RETROSYNFORMER: PLANNING MULTI-STEP CHEMICAL SYNTHESIS ROUTES VIA A DECISION TRANSFORMER

A PREPRINT

Emma Granqvist
Molecular AI, Discovery Sciences,
R&D, AstraZeneca,
Gothenburg, Sweden
&
Department of Computer Science and Engineering
Section for Data Science and AI
Chalmers University of Technology and University of Gothenburg,
Gothenburg, Sweden

Rocío Mercado
Department of Computer Science and Engineering
Section for Data Science and AI
Chalmers University of Technology and University of Gothenburg,
Gothenburg, Sweden

Samuel Genheden
Molecular AI, Discovery Sciences,
R&D, AstraZeneca,
Gothenburg, Sweden

April 10, 2025

ABSTRACT

We present RetroSynFormer, a novel approach to multi-step retrosynthesis planning. Here, we express the task of iteratively breaking down a compound into building blocks as a sequence-modeling problem and train a model based on the Decision Transformer. The synthesis routes are generated by iteratively predicting chemical reactions from a set of predefined rules that encode known transformations, and routes are scored during construction using a novel reward function. RetroSynFormer was trained on routes extracted from the PaRoutes dataset of patented experimental routes. On targets from the PaRoutes test set, the RetroSynFormer could find routes to commercial starting materials for 92% of the targets, and we show that the produced routes on average are close to the reference patented route and of good quality. Furthermore, we explore alternative model implementations and discuss the robustness of the model with respect to beam width, reward function, and template space size. We also compare RetroSynFormer to AiZynthFinder, a conventional retrosynthesis algorithm, and find that our novel model is competitive and complementary to established methodology, thus forming a valuable addition to the field of computer-aided synthesis planning.

Keywords Artificial intelligence · Retrosynthesis · Deep learning · Drug discovery

1 Introduction

Organic chemical synthesis is fundamental to molecular design and discovery, yet designing efficient synthetic routes remains a significant challenge and is often a bottleneck (1). Retrosynthesis, an approach that dates back to the 1960s (2; 3), addresses this by systematically deconstructing target compounds into readily available starting materials. Recent advances in artificial intelligence (AI) and deep learning (DL) have greatly contributed the fields of retrosynthesis prediction and computer-aided synthesis planning (4; 5; 6; 7), leveraging expanding reaction datasets (8; 4). However, there are outstanding challenges, especially for efficient search algorithms in multi-step retrosynthesis planning.

In this work, we introduce the RetroSynFormer, a novel approach to multi-step retrosynthesis prediction guided by a Decision Transformer (DT) (9). By framing retrosynthesis as a sequence modeling problem, RetroSynFormer predicts reaction steps autoregressively, conditioning each decision on previous steps to capture reaction patterns across datasets. Unlike conventional search-based methods, RetroSynFormer inherently learns context-dependent synthesis strategies, enabling the model to utilize the full history of the path it has taken to synthesize a compound at each step and thereby potentially improving route prediction.

We summarize the three main contributions of our work as follows:

- this is the first example of the DT being used for retrosynthesis planning problems, where we recast retrosynthesis optimization as a sequence modeling task;
- this is one of the first demonstration of true multi-step retrosynthesis planning using DL models, as opposed to prior work which has focused on the sequential application of single-step models;
- we thoroughly benchmark our model using meaningful metrics, such as the success rate and top-1 accuracy, providing a detailed comparison to the current SOTA: AiZynthFinder.

2 Background

2.1 Retrosynthesis Prediction

Organic synthesis plays a central role in nearly all chemical industries, from drug discovery to material discovery (1). The aims of organic synthesis are to efficiently manufacture organic compounds through series of chemical reactions; this is a complex problem as each compound can be assembled in multiple ways, creating a huge chemical space and making for a challenging search problem. The synthesis of chemical compounds therefore represents a critical bottleneck in molecular design, a challenge which motivated the conceptualization of retrosynthesis by E.J. Corey in 1967 (3) following earlier developments (2). Retrosynthesis refers to the idea of iteratively breaking down a target compound until all building blocks are readily available starting materials. Since then, emergence of computational technologies and resources has enabled computer-assisted synthesis planning for more productive and efficient retrosynthesis prediction. In recent years, research on retrosynthesis has been further expedited due to the recent developments of AI and DL (4; 5; 6; 7), and the emergence of larger collections of reaction data on which the AI-driven retrosynthesis can be trained (8; 4). This development has enabled chemists to save valuable time and effort when designing synthetic experiments which can lead to finding the right compounds faster (10).

Retrosynthesis prediction is typically separated into two tasks, single-step retrosynthesis prediction and multi-step retrosynthesis planning. In single-step retrosynthesis, the task is to decompose a compound to one or more precursor molecules, or reactants. Multi-step retrosynthesis aims to find a sequence of reactions—a synthesis route—which describes how to make a chemical compound from a set of readily available starting materials. The task is typically approached by iteratively using a single-step model to break down a compound into precursors until all starting materials belong to a set of available building blocks. Single-step models have been extensively researched, and we refer to a recent review for an overview (11). Conversely, multi-step retrosynthesis typically employs a single-step model with a search algorithm such as Monte Carlo tree search (MCTS) (12) (13) or A* search (14; 15); reinforcement learning (RL) has also been proposed for this task (16). However, methods such as MCTS typically only consider the current state (i.e., a single molecule) when making the predictions for the next reaction. Some work has proposed including additional context into models, such as the parent reactions, when making the predictions (17). We hypothesize here that including additional route context in the predictions would be beneficial to retrosynthesis modeling, and that this could result in a model which can learn common patterns or combinations of reactions across the entire route dataset.

2.2 Language Models in Synthesis Planning

In recent years, a variety of transformer models have been widely adopted for various different tasks including machine translation, natural language processing and computer vision. Nonetheless, these models have also gained widespread use in computer-aided synthesis planning. One such example is the Chemformer, a molecular transformer model that has been trained for multiple tasks, including retrosynthesis planning, forward synthesis, and property prediction (18). The Chemformer model has also been integrated with AiZynthFinder for multi-step retrosynthesis planning (6). One other application that uses large language models for multi-step retrosynthesis prediction is DirectMultiStep, which predicts routes as single strings and thus bypasses the need for single-step methods (19). Although an interesting approach, predicting a route as a single string presents a unique set of challenges which are difficult to overcome, such as the inability to condition routes on available starting material and the generation of invalid routes.

2.3 Decision Transformer

Here we present the RetroSynFormer, a novel approach to multi-step retrosynthesis prediction where the search is guided by a DT model (9). The DT framework, originally proposed for RL tasks, reformulates decision-making as a sequence modeling problem, leveraging the success of transformers in natural language processing. Instead of learning a traditional value function or policy, a DT models trajectories of states, actions, and rewards as sequences, predicting future actions autoregressively based on past context. This formulation makes DT particularly well-suited for offline RL settings, where a fixed dataset of trajectories is available, and direct interaction with the environment is costly or impractical.

In retrosynthesis prediction, running additional experiments to test new chemical reactions and explore the search space is often infeasible on a short time-scale, making offline learning essential. By using a DT, our approach conditions action predictions (i.e., reaction templates) on previous states and actions in a retrosynthetic route, allowing it to capture common reaction patterns and dependencies observed in historical data. This enables the RetroSynFormer to effectively generalize across diverse reaction pathways without requiring explicit exploration through new experiments. Starting from a target molecule, the model autoregressively predicts the next reaction step until reaching a stopping criterion, constructing complete retrosynthetic routes in a flexible and data-driven manner. Unlike other sequence-based RL approaches such as the Searchformer(20), which integrates A*-like search with a transformer-based policy, our approach directly models retrosynthetic trajectories without heuristic guidance, making it more flexible for data-driven generalization. As the DT leverages the full trajectory information, making it well-suited for retrosynthesis planning scenarios where long-horizon dependencies are critical, we believe it offers advantages to policy-gradient approaches that may require reward shaping or explicit policy optimization (21).

3 Methodology

RetroSynFormer is a DT model to predict the next reaction (action) in a synthesis route. During inference the model uses a retrosynthesis environment to generate the next reactant molecule(s) (state) and a reward for each action, iteratively generating the retrosynthesis route until one of the stopping criteria is met (Figure 1).

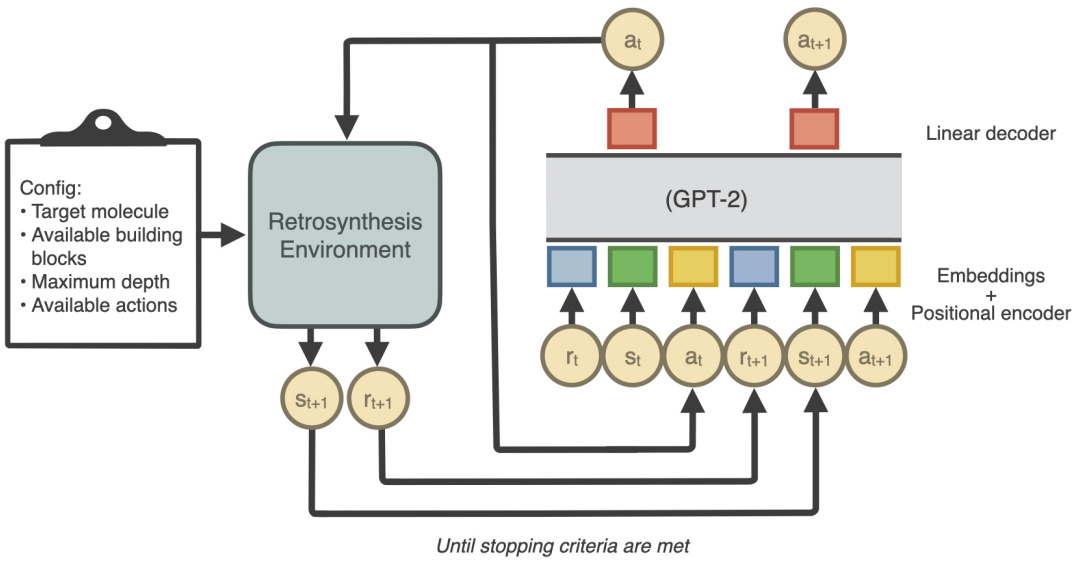


Figure 1: Overview of the RetroSynFormer method. The DT, based on a GPT-2 architecture, predicts an action (reaction template) which is passed to the Retrosynthesis Environment to return the new state along with the corresponding reward. The environment keeps track of the context, such as available building blocks, reaction templates, and maximum depth allowed, as well as the current route, its status, and stop criteria for when to end the predictions.

3.1 Data

RetroSynFormer is trained on a subset of routes from the PaRoutes dataset (4) where the routes have been derived from the reactions in the USPTO dataset provided by Lowe (22). The PaRoutes dataset includes 457,166 routes as JSON data

objects with nested dictionaries, where the molecules are represented as SMILES and the reactions are atom-mapped reaction SMILES with an associated reaction template extracted via RDChiral (23; 24). The full PaRoutes dataset includes a total of 42,551 reaction templates and routes for 175,164 unique targets.

