HuggingMolecules: An Open-Source Library for Transformer-Based Molecular Property Prediction (Student Abstract)

Piotr Gaiński,[1,2] **Łukasz Maziarka,**[1,*] **Tomasz Danel,**[1,2] **Stanisław Jastrzebski**[3]
[1]Jagiellonian University
[2]Ardigen
[3]Molecule.one
piotr.gainski@student.uj.edu.pl

Abstract

Large-scale transformer-based methods are gaining popularity as a tool for predicting the properties of chemical compounds, which is of central importance to the drug discovery process. To accelerate their development and dissemination among the community, we are releasing HuggingMolecules – an open-source library, with a simple and unified API, that provides the implementation of several state-of-the-art transformers for molecular property prediction. In addition, we add a comparison of these methods on several regression and classification datasets. HuggingMolecules package is available at: github.com/gmum/huggingmolecules.

Introduction

Predicting molecule properties is a predominant task in the drug discovery pipeline. A good predictive model is, therefore, a key tool in this process as it can accelerate the whole pipeline as well as prevent costly mistakes in clinical trials. Large pre-trained models based on the transformer architecture are gaining popularity in the molecular property prediction tasks and often become the state-of-the-art methods in this field. We believe that in order to accelerate the development of transformer-based methods in chemistry and to spread their use among practitioners, it is necessary to create one package in which it will be possible to use different models in a simple, consistent way, as in the case of huggingface-transformers in NLP.

In this paper, we propose HuggingMolecules – an open-source python library for simple, consistent usage of different transformer-based methods for molecular property prediction tasks. We also make a strict comparison of transformer-based methods implemented in our library on many molecular property prediction datasets

Available Models

HuggingMolecules includes implementation of 4 transformer-based methods, together with their pre-trained weights: MAT (Maziarka et al. 2020), ChemBERTa (Chithrananda, Grand, and Ramsundar 2020), GROVER (Rong et al. 2020), MolBert (Fabian et al. 2020).

[*]Work done in Ardigen.

Library Architecture

HuggingMolecules consists of two main modules: *src/* and *experiments/* modules. The *src/* module contains abstract interfaces for pre-trained models along with their implementations based on the PyTorch library. This module makes configuring, downloading, and running existing models easy and out-of-the-box. The experiments/ module makes use of abstract interfaces defined in the *src/* module and implements scripts based on the PyTorch Lightning package for running various experiments. This module makes training, benchmarking, and hyper-tuning of models flawless and easily extensible. The neat and abstract design of HuggingMolecules allows to implement and benchmark of new model architectures in a consistent way.

The Src/ Module

Every model implemented in the src/ module has three dedicated classes located in the *configuration*, *featurization* and *models* sub-modules of the *src/* module. The configuration class from the *configuration* module stores information about hyper-parameters of the model. The featurizer class from the *featurization* module is used to transform a SMILES string into representation (a set of features) that can be fed to the model. Finally, the model class from the *models* module contains the implementation of the model itself. The interplay between these three classes is shown on the following code snippet based on the MAT model:

```
config = MatConfig()
featurizer = MatFeaturizer(config)
model = MatModel(config)

batch = featurizer(["CCO", "CO"])
output = model(batch)
```

Moreover, HuggingMolecules provides easy access to the pre-trained weights of the implemented models. It exposes a simple user interface and takes care of downloading and caching the appropriate data. The following code snippet shows how to initialize the MAT model with weights pre-trained on 20M molecules:

```
model = MatModel.\
    from_pretrained("mat_masking_20M")
```

The Experiments/ Module

The core part of the *experiments/* module are three python scripts named *train*, *tune_hyper*, and *benchmark*. The *train* script can be used for convenient training of a model with the PyTorch Lightning package. The *tune_hyper* script performs automated search for optimal training hyper-parameters with the Optuna package that implements state-of-the-art searching heuristics. The *benchmark* script benchmarks a model with a standardized benchmark procedure described in the next section. The scripts are integrated with the abstract interfaces from *src/* module, so any new model implemented by an user in the *src/* module can be used with the scrips after adding a single configuration file.

The *experiments/* module incorporates Therapeutics Data Commons (TDC) framework (Huang et al. 2021) to systematically access the entire range of datasets used both in chemistry and the drug discovery process. The *experiments/* module goes with configuration files for 8 datasets from TDC. The dataset base can be easily extended with any other dataset (even that unavailable in TDC) by adding an appropriate configuration file.

