BERT variations for NER
- Species-800 dataset -

Constantin Gabriel-Adrian
Onescu Iancu-Gabriel
Sociu Daniel
Group: 407 (AI)

Abstract

Given that the task at hand is NER, we must strive to achieve the representation of tokens (words) from the dataset as (numerical) embeddings. With the help of BERT transformer [1], we can accomplish this fairly easily. A transformer is a deep learning model whose goal is solving ”sequence to sequence” tasks i.e. encode-decoder. Transformers are similar with recurrent neural networks (RNN), the main difference being that they don't process the data in a specific order. Thus, a given sequence is analyzed in parallel and the influence or importance of each part of the sequence is determined at every step. Given the fact that by parallelizing this task the initial order is forgotten, the transformer adds for each embedding a positional encoding. In general, transformers predict each token at pretrain. The advantage of BERT over classic transformers is that it can train in a bidirectional fashion using masked LM. This mask is useful for hiding at random each token that is going to be predicted.

About dataset

A corpus for species entities called Species-800 was created based on personally annotated abstracts. There are 800 PubMed abstracts in it that mention recognized organisms. The 800 abstracts were chosen from the following 8 categories: bacteriology, botany, entomology, medicine, mycology, protistology, virology, and zoology, with 100 abstracts from each chosen to improve the corpus taxonomic mention diversity. Species-800 has been annotated with a concentration on species, although higher taxa (including genera, families, and orders) have also been taken into account.

Related work

Tian et. all [6] leverage the syntactic information automatically computed via key-value memory networks (KVMN) and the high understanding capacity of BioBERT to surpass the previous baselines on a variety of datasets including Species-800. They focus specifically on POS labels, syntactic constituents and dependency parsing. Meanwhile, Sharma and Daniel [5] use a pretrained model from the FLAIR framework together with their provided 'pubmed-x' embeddings. They show that these embeddings, even though trained on a much smaller portion of the PubMed corpora (about 5%), are competitive with those of BERT. They also employ a stacked embeddings approach where they use the features generated by BioELMo, which further improve their previous results on all datasets. Kocaman and Talby [3] reimplement the Bi-LSTM-CNN-Char architecture on an Apache Spark framework obtaining an overall f1 score of 80.91%. In the original architecture, the CNN extracts a fixed length feature vector from character-level features and passes it to a forward and backward LSTM which then sums up the output to decide whether the word represents an entity or not. The change they brings consists in the bio-medical embeddings they obtain employing a skip-gram model for learning contextual information. Furthermore a CNN is applied, followed by a 1D Conv and a MaxPooling in order for each word to get a vector representation. On top of that they changed the classic LSTMBlockCell to LSTMBlockFusedCell which have proved to be faster because they use a single TF operation for the entire LSTM. The same authors [2] manage to obtain an f1 score of 82.0% making use of the same architecture together with the same character embeddings as before to which they add Glove/Bert/ELMO embeddings. Phan et. all [4] come up with a model named SciFive that follows the seq-to-seq encoder-decoder architecture and the T5 framework. They train this model on combinations of the C4 corpus and PubMed publications, obtaining in the end a score of 79.45% on the Species-800 dataset.

Proposed approach

We used different variants of BERT for this task.

Given the fact that the input statements have different lengths (number of tokens), to pass them through a transformer we would have to make all of them a specific length. This was done by splitting the longer statements in multiple parts of max length 128, and the rest that didn't reach the set length, we padded them. The tokenizer used splits the tokens in multiple subtokens, therefore only the first token should have a label, therefore the rest of the subtokens were assigned the special label of -100, which is the default ignored value by optimizers. After token splitting, we have replaced all non-alphanumeric characters.

With the tokens good to go, we created a Dataset class to store all our input data which was further passed to our Dataloader. The Dataloader is meant to simplify the workflow of our training process by automatically splits the data in batches. With the data ready to be used thanks to the data processing step, the next step was applying the model. The pre-trained BERT model was not completely prepared for our current task.

As such, one of the 2 most-common transfer learning methods are:

1. **Fine Tuning**: this represents training the model starting from the weights of the pre-trained model and also updating the output head to be corresponding for our task.

2. **Feature Extraction**: which in our case implies to freeze some superior layers which contributes to the creation of some embedding features.

Given the limited compute power available at our disposal (a 3060 Ti with 8GB VRAM), we only explored the second transfer learning method. As such, we explored the following types of BERT:

1. BERT base-uncased

2. BERT base-cased

3. RoBERTa

4. BERTNER

5. BioBERT

Experiments and training methods used

After training the model for 10 epochs, with the batch dimension of 64, this model was experiencing a very bad performance. This was caused by the non-balanced training classes, therefore it had a tendency to predict the dominant classes more which affected the validation score. To mitigate this problem we computed the weights of the classes with the sklearn function compute_class_weight, which were passed to the optimizer that improved the performance of the model by a lot.