Here, we formulate the retrosynthesis task as a classification task and at each step we aim to predict the correct action, i.e., a reaction template. By reducing the number of possible templates, the complexity of the task can be reduced. Therefore, we sorted the templates based on the frequency in the routes and extracted routes with only the most frequently occurring templates. We have created three datasets: *small*, *standard* and *large*, each based on different frequency-cut-offs for the included templates. For the standard dataset, all routes with only the 3,000 most commonly used templates were taken, giving in total 247,531 (54%) routes, and for the *small* and *large* dataset we instead extracted the 1,000 and 6,000 most common templates, respectively (for detailed numbers, see Table 1). To further reduce the number of templates, we followed the procedure of Heid et al. (25) to identify templates that are subgraphs of other templates, which corresponds to the final size of the action space. For the standard dataset this give 1,572 templates. The process of creating the dataset is illustrated in Figure 2a.

The *standard* dataset is used in all experiments while the *small* and *large* datasets are only used for the results in Section 4.6. Note that the *standard* dataset is a subset of the *large* and that the *small* dataset is a subset of the *standard* dataset, as illustrated in Figure 2b.

Synthesis routes are naturally tree-like data structures. In order to easily process them using the DT, we need to convert the tree-like routes into sequences of *states*, *actions*, and *rewards*, illustrated in Figure 3. Where the molecules are the states, the reaction template is the action and for each step, a reward is calculated. When a reaction decomposes a molecular state into more than one intermediate reactant, a branch is formed in the route. To standardize the way branching points are handled in the data, reactant SMILES are sorted according to length and the molecule with the longest SMILES is expanded first, while the other molecule is added to a stack. In this way, we follow a depth-first order when transforming the route into a sequence: only after a given branch is completely rolled out (all leaves are building blocks) do we continue with the next branch.

Finally, the set of available starting materials, i.e., building blocks, is defined by the set of all leaf molecules from the extracted routes. Details can be found in Table 1.

3.2 Data Splitting

The same data splitting is followed three times for each of the *large*, *standard*, and *small* datasets. These datasets enable us to study the effect of training set size on model performance; details for each dataset can be found in Table 1.

Table 1: Number of unique targets and reactions in the training, validation, and test sets for the three main datasets curated in this work. All datasets were created from PaRoutes (4) data.

Dataset	Small	Standard	Large
Templates	588	1,572	2,986
Building blocks	38,521	58,251	72,737
Train set	44,736	67,180	86,048
Valid set	5,362	8,222	10,645
N1 Test set	2,168	4,320	5,631
N5 Test set	1,732	3,569	5,260
Total # unique targets	53,626	82,222	106,452
Total # routes	144,812	326,294	442,844

We split the data by the target compounds rather than by the routes and in this way ensure that there is no overlap between the targets in the hold-out test set and the training and validation sets. First, we created a hold-out test set from the N1 and N5 targets of the PaRoutes dataset (4), and because we subsample the templates, we also needed to subsample the N1 and N5 sets (see Figure 2c). For the *standard* dataset the number of targets are 4,320 and 3,569 for N1 and N5, respectively. Here, the N1 set consists of relatively simple, predominantly linear synthetic routes, reflecting the broader distribution of extracted routes. The N5 set also contains routes, generally longer and more complex, and was designed to be a more challenging benchmark for retrosynthesis prediction.

After removing the N1 and N5 targets and their corresponding routes, we randomly selected 10% (8,222 for the *standard* dataset) of the total number of unique target compounds from the remaining target compounds to create the validation set. Finally, the rest of the target compounds formed the training set (67,180 target compounds for the *standard* dataset).

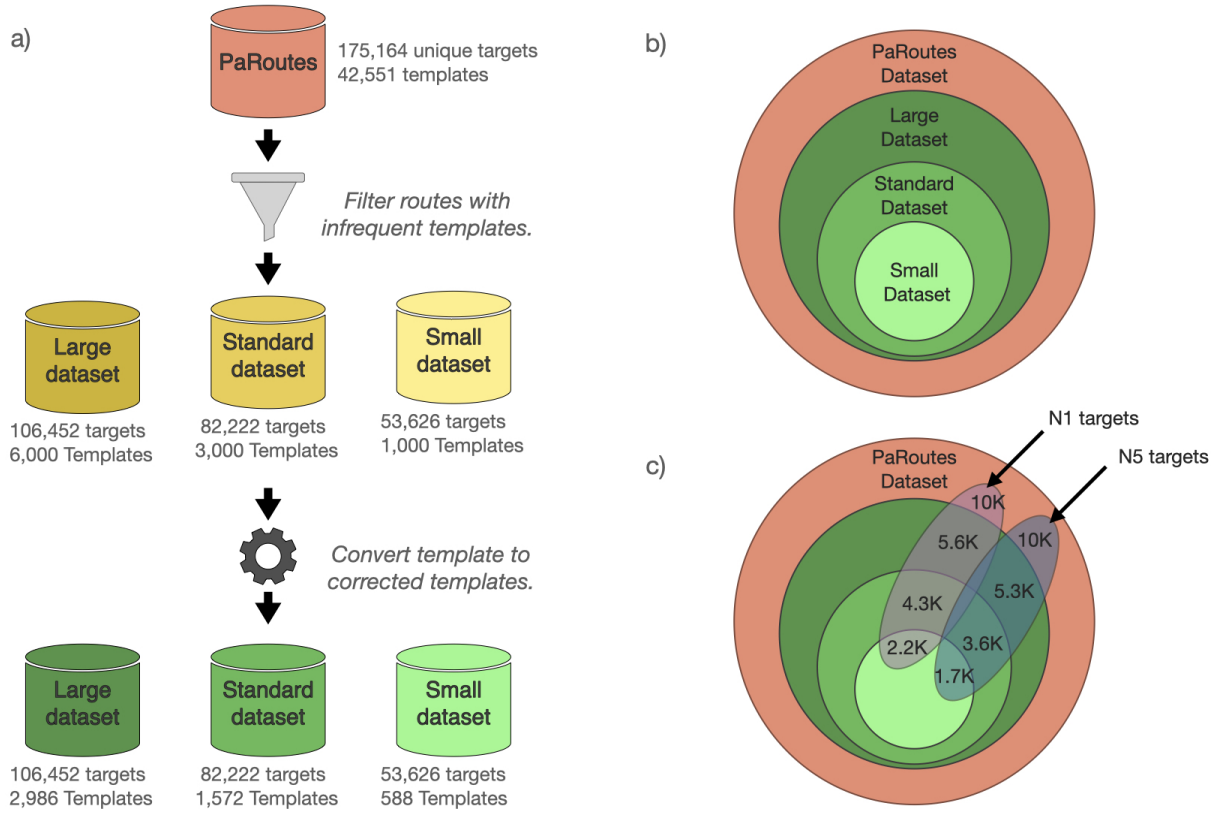


Figure 2: Overview of key data processing steps. a) First, we filter routes from PaRoutes to keep only those using the 3,000 most common templates. After filtering, the reaction templates are further reduced by identifying overlapping templates to give "corrected" templates.(25) b) Illustration of the relationship between PaRoutes and the three datasets, *large*, *standard*, and *small*, showing how the *large* dataset is a subset of the PaRoutes dataset, the *standard* dataset is a subset of the *large* dataset, and the *small* dataset is a subset of the *standard* dataset. c) Illustration showing how the N1 and N5 sets relate to the three datasets in b); as in b) the N1 and N5 sets in the smaller datasets are subsets of the larger ones.

3.3 Decision Transformer

The RetroSynFormer architecture leverages the DT to autoregressively predict the next chemical reaction given all previous reactions, molecules, and rewards. Here, we used the GPT-2 DT model implemented in the Transformers repository (26).

The input to the DT model consists of three vectors: actions, $A = [a_0, \ldots, a_t]$, states, $S = [s_0, \ldots, s_t]$, and rewards, $R = [r_0, \ldots, r_t]$. Each element in A is a one-hot vector encoding a reaction template where $a_i \in \{0, 1\}^{1,572}$ for the *standard* dataset. Each template determines the chemical transformation to apply to the target molecule. Each element in S represents the target molecule(s), so that $s_{i+1} = a_i(s_i)$; here, $s_i \in \mathbb{R}^d$ where d is the length of the molecular fingerprint used. Finally, each element in R is a reward such that $r_i = f(s_i) \in \mathbb{R}$, providing an estimate of the quality of the route at the current time-step, i. The total number of time-steps is denoted by t in a reaction sequence. We used Optuna for hyperparameter optimization to determine the model parameters; details and optimal parameters are provided in Table S7 in the Supplementary Information (SI).

All the presented models have been trained on only one route per target, where, for each target, the route was randomly selected from all available routes.

3.3.1 Actions

The reaction templates used here are the "corrected" templates following Heid et al. (25). The number of available actions is determined by the number of templates that are used in the routes in our dataset. As described in Section 3.1, we are using a set of 1,572 reaction templates for the *standard* dataset, which is also the size of our action space,

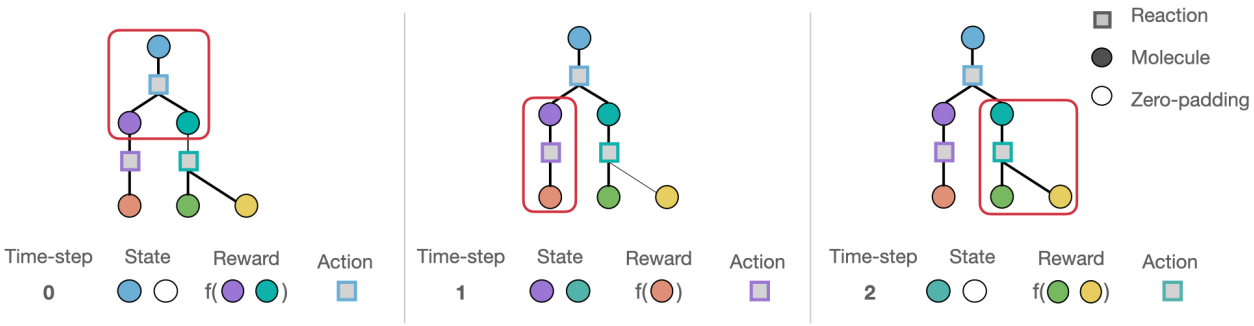


Figure 3: Overview of how reaction routes are converted from trees into sequences. Here, an example route is decomposed into a 3-step sequence. Circles represent molecules (reactants, intermediates, building blocks) and squares represent *actions* (reactions). The *reward* at each time-step is a function of the reactants given by a reaction. A *state* is defined by either a single or pair of molecules, where in the case of a single molecule, zero-padding is used for the second "empty" molecule (represented by the white circle). After each time-step in the sequence, the status of each reactant is evaluated and if the compound is an intermediate (turquoise and purple circles) it will be put in the stack of unexplored states and decomposed in the next time-step. If the compound is a building block (red, green, and yellow circles) then it is the end of a branch and will not be added to the *state*.

