Machine Learning: Science and Technology

TECHNICAL NOTE • OPEN ACCESS

Artificial applicability labels for improving policies in retrosynthesis prediction

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 017001 https://doi.org/10.1088/2632-2153/abcf90

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
27 July 2020

REVISED
20 October 2020

ACCEPTED FOR PUBLICATION
1 December 2020

PUBLISHED
24 December 2020

TECHNICAL NOTE

Artificial applicability labels for improving policies in retrosynthesis prediction

Esben Jannik Bjerrum[1], **Amol Thakkar**[1,2] **and Ola Engkvist**[1]

[1] Molecular AI, Discovery Sciences, BioPharmaceuticals R&D, AstraZeneca, Gothenburg 431 50, Sweden
[2] Department of Chemistry and Biochemistry, University of Bern, Bern CH-3012, Switzerland

E-mail: esben.bjerrum@astrazeneca.com

Keywords: retrosynthesis prediction, policy network, policy guided tree search, reaction rule applicability

Supplementary material for this article is available online

Abstract

Automated retrosynthetic planning algorithms are a research area of increasing importance. Automated reaction-template extraction from large datasets, in conjunction with neural-network-enhanced tree-search algorithms, can find plausible routes to target compounds in seconds. However, the current method for training neural networks to predict suitable templates for a given target product leads to many predictions that are not applicable *in silico*. Most templates in the top 50 suggested templates cannot be applied to the target molecule to perform the virtual reaction. Here, we describe how to generate data and train a neural network policy that predicts whether templates are applicable or not. First, we generate a massive training dataset by applying each retrosynthetic template to each product from our reaction database. Second, we train a neural network to perform near-perfect prediction of the applicability labels on a held-out test set. The trained network is then joined with a policy model trained to predict and prioritize templates using the labels from the original dataset. The combined model was found to outperform the policy model in a route-finding task using 1700 compounds from our internal drug-discovery projects.

1. Introduction

Using computers to assist with retrosynthetic planning is a decades-old idea [1]. The ability to search efficiently for synthetic routes to target compounds is of paramount interest in drug discovery and materials design, where molecular candidates are often first virtually designed by a chemist's intuition or rationale, via computer-aided drug design [2], *de-novo* design [3] or selected via virtual screening [4], but must be synthesized for evaluation in, for example, biological assays and evaluation of their physico-chemical properties to provide information for further optimization. Moreover, finding alternative routes for established chemical synthesis is important as part of process optimization, which allows more resource-efficient or green methods of chemical synthesis to be discovered [5]. Research into, and interest in, automated retrosynthesis planning have been invigorated by the successful application of deep learning and artificial-intelligence approaches [6]. Here, large reaction databases are used as a knowledge base to extract reaction templates that represent generalized reactions. Typically, the reaction templates are extracted as SMIRKS patterns containing the atoms that change during the reaction (the core), and the nearest atoms up to a given bond distance from the reaction core [7]. These patterns can be reversed and used to break a given target product into its proposed reactants. Applying the templates to the products will often give intermediates that are not available from stock or purchasable and further retrosynthetic steps are needed to create a plausible synthetic route to an available starting material. As multiple possible disconnections are available in the template library, this search for synthesis routes can be viewed as a tree-search problem. However, as the reaction-template libraries often exceed 100 000 templates, this quickly leads to an intractable computational problem. Already, with only two steps of disconnection, up to $100\,000^2 = 1\,000\,000\,000\,000$ combinations would need to be evaluated. Using neural networks in conjunction with the Monte Carlo tree-search algorithm can, however, permit efficient tree search, as

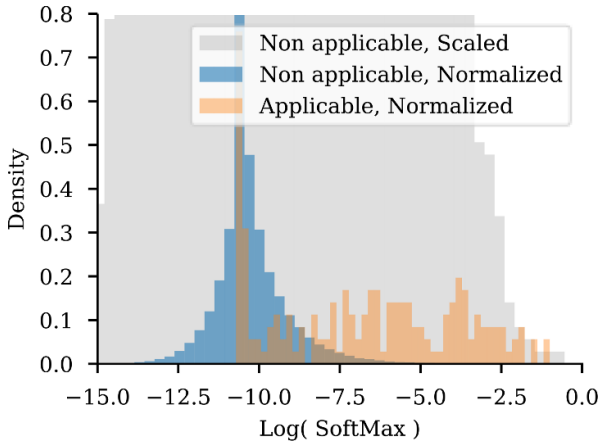

Figure 1. Histograms of probability predictions for an exemplary product. The applicable templates (orange) and the non-applicable templates (blue) appear to be well separated by the policy network. However, the scaled non-applicable histogram (grey, scaled to match the normalized applicable) reveals that the policy network wrongly predicts many non-applicable templates in the top probability range, due to massive differences in the numbers of positive and negative samples.

