Data Augmentation and Pretraining for Template-Based Retrosynthetic Prediction in Computer-Aided Synthesis Planning

As Published: 10.1021/ACS.JCIM.0C00403

Publisher: American Chemical Society (ACS)

Persistent URL: https://hdl.handle.net/1721.1/134636

Version: Original manuscript: author's manuscript prior to formal peer review

Data Augmentation and Pretraining for Template-Based Retrosynthetic Prediction in Computer-Aided Synthesis Planning

Michael E. Fortunato,[*,†] Connor W. Coley,[†] Brian C. Barnes,[‡] and Klavs F. Jensen[†]

†*Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA*

‡*Detonation Science and Modeling Branch, CCDC Army Research Laboratory, Aberdeen Proving Ground, MD, USA*

E-mail: mef231@mit.edu

Abstract

This work presents efforts to augment the performance of data-driven machine learning algorithms for reaction template recommendation used in computer-aided synthesis planning software. Often, machine learning models designed to perform the task of prioritizing reaction templates or molecular transformations are focused on reporting high accuracy metrics for the one-to-one mapping of product molecules in reaction databases to the template extracted from the recorded reaction. The available templates that get selected for inclusion in these machine learning models have been previously limited to those that appear frequently in the reaction databases and exclude potentially useful transformations. By augmenting open-access datasets of organic reactions with artificially calculated template applicability and pretraining a template

relevance neural network on this augmented applicability dataset, we report an increase in the template applicability recall and an increase in the diversity of predicted precursors. The augmentation and pretraining effectively teaches the neural network an increased set of templates that could theoretically lead to successful reactions for a given target. Even on a small dataset of well curated reactions, the data augmentation and pretraining methods resulted in an increase in top-1 accuracy, especially for rare templates, indicating these strategies can be very useful for small datasets.

Introduction

The task of retrosynthetic analysis can be defined as the recursive simplification of a desired target molecule into less complex starting materials. In contrast to the reaction prediction task in the forward direction, where a defined set of reaction conditions should lead to a single distribution of product molecules, a single-step retrosynthetic prediction takes the form of a one-to-many mapping, where the target could theoretically be made through a variety of different individual reaction steps. Retrosynthesis has seen increased attention from the data science and cheminformatics communities recently with a number of machine learning efforts leveraging reaction templates or rules,[1–4] techniques adapted from natural language processing,[5–8] and graph based models.[9,10] However, only the template and rule-based methods are capable of making a connection from the prediction directly back to the source of the template or rule, which is most likely a reaction that was successfully performed in a laboratory. This ability to provide evidence and reasoning behind the prediction of a molecular transformation makes template-based methods an attractive choice for use in software tools designed for synthetic chemists, and guided the choice to pursue this method in this work.

The template-based retrosynthetic methods apply a set of pre-defined molecular transformations to a target molecule. The template rules themselves can originate from hand written rules from expert chemists or reaction databases, however this work employs the

automatic extraction of reaction template transformations from reaction databases using the open-source template extraction code in the rdchiral python package.[11] Because the application of a reaction template to a target molecule will be performed multiple times at each retrosynthetic step, and each step is performed recursively until starting materials have been sufficiently simplified, there is a need to keep the single-step retrosynthetic prediction as computationally efficient as possible. In the extreme, every available template transformation could be applied to every target during retrosynthesis, however this quickly explodes into an intractable number of template applications for retrosynthetic pathways of moderate length. Each template application only takes $\sim 100 \mu s$ on average, however it is common to have $O(10^5)$ reaction templates.[12–14] Using brute-force template application at the time of inference, although each individual prediction would require 0.1–1 second, an exhaustive, but relatively shallow, tree search using a maximum depth of 5 reactions and maximum branching of 10 would require $\sim 10^5$ predictions and this would be impractical to perform in real-time. Machine learning models as template prioritization algorithms offer a computationally efficient way to select a subset of all available templates that are perceived to be most useful or relevant.[2] These machine learning models are trained as multi-class classification models where each class represents a template. The scores output per class from the model offer a way to rank and prioritize templates. Using these machine learning informed predictions can decrease the number of template applications to the top ~ 100 templates instead of $\sim 100,000$, enabling a full retrosynthetic tree search to complete within seconds or minutes instead of hours or days.

The goals of a template recommender or prioritizer can be formulated as follows: first, all templates that could apply to the target molecule should be selected from the superset of available templates, and second, these templates should be prioritized in such a way that the most useful and most likely to be successful get ranked highest. The criteria for "most useful" can be difficult to universally define, however reaction databases curated from experiments successfully carried out in a laboratory contain information about what types

of transformations are most likely to be successful. The current state of the art template prioritizing machine learning models are neural networks that are trained on these types of reaction databases. Training exclusively on this data can successfully achieve the second goal stated above (prioritization of successful candidate reactions), but can often miss out on potentially useful transformations defined in the reaction database due to lack of coverage (incomplete mappings from product molecules to available templates), and rare data (few training examples for certain templates). In fact, it is common practice that templates with very few examples are excluded from the set of templates used for model training, due to the inability for neural networks to learn meaningful relationships between their input and these rare templates.[2,15] While it may be possible to augment training examples with more experimental data (increasing the ability to perform goal number two described above), or design models tasked to perform very specific types of retrosynthetic predictions (such as ring breaking[16]), it is also possible and perhaps easier to augment the information about which templates could theoretically be relevant *in silico* (increasing the ability to perform goal number one described above). The remainder of this manuscript describes methods employed to augment open-access reaction databases with computer generated reaction examples to train a template applicability neural network, and use this to pretrain a template relevance prioritizing model. In addition to evaluating the ability of these models to learn the exact data encoded in the reaction databases, other useful metrics such as template applicability recall and chemical diversity are discussed.

Methods

Data sources and preparation

The two datasets of reactions used in this study were taken from the open-access set of reactions extracted from U.S. patent literature (USPTO) by Lowe.[17] The first set of reactions, herein referred to as the *USPTO-Sm* dataset, represents a small subset of 50,000 pharma-

ceutically relevant reactions curated by Schneider et al.[18] from the entire USPTO set and is often used to benchmark retrosynthetic models. The second set of reactions, herein referred to as the *USPTO-Lg* dataset, includes the entire USPTO set of reactions (approximately 2 million reactions). In both cases, as described in more detail below, only reactions from which a valid reaction template could be extracted were retained, resulting in dataset sizes of 32,099 and 669,683 reactions, respectively.

For each single-product reaction, a retrosynthetic template was extracted from the atom-mapped SMILES string[19] using a modified version of the template extraction code included in the rdchiral Python package.[11] The modification, included in the open source distribution of code used in this work, skipped reactions with large amounts of symmetry that led to difficulties extracting templates. Most template extractions were completed in a fraction of a second, while a few reaction examples did not complete successfully in several minutes. It is likely that this symmetry filter skipped potentially valid reaction examples, however it helped ensure successful extraction and yielded high quality templates. Prior to extraction, any molecules that did not contribute a heavy atom to the product (determined by the provided atom-mapping) were moved to the position of "spectator" so that they did not appear in the template. After extraction, the template was applied to the product molecule from the reaction example, and if the resulting precursor was not found in the originally recorded reactants the reaction example was discarded. This validity check was performed to ensure that the templates being extracted from the reaction database accurately defined the transformation implied by the recorded reaction. However, it is possible that this validity check was overly conservative and filtered out potentially valid reactions and reaction templates. Invalid SMILES strings, multi-product reactions, duplicate reactant/product pairs, unsuccessful template extractions, and invalid templates, were the main contributing factors to the significant decrease in reactions from the 50,000 and nearly two million starting reactions to the final 32,099, and 669,683 filtered reactions for the two datasets, respectively. Retrosynthetic reaction templates were then collapsed into a unique set, grouped by their

un-mapped SMARTS strings, and each unique template was given an identifier (this integer value would ultimately map to the class in the context of the machine learning model). The grouping was performed on the un-mapped SMARTS string because it was possible that two templates ended up representing the same chemical transformation with different atom map numbers.

The reaction examples were split into train, validation, and test sets using a stratified split as described in previous work.[20] In summary, the split helped to ensure that classes (templates) were more equally represented in the training and test sets. In cases where class representation is more evenly distributed across examples, this would not play a major role. However, because many templates that were extracted from the USPTO datasets ended up having very few precedent reactions (sometimes only 1 or 2, particularly with the degree of specificity defined by rdchiral), it would not be uncommon when using a random split that all precedent examples for a given template would have ended up in the train and/or validation set. In this case, a quantitative evaluation of the resulting test set would have omitted the possible inability for the machine learning model to properly recommend the rare template. The stratified split followed an 80/10/10% split, by class, where enough data was available, but for template classes that had fewer than 10 examples, 1 randomly chosen reaction precedent was forced into the test set, 1 randomly chosen reaction precedent was forced into the validation set, and the remainder were put in the training set. If only 2 examples were available, 1 was placed in the test set and 1 was placed in the training set. Lastly, if only 1 example was available, it was randomly placed in one of the train, test, and validation sets with an 80/10/10% probability. The *USPTO-Sm* dataset was divided into train, validation, and test sets with sizes 23,777 (74.1 %), 3,570 (11.1 %), and 4,751 (14.8 %) respectively, and contained 7,765 template classes. The *USPTO-Lg* dataset was divided into train, validation, and test sets with sizes of 498,571 (74.4 %), 73,453 (11.0 %), and 97,659 (14.6 %) respectively, and contained 186,822 template classes. The same data splits were used for the entire pretraining and fine-tuning workflow.

