NTU IR Final Project Report

Team_10

r11922A16 柳宇澤
r11922012 曾訪晴

2023 Fall

1 Competition Results

- Team Name: AndersonLiu
- Rank in test set: 30

2 Introduction

In this project, there are some case reports, D, from real-world medical data. Since these reports contain some personal or confidential information (Protected Health Information, PHI), this project will perform a De-identification process, $Proc_{De}$, to prevent PHI leakage. Secondly, some PHIs could be time information. These PHIs could be written in different patterns according to writers' styles. Hence, this project also needs to perform Time Normalization process, $Proc_{TN}$. Generally, given a case report document $d \in D$, we'll process 2 tasks, De-identification & Time Normalization, on d, which consist of multiple sentences s_i.

2.1 De-identification process $Proc_{De}$

Given a case report document d, after processing $Proc_{De}$, we hope the model will output answer marks M^d_{De}. If a sentence s_i in d contains n_i PHI items to be de-identified, there should also be n_i answer marks m^{s_i} correspondingly. Each $m^{s_i} \in M^d_{De}$ should contain the PHI type and the PHI content.

$$\begin{aligned} M^d_{De} &= \{\ Proc_{De}(s_i) \mid s_i \in d\ ,\ 1 < i < |d|\ \} \\ &= \{\ \{m^{s_i}_j \mid 1 < j < n_i\ \} \mid s_i \in d\ ,\ 1 < i < |d|\ \} \\ &= \{\ m^{s_i}_j \mid s_i \in d\ ,\ 1 < i < |d|\ , 1 < j < n_i\ \} \end{aligned} \tag{1}$$

where $|d|$ means the sentence number of d.

2.2 Time Normalization $Proc_{TN}$

Same as $Proc_{De}$, given a case report document d, after processing $Proc_{TN}$, we hope the model will output answer marks M^d_{TN}. The only difference in Time Normalization process is each $m^{s_i} \in M^d_{TN}$ should contain not only the PHI type and the PHI

OS	Ubuntu 22.04.3 LTS
GPU	NVIDIA RTX A6000 49140MB * 1 NVIDIA RTX A5000 24564MB * 5

Table 1: Equipments used for this project.

content but also normalized PHI content.

$$\begin{aligned} M^d_{TN} &= \{\ Proc_{TN}(s_i) \mid s_i \in d\ ,\ 1 < i < |d|\ \} \\ &= \{\ \{m^{s_i}_j \mid 1 < j < n_i\ \} \mid s_i \in d\ ,\ 1 < i < |d|\ \} \\ &= \{\ m^{s_i}_j \mid s_i \in d\ ,\ 1 < i < |d|\ , 1 < j < n_i\ \} \end{aligned} \tag{2}$$

where $|d|$ means the sentence number of d.

2.3 Multitask View *Proc*

Since we apply multitask training, eq. 1 and eq. 2 could be accomplished by single *Proc*, which is implemented with end-to-end model. Given a case report document d, after processing *Proc*, we hope the model will output answer marks M^d, which contain both De-identification marks & Time Normalization marks.

$$\begin{aligned} M^d &= \{\ Proc(s_i) \mid s_i \in d\ ,\ 1 < i < |d|\ \} \\ &= \{\ m^{s_i}_j \mid s_i \in d\ ,\ 1 < i < |d|\ , 1 < j < n_i\ \} \end{aligned} \tag{3}$$

where $|d|$ means the sentence number of d.

3 Method

To implement eq.1 and eq. 2, we apply a Language Model, Pythia[1], with multitask training in this project. What's more, we also ensemble multiple models' outputs in the final step.

3.1 Equipments

As shown in table 1. Since we trained different model sizes, we used different GPUs depending on those sizes (we'll discuss how we distributed them in sec. 3.4).

3.2 Dataset

As shown in table 2. This dataset D is Health Science Alliance Biobank from Lowy Cancer Research Centre, University of New South Wales(UNSW). There are about 3,000 case reports and medical information in this dataset. Lowy Cancer Research Centre splits D into 3 datasets, D_{train}, D_{valid}, and D_{test}. We also display the 5 most common PHI types in D_{valid} in table 3. This information helps us to efficiently distribute our effort in terms of PHI type in the early contest stage.

	# doc	# sentence	# PHI
train	1,734	117,343	26,545
valid	560	40,501	8,857
test	950	69,465	-

Table 2: Basic information about 3 datasets.

PHI type	# PHI	Proportion
DOCTOR	2,191	24.74%
DATE	1,616	18.25%
IDNUM	1,177	13.29%
HOSPITAL	593	6.7%

Table 3: 5 most common PHI types in D_{valid}

3.3 Data Pre-processing

3.3.1 Split d into s_i

We'll split each $d \in D$ into $|d|$ sentences $\{s_1, s_2, ..., s_{|d|}\}$, where the $|d|$ is depends on how many delimiters exist in d. In the beginning, we treated *'.'* and *'\n'* as the delimiters. However, we noticed there are some DATEs are written in a format like *DD.MM.YYYY*, which causes s_i could be too short and meaningless. Therefore, we replace *'.'* with *'. '*, which is added with 2 spaces.

