Chinese Typo Correction with Taiwan-LLaMa

Hung-Yi Chen

December 27, 2023

Abstract

"在再不分", refers to the misspelling or misuse of two common characters in Chinese language. This error can be annoying to readers and lead to ambiguity in electronic documents and conversations. However, the current typo correction methods in Chinese language fall short in precision. In recent years, the rise of Large Language Models (LLMs) has introduced new solutions to this problem. In this project, a fine-tuned Taiwan-LLaMa v2.0 model and a hybrid dataset comprising real and GPT-generated data were proposed, achieving an overall accuracy score of 98.6 % on the task. This project provided sufficient evidence to demonstrate the efficacy of leveraging LLM and a combined dataset to significantly improve the accuracy of typo correction, offering a promising solution applicable to future tasks.

1 Introduction

Typos are widely prevalent in various languages, and Chinese is no exception. Especially in electronic documents and text, situations where typos occur are more likely to happen. In Chinese writing, errors in the use of the words "再" (zài) and "在" (zài) are quite common, constituting a prevalent linguistic mistake. While these two characters sound similar, they have distinct meanings and usages. For instance, "再" implies repetition or something happening again, as in "再" (goodbye) or "再一次" (try again). On the other hand, "在" denotes a location or a specific time, as seen in phrases like "在家" (at home) or "在八" (at eight o'clock). Given the similarity in pronunciation, many people mistakenly choose inappropriate terms when writing, thereby affecting the accuracy of the intended meaning. Such misuse is not confined to casual conversations but can also manifest in articles, announcements, or other written texts. Therefore, to ensure clarity and precision in our sentences, it's crucial to pay special attention to the selection between "再" and "在." Enhancing our ability to distinguish between these two characters not only elevates the quality of Chinese writing but also helps avoid misunderstandings or confusion due to unclear semantics.

In this project, the primary goal is to differentiate between the commonly confused characters "再" and "在" using Large Language Models (LLM). The project aims to improve its word correction tasks by employing the LoRA technique and fine-tuning Taiwan-LLaMA V2.0, with the aspiration of surpassing the performance of leading LLMs. Furthermore, a crucial aspect of the project involves exploring whether the size and composition of the dataset have an impact on the model's overall performance.

2 Methodology

2.1 Data Preparation

The training and testing datasets consisted of two parts: real data and GPT-generated data. Additionally, data preprocessing was applied on both the training and testing datasets. The following paragraphs will present the details of data preparation in the following sequence: Real Data, GPT-Generated Data and Data Preprocessing.

2.1.1 Real Data

Three Chinese spelling check task datasets (SIGHAN Back-off 2013 2014 2015) [1][2][3] were acquired for the project. Approximately 2200 sentences containing the characters "在" and "再" were

Set/Parameters	lora_r	lora_alpha	batch size
Set A	1	16	1
Set B	16	64	2

Table 1: Two sets of training parameters.

extracted from these datasets and added into the training dataset. In addition, 200 sentences focusing on the more challenging cases such as sentences containing multiple "在" and "再" or classical Chinese sentences were manually generated or found through Internet and evenly distributed into the training and testing dataset.

2.1.2 GPT-Generated Data

To further expand the scale of the training dataset, sentences generated with GPT-4 were included. In the project, GPT-4 was prompted to generate sentences containing the characters "在" and "再". A total of 10000 sentences were generated and distributed into three different scales: 3000, 5000 and 10000 for further experiments.

2.1.3 Data Preprocessing

Data preprocessing was applied on both training and testing dataset to transform the original sentences into the format of sentence completion exercises. The trained model was tasked with determining whether to fill in the blank position with "在" or "再". Moreover, the project incorporated the few-shot learning technique, modifying the training instructions into three different sets: zero-shot, one-shot and two-shot for further observation and analysis.

2.2 Experiment Design

To explore the differences in model performance, three sets of experiments were designed. The following paragraphs will present the details of the experiment design in the following sequence: Training Parameters, Dataset Size and Dataset Composition.