We found out that the best optimizer for fine-tuning is the Adam optimizer with a learning rate of 1e-4. Even though the learning rate is pretty small, we also used the torch.nn.utils.clip_grad_norm_ function to clip the gradients so that we don't suffer of explosive gradients. The main tests we did were about the number of epochs trained and the number of frozen layers.

We tested multiple models and evaluated them on validation and test set.

BERT base

The BERT base model has 12 transformer layers and 110 million parameters. It is pretrained on a big corpus of data, but it is not finetuned to a downstream task by default. The cased version preserves the original casing of words, allowing for the differentiation between uppercase and lowercase letters. This can be useful in tasks where case information is important, such as part-of-speech tagging or entity recognition. On the other hand, the uncased version converts all text to lowercase, resulting in a smaller vocabulary size and potential benefits in scenarios where case distinctions are less relevant, such as text classification or sentiment analysis.

The results obtained are shown in the following table:

	Batch size	Frozen layers	F1-score
BERT base-uncased	64	6	0.816
BERT base-cased	64	6	0.841
BERT base-cased	64	4	*0.85*
BERT base-cased	32	2	0.823

Table 1: Results of the BERT base models.

RoBERTa

Roberta is another transformer-based language model, but it follows a different training approach compared to BERT. Instead of using MLM and NSP objectives, Roberta is trained using a method called ”masked language modeling” (MLM) alone. It also utilizes larger batch sizes and longer training schedules. Roberta achieves state-of-the-art results on various NLP benchmarks and typically outperforms BERT models in terms of accuracy.

The results obtained with 6 frozen layers and a batch size of 64 are as follows:

	Learing rate	Nr epochs	F1-score
RoBERTa	1e-4	10	0.787
RoBERTa	5e-5	20	0.817

Table 2: Results of the RoBERTa model.

BertNER

BertNER is a model that fine-tunes BERT down the task named entity recognition (NER) on annotated NER datasets, enabling accurate recognition of named entities. Therefore this model enriches the NER capabilities of the model, but since our NER class was not in its finetuning, the results were not better.

	Learing rate	Nr epochs	F1-score
BertNER	1e-4	10	0.799
BertNER	5e-5	20	0.828

Table 3: Results of the BertNER model.

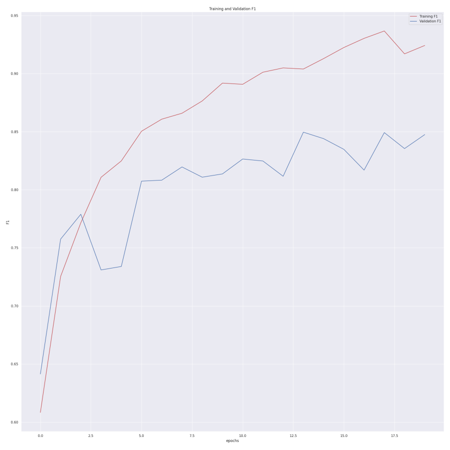


Figure 1: F1 score for train and validation

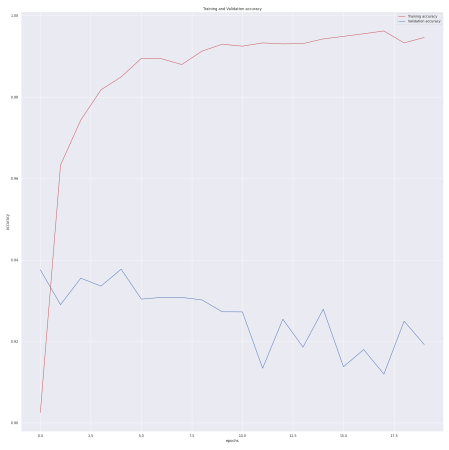


Figure 2: Accuracy score for train and validation

BioBERT cased

BioBERT is a specialized variant of BERT designed specifically for biomedical text analysis. It is pretrained on a large corpus of biomedical literature. BioBERT captures domain-specific knowledge and is commonly used for various biomedical NLP tasks.

	Learing rate	Nr epochs	F1-score
BioBERT cased	1e-4	10	0.837
BioBERT cased	5e-5	20	**0.866**

Table 4: Results of the BioBERT cased model.

