Preference Optimization for Molecular Language Models

Ryan Park
Harmonic Discovery Inc.
Stanford University
rypark@stanford.edu

Ryan Theisen
Harmonic Discovery Inc.
ryan@harmonicdiscovery.com

Navriti Sahni
Harmonic Discovery Inc.
navriti@harmonicdiscovery.com

Marcel Patek
Harmonic Discovery Inc.
marcel@harmonicdiscovery.com

Anna Cichońska
Harmonic Discovery Inc.
anna@harmonicdiscovery.com

Rayees Rahman
Harmonic Discovery Inc.
rayees@harmonicdiscovery.com

Abstract

Molecular language modeling is an effective approach to generating novel chemical structures. However, these models do not *a priori* encode certain preferences a chemist may desire. We investigate the use of fine-tuning using Direct Preference Optimization to better align generated molecules with chemist preferences. Our findings suggest that this approach is simple, efficient, and highly effective.

1 Introduction

In recent years, molecular language models have proven remarkably effective for molecule generation tasks. Such approaches utilize string representations of molecules, such as SMILES [Wei88] or SELFIES [KHN+20], together with standard language modeling architectures to learn and sample from a distribution over chemical structures. These models can then be leveraged for tasks such as drug and material design. For example, the MOSES benchmarking suite finds that a simple LSTM architecture trained on drug-like molecules, represented as SMILES, achieves at or near state-of-the-art performance across a variety of metrics measuring drug-likeness, synthesizability, diversity, and novelty [PZSL+20].

However, molecular language models do not by default encode all properties required for use in many practical settings. For example, a medicinal chemist may be interested in chemical structures containing only a particular substructure, without certain reactive groups, or having other properties such as binding affinity against a given target of interest. While it is possible to perform post-hoc filtering or optimization for these properties after sampling (e.g., using Monte Carlo Tree Search [YZY+17]), it is desirable to have models that can *a priori* encode arbitrary preferences as desired by a user.

In this work, we explore the use of Direct Preference Optimization (DPO) [RSM+23] to encode preferences directly into molecular language models via fine-tuning. Since many of the properties of interest can be directly and efficiently computed (or estimated), it is possible to cheaply generate large labeled, synthetic datasets for this task using a pre-trained language model. We show that this

Workshop on Generative AI and Biology (NeurIPS 2023).

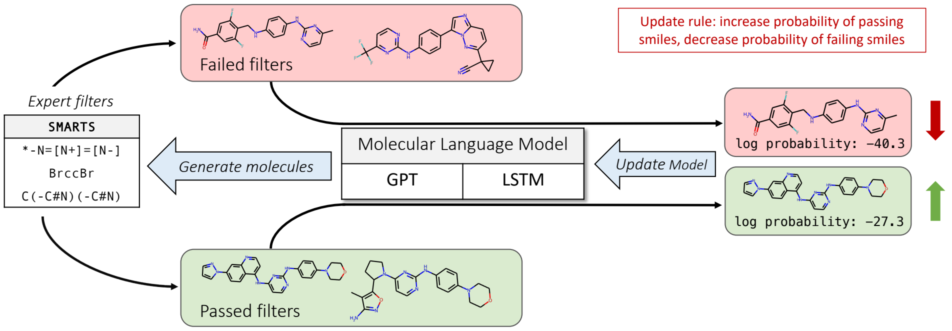

Figure 1: Schematic representation of using DPO to fine-tune a molecular language model to produce molecules that pass medicinal chemist filters.

approach, together with the DPO fine-tuning strategy, enables significant improvement in the quality of generated molecules.

2 Setup and background

2.1 Preference optimization with DPO

DPO was recently introduced as an effective alternative to reinforcement learning (RL) for fine-tuning language models using ordered preference data [RSM+23]. Importantly, unlike RL, DPO does not require learning a separate reward model prior to fine-tuning, and instead facilitates downstream training of a language model directly from preference data. Given a fixed, pre-trained reference model $\pi_{\text{ref}}(s)$, which assigns a probability to a sequence s, and ranked pairs $s_p \succ s_n$, the DPO objective can be written as

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(s_p, s_n)}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(s_p)}{\pi_{\text{ref}}(s_p)} - \beta \log \frac{\pi_\theta(s_n)}{\pi_{\text{ref}}(s_n)}\right)\right],$$

where π_θ is the model to be fine-tuned, and β is a hyper-parameter controlling the extent to which the fine-tuned model deviates from the reference model. Intuitively, the DPO objective shifts probability mass away from negative sequences s_n towards positive sequences s_p, while not deviating too much from the reference model.

