Exploring the Performance of LLaMA 7B-Adapter

Soyoung Oh
EPFL
soyoung.oh@epfl.ch

Abstract

Adapter-based tuning has recently arisen as an efficient alternative to fine-tuning by adding lightweight adapter modules to pre-trained large language models. By adding only a few trainable parameters to adapt to a new downstream task, adapter-based tuning allows to minimize computational and resource footprints while achieving comparable results to fine-tuning. In this report, we conduct several empirical experiments to explore the performance of an adapter module with a backbone of a large language model on several downstream tasks, which include low-resource language task. We demonstrate the efficiency of using an adapter module to significantly enhance the performance of the large language model in benchmarks. Our code implementation is publicly available[1].

1 Introduction

Large Language Models (LLMs) have achieved state-of-the-art results on most natural language processing tasks (Dai et al., 2019; Devlin et al., 2018; Radford et al., 2019; Raffel et al., 2020). Following instructions in natural language, LLMs can generate desired responses to specific downstream tasks within the framework of in-context learning. However, the most potent instruction-following LLMs, such as ChatGPT[2] and GPT-4[3], are currently in a closed-source where their training dataset and source code are not accessible to the users. Therefore, although fine-tuning the LLMs on task-specific datasets usually leads to superior results, the users cannot utilize the LLMs as backbone models for developing fine-tuning methods for specific downstream tasks (Brown et al., 2020; Raffel et al., 2020; Taori et al., 2023).

To mitigate this issue, Taori et al. (2023) introduce the instruction-following language model Alpaca, which is fine-tuned from Meta's LLaMA 7B model (Meta, 2023) with training dataset comprising 52K instruction-following demonstrations generated in the style of self-instruct using text-davinci-003. However, the massive size of LLMs makes updating the entire parameters highly challenging and inefficient. To overcome this challenge, several parameter-efficient fine-tuning techniques (PEFT) have been developed to finetune LLMs with high modelling performance while only requiring the training of only a small subset of parameters (Lialin et al., 2023). One PEFT technique that integrates adapter-based PEFT on LLaMA model (i.e., LLaMA-Adapter) recently made big waves in generative applications (Zhang et al., 2023). This lightweight adaption method efficiently finetunes LLaMA 7B model by introducing 1.2M learnable parameters upon the frozen LLaMA 7B model.

In this report, we conduct several empirical experiments in order to generalize the efficiency of the LLaMA-adapter technique in a wide range of NLP downstream tasks. We evaluate the fine-tuning efficacy on question answering (i.e., ScienceQA (Lu et al., 2022a), ArchivalQA (Wang et al., 2022)) and named entity recognition (Ehrmann et al., 2022) benchmarks. For the named entity recognition (NER) task, we use a historical French dataset to validate the effectiveness of the given technique in processing low-resource dataset. Throughout these experiments, we observe significant improvements on these NLP tasks by efficiently fine-tuning the adapter module with the LLaMA 7B model.

2 Related work

Utilizing LLMs in an efficient and effective manner has become increasingly important. PEFT allows users to train models on a broader range of hardware, including devices with limited computational

[1] https://github.com/dhlab-epfl/historical-adapters
[2] https://chat.openai.com/chat
[3] https://chat.openai.com/chat?model=gpt-4

power. One of the PEFT techniques, adapters for LLMs which contain a few extra trainable parameters, allow efficient fine-tuning of specific tasks without affecting the pre-trained parameters of the LLMs. This framework presents three types of adapters within LLMs: (1) Series Adapter (2) Parallel Adapter (3) Low-Rank Adaptation (LoRA).

Series Adapter. The Series Adapter (Houlsby et al., 2019), also known as a bottleneck adapter, comprises a two-layer feed-forward neural network. This network includes a down-projection matrix, a non-linearity function, an up-projection, and a residual connection between the input and output.

Parallel Adapter. The Parallel Adapter (Pfeiffer et al., 2020) integrates the bottleneck adapters in parallel with the multi-head and feed-forward layers of the transformer block in LLMs. The adapters are incorporated alongside each transformer layer.