As such, for the best model, BioBERT cased with a batch size of 64 and 6 frozen layers, the F1, accuracy and loss are:

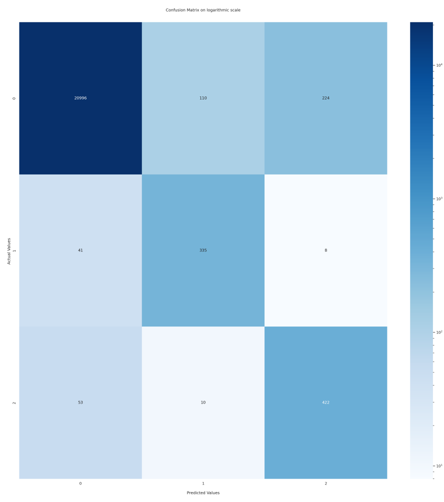


Figure 3: Test Confustion matrix

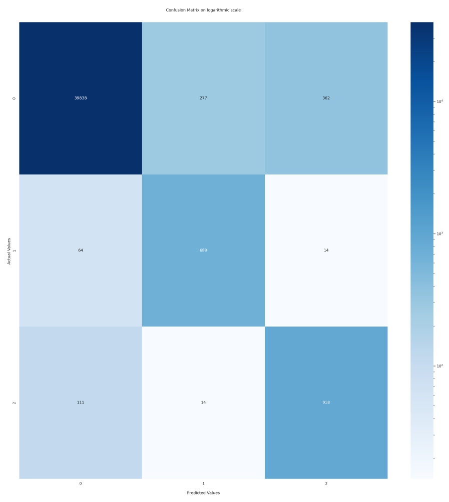
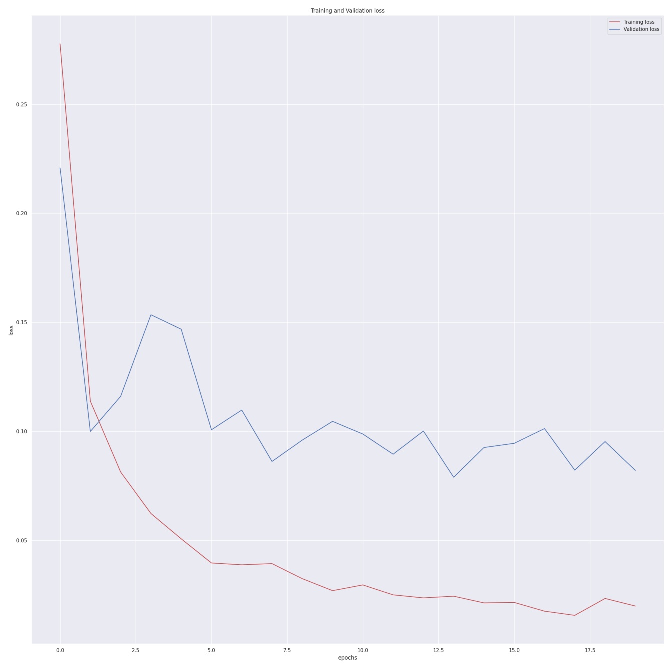


Figure 4: Validitation confusion matrix



Above, we can see the difference between the confusion matrix on the validation set vs on the test set.

Conclusion

The task at hand did not prove difficult to tackle, however it was interesting to observe the volatility of the models based on the upstream task. To our surprise only one of the specialised models managed to beat the base version of BERT. Perhaps this is an indicator that often times simple is better. What caught our attention is the immediate improvement over the previous State-of-the-art models which sometimes utilise the same architecture. However the recent progress in computation power tells us that the extra training that BERT has undergone in the past years can be proven beneficial on older datasets. Perhaps the next steps should involve larger architectures without other upstream specialization. We believe that BERT-large could push the results even further, but with careful tunning, because, as we have seen, a larger number of unfrozen layers doesn't bring an improvement after a certain threshold.

Bibliography

[1] Huggingface. *BERT model.* URL: https://huggingface.co/docs/transformers/model%5C_doc/bert (visited on 01/15/2023).

[2] Veysel Kocaman and David Talby. "Accurate Clinical and Biomedical Named Entity Recognition at Scale." In: (). DOI: https://doi.org/10.1016/j.simpa.2022.100373.

[3] Veysel Kocaman and David Talby. "Biomedical Named Entity Recognition at Scale." In: (). DOI: https://doi.org/10.1007/978-3-030-68763-2_48.

[4] Long N. Phan, James T. Anibal, Hieu Tran, Shaurya Chanana, Erol Bahadıroglu, Alec Peltekian, and Grégoire Altan-Bonnet. "SciFive: a text-to-text transformer model for biomedical literature." In: (). DOI: https://doi.org/10.48550/arXiv.2106.03598.

[5] Shreyas Sharma and Jr. Ron Daniel. "BioFLAIR: Pretrained Pooled Contextualized Embeddings for Biomedical Sequence Labeling Tasks." In: (). DOI: https://doi.org/10.48550/arXiv.1908.05760.

[6] Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. "Improving biomedical named entity recognition with syntactic information." In: (). DOI: https://doi.org/10.1186/s12859-020-03834-6.