The procedure for fine-tuning using DPO is as follows: generate sequences $s_1, \ldots, s_N$ from the reference model π_{ref}, and rank them pairwise. One practical way to do this is to assign a numerical score $y_1, \ldots, y_N$ to each sequence, and construct a positive/negative pair (s_i, s_j) such that $y_i > y_j$. This is the approach we take for training with DPO in the present work.

2.2 Experimental setup

Here, we briefly describe the experimental setup used in the present study. We include additional details in Appendix A; code to reproduce our results is available at https://github.com/Harmonic-Discovery/pref-opt-for-mols.

Model architectures. We perform experiments on two language modeling architectures: a generative pre-trained transformer architecture (GPT) [RNSS18], and a simple but popular LSTM-based architecture used in [SKTW17], closely based on the implementation in the MOSES benchmark suite [PZSL+20]. We refer to the GPT model pre-trained on MOSES as smiles-gpt-base and the pre-trained LSTM model as smiles-rnn-base.

Data. For pre-training, we use the MOSES benchmark dataset [PZSL+20], which consists of 1.9M unique SMILES strings extracted from the ZINC Clean Leads database. For fine-tuning, we also query molecules from the ChEMBL database [MGB+18].

Model	**FracValid**	**FracUnique**	**FracPassesMCF**	**IntDiv**
`smiles-rnn-base`	0.995	0.995	0.532	0.857
`smiles-rnn-mcf-dpo`	0.994 (-0.1%)	0.986 (-0.9%)	0.872 ($+65\%$)	0.849 (-0.9%)
`smiles-gpt-base`	0.995	0.995	0.517	0.856
`smiles-gpt-mcf-dpo`	0.994 (-0.1%)	0.992 (-0.3%)	0.927 ($+79\%$)	0.847 (-1.1%)

Table 1: **Results from MCF fine-tuning experiments.** We observe that for both architectures, DPO fine-tuning significantly increases the rate of filter passing while minimally affecting other metrics.

Metrics. To evaluate models, we use several standard metrics. Specifically:

- **FracValid**: the fraction of generated molecules that represent valid chemical structures (i.e., can be parsed from SMILES strings by the Python package `RDKit`).
- **FracUnique**: the fraction of valid generated molecules that are unique.
- **IntDiv**: the internal diversity of a set of molecules, defined by

$$1 - \binom{M}{2}^{-1} \sum_{i,j;\, i \neq j} T_c(m_i, m_j),$$

where T_c is the Tanimoto similarity, computed using ECFP4 fingerprints of molecules m_i, and M is the number of molecules.

Throughout, all metrics we report are calculated based on 10,000 SMILES strings sampled from the relevant model.

Training. All training was performed using a single NVIDIA A5000 GPU with 24GB of memory.

3 Experiments

We present results from two different experiments using DPO to fine-tune molecular language models. Our focus here is on tasks specifically relevant to drug discovery and, moreover, we exclusively rely on feedback that can be computed automatically.

3.1 Molecule filtering

For our first set of experiments, we consider feedback in the form of a set of filters that exemplify the desiderata a medicinal chemist may want from generated molecules. Specifically for this task, molecules are assigned a binary score in $\{\texttt{PASS}, \texttt{FAIL}\}$ based on the following criteria. A molecule fails the filters if it does not satisfy any one of the following:

1. The molecule does not contain any one of a set of 91 distinct SMARTS filters, representing undesired chemical substructures (e.g., reactive substructures, difficult to synthesize, etc).
2. The molecule has a molecular weight between 300 and 600 Daltons.
3. The molecule has fewer than 2 chiral centers.
4. The molecule contains fewer than 8 rings.

Molecules with the score `PASS` are considered positive examples for the DPO step, and molecules with the score `FAIL` are considered negative examples. Using the pre-trained `smiles-gpt` and `smiles-rnn` models, we sample 100,000 molecules, and perform this filtering in order to obtain training data for DPO. A schematic representation of this workflow is presented in Figure 1.