2.2.1 Training Parameters

Two sets of training parameters, including lora_r, lora_alpha and batch size were compared with the same training datasets. The setting was shown in Table 1. This experiment aimed to identify the more suitable parameter set for this project.

2.2.2 Dataset Size

Three sets of different scales of the training dataset size were contrasted: 3000, 5000 and 10000. The effect of the dataset size on the trained model's performance was expected to be determined in this experiment. Note that the data used in this experiment were all GPT-generated.

2.2.3 Dataset Composition

Three sets of the training dataset compositions were compared: all-GPT-generated, all-real and hybrid. This experiment was designed for deciding the best dataset composition for the task.

2.3 Evaluation Metric

Given the output format of the task in this project in Figure 1, Rouge score and perplexity score are considered inappropriate for the performance evaluation and may lead to inaccurate results. Accuracy score of the two characters "在" or "再" was calculated for this task. Besides, the accuracy score was further separated into accuracy_GPT, accuracy_real and accuracy_overall for more precise evaluation.

```
"instruction": "你是人工智慧助理，以下將提供被挖空的文本，你要對挖空的位置填入「在」或「再」，輸出格式為「答案：再、在」。
                以下為題目:你能不能___多給我一次機會，讓我重新愛你___相互依偎。 答案：",
"output": "再、再",
"prediction": "再、在"
```


Figure 1: The output format of the task.

3 Results

All the experiment results can be seen in Table 2. Note that the top three accuracy levels are indicated with colors arranged from dark to light.

4 Discussion

4.1 Comparison of Model Parameters

As shown in Figure 2, Taiwan-LLaMa v2.0 demonstrates a comparison of training accuracy for GPT-generated training data in zero-shot, one-shot, and two-shot scenarios. The color of the columns indicates the total number of training data: green for 3,000, blue for 5,000 and red for 10,000. The parameters initially used in the model training were as Set A in Table 1. It is found that with the dataset of 3000 training data, the accuracy ranking is: zero-shot >one-shot >two-shot. In contrast, for the other two datasets, a different trend can be observed as: one-shot >zero-shot >two-shot. Upon examining the three datasets, the zero-shot performance with the dataset of 3000 training data emerges as the best, reaching an accuracy score of 0.81. This result is puzzling, as it contradicts the expectation that training with a larger dataset would enhance the performance. Therefore, the experiment underwent multiple parameter adjustments and the parameter set identified as Set B was found. With these parameters, the accuracy scores across all datasets are roughly equal across zero-shot, one-shot, and two-shot scenarios. Moreover, when comparing the best results from each dataset, a clear trend is observed: 10,000 >5,000 >3,000. The performance with 10,000 training data in zero-shot scenario stands out with the best accuracy score of 0.89. Section 4-1 reveals that using parameters from Set B, which achieves the best accuracy score of 0.89, offers a superior performance compared to Set A, which achieves the best accuracy score of 0.81. Consequently, the parameters in Set B were used in other training experiments.

4.2 Comparison of Dataset Size

As shown in Figure 3, the accuracy scores of models trained on GPT-generated data in zero-shot, one-shot, and two-shot scenarios are represented by the following colors: green for the untrained Taiwan-LLaMa v2.0 model; blue, red, and black for the Taiwan-LLaMa v2.0 model trained on 3,000, 5,000, and 10,000 instances of GPT-generated data, respectively; and purple and yellow for GPT 3.5 and GPT 4. In the green section of Figure 3, it is observed that the untrained Taiwan-LLaMa v2.0 model achieves an accuracy score of less than 0.55 in all scenarios. In the blue, red, and black sections, it is found that after training the Taiwan-LLaMa v2.0 model on datasets of 3,000, 5,000, and 10,000 instances, the accuracy scores consistently exceed 0.82. Comparing the highest accuracy scores of each dataset size, the order is 10,000 >5,000 >3,000, with the highest accuracy score of 0.89 achieved on the model trained with 10,000 instances of GPT-generated data. Further comparing the purple and yellow sections representing GPT 3.5 and GPT 4 with the trained Taiwan-LLaMa v2.0 models, the best accuracy scores for GPT 3.5 and GPT 4 are 0.82 and 0.79, respectively. It is evident that both are lower than that of the trained Taiwan-LLaMa v2.0 model. Section 4-2 concludes that training with larger dataset leads to better performance, indicating a positive correlation between dataset size and accuracy score. Moreover, fine-tuning the Taiwan-LLaMa v2.0 model with just 3,000 instances of GPT-generated data is sufficient to surpass the accuracy score of both GPT 3.5 and GPT 4.

