NBME - Score Clinical Patient Notes challenge

3rd Place Model Summary - Raja Biswas (LakeCity)

First of all, thank you NBME and Kaggle for hosting such an amazing competition. It has been an enriching experience for me with a steep learning curve. I would also like to express my sincere gratitude to the Kaggle community for sharing great ideas & engaging discussions.

A1. General Background

Competition Name: NBME - Score Clinical Patient Notes challenge
Team Name: LakeCity
Private Leaderboard Score: 0.89384
Private Leaderboard Place: 3rd

Team Members:
Name: Raja Biswas
Location: Singapore
Email: saun.walker.150892@gmail.com

A2. Personal Background

Academic: I graduated from Indian Institute of Technology, Kanpur (IIT Kanpur) with a bachelor's degree in Civil Engineering (2013). Thereafter, I completed my PhD in computational mechanics from National University of Singapore (2018).

Professional: I am currently working at Evonik (https://corporate.evonik.com/en) as a NLP Data Scientist.

Prior Experience: I have a keen interest in NLP and well-versed in various NLP tasks such as Information Retrieval, NER, Relation Extraction, QA and classification. My exposure to this field definitely helped me to perform well in this competition.

Motivation: I constantly look for solving practical challenges in data science and making meaningful contributions. To this end, I found the problem statement very appealing. Specifically, I wanted to explore the area of semi-supervised learning and engage with the beautiful Kaggle community.

Time Budget: I spent an average of 4 hours per day during the last 2 months of the competition.

A3. Summary

My solution focuses on making optimal use of the unlabeled data provided as a part of the competition. To achieve this goal, I implemented the following strategies:

- **Task Adaptation:** The widely-used Language Models (LM) such as *microsoft/deberta-large* are trained with text from a wide variety of sources e.g. books, wikipedia, crawled data. It has been shown in literature that the LMs can be further adapted to the domain of a target task (e.g. the patient notes for the current task) for performance boost. To this end, I leveraged all patient notes provided and performed MLM (masked language modeling) training. The resulting task-adapted models are well-suited to handle tasks specific to the patient notes.

- **Semi-supervised Learning**: In this competition, I experimented with semi-supervised techniques such as standard Pseudo Labels (PL) and Meta Pseudo Labels (MPL). Both make use of the unlabeled data to learn a better model.
  - Standard Pseudo Labels (PL):
    - A Teacher model is first trained with the provided labeled data
    - The trained Teacher generates pseudo labels (i.e. predicted target from teacher) on the unlabeled data points
    - A Student model is next trained with actual labeled data + pseudo labeled data
    - Due to regularization and data augmentation like effect of pseudo labels, the student learns to become better than the teacher
  - Meta Pseudo Labels (MPL):
    - Achieves state-of-the-art performance in semi-supervised settings by addressing the limitations of standard PL such as inaccurate label propagation / confirmation bias
    - The teacher and student models are trained in parallel
    - Student learns from pseudo labels produced by the teacher
    - Teacher learns from labeled data + the performance improvement feedback from the student (meta-learning)
    - As training progresses, the feedback signal motivates the teacher to continuously improve the quality of pseudo labels such that the dependent student improves as well
    - In this competition, I adapted the meta pseudo labels technique introduced in Meta Pseudo Labels (https://arxiv.org/abs/2003.10580) for the current token classification task.

- **Knowledge Distillation (KD):** I used an ensemble of two models trained using MPL to distill their combined knowledge into another student model. The rationale behind using KD:
  - In literature, there are examples where the student outperforms teachers
  - Adds to model diversity

A4. Training Methods

One key ingredient in my solution is diversity in the way the models are trained. For this competition, I have trained the following models

- DeBERTa Large (4 Models)
- DeBERTa XLarge (2 Models)
- DeBERTa V2 XLarge (2 Models)
- DeBERTa V3 Large (5 Models)

In the following, the I will describe the training details of these models

DeBERTa Large:

- **Steps:**
    1. Task adaptation of the *microsoft/deberta-large* Language Model (LM) using MLM training.
    2. Execution of semi-supervised training with Meta Pseudo Labels (MPL). Here, I used a **soft** version of the meta pseudo labels i.e. raw probability outputs from the teacher are directly used as pseudo labels. The training is performed with all labeled data (~14k) + 180k randomly selected unlabeled data.
    3. Fine tuning of the student model. During MPL, the student model is trained only on unlabelled data using pseudo labels from the teacher. So the student can be further fine-tuned on actual training data for additional performance boost.
    4. Two independent execution steps 2-3, thereby producing two fine-tuned student models. The main difference between these two models: they see different unlabelled data samples and different seeds.
    5. Knowledge distillation. Two models obtained from the above step become teachers during knowledge distillation. Their combined knowledge (mean token predictions) is infused into another DeBERTa Large model. I also used Stochastic Weight Averaging (SWA) during knowledge distillation for better generalization.
    6. Two models are trained with knowledge distillation: (1) using only pseudo labels from the 2 teachers on 250k unlabelled data points (2) using pseudo labels from the 2 teachers on 250k unlabelled data points + 14k labeled data.
- **Models:**
    - 2 models out of step 4
        - Each model takes around 30 hours to train using single P100 GPU (8 hours for MLM + 20 hours for MPL training + 2 hours for fine-tuning)
    - 2 models out of step 6
        - Each model takes around 18 hours to train using single P100 GPU

DeBERTa XLarge:

- **Steps:**
  1. Task adaptation of the *microsoft/deberta-xlarge* Language Model (LM) using MLM training.
  2. Execution of semi-supervised training with Meta Pseudo Labels (MPL). Here, I used a **hard** version of the meta pseudo labels i.e. raw probability outputs from the teacher are converted to hard labels (1 if probability >=0.5, else 0). The training is performed with all labeled data (~14k) + 150k randomly selected unlabeled data.
  3. Fine tuning of the student model. During MPL, the student model is trained only on unlabelled data using pseudo labels from the teacher. So the student can be further fine-tuned on actual training data for additional performance boost.
  4. Two independent execution steps 2-3, thereby producing two fine-tuned student models. The main difference between these two models: they see different unlabelled data samples and different seeds.
- **Models:**
  - 2 models out of step 4
    - Each model takes around 40 hours to train using single P100 GPU (10 hours for MLM + 25 hours for MPL training + 5 hours for fine-tuning)

DeBERTa V2 XLarge:

- **Steps:**
  1. Task adaptation of the *microsoft/deberta-v2-xlarge* Language Model (LM) using MLM training.
  2. Standard fine-tuning of the task adapted model using all labeled data
  3. Execution of semi-supervised training with Meta Pseudo Labels (MPL). Here, I used a **hard** version of the meta pseudo labels i.e. raw probability outputs from the teacher are converted to hard labels (1 if probability >=0.5, else 0). The training is performed with all labeled data (~14k) + 150k randomly selected unlabeled data.
  4. Fine tuning of the student model. During MPL, the student model is trained only on unlabelled data using pseudo labels from the teacher. So the student can be further fine-tuned on actual training data for additional performance boost.
- **Models:**
  - 1 model out of step 2 (10 hours for MLM + 5 hours using single P100 GPU)
  - 1 model out of step 4 (10 hours for MLM + 25 hours for MPL training + 5 hours for fine-tuning)

DeBERTa V3 Large:

- **Steps:**
    1. Task adaptation of the *microsoft/deberta-v3-large* Language Model (LM) using MLM training.
    2. Execution of semi-supervised training with Meta Pseudo Labels (MPL). Here, I used a **hard** version of the meta pseudo labels i.e. raw probability outputs from the teacher are converted to hard labels (1 if probability >=0.5, else 0). The training is performed with all labeled data (~14k) + 180k randomly selected unlabeled data.
    3. Fine tuning of the student model. During MPL, the student model is trained only on unlabelled data using pseudo labels from the teacher. So the student can be further fine-tuned on actual training data for additional performance boost. During fine-tuning, I also used Stochastic Weight Averaging (**SWA**) for better generalization.
    4. Three independent execution steps 2-3, thereby producing three fine-tuned student models. The main difference between these models: they see different unlabelled data samples and different seeds.
    5. **Marked tokens**. This basically involves modifying the feature text with a prefix token. I created 10 new tokens for each case in patient notes e.g. "[QA CASE=0]". I thought some of the feature texts from different cases are very close to each other e.g. Feature 314: nausea, Feature 508: Associated-nausea. Therefore I felt, giving additional context of case number in feature text would help the model to distinguish between these two cases. Two independent execution steps 2-3 with marked tokens, thereby producing two additional fine-tuned student models.