We report results from these experiments in Table 1, where **FracPassesMCF** denotes the fraction of the 10,000 generated molecules that pass the above-defined filters. We observe that the baseline models, `smiles-gpt-base` and `smiles-rnn-base`, attain filter passing rates of 52% and 53%, respectively. After fine-tuning with DPO, these rates improve significantly—by 79% and 65%, respectively. Moreover, these improvements come with little-to-no degradation in the other metrics.

Model	**FracValid**	**FracUnique**	**FracPredActive**	**IntDiv**
`smiles-rnn-chembl`	0.955	0.942	0.096	0.865
`smiles-rnn-EGFR-dpo`	0.951 (-0.4%)	0.871 (-7.5%)	0.602 ($+527\%$)	0.829 (-4.2%)
`smiles-gpt-chembl`	0.939	0.927	0.090	0.864
`smiles-gpt-EGFR-dpo`	0.871 (-7.2%)	0.763 (-17.7%)	0.790 ($+778\%$)	0.799 (-7.5%)

Table 2: **Results from EGFR activity fine-tuning experiments.** We observe that for both architectures, DPO fine-tuning significantly increases the fraction of generated molecules predicted active against the protein target. In this case, however, we observe some degradation in the other metrics.

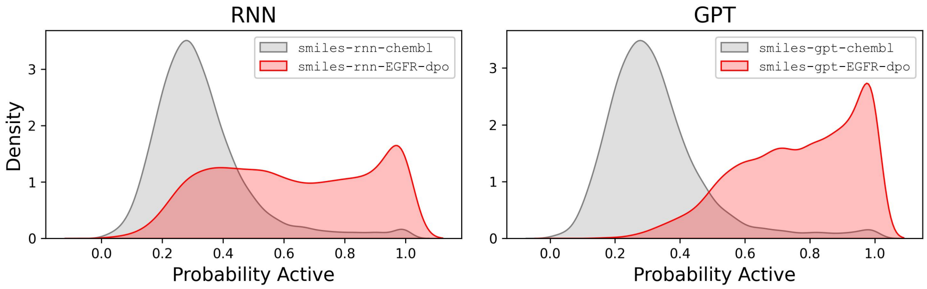

Figure 2: **Predicted probability of activity against EGFR for generated molecules.** After fine-tuning with DPO, the predicted probability of activity against EGFR for generated molecules is shifted significantly towards activity versus the baseline model.

3.2 Predicted bioactivity

In our next set of experiments, we desire to sample molecules with high predicted bioactivity against a given target. Specifically, we focus on the protein kinase EGFR, which is known to be of clinical relevance for a number of diseases, primarily cancer [LMDRT22]. Because the MOSES benchmark set doesn't contain kinase-inhibitor specific drugs, we first fine-tune the `smiles-rnn-base` and `smiles-gpt-base` models on a set of 115,000 kinase inhibitors extracted from the ChEMBL database, to obtain models which we call `smiles-rnn-chembl` and `smiles-gpt-chembl`, respectively. We then train a random forest binary classifier based on 3485 IC_{50} measurements also extracted from ChEMBL. Here we consider a molecule active against the target if its IC_{50} is less than 100nM, and inactive if its IC_{50} is greater than 500nM[1]. This results in 2568 active compounds, and 917 inactive. After holding out 15% of examples for testing, the classifier achieves 95% accuracy. Additional details can be found in Appendix A.3.

Our goal in this section is to sample molecules that are predicted to be active against the EGFR target. We quantify this with the metric **FracPredActive**, measuring the fraction of sampled molecules that are predicted as active against EGFR by the binary classifier. In Table 2, we observe that only approximately 9% of the sampled molecules from `smiles-rnn-chembl` and `smiles-gpt-chembl` are predicted to be active. We then perform DPO fine-tuning by sampling 100,000 SMILES strings from each model, and labeling them as `ACTIVE` or `INACTIVE` using the trained classifier. These examples are then used as positive/negative examples for DPO, resulting in further fine-tuned models `smiles-rnn-EGFR-dpo` and `smiles-gpt-EGFR-dpo`. After fine-tuning with DPO, we find significant improvements in predicted activity, increasing by 527% and 778% for the RNN and GPT models, respectively. However, unlike in the filtering experiments, we observe the improvement in predicted activity comes at the expense of some degradation across the other metrics. We hypothesize that this is due to the limited number of positive examples for this task compared to the filtering experiments (only $\approx 9\%$ of the fine-tuning data versus $\approx 50\%$ for the filtering task).