LoRA. Hu et al. (2021) propose LoRA to introduce trainable low-rank decomposition matrices into LLMs' existing layers, enabling the model to adapt to new data while keeping the previous knowledge fixed on LLMs. In detail, LoRA performs a reparameterization of each model layer, expressed as a matrix multiplication by injecting low-rank decomposition matrices. This reparametrization allows the model to undergo fine-tuning without requiring the computation of the entire dense matrix multiplication. Additionally, by lowering the rank of the matrices, LoRA also aids in reducing the parameter count during the fine-tuning of LLMs.

Under the adapter framework, Zhang et al. (2023) extends the ideas of prefix tuning and adapter method to integrate open-access LLaMA-7B model. As in prefix tuning, the LLaMA-Adapter methods prepends tunable prompt tensors to the embedded inputs. The L topmost transformer layers of the model has its own distinct learned prefix, allowing for more tailored higher-level semantic adaptation. Additionally, LLaMA-Adapter introduces a zero-initialized attention mechanism coupled with gating factor to stabilize the training.

3 Experiments

3.1 Implementation Details

We train LLaMa-Adapter on 8 A100 GPUs for 10 to 30 epochs based on the size of the benchmark. The warmup epochs, batch size, learning rate, and weight decay are set to 2, 64, 9×10^{-3} and 0.02, respectively. In general, we utilize the pre-trained LLaMA model with 7B parameters and $N = 32$ transformer layers as the base model. We set the prompt length $K = 10$ and insert prompts into the last $L = 30$ layers by default. In the generation stage, we adopt *top-p* sampling as the default decoding method with a temperature 0.1 and a *top-p* $= 0.75$.

Furthermore, to assess the enhancement of the LLaMA 7B model through fine-tuning, we perform in-context learning experiments using benchmarks that include zero and 3-shot learnings.

3.2 ScienceQA

Training Data. We train the text-only LLaMA-Adapter on ScienceQA (Lu et al., 2022b), a large-scale multi-modal and multi-choice science question dataset collected from a wide range of domains[4]. Figure 1 shows a single-modality example with text-only input that we used for the fine-tuning.

Results. In Table 1, we compare LLaMA-Adapter with in-context learning of the LLaMA 7B model which includes zero, and three-shot learnings. As described, the LLaMA-Adapter$_T$ attains 78.28% of accuracy with 1.2M updated parameters. Comparing the average performance of in-context learning to fine-tuning, we observe an increase in answering accuracy by 37.96% for zero-shot learning and 34.59% for three-shot learning.

3.3 ArchivalQA

Training Data. We train the LLaMA-Adapter on ArchivalQA (Wang et al., 2022), a large question-answering (QA) dataset consisting of question-answer pairs which are designed for temporal news QA. As in Figure 2, the open-domain question answering pairs, which attempt to answer natural language questions based on a large-scale unstructured document, are used as input for fine-tuning.

Results. We select the baseline results from the previous research (Wang et al., 2022). The following baseline models show the performances of extractive question answering task. That being said, DrQA-Wiki (Chen et al., 2017) combines a search component based on bigram hashing and TF-IDF with a multi-layer recurrent neural network model trained to extract answers from articles. To solve the historical question answering task from ArchivalQA testset, DrQA-Wiki

[4] https://github.com/lupantech/ScienceQA

Model	Tuned Params	Avg	NAT	SOC	LAN	TXT	IMG	NO	G1-6	G7-12
Random Choice (Lu et al., 2022b)	-	39.83	40.28	46.13	29.25	47.45	40.08	33.66	39.35	40.67
Human (Lu et al., 2022b)	-	88.40	90.23	84.97	87.48	89.60	87.50	88.10	91.59	82.42
LLaMA $7B_T$	-	40.32	41.39	29.25	47.09	43.79	37.93	42.16	40.79	39.49
LLaMA $7B_T$ 3-shot	-	43.69	46.09	33.52	47.00	46.97	40.26	46.13	44.90	41.53
LLaMA 7B-Adapter$_T$	**1.2M**	**78.28**	**79.57**	**73.57**	**79.45**	**78.84**	**70.75**	**82.51**	**80.58**	**74.16**

Table 1: **Question Answering Accuracy (%) on ScienceQA's (Lu et al., 2022b) test set.** We report the accuracy of different question classes, including natural science, social science, language science, text context, image context, and no context, grades 1-6, and grades 7-12. The symbol $_T$ denotes a single-modal variant with text-only input.