3.3.2 Generate training pair p_i from s_i

After generating s_i for each d, we'll map each s_i with its answer marks $\{ m_j^{s_i} \}$. Recap mentioned in sec. 2.3, $\{ m_j^{s_i} \}$ could contain both De-identification marks & Time Normalization marks for multitask training. We expect our LM could learn how to adaptively generate two types of marks via these training pairs $p_i = (s_i, \{ m_j^{s_i} \}) \in P$. We'll treat the left part of each p_i as input, while the right part as label during the training process.

3.3.3 *(Optional)* Add prefix sentences into p_i

Note that in the left part of p_i, we also try to add prior sentences of s_i to the left part as input. We call this trying Adding Prefixes. Take adding 1 prefix in the left part of p_i as an example, the processed training pair could be $Prefix(p_i, 1) = (\{ s_{i-1}, s_i \}, \{ m_j^{s_i} \})$. When we add 0 prefix sentence, it means $Prefix(p_i, 0) = p_i$. In this project, we add 1 prefix at most due to equipment limitations.

3.4 Model

We select EleutherAI's Pythia Large Language Model [1] as our main model to solve the problems in this project. Pythia models are pre-trained with Pile dataset [2], which is a 825GiB general-purpose dataset in English. Pythia supplies a collection

of models developed to facilitate interpretability research in 8 sizes. As mentioned in sec. 3.1, we apply different model sizes in this project. There are 4 model sizes we used, **70m**, **160m**, **410m**, and **1.4b**. These sizes represent the parameter number of each model.

3.4.1 Main task

Models should perform what *Proc* (eq. 3) does. Given training pair p_i mentioned in sec. 3.3.2, model should determine left part of p_i contains what PHI and point it out by generating $m_j^{s_i}$. We treat this process as a sequence-to-sequence task. Models could calculate the loss according to the right part of p_i.

3.4.2 Hyperparameters

We finetuned models from step 3000. As referred to in the manual document, Pythia was pretrained with 143000 steps and the learning rate was 10^{-4}. Accordingly, we approximately set the learning rate to 10^{-5} to simulate the scheduler effect that occurred before 3000 steps. We didn't apply a scheduler since the experiments with schedulers didn't perform better.

3.4.3 Training

The training loss is the usual sequence-to-sequence cross-entropy loss. As for the training epoch number, we initially set it to 200. However, according to the validation loss, we found that performance converged early in less than 5 epochs. Please refer to the fig. 1.

We also noticed that the length of inputs significantly affects GPU memory usage when training LLM. For example, our equipment could afford at most $Prefix(p_i,\ 1)$. We can't even add more than 1 sentence for more detailed information.

3.5 Post-processing

3.5.1 Keyword mapping

Our models all performed poorly on predicting PHI types, COUNTRY, and ORGANIZATION. If you need some scores, please refer to tab. 4 and tab. 5 and focus on COUNTRY and ORGANIZATION rows. This phenomenon could be caused by their training data amount, where there are only 3 COUNTRY and 136 ORGANIZATION in training data. Hence, we apply Keyword mapping to alleviate this problem. We found Pycountry.countries[1] for COUNTRY mapping and an organization list from U.S. Securities and Exchange Commission (SEC.gov)[2] for ORGANIZATION mapping. Unfortunately, we only preserved COUNTRY mapping due to the poor performance on ORGANIZATION mapping.

[1]Pycountry codes: https://github.com/pycountry/pycountry

[2]Organization list from SEC.gov: https://www.sec.gov/files/rules/other/4-460list.htm