For all neural network models described below, input molecules were transformed into input features using a Morgan fingerprint[21] bit vector with a fixed-length folding of 2048 bits and a radius of 2 using RDKit[22] including the *useChirality* flag to account for chiral centers. The models were designed to learn a mapping from the fingerprint representation of product molecules to each template extracted from the reaction database that described the transformation to make the product. This follows the work of Segler and Waller[2] where they trained a neural network to recommend reaction templates given the product molecule fingerprint from the reaction which generated that template. The work described here differs in three regards: 1) A single wide hidden layer was selected for the network architecture (see supporting information for details of the hyperparameter scan that led to this selection) 2) The initial weights for the neural network model presented here were not assigned randomly; instead, they were determined by pretraining a neural network with the same architecture on generalized template applicability (described below) and 3) All reaction templates that were successfully extracted were included as classes for the model, regardless of how few precedents each template had. A model similar to that of Segler and Waller (with randomly initialized weights) was also trained, herein referred to as the *baseline* model, to compare to the model pretrained on the augmented dataset, herein referred to as the *pretrained* model.

Template applicability was pre-computed using all product molecules from the reaction dataset in a pairwise manner with all templates as described by Fortunato et al.[20] The RDKit python package[22] was used to brute-force determine the result of a template application for all templates extracted from each dataset on each product molecule present in the dataset. Because of the pairwise nature of the algorithm, the computational cost unfortunately scales with *number of templates*number of molecules*, and for large datasets this can become quite computationally expensive. However, because each individual template application is independent, this can be easily parallelized to take advantage of high performance computing resources. For example, the template application procedure for the *USPTO-Lg* dataset required ∼3,500 CPU hours (∼145 days) but was completed in just over 1 hour in

walltime by highly parallelizing the workflow. The template applicability was determined solely through matching to the template SMARTS string, equivalent to identification of the expected subgraph pattern in the product molecule. This procedure generated additional mappings from product molecules to theoretically relevant templates that were not present in the original dataset. A neural network was then trained to learn these, now one-to-many instead of one-to-one, mappings as a multi-class, multi-label classifier. The resulting weights from this neural network trained on augmented reaction data were used as the initial parameterization before fine-tuning a template relevance recommending neural network, as described previously. A schematic of the baseline template relevance training, applicability pre-training and template relevance fine-tuning is shown in Figure 1.

In brief, the workflow attempts to first teach the neural network which of the entire set of templates could possibly generate precursors, and then the neural network continues to learn which templates are the most relevant, informed by the set of reactions that exist in the reaction databases. Although this initial subgraph isomorphism test could be determined at the time of inference, training a neural network significantly decreases the computational cost for the specific substructures (templates) of interest.

The process follows similarly to how a trained synthetic chemist may approach the problem. First they would, likely subconsciously, eliminate the possibility of using known reactions that do not generate the substructures present in the target molecule. They would then select a few candidate reactions which they believe to be the most likely to be successful from their own knowledge or past experience. In some cases, the chemist may be aware of a *possible* reaction, but may not have performed it as many times in the laboratory. They may then prioritize reactions with which they have experienced success (analogous to reactions that exist in the reaction database), before they attempt a reaction which they think is theoretically possible (analogous to templates deemed as *applicable* during the template applicability process step above).

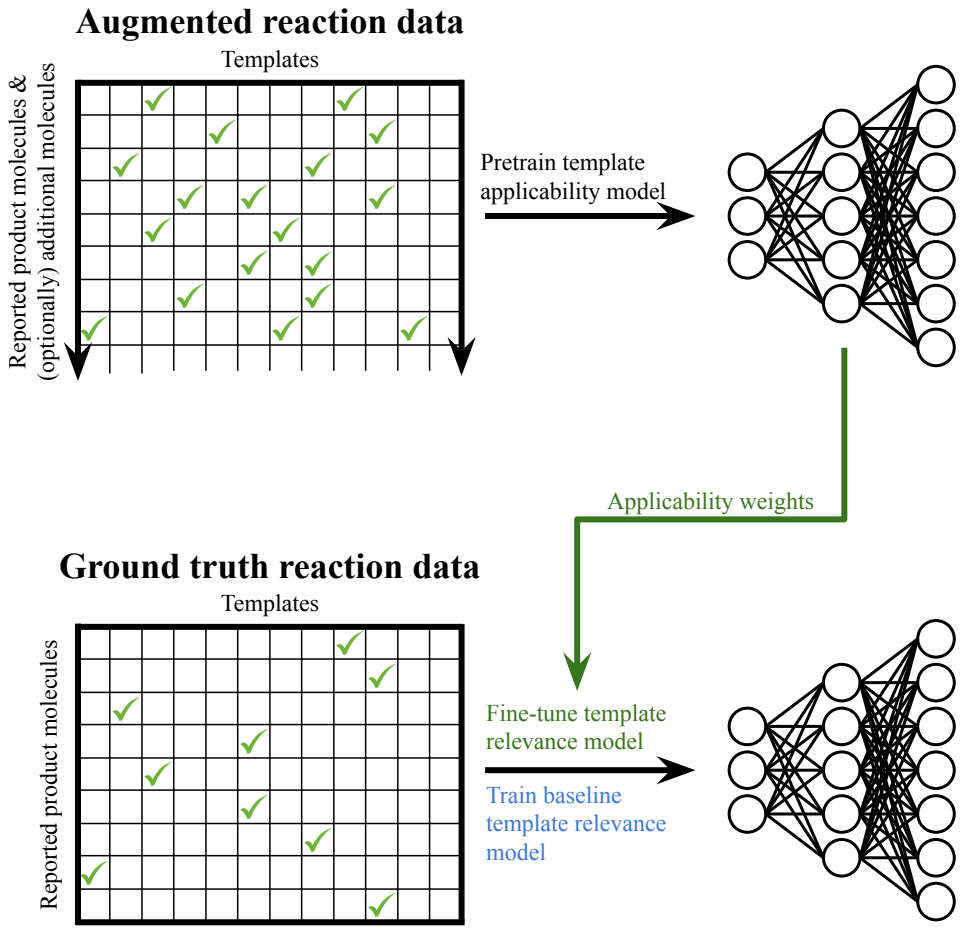

Figure 1: Schematic representing the applicability pretraining and ground truth fine-tuning workflows. First, a model was trained on an augmented set of reaction data, containing both ground truth and computer generated reactions. The weights from this model, which encode generalized template applicability, were then used to initialize a model with the same architecture, and fine-tuning of the model was performed by further training on only the ground truth examples. The baseline model, in comparison, follows the same exact training as the fine-tuing stage, but randomly initializes the network weights instead. The arrows pointing downwards in the augmented reaction data section indicate that the augmentation could be extended by collecting a larger set of example molecules instead of using only the set of product molecules, as was done in this work.

Neural Network Model Details

As described previously, the input to each model was a Morgan fingerprint featurized representation of molecules. This input ultimately took the form of a bit vector (where elements of the vector were either 1 or 0) of length 2048. The output of each model was a vector with a size equal to the number of templates for each datasets: 7,765 and 186,822 for *USPTO-Sm* and *USPTO-Lg*, respectively. Both the baseline model (trained only on ground truth one-to-one mappings from the original datasets) and the applicability model (trained on the augmented, one-to-many molecule-template mappings) were constructed to have the same architecture so that the weights learned from the applicability pretraining could be used to initialize the model for fine-tuning on the original ground truth mappings. A matrix of hyperparameters were scanned, varying the number of fully connected hidden layers {1, 2, 3}, the hidden layer size {1000, 2000, 3000, 4000}, and the number of highway layers[23] {0, 1, 3, 5}, which led to the conclusion that a single, wide, fully connected hidden layer with 0 highway layers achieved the best model performance (see supporting information for details). For each dataset, the size of the hidden layer was limited by the memory available during training, however, in general, the wider the layer the better the performance. Importantly, due to the very large output size of the *USPTO-Lg* dataset, the hidden layer size was limited significantly in comparison to the model trained on the *USPTO-Sm* dataset. Ultimately, the best performing, accessible model architectures were a single hidden layer of sizes 4000 and 2000 for models trained on *USPTO-Sm* and *USPTO-Lg* datasets, respectively. For all models, a ReLU activation was applied at the hidden layer, and a dropout layer[24] with a dropout rate of 0.2 was included following the hidden layer to help prevent over-fitting. All models used the Adam optimizer[25] with a learning rate of 0.001, and optimized crossentropy loss. The contribution to the loss function for each training example was weighted by the inverse template class popularity (number of precedent reaction examples) to help combat the class imbalance that existed, even following data augmentation. The activation functions applied at the output layer varied by model type corresponding to the one-to-one vs one-to-many

mappings the model was trained to learn: the baseline model used a softmax activation and the applicability model used a sigmoid activation. During fine-tuning, because the model was again learning a one-to-one mapping, a softmax activation was applied at the output layer. Models were constructed and trained using TensorFlow[26] and Keras.[27]