demonstrated by the 3N-MCTS algorithm [6]. Here, knowing the association between products from the reactions in the database and the extracted templates, deep neural networks are trained to predict which templates should be used on the target products. These policy networks give a strong prior search direction as well as template prioritization and branch pruning for the subsequent tree-search algorithm used to search for plausible routes. Often, only the top prioritized templates, e.g. the top 50, are used, out of several hundred thousand available templates. This filtering efficiently limits the branching factor in the tree-search algorithm, which would otherwise lead to infeasible computation times. Frequently, the algorithms can construct synthetic routes to target compounds from stock reactants in seconds.

However, while implementing an algorithm for retrosynthesis prediction [8], we noted that these policies often have many false positive templates in the top 50 predictions, leading to a low number of successful templates in the top 50 predictions. These templates lead to wasted computational time trying to apply the templates. We hypothesized that this may be because our dataset contains many false negatives. In principle, if a product is not associated with a template, it does not mean that that reaction could not have been used to make that product, but just that the reaction is not part of the reaction database. When testing the *in silico* application of the templates, we observed that the policy network seems to make a good distinction between the non-applicable and applicable templates (orange and blue histograms in figure 1).

Unfortunately, this discrimination is only clear when the histograms are both normalized individually. Often, only a few hundred templates are applicable for any given product, compared to the hundreds of thousands of non-applicable templates. This large imbalance in the amount of applicable vs. non-applicable templates leads to a situation where the applicable templates are not separated from the non-applicable templates. The grey histogram in figure 1 shows the situation where the histogram of the non-applicable templates has been scaled to match the applicable histogram (figure 1; grey histogram). This enlargement of the thin tail of the non-applicable templates explains why the top 50 prioritized templates often contain many non-applicable templates.

Here, we show how it is possible to train an applicability filter network to remove the non-applicable templates from the policy model predictions. In principle, this can be done with serial application and testing on a CPU but transferring this to a prediction task using a GPU enables efficient embedding in the final algorithm. The combined model was found to have increased route-finding capabilities in a retrosynthetic test using 1700 compounds from our internal electronic lab notebooks.

2. Methods

2.1. Molecular representation

The molecules are represented as Morgan fingerprints, which are an RDKit [9] implementation of extended connectivity fingerprints [10], where a detailed description is available. Briefly explained, the fingerprints use a variation of the Morgan algorithm [11] to create a selection of unique keys based on atomic invariants and their topological arrangements. Each atom is encoded using an invariant code that is derived from seven properties of the atom which are encoded as information by being hashed to a 32 bit key:

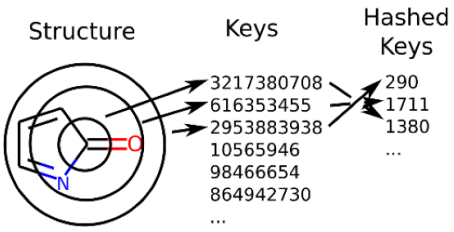

Figure 2. Extended connectivity fingerprints [10], such as Morgan fingerprints [9], encode the atoms and their environments up to a given radius as unique keys. The (often large) keys are then converted to a fixed fingerprint size via a hashing algorithm.

(a) the number of immediate neighbors which are 'heavy' (nonhydrogen) atoms;
(b) the valence minus the number of hydrogen atoms;
(c) the atomic number;
(d) the atomic mass;
(e) the atomic charge;
(f) the number of attached hydrogen atoms (both implicit and explicit);
(g) whether the atom is contained in at least one ring.

The invariants are converted to Morgan keys via an analysis of the neighborhood up to a given radius. Each key thus describes both the atomic properties and its molecular neighborhood. Figure 2 illustrates how a single carbon atom yields three keys, corresponding to radii of zero, one and two, which are subsequently mapped to the final fingerprint bits via a hashing algorithm.