To visualize the shift in the distribution of predicted activity, in Figure 2 we plot the distributions of predicted probability of activity before and after fine-tuning. We observe that DPO effectively shifts the distribution towards higher activity for both architectures, using only synthetically generated data.

[1] For the sake of the classifier considered here, we exclude compounds with ambiguous activity status.

4 Conclusions

We have studied the use of DPO to tune molecular language models in concordance with chemist preferences. Across both experimental settings evaluated here, we find that fine-tuning with DPO is very effective at directing generation towards molecules with desired properties. Moreover, we find that training with DPO is straightforward and computationally low-cost.

Our preliminary investigations suggests many directions ripe for further study; we list a few of particular interest here.

- We observe in some (but not all) experiments that fine-tuning with DPO leads to lower diversity in generated molecules. We hypothesize that this may occur particularly when the labeled datasets do not contain sufficiently many positive examples. Are there ways to mitigate this issue?
- DPO requires that scores be effectively binarized into an ordering $y \succ y'$ (as we have done with bioactivity in Section 3.2). However, many scores relevant for molecular generation tasks are more naturally represented as a continuous scalar, and binarizing these scores may result in loss of important information. Can the DPO technique be generalized to allow for continuous labels?
- As a preliminary investigation, we focused on preference labels that can be computed automatically. However, the recent success of fine-tuning from human feedback in language models suggests manually-curated labels from chemists may yield promising results.

References

[Kar23] Andrej Karpathy. minGPT. https://github.com/karpathy/minGPT, 2023.

[KHN+20] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, oct 2020.

[LMDRT22] Elena Levantini, Giorgia Maroni, Marzia Del Re, and Daniel G Tenen. EGFR signaling pathway as therapeutic target in human cancers. *Seminars in Cancer Biology*, 85:253–275, 2022.

[MGB+18] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R Leach. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47(D1):D930–D940, 11 2018.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PZSL+20] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alan Aspuru-Guzik, and Alex Zhavoronkov. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Frontiers in Pharmacology*, 2020.

[RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[RSM+23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[SKTW17] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating focussed molecule libraries for drug discovery with recurrent neural networks. *CoRR*, abs/1701.01329, 2017.

[Wei88] David Weininger. SMILES, a chemical language and information system. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 02 1988.

[YZY+17] Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. ChemTS: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials*, 18(1):972–976, nov 2017.

	precision	*recall*	*f1-score*	*support*
inactive	0.89	0.92	0.90	126
active	0.97	0.96	0.97	397

Table 3: Test set performance of EGFR binary classifier.

A Additional experimental details

A.1 Training details.

Here we report details of our training procedure, both for pre-training and for fine-tuning.

Architecture details. The RNN model consists of 3 LSTM layers using an embedding dimension of 768, followed by a linear layer mapping to the vocabulary size. The implementation is based on the one provided in the MOSES benchmark suite [PZSL$^+$20]. The GPT model consists of 8 transformer blocks, each with 8 attention heads and an embedding dimension of 256, followed by a final classification head of the dimension of the vocabulary size. The implementation is closely based on minGPT [Kar23].

Pre-training. For pre-training the RNN model on MOSES, we use the Adam optimizer with an initial learning rate of 1e-3, decayed by a factor of 0.1 every 10 epochs, for a total of 80 epochs. For the GPT model, we using the AdamW optimizer with an initial learning rate of 6e-4, cycled using a cosine annealing schedule for 40 epochs. The final models are selected based on the validation loss on the MOSES test set.

Unsupervised fine-tuning. During the unsupervised fine-tuning step (used for the `smiles-rnn-chembl` and `smiles-gpt-chembl` models), we use the same optimizers and number of epochs as in the pre-training stage, except for the GPT model we disable learning rate decay. The final models are selected based on the validation loss on a hold-out set of 15,000 SMILES strings.

DPO. Our implementation of DPO closely mirrors the original implementation provided by the authors in [RSM$^+$23]. We use the RMSProp optimizer with a Lambda learning rate scheduler (as implemented in PyTorch [PGC$^+$17]) for 80 epochs.

A.2 Molecular filtering

SMARTS filters. The SMARTS filters we use are included for reference in Table 4, along with a column "count" indicating the minimum number of times the pattern can be observed for the molecule to fail the filters.

