III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

Tuan Dung Le is currently PhD student in Computer Science at University of South Florida. He received his master's degree in Information Systems from the Hanoi University of Science and Technology in Vietnam. His research focuses on applying natural language processing techniques to extract information from unstructured clinical and medical texts, especially in low-resource settings.

Thanh Duong is currently pursuing a PhD in Computer Science at the University of South Florida, USA. His research focuses on the application of language models to medication-related tasks, particularly in processing electronic health records (EHRs). He has also contributed to projects at the Moffitt Cancer Center, where he worked on developing a system for cancer phenotype extraction.

2. What motivated you to compete in this challenge?

Our research focuses on extracting valuable information from clinical notes, closely aligning with the objectives of this competition. We view participating in this competition as an excellent opportunity to enhance our skills and contribute to our ongoing research on automating the abstraction of information from clinical notes.

3. High level summary of your approach: what did you do and why?

We proposed using prompt-based finetuning to tackle the problem. We created prompt templates based on variable descriptions. For example: *Depress Mood [MASK] Current Mental Illness Treatment [MASK] … [MASK]. NarrativeLE +NarrativeCME*.
The model is trained to predict "yes" or "no" tokens for the [MASK] positions by optimizing the masked language modeling (MLM) objective. The difference between the logits for "yes" and "no" at each [MASK] position is used as the score. A threshold of 0 is applied to classify predictions as "yes" or "no".
For variables such as InjuryLocation and WeaponType1, the model calculates scores for all possible categories (6 locations and 12 weapon types). The category with the highest score (argmax) is predicted as "yes" while all others are predicted as "no".

We create a 5-fold cross validation (CV) using MultilabelStratifiedKFold strategy.
For best private score submission, we used two pretrained models and two different prompt templates.

- Deberta-v3-large + template 1 (Weight 0.63)
- Longformer-large (continued MLM finetuning on training corpus) + template 2 (Weight 0.37)

4. Do you have any useful charts, graphs, or visualizations from the process?

Deberta-v3-large	5 epochs	5 epochs	5 epochs	6 epochs
Fold	LR 2e-5	LR 7e-6	LR 7e-6 add EMA	LR 7e-6 add EMA, R-Drop 0.1
0	84.83	84.62	84.75	85.13
1	85.28	85.27	85.75	85.57
2	84.03	84.08	84.26	83.98
3	84.93	85	85.18	85.55
4	85.28	85.78	85.89	86.16
Average CV	**84.87**	**84.95**	**85.166**	**85.278**
Public score	86.08	85.52	85.14	85.6
Private score	86.05	85.71	85.52	85.53

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

Collate function for faster training/inference: only pad input to max token length of sentences in a batch instead of 1664 tokens

```
def collate(inputs, prefix_token_len):
    # For longformer only
    # mask_len = int(inputs["attention_mask"].sum(axis=1).max()) - prefix_token_len.min()

    # For deberta and others
    mask_len = int(inputs["attention_mask"].sum(axis=1).max())

    for k, v in inputs.items():
        inputs[k] = inputs[k][:, :mask_len]
    return inputs
```

Set global attention mask for Longformer model to all prefix tokens (prompt tokens)

```
        ### Global attention of Longformer model ###
        global_attention_mask = inputs["attention_mask"]
        last_mask_ids = 0
        for i  in range(len(inputs["input_ids"])):
            if inputs["input_ids"][i] == self.mask_token_id:
                last_mask_ids = i
        prefix_token_len = last_mask_ids + 1
        for i in range(prefix_token_len):
            global_attention_mask[i] = 2
        inputs["attention_mask"] = global_attention_mask
```

R-drop loss help improve local CV but does not help public/private score

```
    def compute_kl_loss(p, q, pad_mask=None):
        p_loss = F.kl_div(F.log_softmax(p, dim=-1), F.softmax(q, dim=-1), reduction="none")
        q_loss = F.kl_div(F.log_softmax(q, dim=-1), F.softmax(p, dim=-1), reduction="none")

        if pad_mask is not None:
            p_loss.masked_fill_(pad_mask, 0.)
```

```
        q_loss.masked_fill_(pad_mask, 0.)

    p_loss = p_loss.sum()
    q_loss = q_loss.sum()

    loss = (p_loss + q_loss) / 2
    return loss
```

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): Dual Intel® Xeon® Platinum 8480C Processors
   - GPU (model or N/A): 1 NVIDIA H100 80GB
   - Memory (GB): 80GB
   - OS: Linux
   - Train duration:
     - ~30 minutes (5 epochs) for each fold deberta-v3-large
     - ~1 hour (5 epochs) for each fold longformer-large
   - Inference duration: 45 minutes
     - ~20 minutes for 5 folds deberta-v3-large model
     - ~25 minutes for 5 folds longformer-large model

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

In my experiments, the model occasionally diverges during training. When this happens, I simply restart the training with the seed incremented by 1. Incorporating an exponential moving average (EMA) and R-Drop into the training pipeline significantly improves the stability of the training process.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

No

9. How did you evaluate performance of the model other than the provided metric, if at all?
   In addition to the overall metric, I also looked at the binary f1 score for each binary variable and micro f1 for each categorical variable.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

For this competition, I did a lot of experiments to improve my local CV score.
My best CV score 0.8606 (private 0.8608) came from an ensemble of eight diversified models and optimizing thresholds for specific variables.
The ensemble included five variants of DeBERTa-v3-large, two variants of Longformer-large, and one Flan-T5 XL model. The variants use different prompt templates, with and without further MLM pretrained on training corpus, and adding training techniques such as EMA, R-Drop. My best CV score achieved by a single model (DeBERTa-v3-large, 5-fold) is 0.8528, obtained using both EMA and R-Drop during training.

However, the best private leaderboard score aligned more closely with the best public score rather than my local CV, highlighting discrepancies between local validation and leaderboard performance.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

I would focus on exploring the impact of various prompt templates on the model's performance. My goal would be to automatically generate the most effective templates for this specific task.

12. What simplifications could be made to run your solution faster without sacrificing significant accuracy?

We can drop 5-fold longformer models. My 5-fold deberta-v3-large model can reach a 0.8605 private score.