2.2. Dataset and label generation

A combined dataset of Reaxys, USPTO, Pistachio and our internal data was prepared as described previously [8]. The final template set was filtered for a minimum of two occurrences and grouped based on the identities of the products into a multilabel dataset, as described elsewhere [12]. The final dataset consisted of 4 155 352 products associated with 159 411 templates. For each product, the reaction template was applied, and a label of 'one' was assigned if the reaction template was able to be applied and a label of 'zero' otherwise. The application was parallelized by splitting the dataset into 2048 parts which were submitted for processing on individual computational nodes in the cluster using a simple linux utility for resource management (SLURM) array job. The script included a 60 s time-limit on the application of templates to a given product. If not all templates were able to be applied to a product within the time limit, the computation was stopped, and the product was removed from the final dataset. Finally, the Morgan bit fingerprints of the products for a radius of two, folded to a length of 2048 bits, were calculated using RDKit [9] and stored in SciPy sparse arrays [13]. The choice of fingerprint was fixed to enable compatibility with our existing implementation of the tree-search code [8].

2.3. Neural networks

Neural networks were created using Keras [14] 2.2.4 with TensorFlow 1.12 [15] as a backend. The 'SoftMax' neural networks consisted of five hidden dense layers of 1024 neurons with skip connections to the output of the $n-2$ layer, followed by a 'bottleneck' dense layer of 256 neurons with the ELU activation function. A dropout of 0.35 was applied after each rectified linear unit (ReLU) activation function. The outputs were provided by 159 411 neurons with a kernel L2 regularization of 0.005 before the final SoftMax activation function. The categorical cross-entropic loss function was used. Training was conducted using the original database labels for a total of 150 epochs using the Adam optimizer with default settings, except for the learning rate, which was adjusted during training. The learning rate was adjusted using a scheme with a two-epoch warmup phase from 10^{-3} to a maximum learning rate of 4×10^{-3}. The learning rate was fixed for 15 epochs, followed by a stepwise exponential decay for the remaining epochs. The exponential decay was adjusted so that the final learning rate was 10^{-6}.

The 'Applicability' models had a similar architecture as regards skip connections, but only three ReLU layers and the final layer used a sigmoid activation function with a binary cross-entropic loss function. The applicability model was trained for a total of 50 epochs with a two-epoch warmup period, a ten-epoch fixed learning rate and exponential decay and learning rates as described for the SoftMax model.

Post training, both models were loaded and combined into a new Keras model. The input layer was shared. The final output tensor from the applicability model was binarized with a threshold of 0.1 using TensorFlow operations before being elementwise multiplied with the output tensor from the SoftMax model,

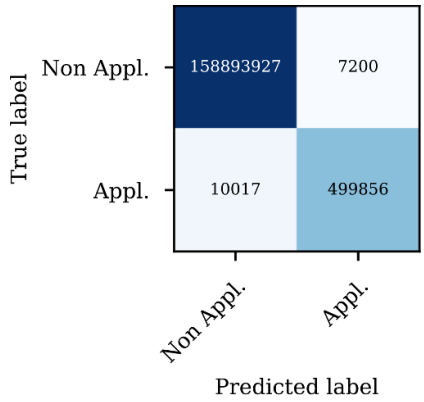

Figure 3. Confusion matrix of the predictions of the applicability labels for 1000 compounds from the test-set.

giving the final predictions of the combined model. The architecture of the model is shown in supplementary figure 1 (available online at stacks.iop.org/MLST/2/017001/mmedia).

2.4. Route-finding

The dataset of 1731 compounds used for a previous analysis was subjected to route finding using the tree-search code as described earlier [8]. If a route was found to stock compounds within 180 s, the route-finding was considered a success and the time was recorded. Both the standard model and the combined model were tested using the same settings for route finding. The exploration/exploitation constant, C, was set to 1.4, and the maximum number of transforms was set to 6. Template expansion was set to a maximum of 25 or a cumulative cutoff of 0.999999, and the time limit in seconds was set to 180. The stocks used were a combination of ACD, Enamine and in-house reagent stocks.

Route finding was performed on cluster nodes equipped with Intel(R) Broadwell Xeon(R) E5-2683 v4 CPUs @ 2.10 GHz with 128 GB RAM, from which 10 GB was reserved for running. Neural network inference was performed on the CPU.