Table 2, The accuracy score of all the experiment training sets.

Experiment 1. LoRa paramteres (p1 / p2)				Experiment 2. real data + GPT-generated data				Reference Data			
	accu_gpt	accu_real	accu_ overall		accu_gpt	accu_real	accu_ overall		accu_gpt	accu_real	accu_ overall
3000, p1, zero-shot	0.92857	0.69725	0.81291	2300 + 0, zero-shot	0.99107	0.89908	0.94508	No-train, zero-shot	0.67857	0.38532	0.53195
3000, p2, zero-shot	0.99107	0.69725	0.84416	2300+ 0, one-shot	0.98214	0.90826	0.94520	No-train, one-shot	0.57143	0.40367	0.48755
3000, p1, one_shot	0.91071	0.55963	0.73517	2300 + 0, two-shot	0.98214	0.97248	0.97731	No-train, two-shot	0.38393	0.27523	0.32958
3000, p2, one-shot	0.97321	0.70642	0.83982	2300 + 3000, zero-shot	0.98214	0.90826	0.94520	GPT3.5, zero-shot	0.93750	0.70642	0.82196
3000, p1, two-shot	0.66964	0.48624	0.57794	2300 + 3000, one-shot	0.98214	0.88991	0.93603	GPT3.5, one-shot	0.94643	0.69725	0.82184
3000, p2 two-shot	0.98214	0.74312	0.86263	2300 + 3000, two-shot	0.99107	0.93578	0.96343	GPT3.5, two-shot	0.94643	0.69725	0.82184
5000, p1, zero-shot	0.84821	0.62385	0.73603	2300 + 5000, zero-shot	0.99107	0.98165	0.98636	GPT4, zero-shot	0.93750	0.64220	0.78985
5000, p2, zero-shot	1.00000	0.68807	0.84404	2300 + 5000, one-shot	0.99107	0.91743	0.95425	GPT4, one-shot	0.82143	0.62385	0.72264
5000, p1, one-shot	0.92857	0.85999	0.85049	2300 + 5000, two-shot	1.00000	0.96330	0.98165	GPT4, two-shot	0.85714	0.67890	0.76802
5000, p2, one-shot	1.00000	0.71560	0.85780	2300 + 10000, zreo-shot	1.00000	0.93578	0.96789				
5000, p1, two-shot	0.67857	0.50459	0.59158	2300 + 10000, one-shot	0.98214	0.91743	0.94979				
5000, p2, two-shot	0.99107	0.70642	0.84875	2300 + 10000, two-shot	0.98214	0.91743	0.94979				
10000, p1, zero-shot	0.64286	0.54128	0.59207								
10000, p2, zero-shot	0.98214	0.79817	0.89015								
10000, p1, one-shot	0.91071	0.56881	0.73976								
10000, p2, one-shot	0.99107	0.77064	0.88086								
10000, p1, two-shot	0.69643	0.51376	0.60510								
10000, p2, two-shot	0.99107	0.77064	0.88086								