The code used for all data preparation, model training, and model evaluation is open-source and can be found online at: https://gitlab.com/mefortunato/template-relevance

Results and Discussion

Pretraining with Generalized Template Applicability

During the pretraining stage, the model was trained on datasets describing one-to-many mappings from product molecules to all templates that were determined to be applicable through brute-force template application. Recall and precision accuracy metrics were used to evaluate the ability of the model to learn this data. Recall measured the ability for the model to recognize all of the applicable templates, while precision measured the ability for the model to limit false positives, or to recommend *only* applicable templates. This model is trained to effectively learn sub-graph isomorphism for a specific set of sub-graphs, and since this can be calculated directly, it is expected that the model should be able to learn the data easily, given sufficient training data. However, the data-augmentation process can still be improved to 1) increase template popularity more evenly across templates, and 2) increase the augmentation coverage so all templates have many augmented examples. Figure 2 shows template popularity for the *USPTO-Sm* dataset before and after data augmentation. The augmentation of the *USPTO-Lg* dataset followed a similar trend (see Figure S6 in supporting information), where the augmentation shifted many templates from "rare" to "popular", however the coverage was not complete and rare templates still remained following augmentation.

Figure 3 shows the validation recall and precision during pretraining on the augmented

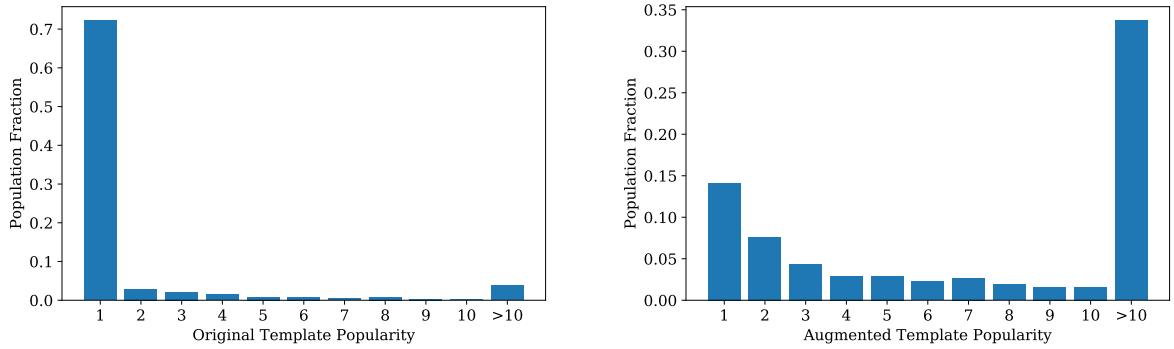

Figure 2: Template popularity for the *USPTO-Sm* dataset before and after data augmentation. The augmentation was successful for many templates, however a fraction still remained under represented.

template applicability dataset (left) and the validation crossentropy loss during training of the baseline model and the fine-tuning stage following pretraining (right). It is of note that the baseline training and fine-tuning training were performed on the same exact dataset (the one-to-one mapping of ground truth reactions), with the only difference being the initial weights of each model. In the case of the baseline model the weights were randomly initialized, whereas during the fine-tuning stage, the weights were initialized using the weights from pretraining on template applicability.

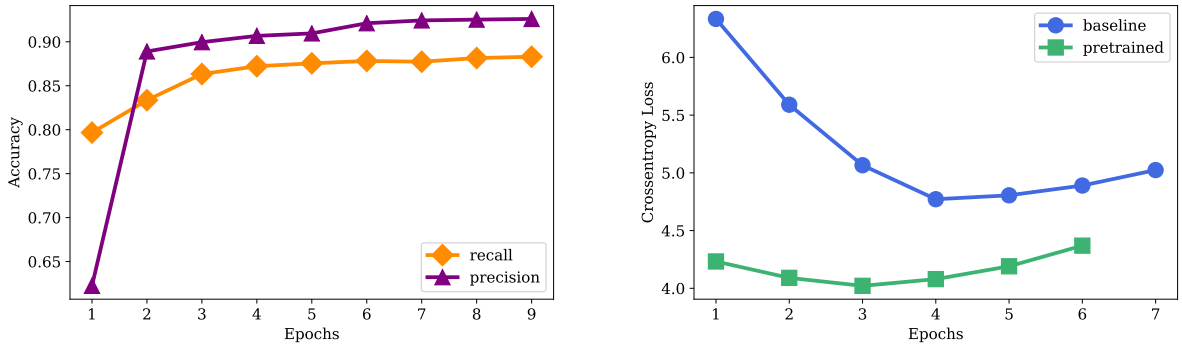

Figure 3: Validation recall and precision during pretraining on the augmented *USPTO-Sm* template applicability dataset (left) and validation crossentropy loss during training of the baseline model and the fine-tuning stage following pretraining for the *USPTO-Sm* dataset (right).

Top-k Template Applicability Recall

The goal of pretraining was to allow the model to learn generalized template applicability. After fine-tuning the model by training on the original ground truth values, it was important to evaluate whether the information learned during pretraining was retained. Top-k template applicability recall attempts to quantify this information retention and is defined as the fraction of all applicable templates that the model ranked in the top-k predicted templates. This metric is bound at 0.0 when k is 0 and 1.0 when k is equal to the total number of template classes. As k increases, the rate at which top-k template applicability recall increases indicates the ability of the model to recommend useful templates with higher rank. In other words, successful pretraining would result in higher template applicability recall for a given k value. Figure 4 shows the top-k applicability recall for baseline and pretrained models for the *USPTO-Sm* and *USPTO-Lg* test sets. In both cases, the increase in applicability recall when pretraining was clear. In practice, retrosynthetic tree search algorithms that make use of reaction templates often search through the top 100-1000 templates, for which the pretrained model would generate more predicted precursors to identify potential reaction pathways.

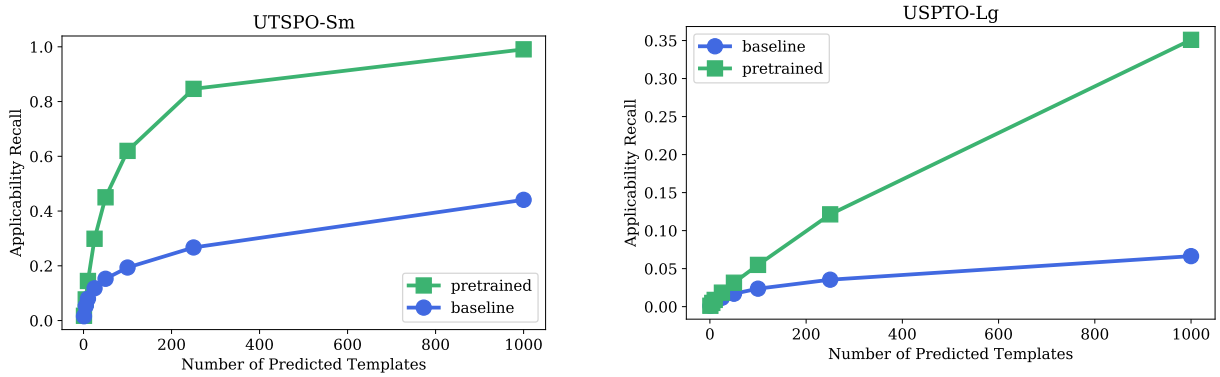

Figure 4: Top-k template applicability recall for the *USPTO-Sm* and *USPTO-Lg* test sets. This metric reports the fraction of all applicable templates that the model recommended in the top-k predicted templates.

See Figure S7 for a related template applicability precision metric for the baseine and pretrained models. As more templates were considered, the precision decreased for both

models, however the precision for the pretrained model was greater than that of the baseline model. In addition to the increase in the number of applicable templates, and therefore predicted precursors, the pool of precursors generated from the pretrained model was, on average, more diverse than that generated from the baseline model. See Figure S8 and associated discussion in the supporting information for more details.