- **Models:**
    - 2 models out of step 4
        - Each model takes around 30 hours to train using single P100 GPU (8 hours for MLM + 20 hours for MPL training + 2 hours for fine-tuning)
    - 2 models out of step 5
        - Each model takes around 30 hours to train using single P100 GPU (8 hours for MLM + 20 hours for MPL training + 2 hours for fine-tuning)

A5. Additional Details

- **Avoiding Memory Overflow:** For MPL training, both Teacher and Student models need to be loaded into memory. To avoid memory overflow issues, I used the usual tricks such as mixed precision training, 8-bit adam optimizer, smaller batch size, freezing of lower layers & gradient checkpointing.
- **Multi-label classification:** I used multi-label token classification head for all the models described in Section A4. Specifically, I created 3 labels to detect if a token is inside the answer span, start of the answer span and end of the answer span. During experimentation, I found that these auxiliary tasks boost model performance and make the models more robust in the spirit of multi-task learning. The classification head encounters the following cases:
    - outside token label -> [0, 0, 0]
    - inside token label which is also start of the answer span -> [1, 1, 0]
    - inside token which is also end of the answer span -> [1, 0, 1]
    - inside token label which is also beginning & end of the answer span i.e. span with only one token -> [1, 1, 1]
    - other inside tokens label -> [1, 0, 0]
- Reinitialization of last 4 to 12 transformer layers. The last transformer layers learn specific aspects of the masked language modeling. Hence, these layers are not particularly helpful for the current task of token classification. I found reinitialization of several top transformer layers stabilizes model training.
- For the multi-label token classification head, I concatenated hidden outputs from the last 6-12 transformer layers.

A6. Ensembling

During inference, each model predicts the probability of a character (in the patient note) to be included in the answer span given a feature text. Thereafter, the final prediction is made by ensembling predictions from individual models. In this task, I used simple blending i.e. weighted average of character wise predictions for ensembling.

A total of 18 model checkpoints is used for inference. This includes 13 models mentioned in Section A4 + 5 folds of DeBERTa v3 Large model weights obtained from this public dataset https://www.kaggle.com/datasets/thanhns/deberta-v3-large-5-folds-public. All checkpoints receive equal weight in ensembling.

I also implemented minor post processing before making the final submission. These mainly include filtering of certain keywords from predicted spans based on error analysis e.g.

- feature 309: duration-2-months: filter out spans containing ‘2 weeks’, ‘2weeks’ in it

A7. Interesting findings

- The effectiveness of task adaptation and pseudo labeling has been one of most interesting findings for me. It reinforces the importance of releasing a large pool of unlabeled task specific data to aid model adaptation through pretraining and leveraging semi-supervised technique. It is particularly useful in low resource settings where the volume annotated data is limited.
- As described in Section A5, I set up the current problem as a multi-label classification task. Based on my experiments, it is particularly useful to design relevant auxiliary tasks, which helps training of the main task. For example, predicting whether a token/character is a part of answer span is the main task, whereas predicting beginning and end of answer spans are the auxiliary tasks.
- For larger backbones such as DeBERTa V2 XLarge, it is important to re-initialize the top transformer layers. It helps in stabilizing the model training.
- The model performance on unseen test data (e.g. private leaderboard) reinforces the effectiveness of Stochastic Weight Averaging (SWA) for better generalization
- I experimented with several ensembling strategies such as simple blending, Weighted Box Fusion (WBF) and Non Maximum Suppression (NMS). Among them, simple blending of model prediction worked best. I think this is due to the combination of micro F1 metric as scoring criteria and multi-span answers.

A8. Model Execution Time

- It takes ~8 hrs to make predictions and generate submission results on the hidden test data from the 18 model checkpoints described in Section A6 (using P100 GPU).
- The training time of these models are described in Section A4.

A9. References

- Don't Stop Pretraining: Adapt Language Models to Domains and Tasks (https://arxiv.org/abs/2004.10964)
- Meta Pseudo Labels (https://arxiv.org/abs/2003.10580)
- Fine-Tuning Pre-trained Language Model with Weak Supervision: A Contrastive-Regularized Self-Training Approach (https://arxiv.org/pdf/2010.07835.pdf)
- Can Students Outperform Teachers in Knowledge Distillation Based Model Comparison? (https://openreview.net/pdf?id=XZDeL25T12l)