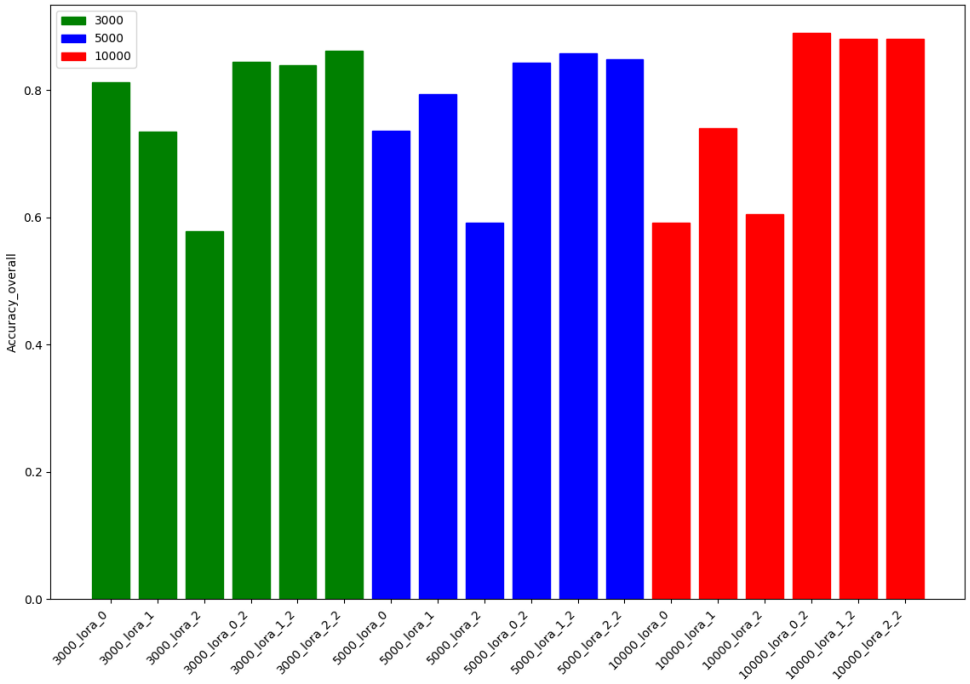

Figure 2: Comparison of Training Accuracy in 0, 1, and 2-Shot Settings for Taiwan-LLaMa v2.0 Using GPT-Generated Data.

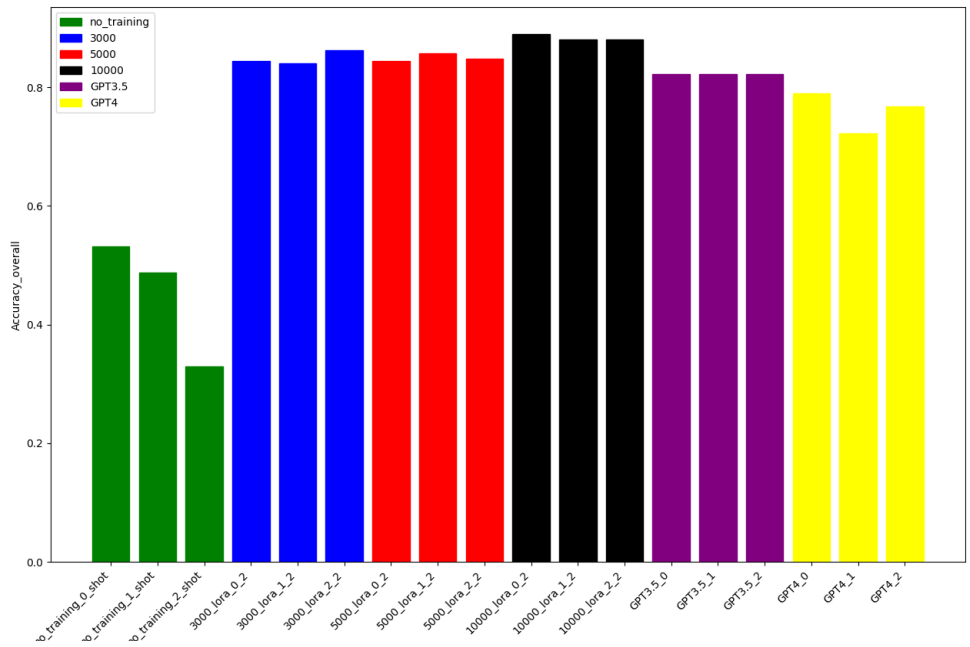

Figure 3: Comparison of Accuracy for Different Models in 0, 1, and 2-Shot Settings Using GPT-Generated Data.