Top-K Accuracy

The top-k accuracy for a classification model is defined as the fraction of examples for which the ground truth appears in the top-k predictions. For example, top-1 accuracy reports the fraction of examples for which the top predicted template was the ground truth. Top-100 accuracy represents the fraction of examples for which the ground truth appeared *anywhere* in the top 100 predictions. In the context of a template prioritizing model, the classes represent templates, and being top-1 accurate meant the model correctly identified the ground truth template for a given product molecule as the best recommendation. In other words, given a product molecule from the reaction database, the model correctly predicted the template that was extracted from the reaction that was recorded to produce the product molecule. This metric evaluates the ability of the model to "learn the dataset" on which it was trained, but, as has been discussed recently,[28] is not the only useful metric to evaluate retrosynthetic models. Figure 5 shows the top-k accuracy of baseline and pretrained models for the *USPTO-Sm* and *USPTO-Lg* datasets. Pretraining increased accuracy on the small data set across the k values examined; however, increases in accuracy were only seen for larger k values on the larger data set. Because the augmented information used during pretraining did not include information about the ground truth, it was somewhat surprising to observe any increase in top-k accuracy for the pretrained models. The boost in accuracy may be explained by an effective "process of elimination", where the information gained during pretraining did not necessarily help the model learn what the correct template should have been, but it could have helped the model learn what templates could not have possibly been the correct

answer. For smaller datasets, where there are fewer templates and therefore fewer potentially applicable templates, this "process of elimination" can help explain the more significant increase in accuracy.

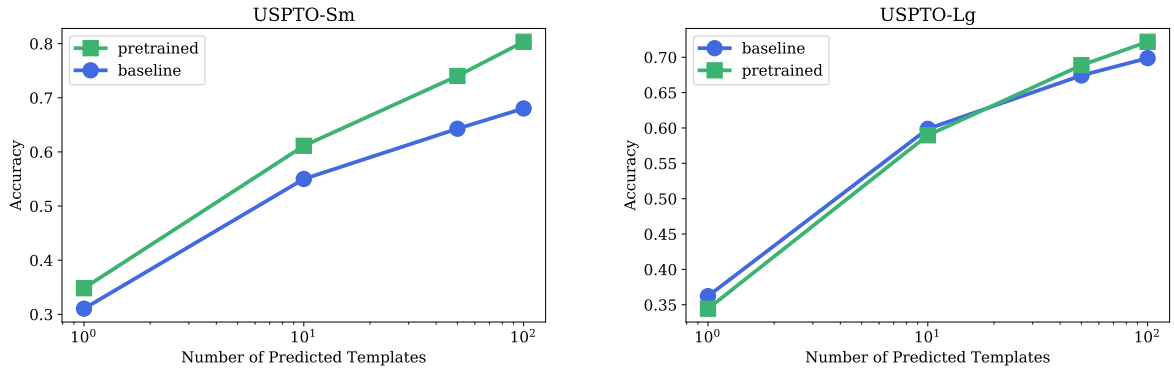

Figure 5: Top-k accuracy for *USPTO-Sm* and *USPTO-Lg* datasets for increasing k values.

Figure 6 shows the breakdown of top-100 accuracy for each data set by the original, unaugmented template popularity. As expected, the accuracy was lower for those templates that had fewer examples. However, pretraining the model on generalized template applicability increased the accuracy for the more rare templates. The effect was once again more significant for the smaller data set, presumably due to the "process of elimination" effect described previously. It was likely that very rare templates represented very specific types of transformations, and if the model was better able to learn when this template could theoretically have been applied it would have been better able to recover the ground truth even without seeing many examples during fine-tuning.

Reciprocal Rank

Figure 7 shows the distribution of reciprocal ranks of true templates from each test example in the *USPTO-Sm* and *USPTO-Lg* datasets. The baseline model ranked many true templates very poorly, whereas the distribution resulting from the pretrained model shows a shift to higher (better) reciprocal rank. The effect is very pronounced for test examples for which there was no precedent example in the ground truth training set, as shown in Figure 8. For

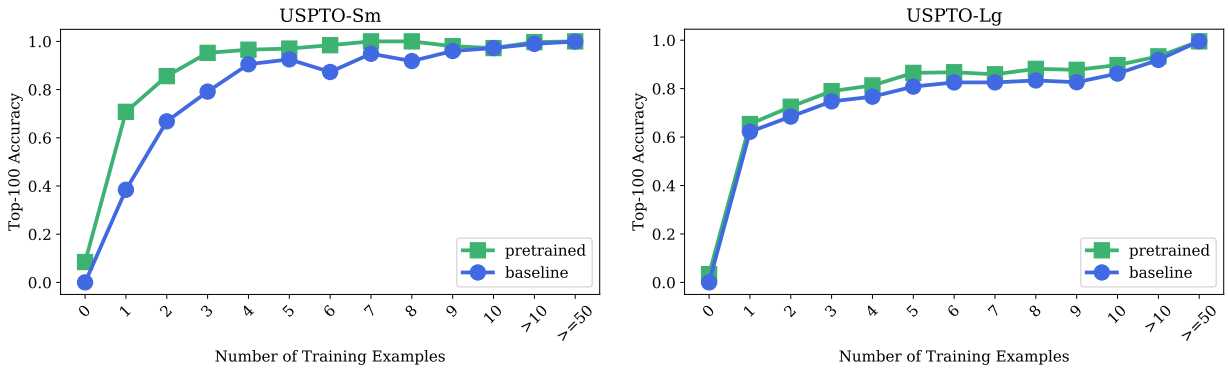

Figure 6: Breakdown of top-100 accuracy for *USPTO-Sm* and *USPTO-Lg* test sets by template popularity in the original datasets.

these test examples, there was only 1 ground truth example and that example only appeared in the test set. Therefore, there were 0 precedent examples during the baseline training and the fine-tuning stage in the pretraining workflow. As expected, for the baseline model, the distribution of reciprocal rank fell near the value of $1/$*number of templates*, because essentially no information was learned about these templates. However, as evidenced by the increase in reciprocal rank for the pretrained model, the information gained during pretraining helped to increase the rank of these ground truth templates.

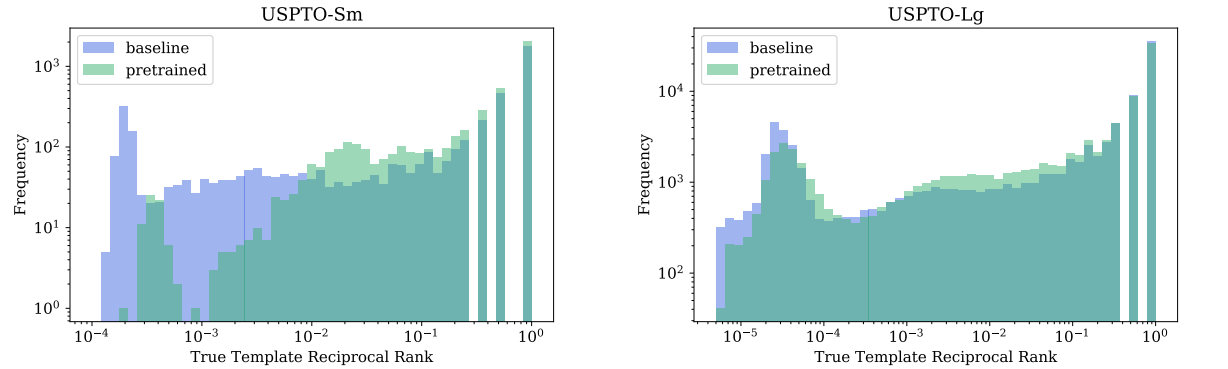

Figure 7: Distribution of reciprocal rank of the true template for all test examples in the *USPTO-Sm* and *USPTO-Lg* datasets.

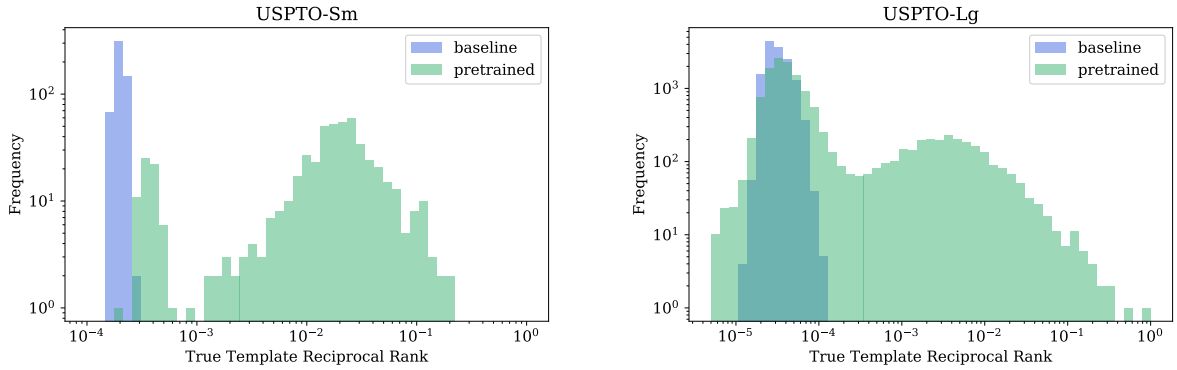

Figure 8: Distribution of reciprocal rank of the true template for very rare template test examples in the *USPTO-Sm* and *USPTO-Lg* datasets. These very rare templates had 0 examples in the baseline/fine-tuning training. However, augmentation and pretraining increased the distribution of reciprocal rank.