3. Results

3.1. Applicability label calculation

The 60 s timeout applied during application-label calculation resulted in the removal of 1277 compounds. The size distribution of the removed compounds is slightly shifted to the higher side, compared to the size distribution of the compounds where labels were successfully calculated (supplementary figure 2). A closer investigation of the compounds that timed out revealed that the templates contained many repeating patterns and functional groups such as alcohol groups (examples in supplementary figure 3). Likewise, the slowest template contained 12 repeating units that matched alcohol groups (supplementary figure 3). The timeout thus appears to be triggered by a combinatorial application problem whereby, for example, all combinations of alcohol groups matching SMARTS in the template needed to be matched to all alcohol groups. For 12 SMARTS and 12 alcohol groups, this led to $12! = 479\,001\,600$ combinations. A further 209 compounds were removed, as they had no calculated applicable templates. Thus, the final dataset for training consisted of 4 153 866 compounds.

3.2. Performance of the applicability filter network

The applicability model converged with a validation loss of 0.0003 and a validation accuracy of 0.9999. Calculating the area under the receiver-operator curve (AUC-ROC) of molecules from the test-set often showed a near-ideal performance of 0.99999. However, both accuracy and AUC-ROC are misleading metrics, due to the highly imbalanced labels in the dataset. This is evident from the confusion matrix shown in figure 3. Here the 158 million correct predictions of non-applicable templates outnumber the wrongly predicted labels by orders of magnitude. If the network predicts that a template is not applicable, it is correct over 99.995% of the time (true negative rate). Rather, as the model is intended to be a filter network, we are only interested in samples that are predicted as applicable. The model's performance is thus better measured using 'Recall' and 'Precision', i.e. measuring the amount of correctly predicted applicable templates and the predicted percentage of false positives, respectively. Recall is also known as the true positive rate and is calculated as $TP/(FN + TP)$, whereas Precision is defined as $TP/(TP + FP)$. Recall measures how many of the

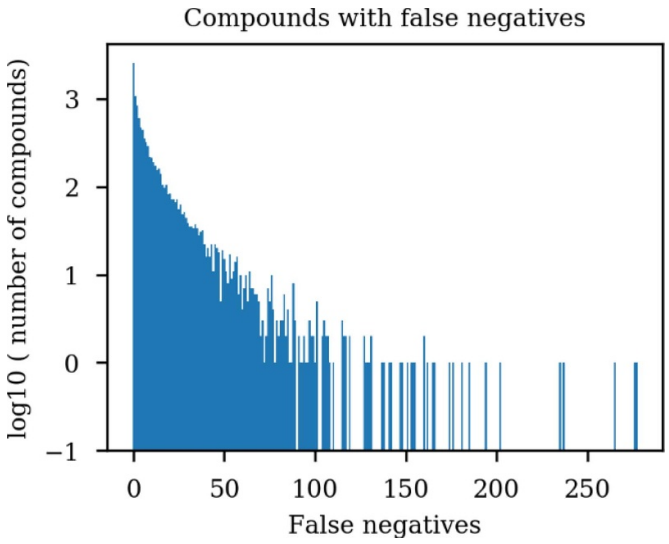

Figure 4. Distribution of the number of false negatives per compounds of the test set.

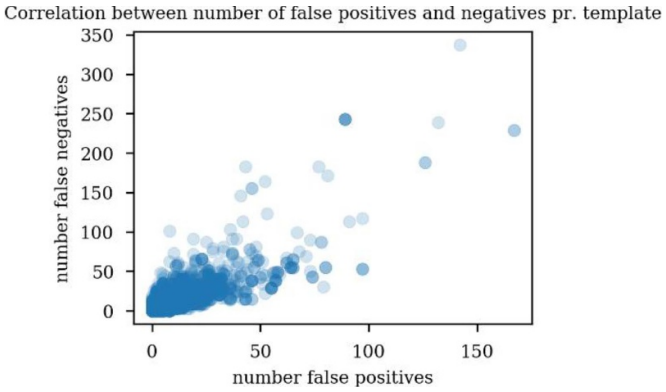

Figure 5. Scatter plot showing the correlation between the number of false positives and false negatives of the test set per template.

applicable templates we recover and Precision measures how many of the templates we predict to be applicable are actually applicable. Impressively, Recall is still high at 0.98 with a Precision of 0.99.