Question: What do these two changes have in common?\n
Bleaching clothes\n
A piece of apple turning brown\n
Options: (A) Both are only physical changes. (B) Both are caused by cooling (C) Both are chemical changes. (D) Both are caused by heating.\n
Answer: The answer is C.

Figure 1: Example of ScienceQA dataset with text-only input.

Question: Who claimed responsibility for the bombing of Bab Ezzouar?\n
Answer: Al Qaeda

Question: When did Tenneco announce it was planning to sell its oil and gas operations?\n
Answer: May 26, 1988

Figure 2: Examples of ArchivalQA dataset.

model uses Wikipedia as the knowledge source. The BERT$_{serini}$-Wiki (Yang et al., 2019) tackles end-to-end question answering by combining BERT (Devlin et al., 2018) with Anserini (Yang et al., 2017) information retrieval toolkit where it uses Wikipedia as the knowledge source, whereas BERT$_{serini}$-NYT (Yang et al., 2019) uses NYT archive.

Compared to the baselines, we did not employ a knowledge source for answer extraction. Through in-context learning with a large language model, we demonstrate the model's effectiveness in utilizing pre-trained knowledge to address a new task. We measure the performance of the models using exact match (EM) and F1 score which are the standard measures commonly used in QA research.

In Table 2, we present a performance comparison among baseline models, the LLaMA 7B model, and the fine-tuned LLaMA-Adapter model. We note that the in-context learning with the LLaMA model yields low performance. However, after fine-tuning the adapter module, we observe a significant improvement in performance, with 11.57% increase in exact match and 16.56% increase in F1 score compared to zero-shot learning. This performance is approximately equivalent to that achieved when the baseline models utilize Wikipedia as a knowledge source (i.e., DrQA-Wiki: 7.53, 11.64; BERT$_{serini}$-Wiki: 10.19, 16.25; LLaMA-Adapter: 11.57, 16.59). In other words, through adapter fine-tuning, LLaMA gains the knowledge to handle the new downstream task for which the model was not originally pre-trained.

Model	EM	F1
DrQA-Wiki	7.53	11.64
DrQA-NYT	38.13	46.12
BERT$_{serini}$-Wiki	10.19	16.25
BERT$_{serini}$-NYT	54.30	66.05
LLaMA 7B	0	0.03
LLaMA 7B 3-shot	0.03	0.56
LLaMA 7B-Adapter	11.57	16.59

Table 2: **Question Answering exact match (EM) and F1 score (%) on ArchivalQA's (Wang et al., 2022) testset.**

3.4 HIPE

Training Data. We train the LLaMA-Adapter on HIPE dataset (Ehrmann et al., 2022), which

is a named entity annotated historical newspaper dataset that consists of newspaper articles written in English, German, and French over 19 and 20 centuries. In this dataset, named entity tags include categories of location (LOC), organization (ORG), person (PER), product (PROD), and time (TIME). As in Figure 3, each sentence token is annotated by the named entity in an Inside-Outside-Beginning (IOB) format. The I- prefix before a tag indicates that the token is inside a named entity. An O tag indicates that a token belongs to no entity. The B- prefix before a tag indicates that the token is the beginning of a chunk that immediately follows another chunk without O tags between them.

Token: [Le, public, est, averti, que, Charlotte, née, Bourgoin, ,, femme, -, de, Joseph, Digiez, ,, et, Maurice, Bourgoin, ,, enfant, mineur, représenté, par, le, sieur, Jaques, Charles, Gicot, son, curateur, ,, ont, été, admis, par, arrêt, du, Conseil, d, ', Etat, du, …]

Tag: [O, O, O, O, O, B-pers, I-pers, I-pers, O, O, O, O, B-pers, I-pers, O, O, B-pers, I-pers, O, O, O, O, O, O, B-pers, I-pers, I-pers, I-pers, O, O, O, O, O, O, O, O, O, B-org, I-org, I-org, I-org, …]

Figure 3: Example of the HIPE dataset.

In order to fine-tune the LLaMA Adapter on this dataset, we convert it according to two different prompt structures, one encoding named entity annotations in JSON format, the other in IOB format. We adapt a structured format prompt for NER task that was constructed by ChatGPT[5]. As Figure 4, the JSON format is considered as a typical prompt format for the generation model according to how prompts are generated by ChatGPT.