Chemical Examples

In addition to the quantitative "data-learning" evaluations described above, the baseline and pretrained models were qualitatively evaluated on two broad classes of molecules: drug-like small molecules and energetic materials. The drug-like small molecules were collected from the World Health Organization (WHO) essential medicines list for 2019,[29] and the energetic materials were curated from the "Chemistry and physics of energetic materials" book[30] and from a Naval Weapons Center Technical Report.[31] After filtering for common name resolution and resulting SMILES string validity, the test set sizes were 328 and 267 molecules for the WHO and energetic sets, respectively. There were many more molecules in the original WHO essential medicines list that were filtered out, however after achieving test sets of approximately the same size, the failed name resolution cases were not explored further. For each test molecule, four summary statistics were calculated: 1) the number of valid precursors generated by the top k predicted templates, 2) the average weighted template popularity for the templates that successfully generated precursors; the average weight was calculated as $template\_popularity/template\_rank$ in order to reward models for ranking rare templates higher, 3) the maximum diversity for any given precursor; the Morgan fingerprint (calculated with the same settings used for input to the machine learning models) Tanimoto

similarity between the test molecule and all precursors generated by the top k templates was calculated, the diversity was defined as $1 - similarity$ for each precursor, then the maximum value was determined, and 4) whether the prediction was a "failure", where a failure was defined as a test case for which the model did not generate any precursors using the top k templates; unique failures were cases where only that model failed, and the other model was able to generate at least 1 valid precursor. This evaluation was performed using the baseline and pretrained models trained on the *USPTO-Sm* dataset of reactions. These summary statistics, averaged across each test set for the top 10 and top 100 predicted templates, are shown in Tables 1 and 2. For both classes of test molecules, the pretrained model, on average, generated more precursors, recommended more rare templates with higher rank, produced more diverse suggestions, and failed less. The chemical structures of targets and predicted precursors for select examples from each test set are shown in Figure 9. The transformations shown are the top valid precursor produced from the prioritized templates. As evident from the template rank depicted for each transformation, this precursor did not always originate from the top predicted template, as it was possible that the models suggested templates that did not generate any precursors. These illustrative examples highlight common ways in which pretraining can practically help template-based retrosynthetic prediction: by decreasing prediction failure and by more frequently predicting useful, rare transformations.

Table 1: WHO Test Summary Statistics

Model (top-k)	Num. precursors	Weighted pop.	Max. diversity	Failures (unique)
Baseline (10)	3.4	113	0.535	61 (35)
Pretrained (10)	7.6	52	0.610	26 (0)
Baseline (100)	9.5	49	0.601	33 (10)
Pretrained (100)	39.5	15	0.703	23 (0)

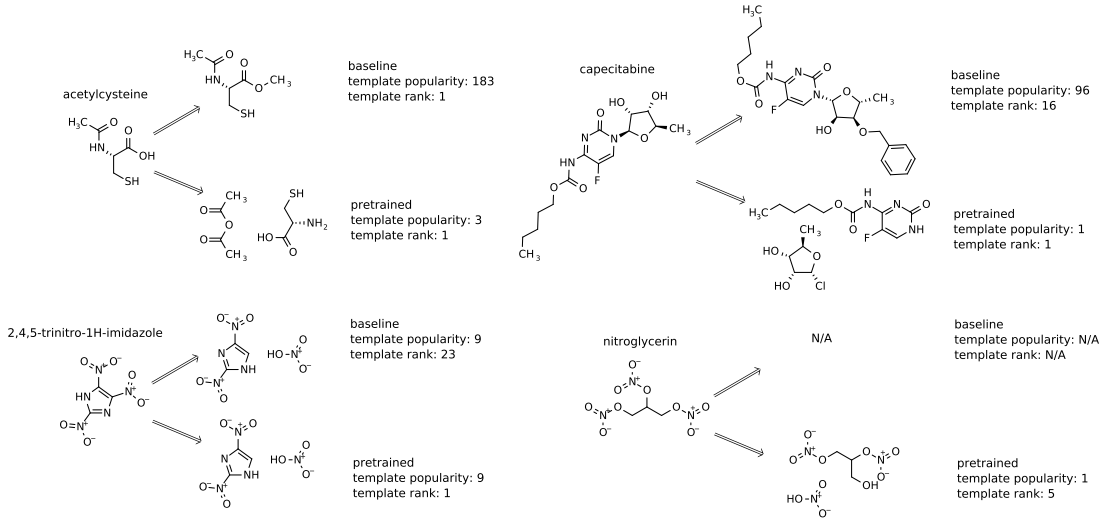

Figure 9: Top retrosynthetic suggestions from the baseline and pretrained models for four test examples: acetylcysteine, capecitabine, 2,4,5-trinitro-1H-imidazole, and nitroglycerin. For acetylcysteine, the top suggestion from the baseline model, a deprotection, demonstrates the implicit bias towards popular templates. The pretrained model, however, recommended a reasonable acetylation of cysteine. For capecitabine, the best suggestion from the baseline model was once again biased towards higher template popularity and was a potentially unproductive deprotection reaction. Additionally, with a template rank of 16, despite this being the first valid precurosr generated, the baseline model prioritized 15 other templates that did not lead to any valid precursors. In the case of 2,4,5-trinitro-1H-imidazole, both models' top suggestion was ultimately a reasonable retrosynthetic nitration template, however the baseline model ranked 22 other templates with higher priority compared to the pretrained model who ranked the template as the top suggestion. Lastly, for nitroglycerin, the baseline model was unable to recommend any template in the top 100 suggestions that led to a valid precursor, however the pretrained model recommended a reasonable, rare template with rank 5.

Table 2: Energetic Materials Test Summary Statistics

Model (top-k)	Num. precursors	Weighted pop.	Max. diversity	Failures (unique)
Baseline (10)	2.5	134	0.416	40 (26)
Pretrained (10)	5.4	80	0.491	14 (0)
Baseline (100)	4.3	80	0.464	19 (10)
Pretrained (100)	18.4	26	0.607	9 (0)

Conclusions

This work described data augmentation and pretraining workflows that can be used to increase the performance of machine learning template-based retrosynthetic prediction models. Datasets of true reactions were augmented with computer generated, hypothetically possible reactions to generate one-to-many mappings from product molecules in the reaction datasets to all applicable templates. An initial applicability neural network model was trained on this augmented dataset and this information was used as a pretrained model for template relevance. Fine-tuning these models using the ground truth reactions further prioritized those reaction templates that represented transformation with evidence that the transformation would be successful if performed in the laboratory. The generalized template applicability information from pretraining was successfully retained during fine-tuning, which led to theoretically applicable templates being prioritized above those that would not lead to any predicted precursors. The pretraining workflow was shown to have a more significant improvement for a smaller dataset of reactions, presumably due to the smaller and more tractable template set used as output classes. The pretrained model can be used a drop-in replacement for template relevance models used for automated tree search algorithms designed for use with reaction templates, and should lead to more, and more diverse, pathways due to the increase in the size and diversity of predicted precursors generated by the top-k recommended templates.

Acknowledgement

The authors thank Betsy Rice and Alex Casey for the curation of the dataset of energetic materials used as a test set in this work. This work was supported in part by a grant of computer time from the DOD High Performance Computing Modernization Program at the ARL DoD Supercomputing Resource Center.

Supporting Information Available

Hyperparameter scanning, template applicability precision, chemical diversity, and expanded chemical examples.

This material is available free of charge via the Internet at `http://pubs.acs.org/`.

References

(1) Coley, C. W.; Rogers, L.; Green, W. H.; Jensen, K. F. Computer-Assisted Retrosynthesis Based on Molecular Similarity. *ACS Cent. Sci.* **2017**, *3*, 1237–1245.

(2) Segler, M. H. S.; Waller, M. P. Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction. *Chem. Eur. J.* **2017**, *23*, 5966–5971.

(3) Badowski, T.; Gajewska, E. P.; Molga, K.; Grzybowski, B. A. Synergy Between Expert and Machine-Learning Approaches Allows for Improved Retrosynthetic Planning. *Angew. Chem. Int. Ed.* **2020**, *59*, 725–730.

(4) Ishida, S.; Terayama, K.; Kojima, R.; Takasu, K.; Okuno, Y. Prediction and Interpretable Visualization of Retrosynthetic Reactions Using Graph Convolutional Networks. *J. Chem. Inf. Model.* **2019**, *59*, 5026–5033.

(5) Liu, B.; Ramsundar, B.; Kawthekar, P.; Shi, J.; Gomes, J.; Luu Nguyen, Q.; Ho, S.;

Sloane, J.; Wender, P.; Pande, V. Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models. *ACS Cent. Sci.* **2017**, *3*, 1103–1113.

(6) Karpov, P.; Godin, G.; Tetko, I. A Transformer Model for Retrosynthesis. *ChemRxiv* **2019**, *10.26434/chemrxiv.8058464.v1*.

(7) Lin, K.; Pei, J.; Lai, L.; Xu, Y. Automatic Retrosynthetic Pathway Planning Using Template-free Models. *ChemRxiv* **2019**, *10.26434/chemrxiv.8168354.v1*.

(8) Chen, B.; Barzilay, R.; Jaakkola, T. S. Path-Augmented Graph Transformer Network. *arXiv* **2019**, *1905.12712*.