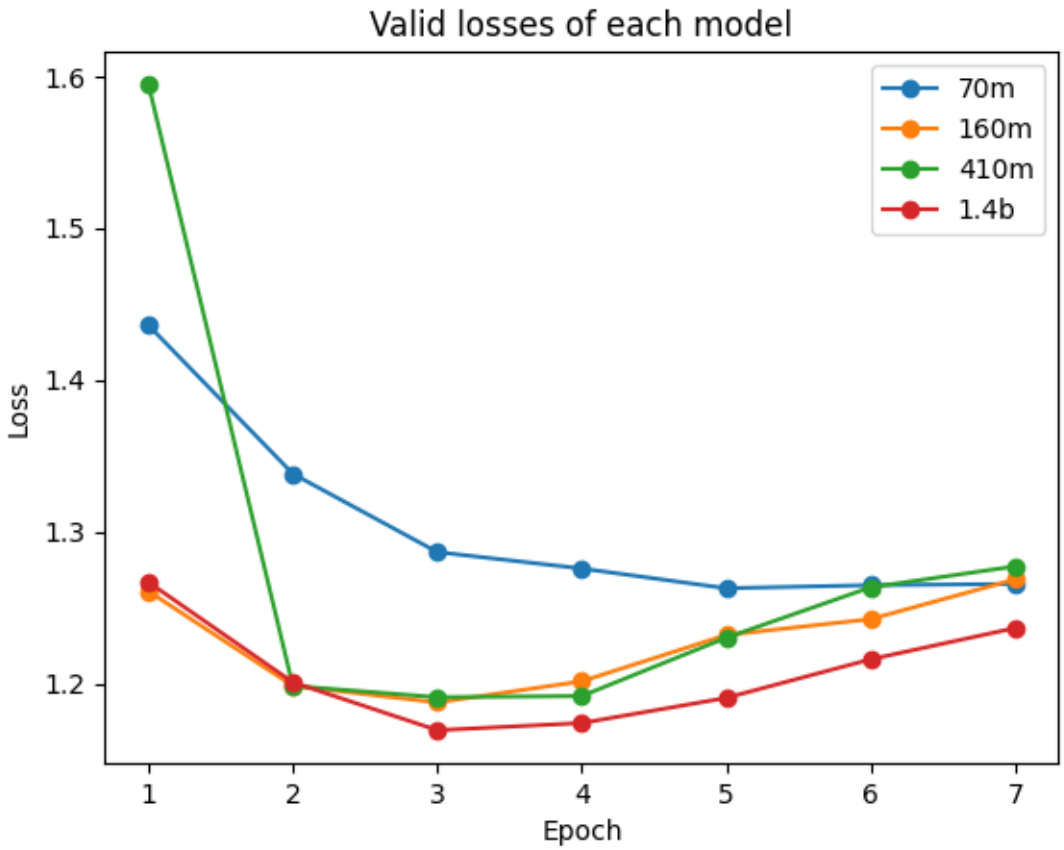

Figure 1: Valid loss trends of each model.

3.5.2 Model ensemble

In sec. 4.2.2, we mentioned that we couldn't observe an almighty model, which performed wonderfully on predicting every PHI type, in our experiments. Usually, models performed well in predicting specific PHI types (refer to tab. 4 and tab. 5). Fortunately, we noticed that even though we trained models on the same setting, they still performed well in several but different PHI-type predictions (refer to model 160m_retry3 and 160m_retry9 in tab. 4). Hence, we implement Model Ensemble by aggregating the best model's prediction for each PHI type. For example, if model A predicted IDNUM well while model B predicted DOCTOR well, we aggregate model A's IDNUM predictions and model B's DOCTOR predictions. We record how we evaluate and select models in a Google sheet[3].

4 Experiments & Discussion

4.1 Experiments

We first tried 410m Pythia model and 1.4b Pythia model. Both models are quite huge, especially for 1.4b. Unexpectedly, 1.4b didn't beat 410m model and was even a little worse than it. 410m earn **0.8309/0.6396** while 1.4b earn **0.8025/0.6249** on De-identification and Time Normalization task scored by average-F1-Macro. Another crucial point that makes us give up 1.4b is the cost of time and GPU memory, which leads to low efficiency and amount of experiments.

We then considered reducing the task complexity. We first appoint 2 different models with same input sentences (left part of p_i) but different PHI labels (right part of p_i). One model only receives De-identification labels while another model only receives Time Normalization labels. This attempt surprisingly worked, this setting *DeNorm* earned **0.8394/0.6661**. Taking advantage of victory, we then tried to treat each PHI type as a single task for a single model. However, this setting *SplitAll* only earned **0.7150/0.6416**.

After that, we tried some techniques like adding learning rate schedulers, adding prefixes, long sentences for inferences (sec. 4.2.1), and data augmentation. Nevertheless, they didn't make any huge progress, or with any consistent trends. We then noticed that some models in the same setting perform well in specific PHI-type predictions. Hence, inheriting the previous concept of task complexity reduction, we only preserved the best predictions for each PHI type predicting task and aggregated them. The details are described in sec. 3.5.2. Though we didn't have enough time to upload our final result to the leaderboard of the validation phase, we finally got**0.7228/0.4924** on the leaderboard of the testing phase, ranked 30.

[3]Model selection for ensemble: https://docs.google.com/spreadsheets/d/1tddZNOPtSl4XWwsowrzAaIiyUHIJ0MBUie6D6jRX58

4.2 Discussion

4.2.1 Short sentence for training, Long sentence for inference

We accidentally infer with $Prefix(p_i,\ 5)$ by using the model trained with $Prefix(p_i, 1)$. The results. We found that the models perform well in predicting PATIENT and STREET. Though we couldn't train with long sentences like $Prefix(p_i,\ 5)$ due to the equipment limitation, we could still provide long sentences in the inference stage to refer to more information. This method could be risky, but it's a worthy way to alleviate equipment limitations.