$A \in \{0,1\}^{t\times 1,572}$, where t is the total number of reactions (time-steps) sampled in a given route. During inference, RDChiral (23) is used to get the reactants for the next state given the template and target molecule. The size of the action space for *small* and *large* datasets is given in Table 1 and corresponds to the number of available templates in the respective dataset.

3.3.2 States

The initial state contains a Morgan fingerprint representation (27) of the target molecule, generated using RDKit (28). The next states contain the Morgan fingerprints for the unexplored intermediate reactant(s) generated by the next reaction. Each state can hold up to two molecules. If only one molecule is in the state (as in the initial state or for uni-molecular reactions), zero padding is added; if there are three or more molecules, only the two largest molecules are included. Importantly, the molecular states are treated as a stack, meaning that the last molecule added to the state will be the first explored in the next step, while any other reactants are put in a stack of all unexplored intermediates. This leads to a depth-first search exploration of the states. When calculating the reward for a given step, all reactants predicted by the reaction are considered. Each molecule is represented by a 1024-bit Morgan fingerprint, such that the total dimension of the state is 2048 bits.

3.3.3 Reward Function

Contrary to other route scoring methods (29; 30) that evaluate the complete route, here we need a reward function that can evaluate the goodness of an intermediate route after each reaction. This poses a challenge, as assessing route quality before it is complete is inherently more difficult and we cannot use any of the commonly used route scorers. Instead, we chose here to include two aspects of the states when calculating the reward at each time-step: 1) the condition of the molecule(s) in the current state and 2) the depth. For a molecule in a route there are three distinct conditions it can meet: 1) a building block, 2) an intermediate, or 3) a dead end. A dead end means that either no reaction template can be applied or that the maximum depth has been exceeded. The second component of the reward function, the depth, simply measures how many transformations the current molecule is away from the target. To determine how much each condition should contribute to the reward and how to incorporate the depth, we performed an optimization using Optuna (31); see Table S8 for details. An example of the route rewards for a 4-step route is illustrated in Figure 4.

3.4 Inference

The DT model can be viewed as a policy which predicts the next action. However, it lacks the chemical context such as knowledge about the building blocks, the route, what actions are applicable to what molecule, when a route is solved, etc., which is needed during inference. Therefore, we have implemented a retrosynthesis environment to be used in combination with the DT, as is typically done for RL. During inference the environment is used to apply the predicted templates to get the new reactants, evaluate the status of the reactants, calculate the reward function, and build and evaluate the synthesis route. In more detail, the probability for each reaction template is predicted by the DT, which

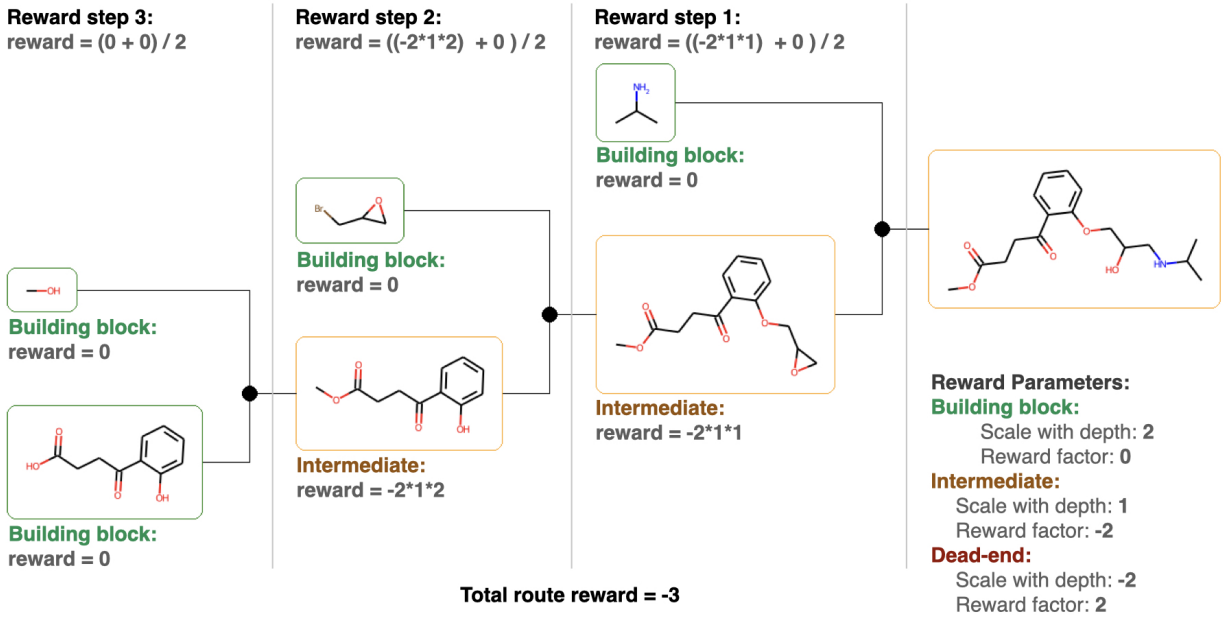


Figure 4: Example of a synthesis route with the associated rewards per reaction, showing how the route reward is calculated from intermediate states. The reward for each step is the average of all reactant molecules in that step and the total route reward is the sum of the reward over all steps.

determines the first applicable template with the highest likelihood as the next action. The action is used to take a step in the environment by RDChiral (23), which returns the reactants given the reaction template (action) and the next molecule (first molecule in the state). Thereafter, the molecules in the new state are sorted based on size and evaluated. If a molecule in the state is part of the building block set, that branch is solved and thus the molecule does not need to be further expanded. However, if there are no applicable actions for that molecule or the route exceeds the maximum depth, the molecule is labeled as a *dead end* and the route is unsolved. If the molecule, however, is determined to be an intermediate, then we add it to the stack of states to be expanded.

The environment also monitors for loops in the route that could stem from recurring intermediates. Loops are not desirable behavior in a route, so if an intermediate occurs more than once in a linear route, the route is terminated and labeled as unsolved.

3.5 Beam Search

Since beam search has proven to be a useful strategy for various natural language tasks, often resulting in improved performance, we have implemented it for the RetroSynFormer. Using beam search, the top n templates given a specific compound (state) are applied at each time-step and evaluated; the same procedure is then applied to each of the n resulting new states, leading to n^2 new states. These n^2 states are sorted based on the cumulative route likelihood and the top n beams kept for the next iteration. This procedure is repeated until either a route is solved or the maximum depth is reached. The beam search enables the RetroSynFormer to consider multiple routes for each target and this way increase the likelihood of finding a solved route. The special case of a beam search using $n = 1$ is equivalent to a greedy search.

3.6 Evaluation Metrics

To assess the model's performance on the retrosynthesis task, we computed standard metrics which evaluate a model's ability to find solved routes for a given target. Specifically, we measured *success rate*, defined as the percentage of targets for which the model identifies a route where all starting materials are available in stock. Additionally, we assessed *top-1* accuracy, which calculates the percentage of targets for which the predicted retrosynthetic route is identical to the ground-truth route. To further compare predicted and target routes, we also evaluated the model using distance metrics to compare the similarity of the predicted routes with the target routes. We calculated the route similarity using the *tree edit distance (TED)*. Since exact TED computation is often infeasible, we used an approximation based on an LSTM model, following Genheden et al. (32).

To further evaluate the predictions made by the model we also calculate the *action accuracy* and *reaction class accuracy*. Here, we compare each action in the predicted routes to the corresponding step in the target route. If the predicted and target routes are not the same length, we exclude the additional actions of the longer route in order to calculate these accuracies.

Finally, we assess the route quality using the *DeepSet route score* described in Yujia et al. (33) and implemented in `rxnutils` (34). This score has incorporated human expert assessments of synthesis routes and can be used to categorizes routes as either "good" for scores between 0 and 5, "plausible" if the score is between 5 and 9, or "bad" if the score is between 9 and 15.

3.7 Baseline

As a baseline, we trained a template-based one-step retrosynthesis model on the reactions from the same routes as the RetroSynFormer model using AiZynthTrain (24). We then performed retrosynthesis experiments with the trained one-step model using AiZynthFinder on the N1 and N5 target sets from PaRoutes(4). In these experiments, we used the same set of building blocks (stock) as in the RetroSynFormer experiments. We used the default AiZynthFinder settings, i.e., 100 iterations of Monte Carlo tree search, and a maximum depth of 6 (the same as RetroSynFormer).

4 Results

The RetroSynFormer was used to predict routes for 1,500 random targets from the N1 test set, and 1,500 random targets from the N5 test set. The training and evaluation steps were repeated three times for statistical analysis. For these experiments, RetroSynFormer was trained on one randomly sampled route per unique target (e.g., 67,180 routes for the *standard* dataset). By default, the RetroSynFormer returns only the first route found via the beam search, although it can also generate multiple routes. Unless otherwise stated, we use a beam width of 50 and return the route with the greatest cumulative likelihood; we call this model *RetroSynFormer50*. As a baseline, AiZynthFinder was trained as described in Section 3.7 and evaluated on the same test sets. For comparison with the baseline, the highest ranked route from AiZynthFinder was extracted for each target.

4.1 Retrosynthesis Performance

There are many approaches for evaluating the performance of retrosynthesis algorithms (4), and one of the most common and also most important metric is the success rate (the percentage of targets for which a solved route is found). This is a prerequisite for evaluating other aspects of the predictions such as the quality of the sampled routes. However, because we have reference routes for the targets as extracted from patents, we can not only calculate if the predicted route is identical to the target route (the accuracy), but also the similarity.

A summary of the RetroSynFormer50 performance on the N1 and N5 targets is presented in Table 2. Although *RetroSynFormer50* results in a slightly worse success rate than the AiZynthFinder baseline, we can see that it performs comparably in many other aspects. For example, the success rate for the N1 test set between the models differs by less than 2%, and the top-1 accuracy differs by only 0.039 and is nevertheless rather low for both models. Compared to the N1 set, the N5 set is more difficult for both models as the success rate and top-1 accuracy is lower. The TED and average route length in Table 2 are only calculated for the solved routes. Here we only see some smaller differences between the models. It seems like the difference between the target and predictions in general are larger for the N5 set and also that on average the routes for the N5 targets are slightly longer, although the routes are on average short.

In Figure 5a we can see the distribution of route lengths for the predictions compared to the targets. We observe that the distributions for the AiZynthFinder routes and RetroSynFormer routes are very similar and, interestingly, that the target routes are in general longer compared to the predicted routes. This suggests that it is possible to find shorter routes with the available stock than what was used for the original patent routes. In Figure 5b, we plot the distribution of the route reward in the predicted routes from RetroSynFormer compared to the target routes. The predicted routes show a wider distribution than the target routes, and the median reward is rather different. Equivalent figures for results on the N5 set are available in Figures S8a and S8b - and we observe the similar trends as for the N1 set.