(9) Dai, H.; Li, C.; Coley, C.; Dai, B.; Song, L. Retrosynthesis Prediction with Conditional Graph Logic Network. Adv. Neural. Inf. Process. Syst. 2019; pp 8870–8880.

(10) Liu, X.; Li, P.; Song, S. Decomposing Retrosynthesis into Reactive Center Prediction and Molecule Generation. *bioRxiv* **2019**, *677849*.

(11) Coley, C. W.; Green, W. H.; Jensen, K. F. RDChiral: An RDKit wrapper for handling stereochemistry in retrosynthetic template extraction and application. *J. Chem. Inf. Model.* **2019**, *59*, 2529–2537.

(12) Thakkar, A.; Kogej, T.; Reymond, J.-L.; Engkvist, O.; Bjerrum, E. J. Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain. *Chem. Sci.* **2020**, *11*, 154–168.

(13) others,, et al. A robotic platform for flow synthesis of organic compounds informed by AI planning. *Science* **2019**, *365*, eaax1566.

(14) Segler, M. H.; Preuss, M.; Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* **2018**, *555*, 604–610.

(15) Molga, K.; Gajewska, E. P.; Szymkuć, S.; Grzybowski, B. A. The logic of translating chemical knowledge into machine-processable forms: a modern playground for physical-organic chemistry. *React. Chem. Eng.* **2019**, *4*, 1506–1521.

(16) Thakkar, A.; Selmi, N.; Reymond, J.-L.; Engkvist, O.; Bjerrum, E. J. Ring Breaker: Assessing Synthetic Accessibility of the Ring System Chemical Space. *ChemRxiv* **2019**, *10.26434/chemrxiv.9938969.v2*.

(17) Lowe, D. M. Extraction of chemical structures and reactions from the literature. Ph.D. thesis, University of Cambridge, 2012.

(18) Schneider, N.; Lowe, D. M.; Sayle, R. A.; Landrum, G. A. Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity. *J. Chem. Inf. Model.* **2015**, *55*, 39–53.

(19) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36.

(20) Fortunato, M. E.; Coley, C. W.; Barnes, B. C.; Jensen, K. F. 21st Biennial APS Conference on Shock Compression of Condensed Matter, submitted.

(21) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107–113.

(22) Landrum, G. RDKit: Open-source cheminformatics. `https://www.rdkit.org/`, 2006.

(23) Srivastava, R. K.; Greff, K.; Schmidhuber, J. Training very deep networks. Adv. Neural. Inf. Process. Syst. 2015; pp 2377–2385.

(24) Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.

(25) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, *1412.6980*.

(26) Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv* **2016**, *1603.04467*.

(27) Chollet, F. Keras. 2015; `https://keras.io`.

(28) Schwaller, P.; Petraglia, R.; Zullo, V.; Nair, V. H.; Haeuselmann, R. A.; Pisoni, R.; Bekas, C.; Iuliano, A.; Laino, T. Predicting Retrosynthetic Pathways Using a Combined Linguistic Model and Hyper-Graph Exploration Strategy. *ChemRxiv* **2019**, *10.26434/chemrxiv.9992489.v1*.

(29) World Health Organization, *World Health Organization model list of essential medicines: 21st list 2019*; 2019; `https://www.who.int/medicines/publications/essentialmedicines/en/`.

(30) Storm, C.; Stine, J.; Kramer, J. In *Chemistry and Physics of Energetic Materials*; Bulusu, S. N., Ed.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1990; pp 605–639.

(31) Wilson, W. S.; Bliss, D. E.; Christian, S. L.; Knight, D. J. *Explosive properties of polynitroaromatics*; 1990.

Supporting information for: Data Augmentation and Pretraining for Template-Based Retrosynthetic Prediction in Computer-Aided Synthesis Planning

Michael E. Fortunato,*,† Connor W. Coley,† Brian C. Barnes,‡ and Klavs F. Jensen†

†*Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA*

‡*Detonation Science and Modeling Branch, CCDC Army Research Laboratory, Aberdeen Proving Ground, MD, USA*

E-mail: mef231@mit.edu

Hyperparameter Scan

Various parameters of the model architecture, which were common to both the baseline and pretrained models, were scanned over and are summarized in Table S1. For computationally efficiency, the scan was only performed for the smaller *USPTO-Sm* dataset. The model loss, top-1 accuracy, and top-100 accuracy were used as metrics to determine the best model architecture.

Figure S1 shows the top-1 accuracy, top-100 accuracy, and crossentropy loss for model architectures averaged as a function of the number of highway layers. As the number of

Table S1: Neural Network Hyperparameters Scanned

Parameter	Values scanned
Num. dense layers	{1, 2, 3}
Dense layer size	{1000, 2000, 3000, 4000}
Num. highway layers	{0, 1, 3, 5}

highway layers were increased, on average, the model performance decreased.

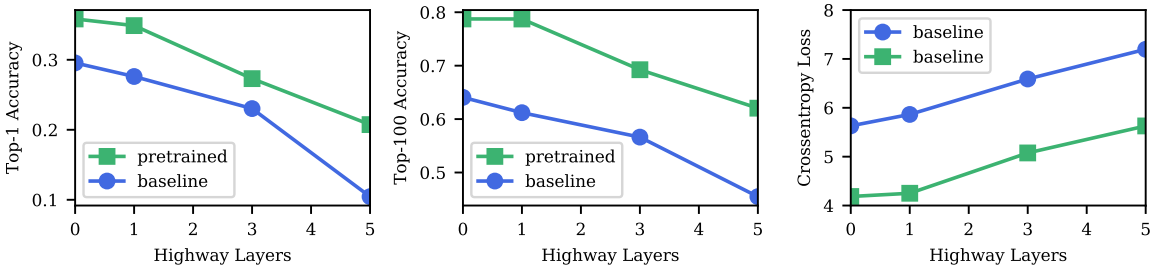

Figure S1: Top-1 accuracy, top-100 accuracy, and crossentropy loss for baseline and pre-trained models averaged as a function of the number of highway layers.

Figure S2 shows the top-1 accuracy, top-100 accuracy, and crossentropy loss for model architectures averaged as a function of the number of dense layers. As the number of fully connected hidden layers increased, the model performance, on average, decreased.

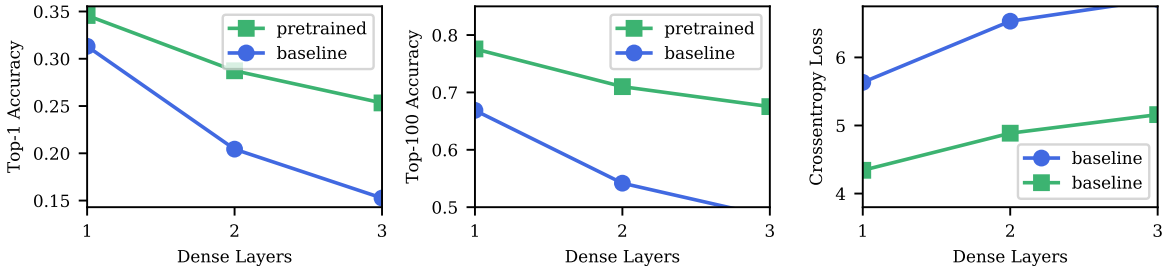

Figure S2: Top-1 accuracy, top-100 accuracy, and crossentropy loss for baseline and pre-trained models averaged as a function of the number of dense layers.

Figure S3 shows the top-1 accuracy, top-100 accuracy, and crossentropy loss for model architectures averaged as a function of the size of the dense layer(s). The average decrease in model performance as the hidden size increased is an artifact of including the low performing models with multiple dense layers and highway layers. Figure S4 shows only models with 1

dense layer and 0 hidden layers. When selecting only these best model architectures, it can been seen that as the hidden layer size increased, the crossentropy loss decreased. Definitive trends were less clear for top-k accuracy.

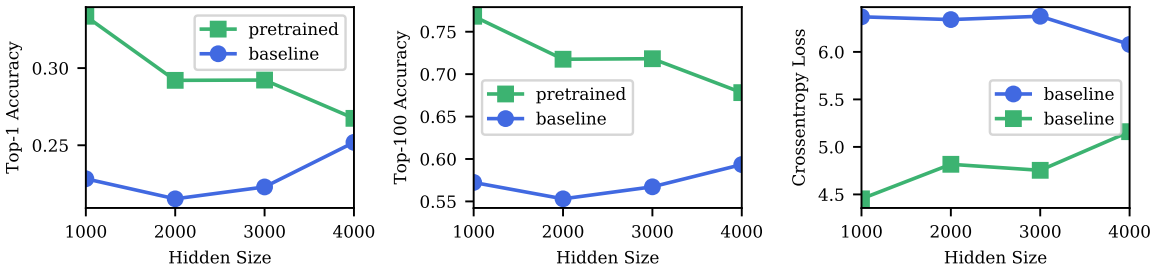

Figure S3: Top-1 accuracy, top-100 accuracy, and crossentropy loss for baseline and pretrained models averaged as a function of the size of the hidden layers.