You are working as a named entity recognition expert and your task is to label a given text with named entity labels.
Your task is to identify and label any named entities present in the text.
The named entity labels that you will be using are TIME (time), LOCATION (location), PERSON (person), ORGANIZATION (organization), and PRODUCT (product).

NOTE: Your output format should be a JSON format, where each data consists of a word from the input text and its corresponding named entity label.

INPUT: Le public est averti que Charlotte née Bourgoin, femme – de Joseph Digiez, et Maurice Bourgoin, …

OUTPUT: [{'entity': 'PERSON', 'text': 'Charlotte née Bourgoin'}, {'entity': 'PERSON', 'text': 'Joseph Digiez'}, …]

Figure 4: Example of the JSON format prompt.

[5]https://twitter.com/samarthrawal/status/1654471254592692230?s=20

Also, we adapt a prompt structure from previous research (Lai et al., 2023) that is developed for ChatGPT to do the NER task (i.e., PromptNER = [task description; output format note; input sentence], which involves a task description to explain the task and list entity tags of interest. We modify the prompt according to our specific interests, as in Figure 5.

A key difference from the typical NER task to our prompt-based experiment is that we explicitly ask the language model to generate named entity tags.

You are working as a named entity recognition expert and your task is to label a given text with named entity labels.
Your task is to identify and label any named entities present in the text.
The named entity labels that you will be using are TIME (time), LOC (location), PER (person), ORG (organization), and PROD (product).
You may encounter multi-word entities, to make sure to label each word of the entity with the appropriate prefix ("B" for the first word of the entity, "I" for any non-initial word of the entity).
For words which are not part of any named entity, you should return "O".

NOTE: Your output format should be a list of tuples, where each tuple consists of a word from the input text and its corresponding named entity label.

Input: ['Le', 'public', 'est', 'averti', 'que', 'Charlotte', 'née', 'Bourgoin', ',', 'femme', '-', 'de', 'Joseph', 'Digiez', …]

Output: [('Le', 'O'), ('public', 'O'), ('est', 'O'), ('averti', 'O'), ('que', 'O'), ('Charlotte', 'B-PER'), ('née', 'I-PER'), ('Bourgoin', 'I-PER'), …]

Figure 5: Example of the ChatGPT prompt.

Results. The results of fine-tuned adapter with JSON format prompt for NER task are presented in Table 3. To retrieve the results based on CoNLL evaluation metric (Sang and De Meulder, 2003), we perform post-processing on the generations, which involves replacing characters and adding characters to conform to the desired format. The metrics include Precision, Recall, and $F_{\beta=1}$ score for five named entity categories. Under the zero- and three-shot settings, all categories have 0.0 scores[6]. However, by fine-tuning the adapter module with the train set, the overall scores across all named entity categories are improved by 40.76%, 42.31%, and 41.52%.

The improvement due to fine-tuning is consistently evident in the second prompt, as depicted in Table 4. Notably, the performance of LLaMA 7B

[6]More details are presented in Appendix A

LLaMA 7B-Adapter	Precision	Recall	$F_{\beta=1}$
LOC	53.21	53.40	53.30
ORG	14.29	10.00	11.76
PER	33.83	40.64	36.92
PROD	0	0	0
TIME	4.55	7.55	5.67
overall	40.76	42.31	41.52

Table 3: **NER results (%) (Sang and De Meulder, 2003) on the HIPE French test set of the LLaMa 7B-Adapter fine-tuned over the training data converted to JSON-format prompts.**

on the NER task without fine-tuning is remarkably low, scoring 0.0 for both zero-shot and three-shot learning settings. Following the fine-tuning process, the model's performance demonstrates significant enhancements, with performance gains of 45.03%, 23.75%, and 31.10% observed across all five named entity categories.

