The bioMETRO-LM project

Short project description

The bioMETRO-LM project aims to train (& release) a wide variety of METRO-like models on a variety of biochemical sequences.

Data

Since the project is focused on the language modeling aspect, it can be readily applied to a variety of biochemical sequence sources. Data retrieval and preprocessing should be handled as part of the *bio-datasets* project, while model evaluation should be handled as part of the *bio-eval* project).

We could leverage the language model's internal representation for general protein fitness & structure prediction (see ESM, ProtTrans, ProteinBERT and CARP, AminoBERT). For this use case, we would most likely be training on either UniRef50 or UniRef90.

When it comes to proteins we could also train more "specialized" models, for example for antibody sequences (see AntiBERTa, trained on a clustered set of sequences sourced from the Observed Antibody Space) or T-cell receptor sequences (see TCR-BERT, trained on sequences sourced from TCRdb).

For protein sequences, antibody sequences and TCR sequences we most likely want to include both the original sequences and their reverse.

We could also apply this kind of modeling to small molecule sequences (see MolFormer, for example). In this case, the main datasets would be PubChem and ZINC, and we would most likely want to leverage the SELFIES molecular string representation over the more commonly used SMILES.

Methods

The main method we want to leverage is the replaced token detection (RTD) objective first introduced in ELECTRA. The objective has seen significant success in NLP, having been picked up by many followup models (including COCO-LM, DeBERTaV3 and most recently METRO) due to its increased sample efficiency compared to BERT's masked language modeling (MLM) objective. In particular, by forcing the model to decide whether each and every token in the sequence has been replaced or not, with RTD we are providing the model with a learning signal for *all* tokens in the dataset, unlike MLM's case (where it's usually provided for 15% of tokens). This fact is particularly important for datasets that are not exceedingly large, like most biochemical ones.

When it comes to positional embeddings we likely want to ablate both Rotary and AliBi.

FlashAttention provides a highly performant IO-aware self-attention implementation. It's however not yet easy to install, and it limits each head's dimension to 64 (though this one in particular should not be an issue, given that the largest METRO models also use this head size). It's likely worth considering.

We most likely want to use Maximal Update Parametrization (muP) for hyperparameter tuning. Properly tuning the models will be important, as some previous models have been such as ProtTrans-XL & XXL have been shown to perform badly with respect to their size.

We most likely want to use either PyTorch's Fully Sharded Data Parallel (FSDP) API or ALPA (in case of a JAX codebase) to make sure models can be scaled up if needed. In case we do scale up the model size, we should keep in mind that most downstream users won't have access to large-scale computing resources, and that as such we should make sure that inference from all models is possible through Colab instances.

Project goals

The main project goal should be that of building a scalable codebase for METRO-like models, in order to allow the training of a variety of models on various kinds of biochemical sequences. We should attempt to release the best single (protein, antibody, TCR or molecule) sequence models at the time of release.