The Learning Agency Lab - PII Data Detection

1st Vlad-Cristian Matei
Computer Science Department
University Politehnica of Bucharest
Bucharest, Romania
vlad_cristian.matei@stud.acs.upb.ro

I. The first attempts

The first attempt was represented by training a valid model that could achieve at least a mediocre result. That's roughly what I thought, that it would be easiest to use a pre-trained BERT model for solving the NER task, and also, it would be great if I could import something from the transformers library as part of the Huggingface project.

So, I chose the BERT architecture, in its base form, for training the dataset.

A. BERT-base-ner

BERT, or Bidirectional Encoder Representations from Transformers [1], is a game-changing model in NLP. Developed by Google in 2018, it stands out for its bidirectional understanding of word context, unlike previous models. Pre-trained on vast datasets like Wikipedia and BookCorpus, BERT learns language nuances through masked language modeling. Its adaptability allows fine-tuning for specific tasks like sentiment analysis and Named Entity Recognition (NER), setting new standards in NLP performance and bridging the gap between language complexity and task specificity.

Regarding the dataset preprocessing and fine-tuning process, I aimed to adhere closely to the points specified in the literature [1]. I utilized a BERT family Tokenizer for tokenization, handled special tokens to mark the beginning and end of sequences, and addressed subword tokens appropriately. Additionally, I implemented a learning scheduler to adjust the learning rate during training, and applied the technique of freezing layers to manage the fine-tuning process.

II. Best Solution

When aiming for significantly improved results, I turned to discussions and proposals on Kaggle and sought inspiration from various public notebooks. My goal was to enhance these notebooks through my own creativity and ideas, leveraging them as a source of inspiration for my work. So, I managed to find some discussions, such as this one [2], presenting some results that include an architecture based on deBERTa.

[1]https://medium.com/@ahmetmnirkocaman/mastering-named-entity-recognition-with-bert-a-comprehensive-guide-b49f620e50b0
[2]https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data/discussion/493608

A. DeBERTa

It's common knowledge that BERT is built on the attention mechanism from the Transformer architecture, a cornerstone in modern NLP. However, new ideas emerge continuously in machine learning. One notable innovation, introduced in 2021, enhanced attention as "Disentangled attention," leading to DeBERTa [2]. Despite incorporating just a couple of new architectural principles, DeBERTa shows significant improvements on major NLP benchmarks compared to other large models.

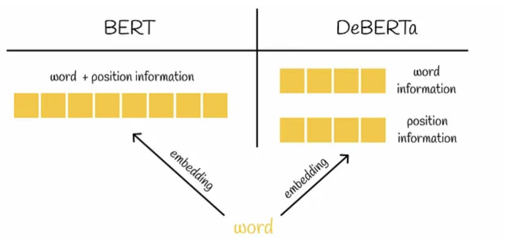

Fig. 1. Embedding construction in BERT and DeBERTa. Instead of storing all the information in a single vector, DeBERTa uses two separate vectors to store word and position embeddings. [3]

1) Disentangled attention: DeBERTa introduces the concept of disentangled attention, aiming to overcome potential information loss in the original Transformer block. Unlike the single vector representation in Transformers, DeBERTa stores information in two separate vectors for each token, addressing both content and position. The attention computation mechanism is adjusted to explicitly consider the relations between token content and position.

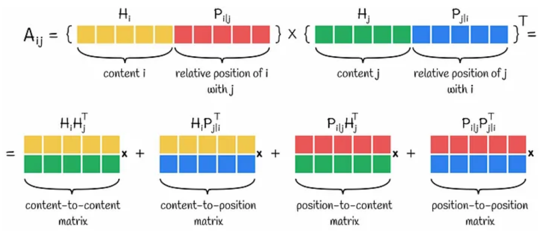
$A_{ij} = \{ H_i \; P_{i|j} \} \times \{ H_j \; P_{j|i} \}^T =$
content i
relative position of i with j
content j
relative position of j with i
$= H_iH_j^T + H_iP_{j|i}^T + P_{i|j}H_j^T + P_{i|j}P_{j|i}^T$
content-to-content matrix
content-to-position matrix
position-to-content matrix
position-to-position matrix

Fig. 2. Computation of cross-attentin score between two embedding vectors. [3]

This adjustment involves decomposing the calculation of cross-attention scores into four pairwise multiplications of subvectors. These multiplications result in matrices representing different combinations of content and position information. Notably, the position-to-position matrix is discarded as it lacks content details. The remaining matrices contribute to the final output attention matrix, computed similarly to the original Transformer.