The speed of prediction was compared by the prediction of 1000 compounds on a workstation equipped with an RTX 2080 Ti graphics card. SoftMax alone had a prediction time of 4.4 ms on average, and the applicability model was faster with an average prediction time of 2.4 ms. The combined model had an average prediction time of 6.1 ms. This reflects the number of parameters in the networks, with a possible overhead from I/O to the GPU. In comparison, the application time required to check the applicability of approximately 159 000 templates one by one was, on average, approximately 2 s.

3.3. Error analysis

The distribution of errors follows a distribution whereby most compounds have a few errors, whereas a few compounds have many errors. The distribution of false negatives per compound is shown in figure 4. False negatives follow a qualitatively similar distribution, as do the errors per template (SI figures 5–7). There was no correlation between false negatives and positives per compound, whereas there was a correlation between the numbers of false positives and negatives per template (figure 5). It thus appears that some templates are more error prone than others. Examination of the most error-prone templates showed many templates where two similar reaction sites in the product are involved in the template's application (see the example in figure 6). This is probably due to the use of bit fingerprints, where similar fingerprint bits for each duplicate atom make them indistinguishable from the unduplicated version. The neural network thus has no chance of discriminating between a compound with duplicated features and one without, and the template cannot be applied to the product without duplicated features.

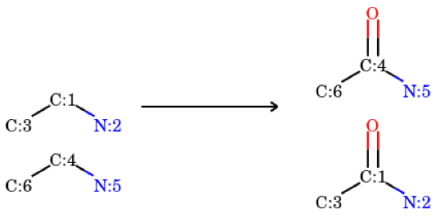

Figure 6. Example of a problematic template displaying duplicated features giving a high number of errors on the test set.

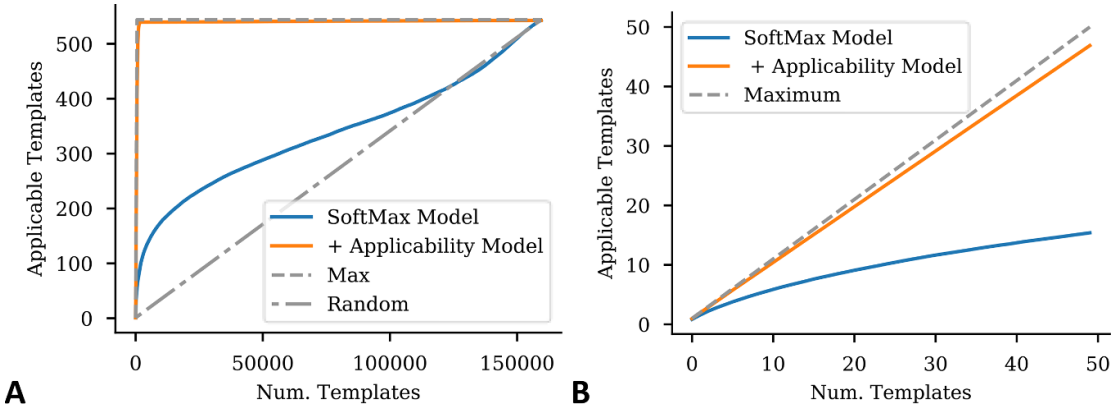

Figure 7. Prioritization of templates by the SoftMax and combined models. The templates were sorted according to the model's prediction and the cumulative number of applicable templates recorded. The curves are an average of 1000 compounds. (a) The full curve. (b) Early enrichment of the top 50 templates.

3.4. Prioritization of templates

An important part of the policy network's function is to enrich the top predictions with plausible and applicable templates. Figure 7 shows a comparison between the standard SoftMax model and the combined model. Overall, the combined model is closer to the theoretical maximum. However, it is the early enrichment that is of particular interest, as shown in panel B. Here, filtering for applicability leads to a large increase in the number of applicable templates in the top 50. These can then be considered during the tree-search algorithm. As the applicability prediction is binarized before multiplication with the SoftMax prediction, the order of the applicable templates is not changed.

The combined model thus shows a very good separation between applicable and non-applicable templates, as is evident from figure 8. There is a clear separation between the two categories of templates and even in the non-normalized instance, there are only a few instances of non-applicable templates predicted in the range that contains applicable templates (small shadows between log SoftMax −10 and −5). Most of the non-applicable templates have been assigned a value of 10^{-18} if they were set to zero, and thus appear in the plot as a single bar at −18.