A.3 EGFR activity optimization

Binary classifier. The binary classifier is trained using the default parameters of the `RandomForestClassifier` class in `scikit-learn` [PVG$^+$11]. To featurize molecules, we use 1024-bit Morgan fingerprints, which we compute using the `RDKit` package in Python. As we mention in Section 3.2, we hold out 15% of our data for testing, resulting in 523 testing examples. On this hold-out set, the model achieves a raw accuracy of 95%; we report additional performance metrics in Table 3. We remark that the classifier used here is largely for demonstration, and that, depending on the scenario of interest, better predictors may be available.

A.4 Scaffold conditioning

B Model samples

Here we include some samples from the molecular language models referenced throughout the paper.

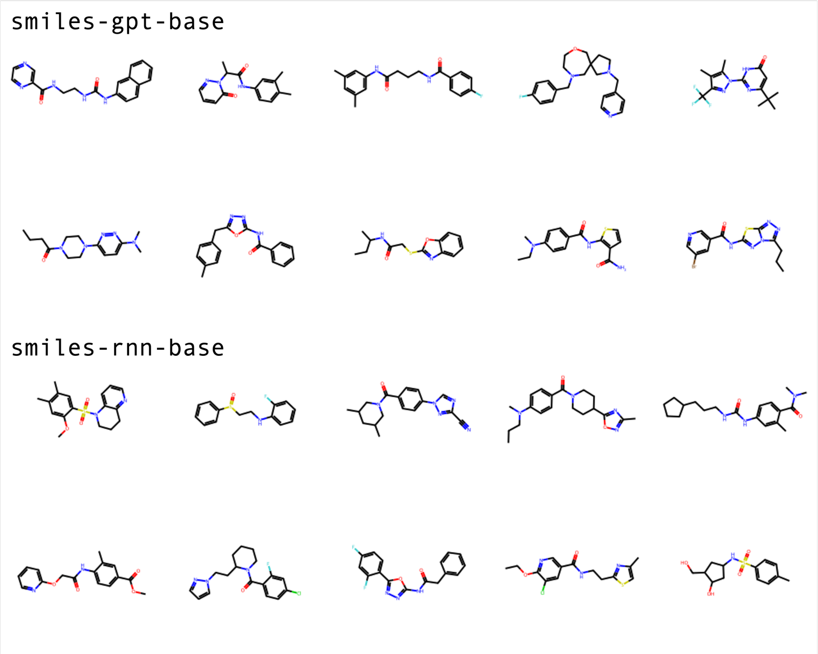

Figure 3: Samples from baseline GPT and RNN models, pre-trained on MOSES.

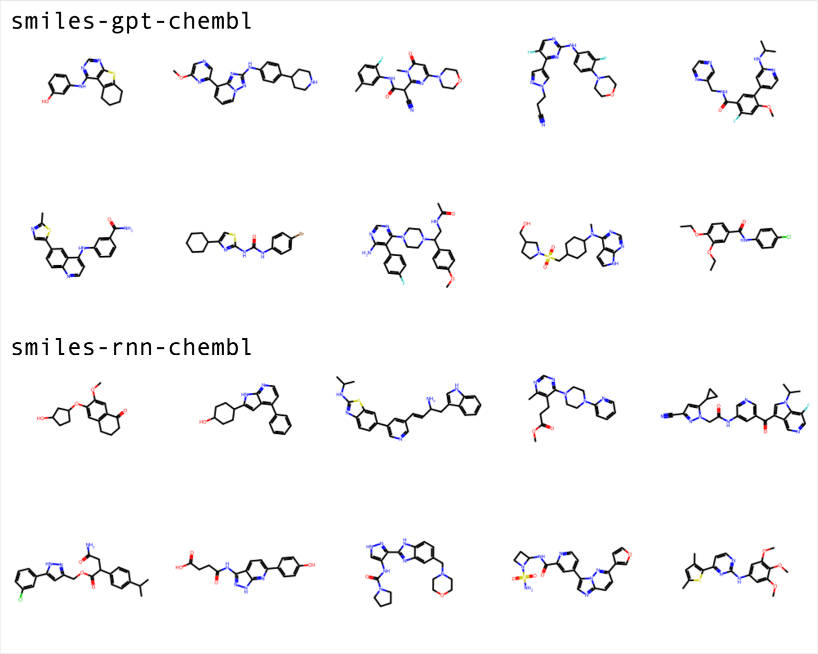

Figure 4: Samples from ChEMBL GPT and RNN models, pre-trained on MOSES, and fine-tuned on ChEMBL set of kinase inhibitors.