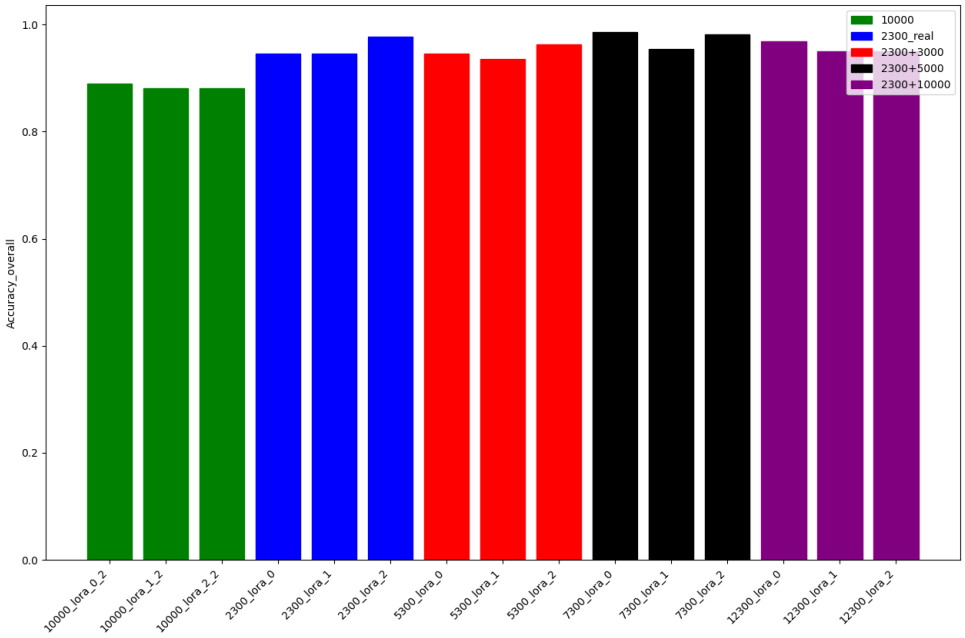

Figure 4: Comparison of Accuracy in 0, 1, and 2-Shot Settings Using Different Generated Data for Taiwan-LLaMa v2.0.

4.3 Comparison of dataset composition

As shown in Figure 4, the comparison of accuracy score of Taiwan-LLaMa v2.0 trained on different generated datasets in zero-shot, one-shot, and two-shot scenarios is represented by the following colors: green for training solely with 10,000 GPT-generated data; blue for training exclusively with 2,300 real data; and red, black, and purple for training with a combination of 2,300 real data plus 3,000, 5,000, and 10,000 GPT-generated data, respectively. In this experiment, the primary focus is on how three types of dataset compositions affect the model. In the green section, which is trained with only GPT-generated data, achieves an accuracy score of 0.89. In the blue, red, black, and purple sections, where real data are used for training, the accuracy scores are consistently higher than the model trained solely with GPT-generated data. Finally, among all sections utilizing real data, the black section, which combines 5,000 GPT-generated data with 2,300 real data, performs the best, achieving an accuracy score of 0.98. In Section 4-3, it is clear that training with a small amount of real data (2,300 instances) results in higher accuracy score compared to those training with a larger purely GPT-generated datasets. Furthermore, training with a hybrid dataset proves to be more effective than training solely with either GPT-generated or real data. In this project, a combination of 5,000 GPT-generated data with 2,300 real data, approximately a 2:1 ratio, achieves an overall best accuracy score as high as 0.98.

5 Conlusion

In this project, a fine-tuned Taiwan-LLaMa V2.0 model is presented, achieving an overall accuracy score of 0.986 distinguishing the two common misspelled characters in Chinese, “在” and “再. Moreover, two key observations emerge in the training experiments. First, the performance of the model trained with a real dataset of 2300 instances surpasses that of models trained solely with larger GPT-generated dataset, highlighting the significant impact of the involvement of real data in the training progress. Second, the performance of the model trained with a hybrid dataset can further enhance the accuracy score, indicating that the involvement of GPT-generated data can generalize the capability of the trained model in the word correction task.