Configuration files with meta-data required to run a script are organized in a hierarchy and injected into the script by the Gin Config package. It makes extending the *experiments/* module intuitive and easy and allows to configure powerful scripts with just a few flags in the command line. The following bash snippet shows how to benchmark the MAT model pre-trained on 20M molecules on the FreeSolv dataset:

```
python -m scripts.benchmark /
-m mat -d freesolv /
--model.pretrained_name mat_masking_20M
```

Benchmark

We benchmark all models implemented in HuggingMolecules on 9 different datasets (FreeSolv, Caco-2 , Clearance, QM7, HIA, Bioavailability, PPBR, BBBP, Tox21-NR-AR) using the *benchmark* script from the *experiments/* module. The benchmark procedure is: on the given dataset we train the given model on 10 learning rates and 6 data splits (60 trainings in total). Then we choose the learning rate that optimizes an averaged (on 6 data splits) validation metric (metric computed on the validation dataset, e.g. RMSE or ROC AUC). The final result of the benchmark is the average value of the metric computed on the test set for the chosen learning rate.

We tested 3 versions of MAT: pre-trained on 200k, 2M, and 20M of molecules; and 2 versions of GROVER: base version and large version. Additionally, we tested 3 different versions of D-MPNN (Yang et al. 2019): a vanilla version (referred as D-MPNN), a version with *rdkit_2d_normalized* features generator (D-MPNN 2d) and a version with *morgan_count* features generator (D-MPNN mc).

Rank plots of obtained results are presented in Figure 1. From the obtained results we can see that graph-based transformers (MAT and GROVER) surpass textual transformers that operate directly on SMILES (ChemBERTa, MolBERT) and the non-transformer state-of-the-art D-MPNN model.

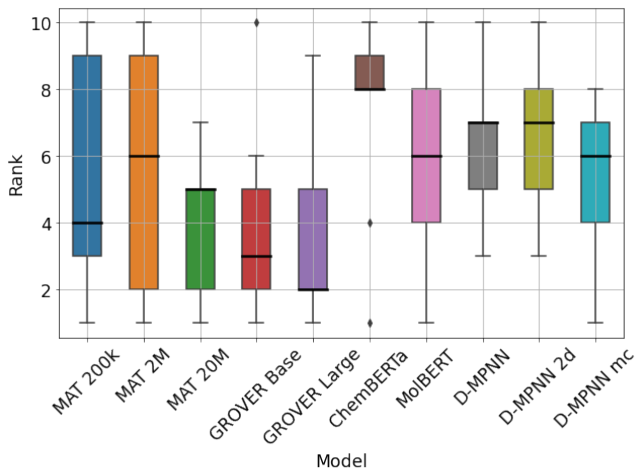

Figure 1: Rank plot of benchmarked models.

Acknowledgments

This research was funded by the Priority Research Area Digiworld under the program Excellence Initiative – Research University at the Jagiellonian University in Kraków. The work of P. Gaiński was supported by the POB DigiWorld and the work of Ł. Maziarka was supported by the National Science Centre (Poland) grant no. 2019/35/N/ST6/02125. We would like to thank NVIDIA for supporting us with the computational resources required to complete this work.

References

Chithrananda, S.; Grand, G.; and Ramsundar, B. 2020. ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction. *arXiv preprint arXiv:2010.09885*.

Fabian, B.; Edlich, T.; Gaspar, H.; Segler, M.; Meyers, J.; Fiscato, M.; and Ahmed, M. 2020. Molecular representation learning with language models and domain-relevant auxiliary tasks. *arXiv preprint arXiv:2011.13230*.

Huang, K.; Fu, T.; Gao, W.; Zhao, Y.; Roohani, Y.; Leskovec, J.; Coley, C. W.; Xiao, C.; Sun, J.; and Zitnik, M. 2021. Therapeutics Data Commons: Machine Learning Datasets and Tasks for Therapeutics. arXiv:2102.09548.

Maziarka, Ł.; Danel, T.; Mucha, S.; Rataj, K.; Tabor, J.; and Jastrzebski, S. 2020. Molecule attention transformer. *arXiv preprint arXiv:2002.08264*.

Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; and Huang, J. 2020. Self-Supervised Graph Transformer on Large-Scale Molecular Data. *Advances in Neural Information Processing Systems*, 33.

Yang, K.; Swanson, K.; Jin, W.; Coley, C.; Eiden, P.; Gao, H.; Guzman-Perez, A.; Hopper, T.; Kelley, B.; Mathea, M.; et al. 2019. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8): 3370–3388.