3.5. Route-finding capability

The two models were tested regarding route-finding capability on a dataset of 1731 in-house project compounds used in a previous study [8]; the results are shown in figure 9. The two models show a similar performance for the first 40 s, whereas the combined network outperforms the standard model after 50 s of runtime. As the prioritization is not changed by the combined network, the same templates are most likely to be tried in the early phase of the tree search when routes are found for easy compounds, whereas the higher number of applicable templates available after prediction by the combined model allows the algorithm to find routes in the later phase of the tree search for more difficult cases. The SoftMax policy and applicability-filtered policy led to 87% and 91% of solved routes, respectively. Thus, 13% of the routes were not able to be solved with the SoftMax policy, which decreased to 9% with the applicability-filtered policy. This constitutes a 30% decrease in the unsolvable compounds. The slightly longer prediction time of the combined model (6.1 ms vs 4.4 ms) may explain the slight performance advantage the standard model has in the early phase of the tree search.

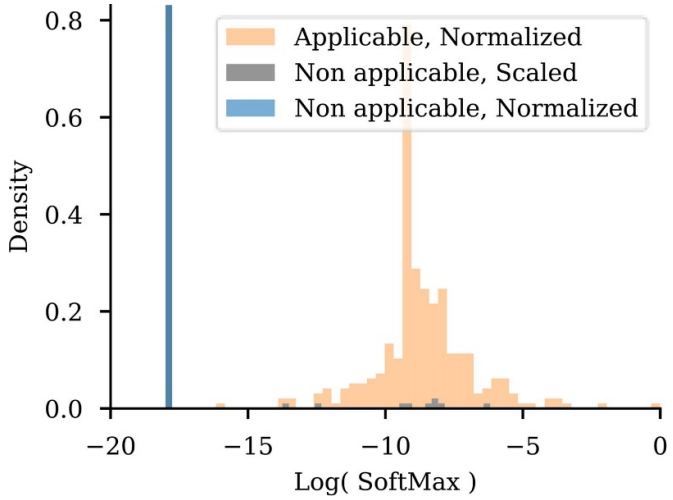

Figure 8. Histogram comparison of the SoftMax predictions of the combined model. Predictions of zero were set to 10^{-18} before logarithmic conversion. Only a few non-applicable templates are intermixed with the applicable ones (small shadows between log SoftMax −10 and −5). *Y*-axis truncated at 0.8. Compare with figure 1.

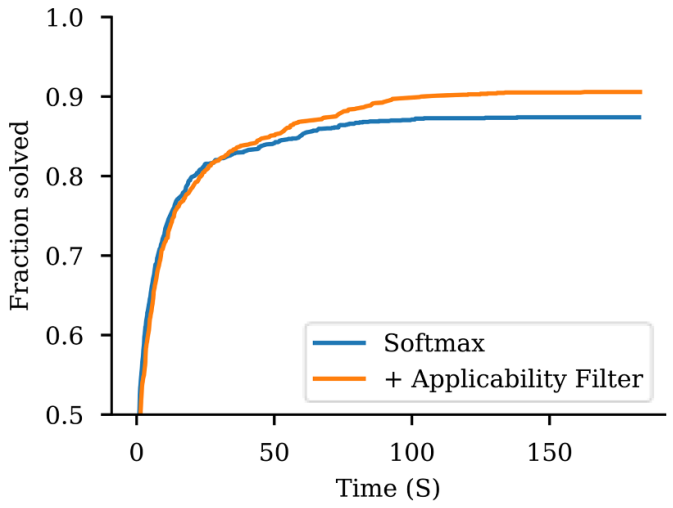

Figure 9. Route-finding capability of the two models for a dataset of 1731 in-house project compounds. The combined model shows an increased route-finding capability for run times above 50 s.