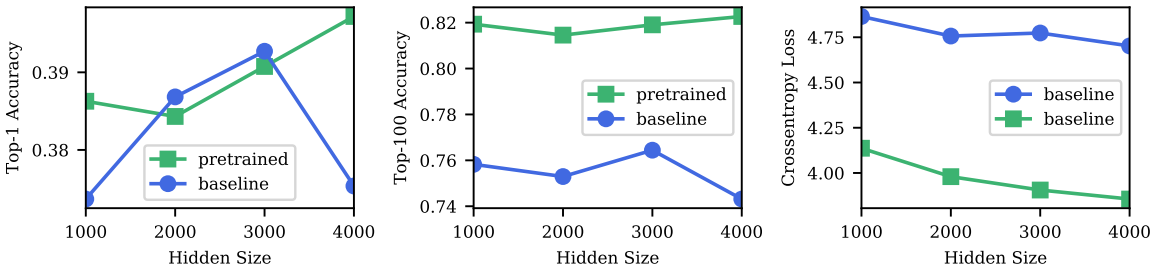

Figure S4: Top-1 accuracy, top-100 accuracy, and crossentropy loss for baseline and pretrained models averaged as a function of the size of the hidden layers, only including models with 1 dense layer and 0 highway layers.

Figure S5 shows the relative improvement gained from pretraining for all model architectures with 0 highway layers. Of note, pretraining increased model performance across all model architectures, even in those cases where both models performed relatively poorly.

Table S2 summarizes some of the best performing individual baseline models, as well as some poorly performing model architectures for comparison, from the hyperparameter scan.

Table S3 summarizes some of the best performing individual pretrained models, as well as some poorly performing model architectures for comparison, from the hyperparameter scan.

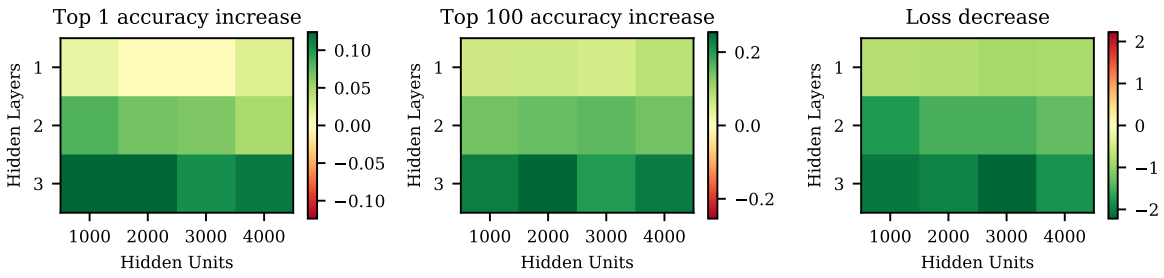

Figure S5: Relative improvement to top-1 accuracy, top-100 accuracy, and crossentropy loss gained from pretraining for all model architectures with 0 highway layers.

Table S2: Best Performing Baseline Model Architectures

Model Architecture	Loss	Top-1 Acc.	Top-100 Acc.
1x(4000)-0x(highway)	**4.70**	0.375	0.743
1x(3000)-0x(highway)	4.77	**0.393**	**0.764**
1x(4000)-1x(highway)	4.80	0.372	0.733
1x(4000)-3x(highway)	5.36	0.359	0.719
2x(4000)-0x(highway)	5.51	0.305	0.643
3x(4000)-0x(highway)	6.26	0.210	0.526
1x(4000)-5x(highway)	6.98	0.191	0.516

Template Popularity Augmentation for USPTO-Lg

Figure S6 shows the increase in template popularity for the *USPTO-Lg* dataset following data augmentation. As with the *USPTO-Sm* dataset, a larger fraction of templates were suucessfully augmented with additional information, but there still remained some templates with very few examples.

Top-k Applicability Precision

Figure S7 shows the top-k applicability precision for the *USPTO-Sm* and *USPTO-Lg* test sets. Top-k applicability precision measures what fraction of the top-k predicted templates generated precursors when applied to the molecule provided as input to the model. A higher value when k=1 meant that the top recommended template more often was useful and generated a valid precursor, which was observed for the pretrained model for both test sets. As more templates were considered (increasing k value on the x-axis), the fraction of

Table S3: Best Performing Pretrained Model Architectures

Model Architecture	Loss	Top-1 Acc.	Top-100 Acc.
1x(4000)-0x(highway)	**3.86**	**0.397**	**0.822**
1x(4000)-3x(highway)	3.88	0.362	0.812
1x(3000)-0x(highway)	3.91	0.390	0.820
2x(3000)-0x(highway)	4.11	0.362	0.790
1x(3000)-5x(highway)	4.24	0.353	0.783
3x(2000)-1x(highway)	4.34	0.307	0.766

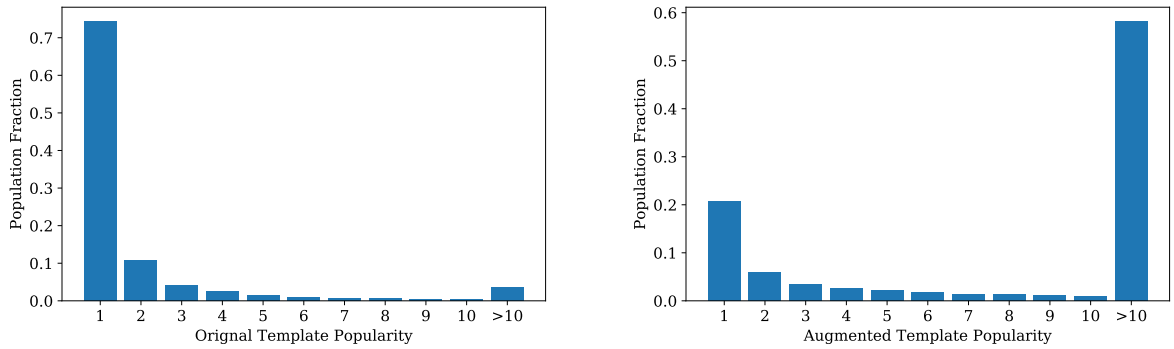

Figure S6: Template popularity for the *USPTO-Lg* dataset before and after data augmentation. The augmentation was successful for many templates, however a fraction still remained under represented.

top-k templates that were applicable decreased, as some templates that did not generate precursors were recommended. However, this fraction of non-applicable templates was lower for the pretrained model, or in other words the pretrained model showed a higher template applicability precision. The lower value for applicability precision for a given k value for the *USPTO-Sm* test set compared to the *USPTO-Lg* test set is an effect resulting from the increased template set size for the *USPTO-Lg* dataset, and the increased average number of applicable templates per molecule.

Chemical Diversity

In addition to the increase in top-k template applicability recall and precision, it was of interest to evaluate the increase in the chemical diversity within the pool of predicted precursors

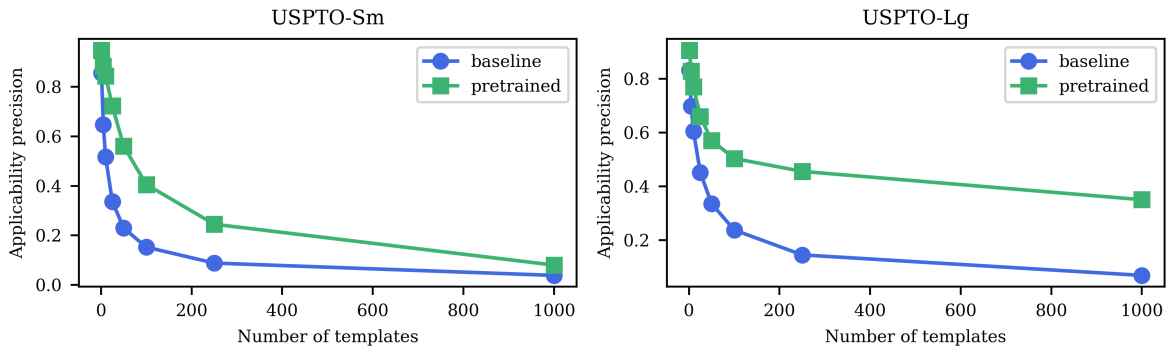

Figure S7: Top-k template applicability precision for the *USPTO-Sm* and *USPTO-Lg* test sets. This metric reports the fraction of top-k predicted templates that "apply" to the molecule provided as input to the model.

from each model. Generating a larger pool of precursors would not be useful if the additionally predicted molecules were not meaningfully different from those already generated. To quantify chemical diversity, the similarity between all precursors in a set of molecules resulting from a single prediction using the top 100 prioritized templates was calculated in a pairwise manner using the well-known and commonly used Tanimoto similarity metric and Morgan fingerprints. The parameters used to generate Morgan fingerprints were the same as described previously to generate input for machine learning models. The diversity of this pool of precursors was defined as one minus the mean pairwise similarity. Importantly, the diagonal terms of the resulting two-dimensional pairwise similarity matrix were omitted from calculation as they were always 1.0. Figure S8 shows the distributions across the test sets of calculated diversity for the *USPTO-Sm* and *USPTO-Lg* datasets. In both cases, pretraining increased the average chemical diversity when compared to the baseline model. The peaks at 0.0 for the baseline models, which were present in both the *USPTO-Sm* and *USPTO-Lg* datasets, represent test examples where the set of precursors generated from the top 100 templates were identical molecules. The disappearance of this peak in the pretrained model is of note, and meant that pretraining increased the number of distinct precursors and therefore, by definition, increased the chemical diversity of the pool of predicted precursors.