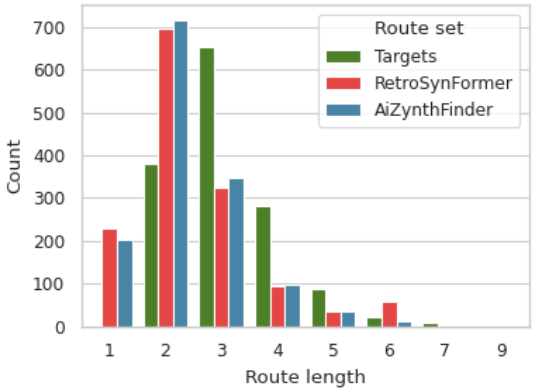
4.2 Complementarity of Retrosynthesis Approaches

As demonstrated, the RetroSynFormer is almost comparable to AiZynthFinder with respect to the success rate. A natural question that arises is if the solved targets are the same across the models, or if the targets solved by each model are complementary. As described above, the two models have been evaluated on the same N1 and N5 target sets, and the number of targets solved by each model is reported in Table 3. Here, we can see that 88.5% of the N1 routes are

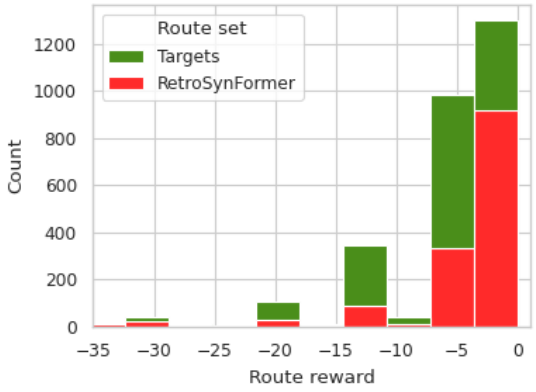
Table 2: Performance of the RetroSynFormer compared to the AiZynthFinder baseline on retrosynthesis planning tasks using a diverse suite of evaluation metrics. Arrows indicate the direction of better performance for each metric.

	RetroSynFormer50*		AiZynthFinder	
Test set	N1	N5	N1	N5
Success rate (%) ↑	0.924	0.899	0.939	0.924
Top-1 accuracy ↑	0.106	0.058	0.143	0.071
Mean time per route(s) ↓	68.1	81.9	5.45	5.8
Mean TED ↓	5.58	7.08	5.40	7.12
Mean # reactions per route ↓	2.34	2.55	2.52	2.92

* RetroSynFormer results show averages over three runs using beam width 50. Standard deviation for the RetroSynFormer models is ≤ 0.004 for the success rate, ≤ 0.001 for the top-1 accuracy, ≤ 1.7 for the mean time per route, ≤ 0.05 for the mean tree edit distance (TED), and ≤ 0.01 for the mean # reactions per route.



(a) Histograms of N1 route lengths, measured as the number of actions per route, for each route set.



(b) Stacked bar plot showing the distribution of N1 route rewards in the target versus RetroSynFormer-predicted routes.

Figure 5: Route characteristics for the N1 test set target routes compared to the RetroSynFormer and AiZynthFinder solved predictions for the same targets. a) The median reward for the route length is 3 for the target routes and 2 for the RetroSynFormer and AiZynthFinder routes, indicating solved routes are generally shorter than those in the reference dataset. b) The median reward for the route reward is -6 for the target routes and -2 for the RetroSynFormer routes.

solved by both models and that 3.9% and 5.4% are solved by only RetroSynFormer and AiZynthFinder, respectively. This means that only 32 routes (2.1%) could not be solved by either model. Similarly for the N5 set, only 3.1% of targets could not be solved by any model. To conclude, by combining both methods we can solve 97-98% of the routes and potentially reduce the error compared to using an individual model.

Table 3: Number and percentage of the N1 and N5 targets which are solved by only RetroSynFormer, only AiZynthFinder, both models, and no model. There are 1,500 targets in each of the N1 and N5 sets.

	N1		N5	
Solved By	Count	Percent	Count	Percent
Only RetroSynFormer	59	3.9%	67	4.5%
Only AiZynthFinder	81	5.4%	112	7.5%
Both	1,328	88.5%	1,274	84.9%
None	32	2.1%	47	3.1%

In Figure 6 we compare an example route generated by RetroSynFormer to the route generated by AiZynthFinder for the same substituted benzamide target from patent US-8680159-B2, a patent for bradykinin 1 receptor modulating compounds. In this example, RetroSynFormer generates a route in two steps and is very efficient. The route starts with a reductive amination, a step with some support from the literature (e.g., US-8519124-B2 or WO20/103896 patents), followed by a deprotection of the amine group. The patented target route also ends with a reduction of a protected aminocyclohexanone and naphthyridine, followed by deprotection, but takes some steps to build up the aminocyclohexanone intermediate from cheaper starting materials. AiZynthFinder, on the other hand, fails to employ the template for the reductive amination, and thus is unable to break the bond between the naphthyridine and cyclohexane rings. After many unproductive steps, the search hits the maximum tree depth and stops at a starting material not in the commercial stock. Nevertheless, we want to stress that this example only serves to illustrate a scenario where RetroSynFormer produces a route complementary to AiZynthFinder, and we could likely identify other examples where AiZynthFinder is better suited to solve a given target than RetroSynFormer.

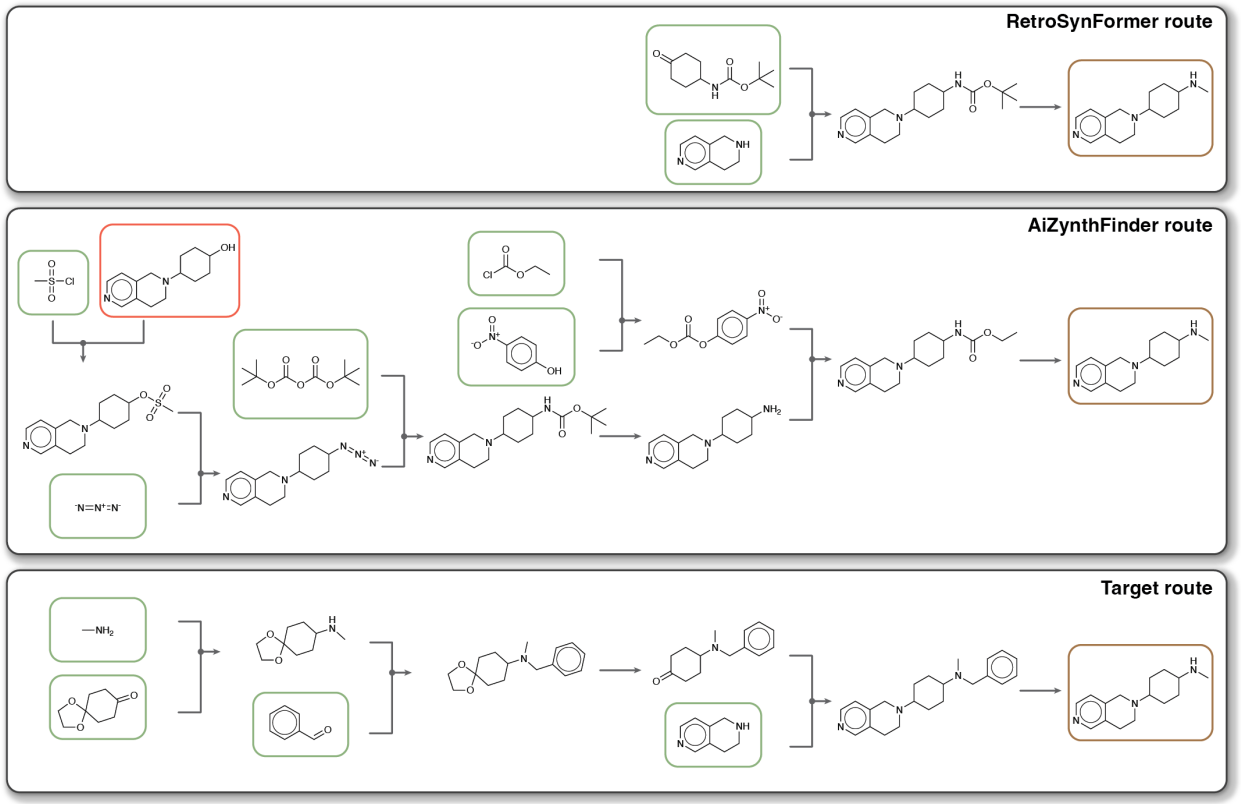


Figure 6: Comparison of an example route generated by RetroSynFormer for a substituted benzamide target from patent US-8680159-B2 (top panel) to the route generated by AiZynthFinder (middle panel); the target route is shown for reference (bottom panel). The final product is the same in all three panels and denoted with a golden box. The green boxes indicate purchasable building blocks and the red box indicates a dead-end state that is not a purchasable building block (i.e., maximum route depth was reached).

4.3 Analysis of Chemistry in Predicted Routes

To gain a deeper understanding of the model behavior, we further analyzed the reactions in the *solved* predicted routes, i.e., the routes that terminated in purchasable (in-stock) starting material(s). In Table 4, we show the that the total number of reactions observed in the solved routes is on the order of 3,000 compared to 4,000 in the target routes, and observation that is also reflected by the shorter average route lengths of RetroSynFormer-predicted routes (Table 2). Furthermore, these reactions are represented by about 650 and 700 unique templates for the RetroSynFormer and AiZynthFinder, respectively, which is fewer than the approximately 800 unique templates in the target routes. However, the unique number of reaction classes is about 60 for both the predicted and target routes. This shows that there is a great redundancy in the templates, i.e., several templates represent similar reactions, such that the model can predict different templates that lead to identical disconnections.

In Figures S9a and S10a we plot histograms of the 15 most common reaction templates in the N1 and N5 target routes, respectively, and compare them to the histograms from the predicted routes. We can observe that there is a significant discrepancy—popular templates in the target routes are not necessarily the most frequently used templates in the predicted routes. Furthermore, we plot histograms of the 15 most common reaction classes in the targets routes in Figures S9b and S10b, and we observe that there is significantly less discrepancy in the templates between the reference N1 and N5 routes compared to those in the predicted routes for same sets. This is also reflected in the higher class accuracies for the RetroSynFormer and AiZynthFinder compared to the action accuracies (Table 4). In general, both the action and class accuracies are lower for the RetroSynFormer than for AiZynthFinder, and, generally speaking, the accuracy is lower for the N5 set than for the N1 set.

We also looked at the solved routes separately and calculated the route accuracy for solve routes (Table 4). Naturally, we observe that this accuracy is slightly higher than when we also include unsolved routes (Table 2). If we disregard the order of the steps, we also observe a slight increase in accuracy (e.g., +0.05 for route accuracy on the N1 set using the RetroSynFormer), indicating that although some of the predicted routes are overall correct, the exact order of the individual actions might differ compared to the target routes.