4. Discussion

The filter network shows a suspiciously high discriminatory power, but it must be kept in mind that the dataset is essentially noise free, as it is artificially created. Moreover, the information needed for discrimination is encoded directly into the molecular representation. Figure 10 shows an example SMARTS definition from the product side of a template. The atoms are specified in square brackets via atomic type and eventual degree, hydrogen count and charge. The structural topology is mapped via bonds (– single, = double), and the branching is specified with parentheses. The degree of an atom is defined by its number of directly bonded neighbors and is independent of bond orders; however, explicit hydrogen atoms are included. Thus, as explicit hydrogen atoms are usually not present in the templates, this will correspond to the first atomic property included in the Morgan atomic invariants, as described in the 'methods' section. The number of hydrogens attached to the atom (implicit and explicit) can be inferred from the other atomic properties which compose the atom's invariant keys in the Morgan algorithm. The charge is also a direct part of the atom's invariant calculation. Therefore, there is a direct mapping between the atomic invariants and the specification of the atom in SMARTS. This means that if there is a key in the Morgan fingerprint of a product that is not mapped in SMARTS, the template can be immediately discarded. Furthermore, the structural information will also be included in the Morgan fingerprint, which can be important for discrimination. The SMARTS definition shown in figure 10 will not match the structure in figure 2, even

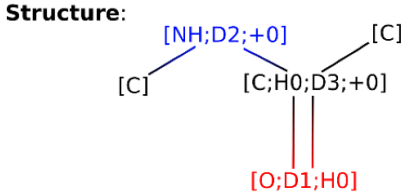

Figure 10. Example of a SMARTS definition, both in linear form and mapped to a molecular graph. The atomic definitions in square brackets specify the number of hydrogen atoms (H) and the degree (connections to heavy atoms and formal charge).

though it seemingly contains the correct atoms, there is a single bond specified between the carbon and nitrogen atoms to the left, which is a double bond in the structure in figure 2.

In our initial attempts at network architectures, the model had shared layers and only diverged at the final output layers. We did this because our hypothesis was that the learned features would be synergistic to both tasks, in a similar way to that observed for multi-task learning models. However, this turned out not to work as expected. The network hyperparameters were able to be tuned to have either good prioritization (accuracy of the SoftMax model) or a good filtering capability (recall of the applicability prediction), neither of which was able to perform as well as models trained on their specific kind of label on their own. Moreover, we observed that a high recall was important for the route-finding capability. A recall as high as 0.9 still led to a decreased route-finding capacity in comparison with the standard SoftMax model, probably because some of the 10% of removed templates were crucial to finding routes to some compounds. These observations are in contrast to the findings of a recent paper [16], where artificial labels were used to pretrain a network that was then subsequently fine-tuned directly on the database labels to yield the final policy model. The benefits of transfer learning suggest that at least some level of synergy is to be expected, which was not observed here, and led us to the final training regime of separate training and combination into a compound model post-training with explicit TensorFlow operations.

The results show that it is possible to improve the route-finding capability of template-based policy-guided retrosynthetic tree-search algorithms using artificially created labels. As the template applicability is binarized, it does not change the order of the applicable templates predicted with the SoftMax layer. There would be no expected difference between the outcome if instead of the top 50 predictions of the standard model, the number was expanded to perhaps the top 2000, and the non-applicable templates sorted out. This would, however, lead to many wasted template trials during processing of the tree-search algorithm and consequent prolonged runtimes. The solution proposed here is a plug-and-run solution for our existing route-finding software that has a minimal computation-time overhead.

In this study, we did not assess the plausibility and quality of the proposed routes. The higher number of templates tested for each expansion carries an inherent risk that, reactions are suggested that are applicable *in silico*, but chemically infeasible. This would then need to be counteracted with a feasibility model, as also included in the original 3N-MCTS algorithm [6].

It is also worth noting that the applicability-filter network proposed here is a very strong and accurate filter, which could in turn enable even more exhaustive searches for retrosynthetic routes. As an alternative, the standard policy network could be skipped entirely. As the applicability filter already limits the branching factor to a few hundred applicable reactions, these could be passed through a feasibility test before being used in the tree-search algorithm and still give realistic search times for exhaustive searching. As the filtering is not based on existing reactions, the occurrence of a template reflecting the popularity of the associated reactions in the dataset would not influence the filtering. However, as more frequently occurring reaction types can give rise to several different templates, there is a chance that a given product will still be associated with given reaction classes more often, as there simply are more templates to match. On the other hand, the different templates arise due to differences in the environment around the reaction core and will not have the same applicability. More seldomly occurring reaction templates could thus get a more even chance of being used, with a higher number of alternative routes being suggested. This would probably be more useful in a process optimization setting than a medicinal chemistry scenario, as the former will prioritize the difference of the routes considered over well-known reactions that will just give enough compound to be tested. This clear division between a network that limits the number of choices to the applicable, and a second feasibility network that tests these results one-by-one, considering the full scope of the reaction, could lead to an even better route-finding capability where the quality of the route finding can be directly tuned by the settings and

the performance of the feasibility filter. If the template quality is high and already contains rules that limit the scope of the application, similarly to the expert rules used in Chematica (now Synthia™) [17], the applicability filter could significantly speed up the search time, with no impact on the quality of the final routes.