LLaMA 7B-Adapter	Precision	Recall	$F_{\beta=1}$
LOC	58.25	29.60	39.25
ORG	13.55	6.06	8.37
PER	35.77	23.64	28.47
PROD	8.33	1.63	2.73
TIME	0	0	0
overall	45.03	23.75	31.10

Table 4: **NER results (%) (Sang and De Meulder, 2003) by using *nervaluate*[a] on the HIPE French test set of the LLaMa 7B-Adapter fine-tuned over the training data converted to IOB-format prompts (Lai et al., 2023).**

[a] https://github.com/MantisAI/nervaluate

Additionally, we provide a description of a few generation examples demonstrating three-shot learning using the LLaMA 7B model in Appendix A. It's worth noting that the model generates code templates, as illustrated in Figure 7, and also produces personal information-related content, as shown in Figure 6. These examples highlight the importance of data preprocessing to address privacy concerns.

4 Conclusions

In this report, we delineate the advantages of employing the LLaMA-Adapter across various downstream NLP tasks. Through the adjustment of several additional parameters within the adapter module's layers, the LLaMA-Adapter achieves superior performance compared to the original model, namely LLaMA 7B under in-context learning scenario. This outcome can be generalized for closed-domain question answering, open-domain question answering, and named entity recognition tasks under a low-resource language setting.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Maud Ehrmann, Matteo Romanello, Sven Najem-Meyer, Antoine Doucet, and Simon Clematide. 2022. Extended overview of HIPE-2022: Named Entity Recognition and Linking in Multilingual Historical Documents. In *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum*, volume 3180. CEUR-WS.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Viet Dac Lai, Nghia Trung Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. 2023. Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning. *arXiv preprint arXiv:2304.05613*.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022a. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022b. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

AI Meta. 2023. Introducing llama: A foundational, 65-billion-parameter large language model. *Meta AI. https://ai. facebook. com/blog/large-language-model-llama-meta-ai*.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. *arXiv preprint arXiv:2005.00052*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Jiexin Wang, Adam Jatowt, and Masatoshi Yoshikawa. 2022. Archivalqa: A large-scale benchmark dataset for open-domain question answering over historical news collections. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3025–3035.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 1253–1256.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. 2023. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

A Appendix

```
# -*- coding: utf-8 -*-
    # @author: Yuxuan Liu
    # @date: 2017-03-27
    # @time: 17:58
    # @version: 1.0
    # @desc:
    # @note:
    # @see: https://leetcode.com/problems/named-entity-recognition/
    # @link: https://leetcode.com/problems/named-entity-recognition/
-------------------------------------------------------------------------------------------------
# -*- coding: utf-8 -*-
    # @author: Yannick
    # @date: 2016-03-02
    # @time: 16:30
    # @version: 1.0
    # @note:
    # @see: https://www.hackerrank.com/challenges/named-entity-recognition/problem
    # @see: https://www.hackerrank.com/challenges/named-entity-recognition/editorial
```

Figure 6: Examples of generation using a JSON format prompt in the context of 3-shot learning.

```
 # Initialize variables
        self.word_list = []
        self.word_list_length = 0
        self.word_list_index = 0
        self.word_list_index_length = 0
        self.word_list_index_length_max = 0
        self.word_list_index_length_max_max = 0
        self.word_list_index_length_max_max_max = 0
        self.word_list_index_length_max_max_max_max = 0
        self.word_list_index_length_max_max_max_max_max = 0
        self.word_list_index_length_max_max_max_max_max_max = 0
        self.word_list_index_length_max_max_max_max_max_max_max = 0
        self.word_list_index_length_max_max_max
-------------------------------------------------------------------------------------------------
# Initialize variables
        self.entity_labels = ['TIME', 'LOC', 'PER', 'ORG', 'PROD']
        self.entity_labels_lowercase = self.entity_labels.lower()
        self.entity_labels_uppercase = self.entity_labels.upper()
        self.entity_labels_special = self.entity_labels_uppercase +
self.entity_labels_lowercase
        self.entity_labels_special_lowercase = self.entity_labels_special.lower()
        self.entity_labels_special_uppercase = self.entity_labels_special.upper()
        self.entity_labels_special_special = self.entity_labels_special_uppercase +
self.entity_labels_special_lowercase
        self.entity_labels_special_special_lowercase =
self.entity_labels_special_special.lower()
        self.entity_labels_special_special_uppercase =
self.entity_labels_special_special.upper()
        self.entity_labels_special_special_special =
self.entity_labels_special_special_upper
```

Figure 7: Examples of generation using a ChatGPT prompt in the context of 3-shot learning.