Finally, we estimated route quality by the recently proposed DeepSet route score (33) for both the predicted routes and their targets. We observe that scores for the RetroSynFormer are marginally better (lower) than for AiZynthFinder on both the N1 and N5 datasets. However, as averages for predicted and patented references routes in both datasets are $<$3.5, they can all be classified as “good". This indicates that on average the routes can be used as-is to plan the wet-lab experiments and that they do not require modification by an experienced scientist.

Table 4: Summary statistics, action/class accuracies, and route scores for the *solved* routes exclusively. Arrows indicate the direction of better performance for each metric.

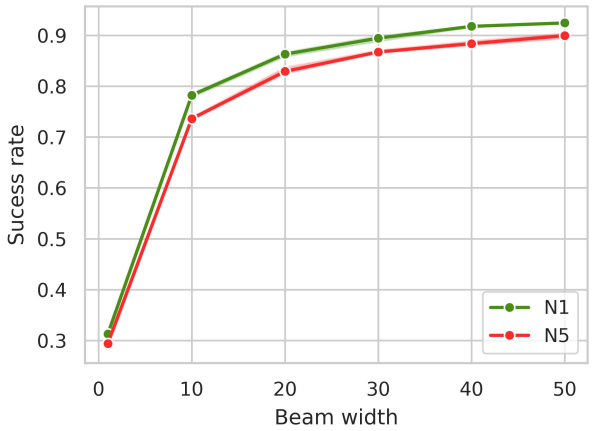
	RetroSynFormer50		AiZynthFinder		Targets	
	N1	N5	N1	N5	N1	N5
Number of total reactions	3,047	3,189	3,234	3,556	4,093	4,567
Number of unique templates	669	650	712	706	795	812
Number of unique reaction classes	63	61	62	63	62	61
Action accuracy ↑	0.266	0.223	0.312	0.239	-	-
Class accuracy ↑	0.374	0.321	0.415	0.333	-	-
Route accuracy ↑	0.127	0.084	0.163	0.096	-	-
Unordered route accuracy ↑	0.177	0.125	0.216	0.147	-	-
DeepSet route score (33) ↓	3.139	3.313	3.336	3.531	3.137	3.411

4.4 Model Exploration 1: Beam Search

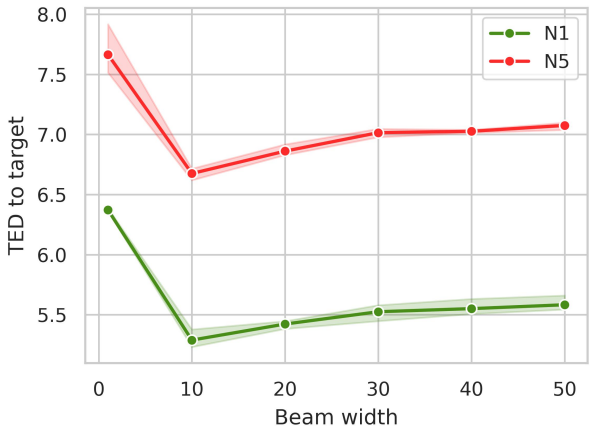
The results presented above were achieved using a beam width of 50. A high beam width naturally results in a larger search space and thus a higher probability of success; however, for the same reason, it also increases the search time. Thus, there is a trade-off between optimizing success rate and search time. Using the trained model, we have evaluated the effect of the beam width on route predictions for the targets in the test set (Figure 7). We can here clearly see that the success rate increases logarithmically with the beam width, as expected. However, when increasing the beam width, the search time increases exponentially and becomes a limiting factor for choosing the beam width. Interestingly, we don’t see the expected increase in top-1 accuracy or decrease in TED with increasing beam width. Both of these metrics show an optimal beam width of 10 and they deteriorate slightly for high beam widths. This shows that there is not a clear correlation between the success rate and the top-1 accuracy, and highlight the importance of considering both metrics for retrosynthesis.

4.5 Model Exploration 2: Reward Function

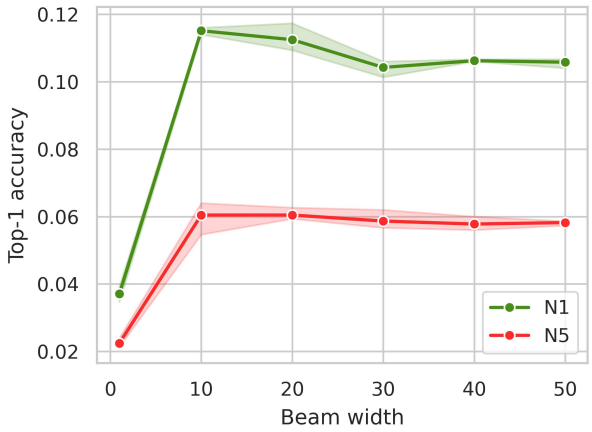
A hyperparameter search was performed as described in Section 3.3.3. Surprisingly, we observed that the model was not very sensitive to the choice of reward function. In Table 5 we demonstrate this by showing the success rate and top-1 accuracy for models trained using alternative reward functions. Here, we have changed the signs and removed the scaling with depth parameter. Details for all the reward functions can be found in Table S9. The results show that the default reward used in the main results presented does indeed give the highest success rate. However, the difference between the other rewards are in general small and if the top-1 accuracy is considered instead, the best reward function is another one denoted *Remove Scaling Building Block Reward*. This is because the reward was optimized with regards to the success rate rather than the top-1 accuracy.



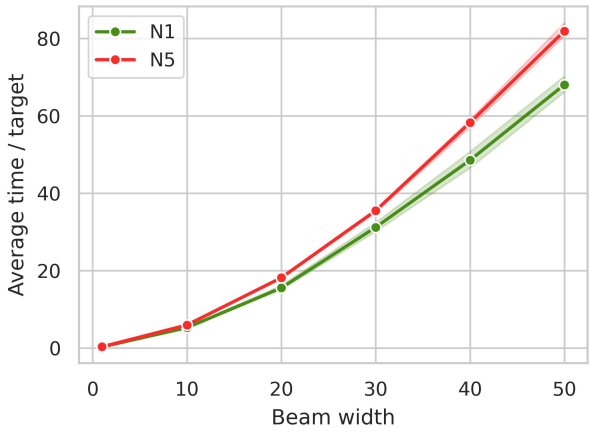
(a) Success rate against beam width.



(b) Tree edit distance against beam width.



(c) Top-1 accuracy against beam width.



(d) Average time per target against beam width.

Figure 7: Effect of increasing the beam width on RetroSynFormer performance on the N1 and N5 sets. a) Success rate versus beam width. b) Tree edit distance (TED) versus beam width. c) Top-1 accuracy versus beam width. d) Average search time / target versus beam width. Error bars indicate the standard deviation of three separate model predictions in all plots.

4.6 Model Exploration 3: Action Space Size

All results presented above have used the *standard* dataset which includes 1,572 templates. To show that the approach is also useful for other action spaces that are of larger or smaller sizes, we have trained RetroSynFormer and AiZynthFinder with two other datasets, denoted *small* and *large* (see Section 3.1 and Figure 2). For the evaluation, we constructed different N1 and N5 test sets for the different action space sizes by randomly sampling 1,500 routes from each N1 and N5. This is done to ensure that we evaluate the models on targets with reference routes consisting of the full template space. The results of this evaluation can be found in Table 6. The results indicate that the model is able to solve the majority of the targets for all action space sizes and that the success rate does not differ significantly depending on the template set. We can also observe that the top-1 accuracy decreases when the number of available templates increases. This is not surprising as when there are more disconnections to choose from, the probability of predicting the correct one decreases and it is likely that there are more reaction templates that are plausible and/or very similar. In addition, the additional templates which are added when scaling up the template space will be less frequently used and might be only used in very few routes, something which may also explain the drop in top-1 accuracy.

We further sub-sampled the targets in the N1 and N5 sets to select the targets that are common between the *small*, *medium*, and *large* datasets; there are 232 and 248 such targets for N1 and N5, respectively. The performances

Table 5: Different reward functions and their impact on the model performance.

Reward function	Success rate		Top-1 accuracy	
	N1	N5	N1	N5
Default	**0.924**	**0.899**	0.106	0.058
Increasing building block reward	0.916	0.889	0.099	0.054
Decreasing building block reward	0.916	0.891	0.107	0.056
Remove scaling building block reward	0.912	0.890	**0.110**	**0.064**
Flipping sign intermediate score	0.914	0.890	0.106	0.054
Remove scaling intermediate	0.920	0.898	0.105	0.060
Flipping sign dead end score	0.921	**0.899**	0.107	0.058
Remove scaling dead end	0.915	0.898	0.103	0.059

* RetroSynFormer results show averages over three runs. Standard deviations for all RetroSynFormer models are ≤ 0.006 for success rate and ≤ 0.005 for top-1 accuracy.

of RetroSynFormer and AiZynthFinder on those targets are shown in Table S10. Interestingly, the largest drop in performance is observed for the models trained on the *small* dataset, where the success rate of RetroSynFormer drops by almost 10% compared to the entire *small* target set. The success rate of AiZynthFinder trained on the *small* dataset does not show the same drop in success rate, but the top-1 accuracy is noticeable lower.

Table 6: Performance of retrosynthesis models for different template sets.

Dataset	Model	Test set	Success rate	Top-1 Accuracy	TED	Avg. route length
Small	RetroSynFormer50	N1	**0.950**	0.182	4.426	2.291
		N5	0.833	0.101	5.428	2.472
	AiZynthFinder	N1	0.923	**0.223**	4.065	2.343
		N5	**0.917**	**0.125**	7.121	2.923
Standard	RetroSynFormer50	N1	0.924	0.106	5.5836	2.337
		N5	0.899	0.058	7.075	2.548
	AiZynthFinder	N1	**0.939**	**0.143**	5.398	2.517
		N5	**0.924**	**0.071**	7.120	2.923
Large	RetroSynFormer50	N1	0.929	0.082	6.168	2.276
		N5	0.887	0.045	7.54	2.531
	AiZynthFinder	N1	**0.939**	**0.115**	6.112	2.538
		N5	**0.925**	**0.073**	7.713	3.019

* RetroSynFormer results show averages over three runs. Standard deviations of all RetroSynFormer models are ≤ 0.004 for success rate, ≤ 0.003 for top-1 accuracy, ≤ 0.09 for TED, and ≤ 0.01 for route length.

5 Discussion

We have presented a novel approach, RetroSynFormer, for retrosynthesis prediction that uses the recently developed DT to generate synthesis routes conditioned on a target compound. The DT model is one of a few DL models that recasts an RL problem as a sequence modeling problem. We have herein shown for the first time that it is possible to adapt the DT model for problems in the chemical domain. Nevertheless, RetroSynFormer shares many implementation details with established retrosynthesis methods: it uses a fixed set of templates to break down molecules into reactants—just as template-based single-step models—and it uses fixed stock of building blocks to indicate the termination of retrosynthesis pathways—as do the majority of published multi-step retrosynthesis algorithms. In contrast, although RetroSynFormer predicts the next reaction autoregressively, it utilizes at each step the entirety of the previously predicted route and thus has the potential to optimize the sequence of reactions over a larger context window than is possible with existing algorithms.