References

[1] Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. Introduction to SIGHAN 2015 bake-off for Chinese spelling check. In Liang-Chih Yu, Zhifang Sui, Yue Zhang, and Vincent Ng, editors, *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37, Beijing, China, July 2015. Association for Computational Linguistics.

[2] Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. Chinese spelling check evaluation at SIGHAN bake-off 2013. In Liang-Chih Yu, Yuen-Hsien Tseng, Jingbo Zhu, and Fuji Ren, editors, *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.

[3] Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. Overview of SIGHAN 2014 bake-off for Chinese spelling check. In Le Sun, Chengqing Zong, Min Zhang, and Gina-Anne Levow, editors, *Proceedings of the Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 126–132, Wuhan, China, October 2014. Association for Computational Linguistics.

Appendix

Table 3, The Rouge score and accuracy score of all training sets												
	r-1 r	r-1 p	r-1 f	r-2 r	r-2 p	r-2 f	r-l r	r-l p	r-l f	accu_gpt	accu_real	accu_overall
10000, zero-shot	0.86833	0.91499	0.87517	0.55750	0.54417	0.54819	0.74499	0.79167	0.75183	0.64286	0.54128	0.59207
10000, zero-shot 2	0.97500	0.93999	0.95299	0.85125	0.83958	0.84364	0.97167	0.93667	0.94967	0.98214	0.79817	0.89015
10000, one-shot	0.90917	0.91249	0.90217	0.71875	0.70500	0.70999	0.85083	0.85417	0.84383	0.91071	0.56881	0.73976
10000, one-shot 2	0.97000	0.93667	0.94899	0.84333	0.82917	0.83448	0.96000	0.92667	0.93899	0.99107	0.77064	0.88086
10000, two-shot	0.87833	0.90333	0.87867	0.64458	0.63167	0.63590	0.79833	0.82333	0.79867	0.69643	0.51376	0.60510
10000, two-shot 2	0.97833	0.94499	0.95799	0.84375	0.83417	0.83752	0.96167	0.92833	0.94133	0.99107	0.77064	0.88086
12300, zreo-shot	0.97333	0.97167	0.96899	0.85958	0.85875	0.85857	0.97000	0.96833	0.96567	1.00000	0.93578	0.96789
12300, one-shot	0.97417	0.96749	0.96649	0.84250	0.83625	0.83836	0.97083	0.96417	0.96317	0.98214	0.91743	0.94979
12300, two-shot	0.98333	0.97499	0.97599	0.86125	0.85875	0.85931	0.97999	0.97167	0.97267	0.98214	0.91743	0.94979
2300, zero-shot	0.84333	0.94499	0.87967	0.68792	0.70083	0.69262	0.83999	0.94167	0.87633	0.99107	0.89908	0.94508
2300, one-shot	0.84167	0.95499	0.88267	0.67417	0.69500	0.68207	0.83499	0.94833	0.87599	0.98214	0.90826	0.94520
2300, two-shot	0.82083	0.97249	0.87999	0.64583	0.66875	0.65395	0.82083	0.97249	0.87999	0.98214	0.97248	0.9773[illegible]
3000, zero-shot	0.89917	0.92917	0.90317	0.69542	0.68750	0.68949	0.85083	088083	0.85483	0.92857	0.69725	0.81291
3000, zero-shot 2	0.93167	0.95167	0.93367	0.80542	0.80125	0.80248	0.92499	0.94499	0.92699	0.99107	0.69725	0.84416
3000, one_shot	0.93250	0.91749	0.91483	0.71208	0.70200	0.70502	0.86583	0.85083	0.84817	0.91071	0.55963	0.73517
3000, one-shot 2	0.92667	0.95417	0.93283	0.81000	0.80583	0.80724	0.92000	0.94749	0.92617	0.97321	0.70642	0.83982
3000, two-shot	0.90750	0.91249	0.89917	0.61833	0.60042	0.60698	0.79417	0.79917	0.78583	0.66964	0.48624	0.57794
3000, two-shot 2	0.93749	0.96583	0.94417	0.81708	0.81708	0.81631	0.93417	0.96249	0.94083	0.98214	0.74312	0.86263
5000, zero-shot	0.87333	0.92999	0.88467	0.67458	0.66833	0.66967	0.83167	0.88833	0.84299	0.84821	0.62385	0.73603