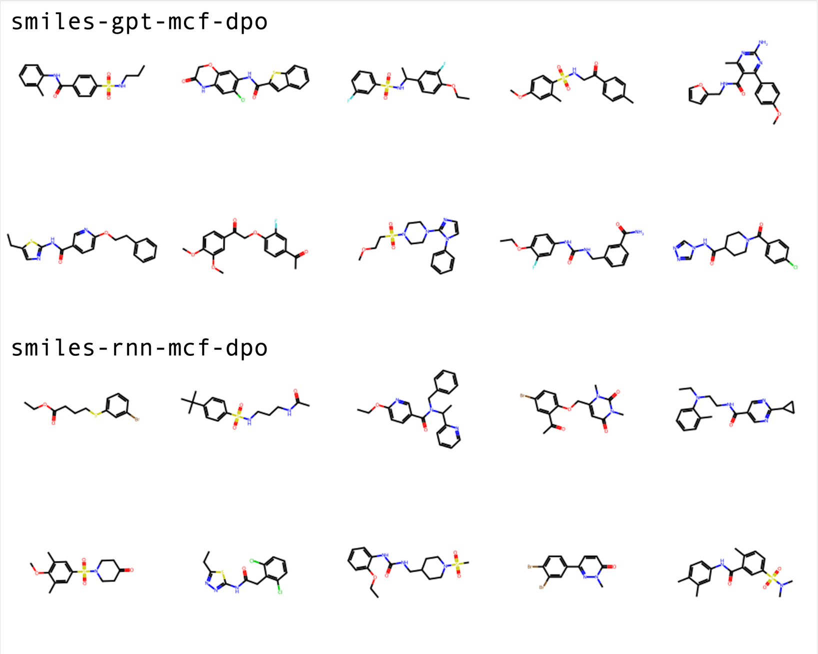

Figure 5: Samples from ChEMBL GPT and RNN models, pre-trained on MOSES, and fine-tuned using DPO using medicinal chemist filtering, as described in Section 3.1.

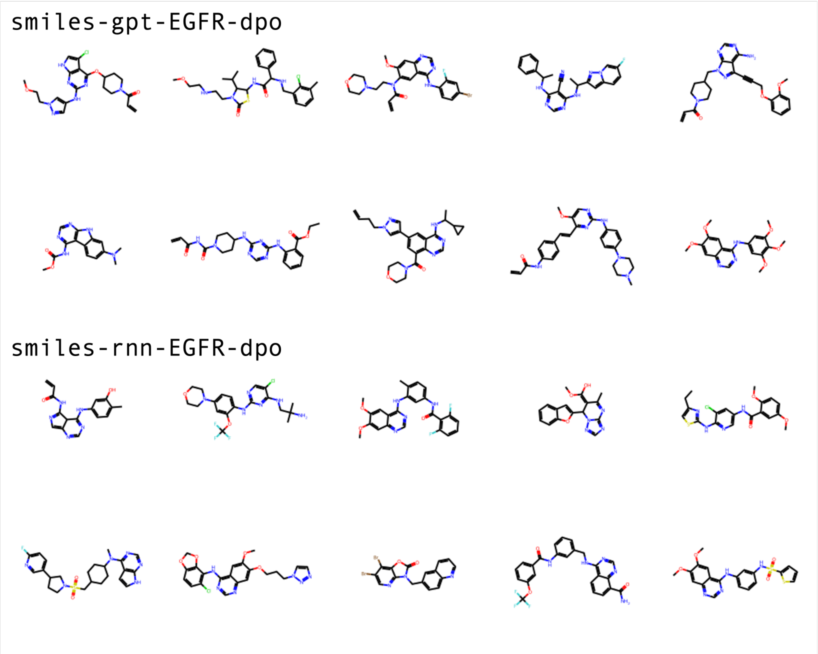

Figure 6: Samples from ChEMBL GPT and RNN models, pre-trained on MOSES, fine-tuned on ChEMBL, and further fine-tuned using DPO using predicted activity against EGFR, as described in Section 3.2.