We have benchmarked the performance of RetroSynFormer in several ways by comparing it to AiZynthFinder, a common, template-based method that represents a conventional retrosynthesis approach. In terms of success rate, i.e., for how many targets the retrosynthesis produces a route that leads to commercial building blocks, the two approaches are comparable (see Table 2). AiZynthFinder outperforms RetroSynFormer by a few percentage points, but it is unclear if this difference is of practical importance. When comparing the predicted routes to the reference patent routes for each target, AiZynthFinder more often reproduces the patented routes—but if we look at the route similarity as computed by TED, RetroSynFormer is indistinguishable from AiZynthFinder on average. The two approaches also find routes of comparable length, and the average route score is similar. This indicates that RetroSynFormer produces routes that are different than the patented route but of similar quality as the routes produced by AiZynthFinder. Encouragingly, the predicted routes by either algorithm are of similar quality to the patented reference routes, and can in both cases be classified as "good" according to a recently established route scoring method (33). Furthermore, we have shown that the two approaches to retrosynthesis are complementary, and together they can predicted routes to commercial starting material for $>$97% of the targets investigated (see Table 3). Such a result has been observed for other retrosynthesis models with AiZynthFinder before (6) and point to a real use-case for RetroSynFormer where the combination of different algorithms has a higher chance of producing valuable results and can be used in a staged fashion.

The exact reproducibility of the reference patented route is also not a necessity because of the redundancy in the template set, i.e., different templates could translate into identical or near identical disconnections. We have shown that both RetroSynFormer and AiZynthFinder show a greater discrepancy to patented routes if one looks at the exact predicted template rather than the reaction classes represented by the templates. If we instead evaluate route quality based on class accuracy or disregard the order of the reactions in the route, we generally observe greater agreement with the patented routes (Table 4). This complexity of calculating route similarity was recently discussed in Genheden et al. (29). Furthermore, we have again shown that only a fraction of the templates are practically needed to find synthesis routes, e.g., RetroSynFormer uses only 669 out of the 1572 available templates (standard dataset) to find synthesis routes for the N1 targets. This has been shown previously for AiZynthFinder (35) and here we have shown it again for RetroSynFormer. Hence, it is clear that we either 1) should evaluate retrosynthesis planning on a different target set where more templates are needed, or 2) need better retrosynthesis models that can better employ rarely used templates.

Designing a novel algorithm is not straightforward, and herein we have highlighted a few explorations on the model design. First, beam search was essential for finding routes to commercial materials; in Figure 7 we show that with a greedy algorithm we only reach about a 30% success rate. Unfortunately, the scaling factor of the beam search is considerable, especially compared to a search algorithm such as the one in AiZynthFinder where additional iterations come at basically a constant cost. However, the success rate increases more steeply with increased beam width in the RetroSynFormer than with additional iterations in AiZynthFinder (4). This indicates that the effort of increased beam width could make a practical difference to the produced synthesis routes, compared to AiZynthFinder where it takes many additional iterations to find additional solutions. We acknowledge that there is still a lot of engineering improvements possible to increase the efficiency of RetroSynFormer, which is currently much slower than AiZynthFinder. We would like to argue that currently the reported times in Table 2 are not a fair comparison as the performance of AiZynthFinder has been optimized over the course of $>$5 years.

Furthermore, a key challenge in developing RetroSynFormer has been the design of the reward function. As presented in Section 4.5, we observed that changing the reward function in what seems to be a suboptimal way does not have a significant negative impact on the results, which may seem counter-intuitive. First of all, it is not straightforward to evaluate synthesis routes in a step-by-step fashion as we do here as it is not evident how good a single reaction is until we have the full route. We thus believe that further investigation of different reward set-ups could potentially be beneficial and it is possible that one could design a better reward function than the one used here. One possibility, for instance, could involve using a machine learning model to estimate the future reward of a partially constructed route (14). As a final exploration, we increased and decreased the action space and found that the performance of the

RetroSynFormer compared to AiZynthFinder does not change noticeably in these experiments. It will be up to future work to investigate if the model scales well to the sizes available in proprietary datasets (24). Considering the low fraction of templates used in practice, one could nevertheless argue that having the ability to scale to larger template spaces is not a requirement for a useful retrosynthesis algorithm.

Herein, we have chosen to evaluate the RetroSynFormer on targets from the PaRoutes dataset because we are interested in developing a novel algorithm and benchmarking it. In other words, we are more interested in exploring the capabilities of the DT, i.e., does it work at all, and what are the limitations, rather than improving the state-of-the-art success rate by a few percentage points. The PaRoutes dataset is ideal in this scenario as it is robust, has been used in several previous publications, and also provides reference patented routes. An obvious follow-up to this would be to evaluate RetroSynFormer on other target datasets, e.g., ChEMBL (36), GDB (37), or compound ideas from internal drug discovery projects. Because the conventional AiZynthFinder is already successfully deployed in drug discovery projects (10), we would like to identify areas where a different algorithm like RetroSynFormer could bring additional value. We have started in this work to identify that RetroSynFormer is in many ways complementary to AiZynthFinder, but more work is needed. It would be especially interesting to show the advantage of the memory inheritance feature of the Decision Transformer, where we may base the prediction of the next action on all previously predicted actions (beyond a single route). To enable this, we would need to design a special target set because it is likely that this feature would only be important for certain classes of compounds.

6 Conclusion

We have presented RetroSynFormer, a novel approach to retrosynthesis that treats the task as a sequence modeling problem. We have demonstrated that the model can find a synthesis route for 92% of the N1 targets and 90% of the N5 targets from the PaRoutes suite of retrosynthesis benchmarks, and that it can be used complementarily with AiZynthFinder to solve >97% of targets. Our method is unique in its approach as it is the first time a DT has been used in the chemical domain. The RetroSynFormer treats retrosynthesis prediction as a sequence modeling task and conditions its predictions based on previous reactions, targets, and rewards, thus suggesting disconnections using greater global awareness than more conventional approaches. With some additional developments and benchmarks, especially on code optimization—we believe that RetroSynFormer will form a valuable tool in computer-aided synthesis planning.

Acknowledgments

EG and RM acknowledge the funding provided by the Wallenberg AI, Autonomous Systems, and Software Program (WASP), supported by the Knut and Alice Wallenberg Foundation. The computations and data storage were partially enabled by resources provided by Chalmers e-Commons. The computations and data storage were partially enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725. We thank Mike Preuss for fruitful discussions.

Competing Interests

EG and SG are employees of AstraZeneca. The authors declare no competing interests.

Software and Data Availability

The code will be made available on Github upon publication.

The PaRoutes datasets used in this study are publicly available and can be accessed at https://zenodo.org/records/7341155. Further details on the PaRoutes data sources are provided in the original publication (4).

References

[1] K. Nicolaou, "Organic synthesis: the art and science of replicating the molecules of living nature and creating others like them in the laboratory," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 470, no. 2163, p. 20130690, 2014.

[2] G. E. Vleduts, "Concerning one system of classification and codification of organic reactions," *Information Storage and Retrieval*, vol. 1, pp. 117–146, 1963.

[3] E. J. Corey and W. T. Wipke, "Computer-assisted design of complex organic syntheses: Pathways for molecular synthesis can be devised with a computer and equipment for graphical communication.," *Science*, vol. 166, no. 3902, pp. 178–192, 1969.

[4] S. Genheden and E. Bjerrum, "PaRoutes: towards a framework for benchmarking retrosynthesis route predictions," *Digital Discovery*, vol. 1, no. 4, pp. 527–539, 2022.

[5] K. Maziarz, G. Liu, H. Misztela, A. Kornev, P. Gaiński, H. Hoefling, M. Fortunato, R. Gupta, and M. Segler, "Chimera: Accurate retrosynthesis prediction by ensembling models with diverse inductive biases," *arXiv preprint arXiv:2412.05269*, 2024.

[6] A. M. Westerlund, S. Manohar Koki, S. Kancharla, A. Tibo, L. Saigiridharan, M. Kabeshov, R. Mercado, and S. Genheden, "Do Chemformers dream of organic matter? evaluating a transformer model for multistep retrosynthesis," *Journal of Chemical Information and Modeling*, vol. 64, no. 8, pp. 3021–3033, 2024.

[7] M. Cretu, C. Harris, J. Roy, E. Bengio, and P. Liò, "SynFlowNet: Towards molecule design with guaranteed synthesis pathways," in *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024.

[8] S. M. Kearnes, M. R. Maser, M. Wleklinski, A. Kast, A. G. Doyle, S. D. Dreher, J. M. Hawkins, K. F. Jensen, and C. W. Coley, "The open reaction database," *Journal of the American Chemical Society*, vol. 143, no. 45, pp. 18820–18826, 2021.

[9] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15084–15097, 2021.

[10] J. D. Shields, R. Howells, G. M. Lamont, L. Yin, A. Madin, C. Reimann, H. Rezaei, T. Reuillon, B. Smith, C. Thomson, Y. Zheng, and R. E. Ziegler, "AiZynth impact on medicinal chemistry practice at AstraZeneca.," *RSC Medicinal Chemistry*, vol. 15 4, pp. 1085–1095, 2024.

[11] Z. Zhong, J. Song, Z. Feng, T. Liu, L. Jia, S. Yao, T. Hou, and M. Song, "Recent advances in deep learning for retrosynthesis," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 14, no. 1, p. e1694, 2024.

[12] S. Genheden, A. Thakkar, V. Chadimová, J.-L. Reymond, O. Engkvist, and E. Bjerrum, "AiZynthFinder: a fast, robust and flexible open-source software for retrosynthetic planning," *Journal of Cheminformatics*, vol. 12, no. 1, p. 70, 2020.

[13] M. H. Segler, M. Preuss, and M. P. Waller, "Planning chemical syntheses with deep neural networks and symbolic AI," *Nature*, vol. 555, no. 7698, pp. 604–610, 2018.

[14] B. Chen, C. Li, H. Dai, and L. Song, "Retro*: learning retrosynthetic planning with neural guided A* search," in *International Conference on Machine Learning*, pp. 1608–1616, PMLR, 2020.

[15] S. Xie, R. Yan, P. Han, Y. Xia, L. Wu, C. Guo, B. Yang, and T. Qin, "RetroGraph: Retrosynthetic planning with graph search," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2120–2129, 2022.

[16] J. S. Schreck, C. W. Coley, and K. J. Bishop, "Learning retrosynthetic planning through simulated experience," *ACS Central Science*, vol. 5, no. 6, pp. 970–981, 2019.

[17] S. Liu, Z. Ying, Z. Zhang, P. Zhao, J. Tang, L. Lin, and D. Wu, "Metro: Memory-enhanced transformer for retrosynthetic planning via reaction tree," 2022.