5000, zero-shot 2	0.94249	0.95749	0.94217	0.79s92	0.78875	0.78898	0.92249	0.93749	0.92217	1.00000	0.68807	0.84404
5000, one-shot	0.90999	0.90667	0.89717	0.72625	0.72042	0.72129	0.86333	0.85999	0.85049	0.92857	0.85999	0.85049
5000, one-shot 2	0.93833	0.95083	0.93683	0.80958	0.80167	0.80402	0.92833	0.94083	0.92683	1.00000	0.71560	0.85780
5000, two-shot	0.90749	0.91167	0.89783	0.64750	0.63667	0.64019	0.82083	0.82499	0.81117	0.67857	0.50459	0.59158
5000, two-shot 2	0.93833	0.95833	0.93867	0.80292	0.79917	0.79962	0.92833	0.94833	0.92867	0.99107	0.70642	0.84875
5300, zero-shot	0.94417	0.98167	0.95399	0.81250	0.81500	0.81267	0.94417	0.98167	0.95399	0.98214	0.90826	0.94520
5300, one-shot	0.95167	0.97667	0.95999	0.82958	0.82792	0.82774	0.94499	0.97000	0.95333	0.98214	0.88991	0.93603
5300, two-shot	0.96999	0.97999	0.97099	0.85750	0.85250	0.85407	0.96999	0.97999	0.97099	0.99107	0.93578	0.96343
7300, zero-shot	0.96083	0.97417	0.96349	0.85208	0.84958	0,84940	0.96083	0.97417	0.96349	0.99107	0.98165	0.98636
7300, one-shot	0.96833	0.97333	0.96699	0.85792	0.85167	0.85369	0.96833	0.97333	0.96699	0.99107	0.91743	0.95425
7300, two-shot	0.95999	0.97499	0.96399	0.85500	0.85249	0.85290	0.95999	0.97499	0.96399	1.00000	0.96330	0.98165
GPT3.5, zero-shot	0.98333	0.92042	0.93662	0.76250	0.76875	0.76348	0.93500	0.87208	0.88829	0.93750	0.70642	0.82196
GPT3.5, one-shot	0.97833	0.90417	0.92355	0.78958	0.77833	0.78035	0.95333	0.87917	0.89855	0.94643	0.69725	0.82184
GPT3.5, two-shot	0.97249	0.91708	0.92664	0.77333	0.76571	0.76684	0.93917	0.88631	0.89539	0.94643	0.69725	0.82184
GPT4, zero-shot	0.95000	0.94208	0.93795	0.77958	0.77500	0.77627	0.92167	0.91375	0.90962	0.93750	0.64220	0.78985
GPT4, one-shot	0.86499	0.75967	0.79351	0.71042	0.59400	0.63810	0.85499	0.75133	0.78446	0.82143	0.62385	0.72264
GPT4, two-shot	0.89833	0.84500	0.85826	0.75958	0.68525	0.71402	0.89167	0.83917	0.85207	0.85714	0.67890	0.76802
No-train, zero-shot	0.36917	0.80173	0.46913	0.02833	0.03000	0.02869	0.35916	0.79173	0.45913	0.67857	0.38532	0.53195
No-train, one-shot	0.67833	0.88367	0.71420	0.27708	0.29955	0.28431	0.58999	0.79524	0.62587	0.57143	0.40367	0.48755
No-train, two-shot	0.50417	0.88999	0.59650	0.08333	0.09500	0.08767	0.42083	0.80667	0.51317	0.38393	0.27523	0.32958