SMARTS	Count
*-N=[N+]=[N-]	1
BrccBr	1
C(-C#N)(-C#N)	1
C(-C#N)C(-C#N)	1
C1(=O)[C,c]:[C,c]C(=O)[C,c][C,c]1	1
C14 * * * * C 1 * * C2 C3 * * * C 3 * * C 2 4	1
C 1 C C 2 C C [#6] [#6] C 2 C C 1	1
N=[N;!R]	1
NC#[NX1]	1
S#N	1
S(=[N])	1
S[N;!R][N,n]	1
[!$(C=O)][NX3]-[OH]	1
[!$(C=O)][n,N]-[OH]	1
[#6;!R][CX3;$([!R][#6])](=[O])[#6;!R][#6](=[O])	1
[#6]1=[#6]-[#6]-1	1
[#6]1=[#6]-[#7,#6]=[#6]-1	1
[#6][#16][#16][#6]	1
[#7,S][C;!R](=S)N	1
[#7] [#6]([Cl,Br])([#7])	1
[#8,#7,#16]-[CH,CH2;!R]-[#8,#7,#16][CX4,c;!$(C(=O))]	1
[#8;!$(O[CH2][CH2][O,N])]	6
[$([#7+][OX1-]),$([#7v5]=[OX1]);!$([#7]([O]) [O]);!$([#7]=[#7])]	1
[$([CX3]=[CX3]([H,!C])[CX2]#[NX1]);!R]	1
[$([HO]-N-C(=O))]	1
[$([N]=[CX3]c);!R]	1
[*r5R1]1[CR2]2[CR1][CR1][CR1][CR1][CR2]2[*r5R1][*r5R1]1	1
[B;!$([B][OX1])]	1
[B]([OX2]C)([OX2][C])	1
[C,O][N+,N](=[OX1])	1
[C;H1](=O)	1
[CH2]=[CH]-[N,O,S]	1
[CH]([OH])[CH]([OH])[CH]([CH2][OH])	1
[CH]([OH])[CH]([OH])[CH]([OH])	1
[CH]=[CH]-N(=O)(O)	1
[C]!@[C]!@[C]!@[C]!@[C]	1
[C](=[O,S])[CH2][Br,Cl,I]	1
[C]([Br,Cl])([Br,Cl])	1
[C]=[#16+][O-]	1
[C]=[CH]-[O,NH,NH2]	1
[Cl,Br,I]	5
[I,Se,se,Si,P]	1
[I,Se,se,Si]	1
[N!$(N-O)]=O	1
[N+,N](=[OX1])[OX1]	2
[N+,n+;!$([N+,N](=[OX1])[OX1])][C,c]	1
[N+]#[C-]	1
[N+](=O)[O-]	1
[N,!R][C](=[N,S])	1
[N,O,S][F,Cl,Br]	1
[N,O]([SX2,SX3])	1
[N,S,O,Br][C]=[#6]	1

[N,S][c,C]([SH,S-1])[#7]	1
[N,n](-[O])	1
[N,n]([O])	1
[N;!$(N=O)]-O	1
[N;!$(N=O)]-[OH]	1
[NX3;!R][NX2]=[*]	1
[NX3][CH2][CH2][Cl,Br]	1
[NX3][NX3;!R]	1
[N]-[NN;R]	1
[N]-[n]	1
[O,N,S,Cl,F,Br][CH,CH2][O,N,S,Cl,Br][CX4,c;!$(C(=O))]	1
[O,N,S][CH,CH2;!R][O,N,S][CX4,c;!$(C(=O))]	1
[O][O]	1
[PH]([c,C,N])	1
[PX4D4]([OX1,N])([O,N])	1
[P](=O)([O,N])	1
[P]([O,N,C,S])	1
[S+1;!#16#8]	1
[S,C;+1]	1
[S,C](=[O,S])[F,Br,Cl,I]	1
[S;!$(S[OX1-1])]	3
[S;D4](=[OX1])(=[OX1])[O;H1,O-]	1
[SH]	1
[S][S]	1
[Se][Se]	1
[c]1C(=O)[NX3][CH2][c]1	1
[c]1C(=O)[NX3][C](=O)[c]1	1
[c]1[c][s,n,o,c][s,n,o,c]1	1
[n+]-[O-]	1
[o,O;+1]	1
[r3O,r3N,r3S]	1
[r7]	1
[r8-]	1
a-[C]=[CH2]	1
a[CH,CH2][Cl,Br]	1
a[CH2][CH2;!R][CH2][CH2]C*	1
a[C]=[CH2]	1
a[OH]	3
c[Br]n	1

Table 4: SMARTS patterns used for filtering in Section 3.1.