Figure S9 shows extra chemical examples comparing the top predicted precursor from

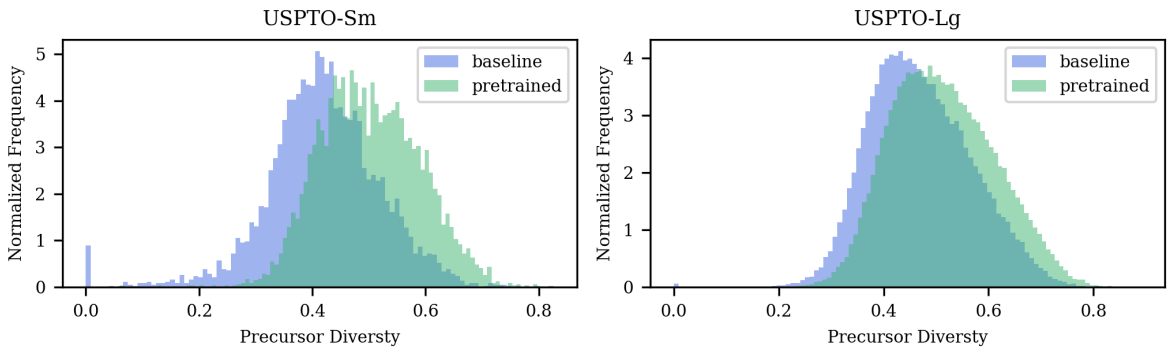

Figure S8: Distributions of Tanimoto calculated diversity from the pool of precursors generated using the top 100 templates predicted by each machine learning model for the *USPTO-Sm* and USTPO-Lg test sets.

the application of the top 100 prioritized templates from the baseline and pretrained models. The two examples on the left were taken from the WHO test set, and the two on the right were taken from the energetic materials test set. Generally speaking, the pretrained model was better able to recognize useful, rare templates with higher rank, and in some cases, the baseline model was unable to find any aplicable templates in the top 100.

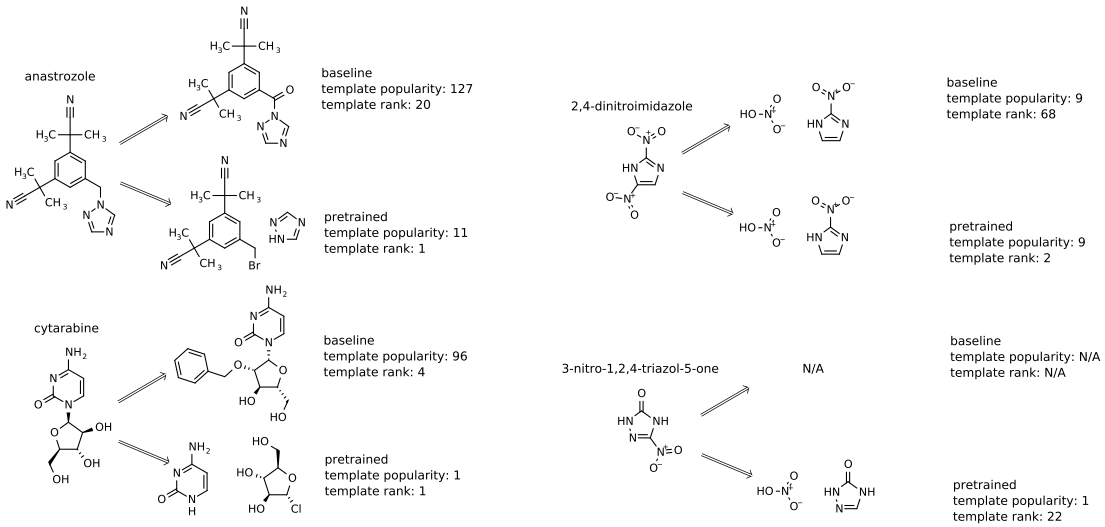

Figure S9: Chemical examples comparing the top predicted precursor from the application of the top 100 prioritized templates from the baseline and pretrained models.

Pretraining with All Template Applicability Information

For the results presented in the main manuscript, the same train/test splits were used throughout the entire pretraining and fine-tuning workflow. The meant that the template applicability information for molecules in the test set was not explicitly learned during pretraining, and the performance boosts were reflective of the generalization from the train set to the test set. As an alternative experiment described in this section, all of the applicability information was included during pretraining, and the original train/test split was used during fine-tuning. First, the applicability pretraining was performed on only the train set, and training was terminated when the validation loss began to increase. Then, a new applicability model was trained using all pre-computed template applicability information (train, validation, and test sets, combined) for the number of epochs determined previously. During fine-tuning, the original data splits were used, so that the model, following the entire pipeline, learned, explicitly, template applicability for test set molecules, but never learned the ground truth template. The crossentropy loss during training is shown in Figure S10, the top-k accuracy as a function of k is shown in Figure S11, the top-100 accuracy as a function of template popularity is shown in Figure S12, and the top-k template applicability is shown in Figure S13. It is difficult to directly extrapolate this to a real use-case, but this shows promise for greatly expanding the data augmentation to include as many molecules as possible. These results indicate that learning template applicability as accurately as possible can help increase template relevance prediction.

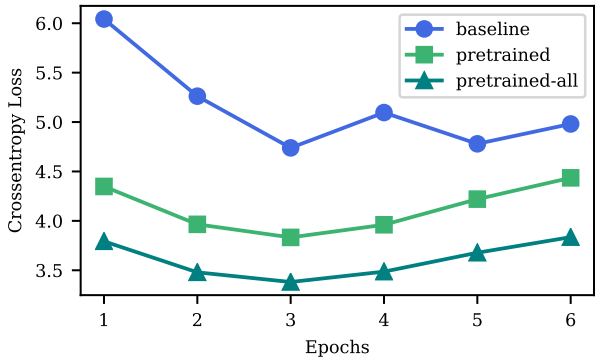

Figure S10: Crossentropy loss during training the baseline and pretrained models compared to pretraining using all template applicability information.

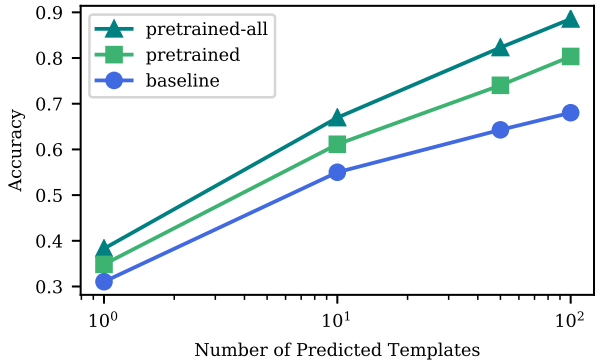

Figure S11: Top-k accuracy as a function of k, the number of templates, for baseline and pretrained models compared to pretraining using all template applicability information.

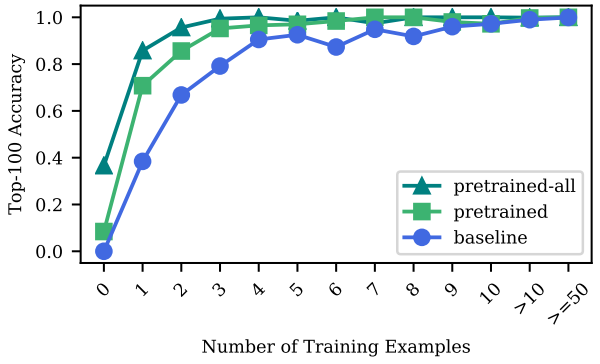

Figure S12: Top-100 accuracy as a function of original template popularity for baseline and pretrained models compared to pretraining using all template applicability information.

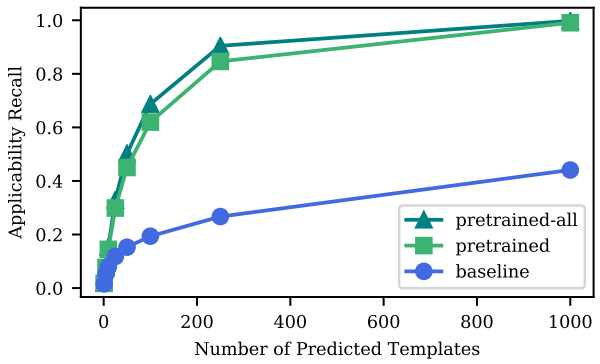

Figure S13: Top-k applicability recall as a function of k, the number of templates, for baseline and pretrained models compared to pretraining using all template applicability information.