[18] R. Irwin, S. Dimitriadis, J. He, and E. J. Bjerrum, "Chemformer: a pre-trained transformer for computational chemistry," *Machine Learning: Science and Technology*, vol. 3, no. 1, p. 015022, 2022.

[19] Y. Shee, H. Li, A. Morgunov, and V. Batista, "DirectMultiStep: Direct route generation for multi-step retrosynthesis," *arXiv preprint arXiv:2405.13983*, 2024.

[20] L. Lehnert, S. Sukhbaatar, D. Su, Q. Zheng, P. Mcvay, M. Rabbat, and Y. Tian, "Beyond A*: Better planning with transformers via search dynamics bootstrapping," *arXiv preprint arXiv:2402.14083*, 2024.

[21] P. Bhargava, R. Chitnis, A. Geramifard, S. Sodhani, and A. Zhang, "When should we prefer decision transformers for offline reinforcement learning?," *arXiv preprint arXiv:2305.14550*, 2023.

[22] D. Lowe, "Chemical reactions from US patents (1976–sep2016)," *figshare*, 2017.

[23] C. W. Coley, W. H. Green, and K. F. Jensen, "RDChiral: An RDKit wrapper for handling stereochemistry in retrosynthetic template extraction and application," *Journal of Chemical Information and Modeling*, vol. 59, no. 6, pp. 2529–2537, 2019.

[24] S. Genheden, P.-O. Norrby, and O. Engkvist, "AiZynthTrain: robust, reproducible, and extensible pipelines for training synthesis prediction models," *Journal of Chemical Information and Modeling*, vol. 63, no. 7, pp. 1841–1846, 2023.

[25] E. Heid, J. Liu, A. Aude, and W. H. Green, "Influence of template size, canonicalization, and exclusivity for retrosynthesis and reaction prediction applications," *Journal of Chemical Information and Modeling*, vol. 62, no. 1, pp. 16–26, 2021.

[26] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 38–45, Association for Computational Linguistics, Oct. 2020.

[27] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[28] G. Landrum, "RDKit: Open-source cheminformatics." `https://www.rdkit.org`.

[29] S. Genheden and J. D. Shields, "A simple similarity metric for comparing synthetic routes," *Digital Discovery*, vol. 4, no. 1, pp. 46–53, 2025.

[30] J. Li, L. Fang, and J.-G. Lou, "Retro-BLEU: quantifying chemical plausibility of retrosynthesis routes through reaction template sequence analysis," *Digital Discovery*, vol. 3, no. 3, pp. 482–490, 2024.

[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

[32] S. Genheden, O. Engkvist, and E. Bjerrum, "Clustering of synthetic routes using tree edit distance," *Journal of Chemical Information and Modeling*, vol. 61, no. 8, pp. 3899–3907, 2021.

[33] G. Yujia, M. Kabeshov, T. H. D. Le, S. Genheden, G. Bergonzini, O. Engkvist, and S. Kaski, "An expert-augmented deep learning approach for synthesis route evaluation," *ChemRxiv*, 2025.

[34] C. Kannas and S. Genheden, "Rxnutils–a cheminformatics python library for manipulating chemical reaction data," 2022.

[35] L. Saigiridharan, A. K. Hassen, H. Lai, P. Torren-Peraire, O. Engkvist, and S. Genheden, "AiZynthFinder 4.0: developments based on learnings from 3 years of industrial application," *Journal of Cheminformatics*, vol. 16, no. 57, 2024.

[36] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, and J. P. Overington, "ChEMBL: a large-scale bioactivity database for drug discovery," *Nucleic Acids Research*, vol. 40, pp. D1100 – D1107, 2011.

[37] S. Bühlmann and J. Reymond, "ChEMBL-likeness score and database GDBChEMBL," *Frontiers in Chemistry*, vol. 8, 2020.

Supplementary Information

RetroSynFormer: Planning multi-step chemical synthesis routes via a Decision Transformer

Authors: Emma Granqvist, Rocío Mercado, and Samuel Genheden

A Optimization and Reward Settings

In Tables S7–S8 we describe the search space for the DT and reward parameters, which were optimized using Optuna. In Table S9 we provide details on the reward parameters for the alternative reward functions described in Table 5.

Table S7: Search space for the DT parameters and the optimal values used for training the DT model.

Model parameter	Search range	Optimal value
`activation_function`	{`relu`, `silu`, `gelu`, `tanh`, `gelu_new`}	`relu`
`action_tanh`	{`true`, `false`}	`false`
`attn_pdrop`	$\{\{0.01x \mid x \in \mathbb{Z}, 1 \leq x \leq 20\}\}$	0.02
`embd_pdrop`	$\{\{0.01x \mid x \in \mathbb{Z}, 1 \leq x \leq 20\}\}$	0.2
`hidden_size`	$\{32, 64, 128, 256, 512, 1024, 2048\}$	256
`n_heads`	$\{2x \mid x \in \mathbb{Z}, 0 \leq x \leq 32\}$	4
`n_layers`	$\{2x \mid x \in \mathbb{Z}, 0 \leq x \leq 32\}$	26
`resid_pdrop`	$\{\{0.01x \mid x \in \mathbb{Z}, 1 \leq x \leq 20\}\}$	0.08
Settings		Value
# epochs		300
Maximizing		Success rate

Table S8: Search space for the reward parameters and the optimal values used for training the DT model.

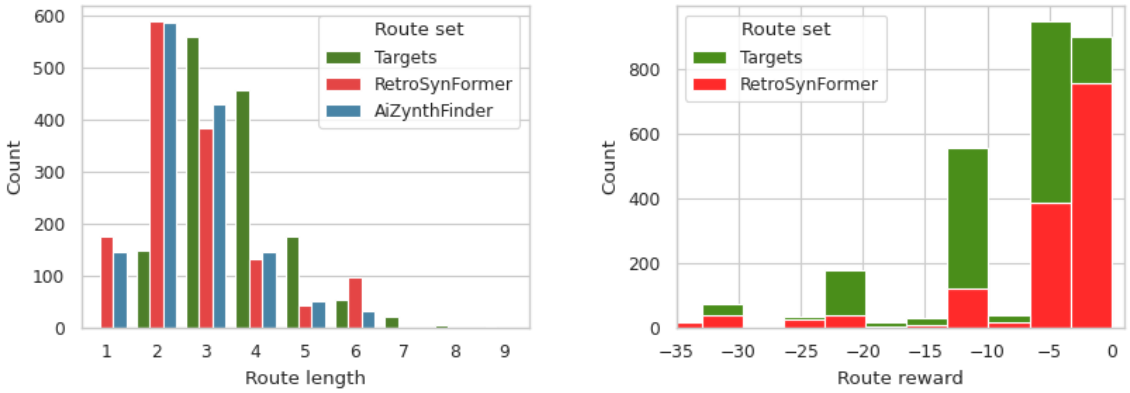
Hyperparameters	Search range	Optimal value
Building block reward	$\{0, 0.001, 0.01, 0.1, 0.25, 0.5, 1, 2, 4\}$	0
Building block scale with depth	$\{0, 0.01, 0.1, 0.5, 1, 2\}$	2
Intermediate reward	$\{0, -0.001, -0.01, -0.1, -0.25, -0.5, -1, -2, -4\}$	-2
Intermediate scale with depth	$\{0, 0.01, 0.1, 0.5, 1, 2\}$	1
Dead-end reward	$\{0, -0.001, -0.01, -0.1, -0.25, -0.5, -1, -2, -4\}$	-2
Dead-end depth	$\{0, 0.01, 0.1, 0.5, 1, 2\}$	2
Settings		Value
# epochs		100
Maximizing		Success rate

Table S9: The reward parameters for alternative rewards.

Label	Building block		Intermediate		Dead End	
	Reward	Scale	Reward	Scale	Reward	Scale
Default	0	-2	1	-2	-2	2
Increasing Building Block Reward	2	-2	1	-2	-2	2
Decreasing Building Block Reward	-2	-2	1	-2	-2	2
Remove Scaling Building Block Reward	2	0	1	-2	-2	2
Flipping Sign Intermediate Score	0	-2	1	2	-2	2
Remove Scaling Intermediate	0	-2	0	-2	-2	2
Flipping Sign Dead End Score	0	-2	1	-2	2	2
Remove Scaling Dead End	0	-2	1	-2	-2	0

B Complimentary Results

In Figure S8 we describe the route characteristics for the routes generated by the RetroSynFormer and AiZynthFinder on the N5 set. In Figures S9 and S10 we present histograms for the most common reaction templates in the N1 and N5 sets, respectively. Finally, in Table S10 we compare the performance of the RetroSynFormer and AiZynthFinder on the targets that are shared across the N1 and N5 test sets, respectively.



(a) Histograms of N5 route lengths, measured as the number of actions per route, for each route set.

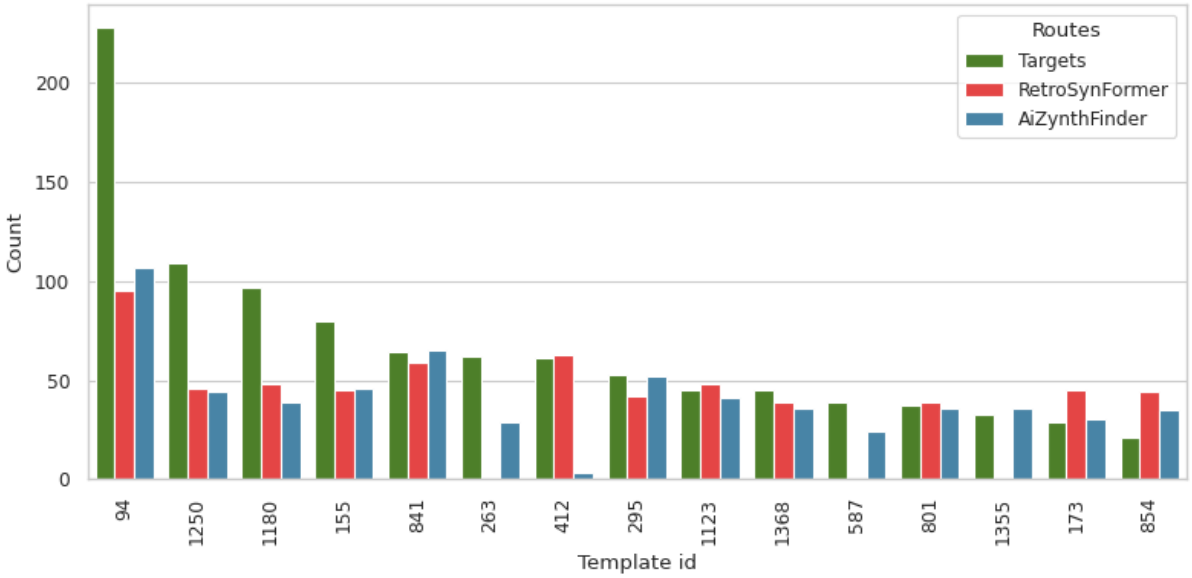
(b) Stacked bar plot showing the distribution of N5 route rewards in the target versus RetroSynFormer-predicted routes.