B. Enhanced mask decoder

In the context of DeBERTa, the enhanced mask decoder addresses the limitation of disentangled attention by considering absolute positioning, in addition to content and relative positioning. The absence of absolute positioning information can hinder accurate predictions, as demonstrated by examples like ”a new store opened beside the new mall.” Human understanding of grammatical word order relies on absolute positions, aiding in sentence completion. DeBERTa integrates absolute positioning after all Transformer layers, enhancing model performance by combining it with relative positioning. This approach improves the model's ability to learn and utilize information about the context and structure of language.

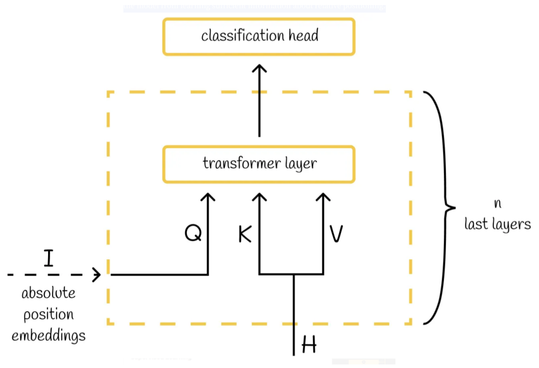

Fig. 3. Enhanced mask decoder in DeBERTa. [3]

C. Other Solution Ideas

Thus, after choosing to train DeBERTa-large, I researched forums and found information indicating that experimentally, the best results have been achieved with max_tokenization = 1024 or higher if resources were available [3]. Additionally, results could be improved by setting a confidence threshold for labels. This enhances recall by filtering out predictions labeled ”O” with lower probabilities and selecting predictions with the highest probabilities from the remaining labels. [4] Lastly, a few percentage points could be gained by adding extra datasets to augment the primary dataset in capturing information. One dataset was highly recommended on forums [5], while descriptive analyses existed for others [6]. I noticed the dataset was imbalanced, so I creatively attempted to compose a unique dataset from the recommended parts on forums, but in an intuitive manner chosen to help balance the dataset. However, I also read discussions where this method seemed to be considered risky and could potentially harm the final results [7]. Finally, in my solution, I applied the recommendation found on several forums, namely using weighted probabilities provided by the model along with regex patterns for classes with very few examples in the dataset [8].

[3]https://www.kaggle.com/code/emiz6413/deberta-v3-single-model-lb-0-966

[4]https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data/discussion/470978

[5]https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data/discussion/472221

[6]https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data/discussion/479491

III. Challenges

One of the biggest challenges was getting familiar with the Kaggle platform. Understanding the best way to run notebooks, efficiently using resources, managing sessions, and versions was crucial. Another challenge was handling a Named Entity Recognition (NER) task, which I encountered for the first time. I had to learn how to perform quality tokenization, which involved research and repeated attempts. In the end, to secure a higher position on the leaderboard, I had to study DeBERTa, a model I learned about upon joining this competition.

IV. Milestone II

What I attempted to do was to use LoRA (Low-Rank Adaptation) [4]. By employing this method during training, I was able to train models comparable to the best model trained for milestone 1, while benefiting from increased training speed, the ability to create more checkpoints, and utilizing a solution like load best model at end. By using LoRA, I was able to evaluate the model's performance by training over multiple epochs and folds, and by analyzing partial results. This approach, added in this milestone, resulted in an improvement. The method was inspired by this source [9]. Conclusively, I created checkpoints every 250 steps, compared to 5000 steps before. I trained for 8 epochs instead of 5, experimented with a batch size of 2048 compared to 1024 previously, trained completely on 4 folds instead of 3, and closely monitored the evolution of metrics throughout the training. Ultimately, the final result obtained after submission was 0.958.

V. Future Directions

As for future directions, I'm considering the results I could have achieved by attempting model distillation from the most successful models, followed by a multi-tasking approach or weighted combination of predictions from multiple models. Additionally, I'm contemplating whether better results could have been obtained by adding a Conditional Random Fields (CRF) layer on top of the final layer from DeBERTa. Ultimately, a future direction involves conducting more in-depth research to surface new methods, combinations of ideas, and experiments that could lead to further advancements.

[7]https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data/discussion/478227

[8]https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data/discussion/486838

[9]https://www.kaggle.com/code/emiz6413/train-deberta-large-on-kaggle-kernel-with-peft

VI. Results

Model	Score	Folds	Epochs	Size	ExtraData
BERT-base-NER	0.7	1	3	512	
deBERTa-base	0.920	4	3	512	
deBERTa-v3-large	0.956	4	3	1024	2
deBERTa-v3-large-Lora	0.958	1	8	1024	1

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[2] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," 2021.

[3] Large language models: Deberta - decoding enhanced bert with disentangled attention. Accessed on 19.4.2024. [Online]. Available: https://towardsdatascience.com/large-language-models-deberta-decoding-enhanced-bert-with-disentangled-attention-90016668db4b

[4] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.