data_run1}.union(data_run2.keys()))
data_points = []
for label in labels:
    run1_data = next((row for row in data_run1 if row[0] == label), None)
    run2_data = data_run2.get(label, {"number": None, "f1": None})

    if run1_data:
        data_points.append((label, run1_data[1], run1_data[2], "Baseline"))
    if run2_data["number"] is not None:
        data_points.append((label, run2_data["number"], run2_data["f1"], "Experiment "))

x = [point[1] for point in data_points]
y = [point[2] for point in data_points]
colors = [labels.index(point[0]) for point in data_points]
runs = [point[3] for point in data_points]

plt.figure(figsize=(12, 8))
scatter = plt.scatter(x, y, c=colors, cmap='tab10', s=100, edgecolor='k', alpha=0.8)


for i, point in enumerate(data_points):
    plt.annotate(f"{point[0]} ({point[3]})", (x[i], y[i]), textcoords="offset points", xytext=(5, 5),
ha='center')


plt.xlabel("Count")
plt.ylabel("F1 Score")
plt.title("F1 Score vs. Count (Across Two Runs)")
handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=scatter.cmap(scatter.norm(c)),
markersize=10) for c in range(len(labels))]
plt.legend(handles, labels, title="Entity Types", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

References:

1. Sahoo, Pranab, et al. "A systematic survey of prompt engineering in large language models: Techniques and applications." arXiv preprint arXiv:2402.07927 (2024).
2. Hu, J., Liu, W., & Du, M. (2024). Strategic Demonstration Selection for Improved Fairness in LLM In-Context Learning. arXiv preprint arXiv:2408.09757.