Figure S8: Route characteristics for routes for the N5 test set target routes compared to the RetroSynFormer and AiZynthFinder solved predictions. a) The median reward for the route length is 4 for the target routes and 2 for the RetroSynFormer and AiZynthFinder routes. b) The median reward for the route reward is -8 for the target routes and -2 for the RetroSynFormer and AiZynthFinder routes.

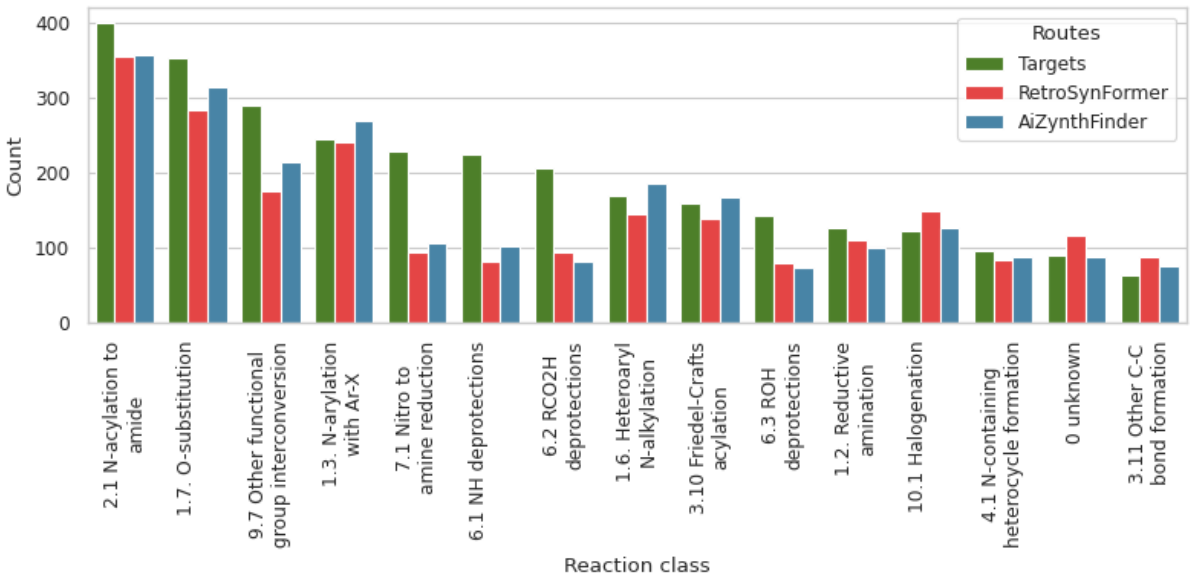
Table S10: Performance of retrosynthesis models for different template sets on the 232 N1 targets and 248 N5 targets that are common among across the test sets in Table 6.

Dataset	Model	Test set	Success rate	Top-1 Accuracy	TED	Avg. route length
Small	RetroSynFormer50	N1	0.805	0.098	6.033	2.609
		N5	0.839	0.094	5.294	2.37
	AiZynthFinder	N1	0.914	0.142	4.746	2.590
		N5	0.907	0.093	6.423	2.902
Standard	RetroSynFormer50	N1	0.953	0.102	5.314	2.309
		N5	0.898	0.052	6.901	2.397
	AiZynthFinder	N1	0.935	0.121	5.031	2.396
		N5	0.911	0.073	7.059	2.726
Large	RetroSynFormer50	N1	0.958	0.091	5.499	2.147
		N5	0.917	0.031	7.136	2.296
	AiZynthFinder	N1	0.940	0.099	5.293	2.362
		N5	0.927	0.065	7.262	2.748

* RetroSynFormer results show averages over three runs. Standard deviations for all RetroSynFormer models are ≤ 0.02 for success rate, ≤ 0.02 for top-1 accuracy, ≤ 0.15 for TED, and ≤ 0.05 for route length.

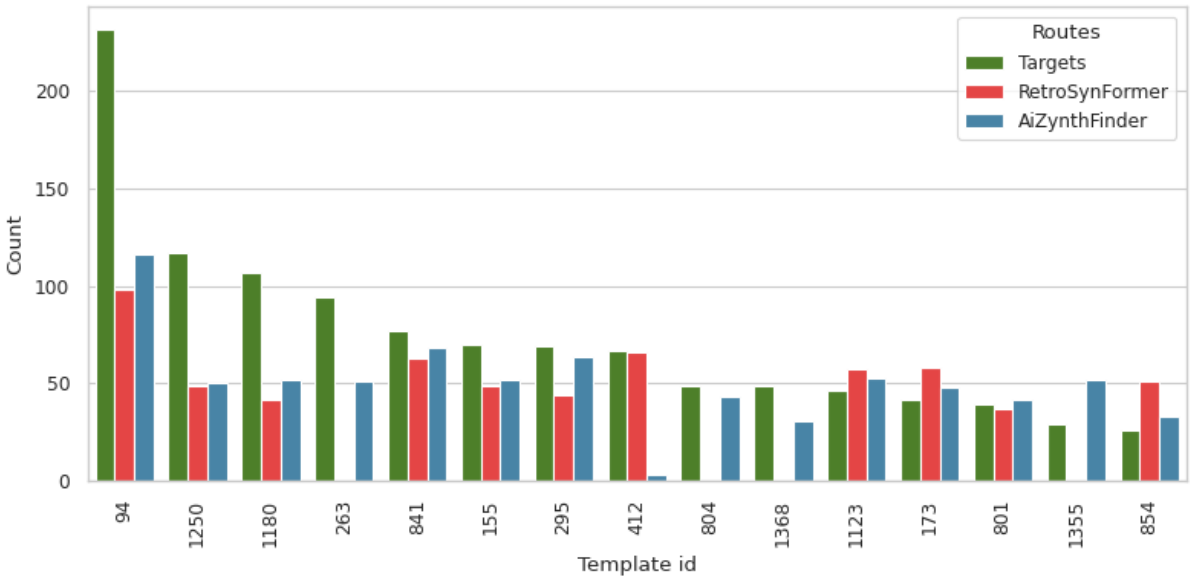


(a) Reaction template frequency distribution.

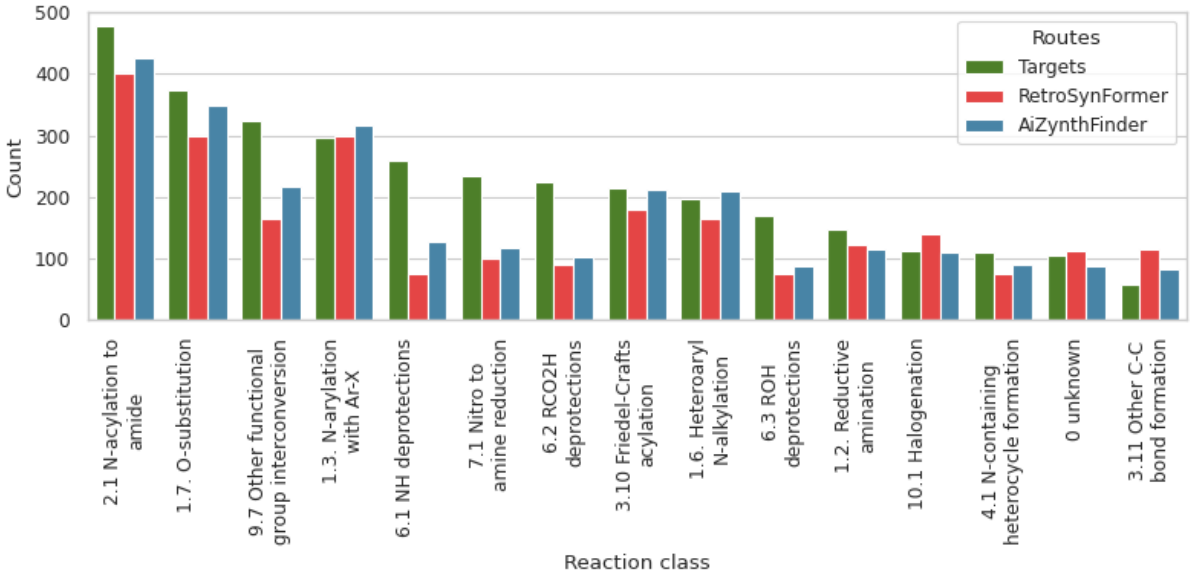


(b) Reaction classes frequency distribution.

Figure S9: Histograms of predicted reaction templates and classes for the N1 test set. a) Comparison of the counts for the 15 most common predicted templates compared to ground truth and baseline. Illustrations of the template IDs can be found in Table S11. b) Comparison of the counts for the 15 most common predicted reaction classes compared to ground truth and baseline.



(a) Reaction template frequency distribution.



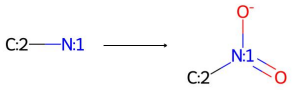
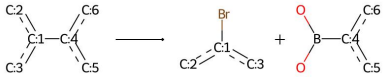
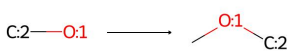
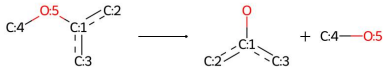
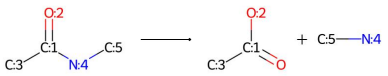
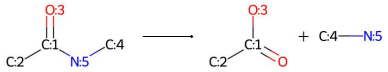
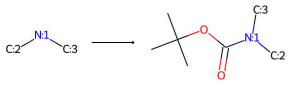
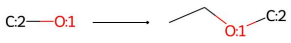
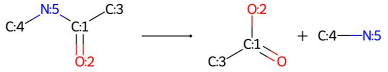
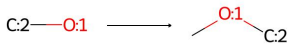
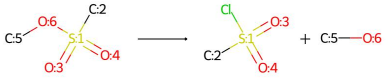
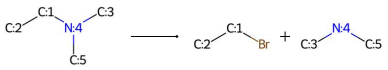
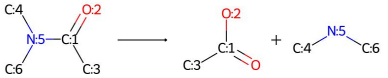
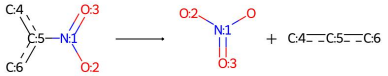
(b) Reaction classes frequency distribution.

Figure S10: Histograms of predicted reaction templates and classes for the N5 test set. a) Comparison of the counts for the 15 most common predicted templates compared to ground truth and baseline. Illustrations of the template IDs can be found in Table S11. b) Comparison of the counts for the 15 most common predicted reaction classes compared to ground truth and baseline.

C Reaction Templates

In Table S11, we present an illustration of the most commonly occurring templates in the predicted reactions from the RetroSynFormer.

Table S11: Visualization of the most common templates mentioned in Figure S9a and S10a.

Template id	Template	Template id	Template
94		841	
155		854	
173		1123	
263		1180	
412		1250	
587		1355	
801		1368	
804