4.2.2 Does the almighty model exist?

In our experiments, we observed models could perform well in specific PHI types, but not generally well in every PHI type. Maybe the whole task could be too difficult for a single model to complete. Even though it performed poorly in general task, it still learned something to solve the subtasks perfectly.

4.2.3 Influence of model size

Does a huge model perform better than a small model if the task is too complicated to handle and forces the model to focus on learning subtasks? Move our attention to the 160m model in tab. 4, we noticed that it beats 410m and even 1.4b models in many PHI types. Hence, bigger is not always better at this time. We need to balance the training cost and probability of training a good subtask model. If we keep training a huge model, we may lose many chances of training to meet a good subtask model.

	70m	**160m_retry3**	**160m_retry9**	**410m**	**1.4b**
IDNUM	0.9654	**0.9761**	0.9669	0.9655	0.9728
MEDICALRECORD	0.9884	**0.9938**	0.9885	0.9734	0.9814
PATIENT	0.8768	**0.8935**	0.844	0.8472	0.8254
STREET	0.8079	**0.8608**	0.6797	0.7539	0.638
CITY	0.9488	**0.9494**	0.9435	0.8935	**0.9494**
STATE	0.9821	**0.9871**	**0.9871**	0.9754	0.9855
ZIP	0.8642	0.8659	0.7612	0.8536	**0.8831**
DEPARTMENT	0.928	0.925	0.9257	**0.9407**	0.9275
HOSPITAL	0.9219	0.9586	**0.9602**	0.9325	0.9387
DOCTOR	0.9114	0.9347	**0.9382**	0.9091	0.9154
ORGANIZATION	0.5833	0.4687	**0.68**	0.6275	0.4
AGE	0.8462	0.9057	0.9346	0.9346	**0.9541**
PHONE	0	0	0.6667	**1**	**1**
COUNTRY	0	0	0	0	0
LOCATION-OTHER	0.4	0.5714	0.4	**0.8571**	0.6667
DATE	0.774	0.7755	**0.7801**	0.7754	0.7756
TIME	0.8779	0.8249	**0.9018**	0.8768	0.8352
DURATION	0.6667	**0.7273**	0.6667	0.2667	0.6667

Table 4: F1-Macro scores of each PHI type predicted by different model sizes.

	DeNorm_410m	SplitAll_410m	Prefix1_410m	Prefix1_410m_InputPrefix5
IDNUM	0.9736	**0.9815**	0.9678	0.9577
MEDICALRECORD	**0.9841**	0.9449	0.8589	0.7914
PATIENT	0.8697	0.8927	0.8685	**0.9073**
STREET	0.8336	0.793	0.8989	**0.9399**
CITY	0.9489	**0.955**	0.9343	0.9245
STATE	**0.9805**	0.979	0.979	0.9773
ZIP	0.8344	0.8382	**0.8802**	0.8764
DEPARTMENT	**0.9358**	0.9333	0.9229	0.9201
HOSPITAL	**0.9605**	0.9536	0.9502	0.9201
DOCTOR	0.9035	**0.9168**	0.8798	0.8174
ORGANIZATION	**0.566**	0.4091	0.5	0.549
AGE	**0.9444**	0.9057	0.9381	0.8485
PHONE	**1**	0	**1**	**1**
COUNTRY	0	0	0	0
LOCATION-OTHER	**0.8571**	0.2222	**0.8571**	0
DATE	0.7743	**0.7956**	0.7854	0.6778
TIME	0.8606	**0.8794**	0.8372	0.6953
DURATION	**0.3636**	0.25	0	0

Table 5: F1-Macro scores of each PHI type predicted by different model manipulations, including task complexity reduction, adding prefixes, and long sentences for inference mentioned in sec. 4.2.1.

5 Conclusion

In this project, we proposed an end-to-end multitask LLM model to solve De-identification and Time Normalization tasks. Though we couldn't train an almighty model, we still alleviated this problem by aggregating several subtask models. This is kind of a way to overcome the equipment limitation that restricts us from training huge models. The biggest challenge is the low data amount of certain PHI types, such as COUNTRY and ORGANIZATION. We think that if we could have more time to survey more related geographic data, the models would perform much better and more robustly. A limitation to notice is that during the inference, our method could be too time-consuming since we need to collect several models' predictions and aggregate them. If we continue this project and are eager to improve this method, this could be a crucial point to address.

6 Contribution

- R11922A16 柳宇澤 Experiments, Model ensemble, Report writing
- R11922012 曾訪晴 Data preprocess, Experiments, Report writing

References

[1] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

[2] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020.