5. Conclusions

Artificial applicability labels can be used to increase the route-finding capability of template-based policy-guided tree-search algorithms for retrosynthetic route finding. The trained applicability-filter network has near-perfect performance and can be combined directly with existing SoftMax models with minimal computational overhead due to the high parallelism of the GPU. The applicability filter represents a new way of limiting the branching factor of the tree-search algorithm which is independent of the popularity of the reactions and can be used in alternative algorithmic designs with better templates or feasibility models.

Funding

Amol Thakkar is supported financially by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie Grant Agreement No. 676434, 'Big Data in Chemistry' ('BIGCHEM' http://bigchem.eu).

Data availability statement

No new data were created or analyzed in this study.

ORCID iDs

Esben Jannik Bjerrum https://orcid.org/0000-0003-1614-7376
Amol Thakkar https://orcid.org/0000-0003-0403-4067
Ola Engkvist https://orcid.org/0000-0003-4970-6461

References

[1] Corey E J and Wipke W T 1969 Computer-assisted design of complex organic syntheses *Science* **166** 178–92
[2] Muegge I, Bergner A and Kriegl J M 2017 Computer-aided drug design at Boehringer Ingelheim *J. Comput. Aided. Mol. Des.* **31** 275–85
[3] Kotsias P-C, Arús-Pous J, Chen H, Engkvist O, Tyrchan C and Bjerrum E J 2020 Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks *Nat. Mach. Intell.* **2** 254–65
[4] Chen H, Kogej T and Engkvist O 2018 Cheminformatics in drug discovery, an industrial perspective *Mol. Inform.* **37** 1800041
[5] Roschangar F, Sheldon R A and Senanayake C H 2015 Overcoming barriers to green chemistry in the pharmaceutical industry—the green aspiration level™ concept *Green Chem.* **17** 752–68
[6] Segler M H S, Preuss M and Waller M P 2018 Planning chemical syntheses with deep neural networks and symbolic AI *Nature* **555** 604–10
[7] Coley C W, Green W H and Jensen K F 2019 RDChiral: an RDKit wrapper for handling stereochemistry in retrosynthetic template extraction and application *J. Chem. Inf. Model.* **59** 2529–37
[8] Thakkar A, Kogej T, Reymond J-L-L, Engkvist O and Bjerrum E J 2019 Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain *Chem. Sci.* **11** 1
[9] 2019 RDKIT: open source cheminformatics (available at: www.rdkit.org) (Accessed: 12 September 2019)
[10] Rogers D and Hahn M 2010 Extended-connectivity fingerprints *J. Chem. Inf. Model* **50** 742–54
[11] Morgan H L 1965 The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service *J. Chem. Doc.* **5** 107–13
[12] Thakkar A, Selmi N, Reymond J-L, Engkvist O and Bjerrum E J 2020 'Ring Breaker': assessing synthetic accessibility of the ring system chemical space *J. Med. Chem* 63 8791–8808
[13] Virtanen P *et al* 2020 SciPy 1.0: fundamental algorithms for scientific computing in Python *Nat. Methods* **17** 261–72
[14] Chollet F 2015 Keras (https://keras.io/)
[15] Abadi M *et al* 2016 TensorFlow: a system for large-scale machine learning *Proc. 12th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2016* May 265–83
[16] Fortunato M E, Coley C W, Barnes B C and Jensen K F 2020 Data augmentation and pretraining for template-based retrosynthetic prediction in computer-aided synthesis planning *J. Chem. Inf. Model.* **60** 3398–407
[17] Klucznik T *et al* 2018 Efficient syntheses of diverse, medicinally relevant targets planned by computer and executed in the laboratory *Chem* **4** 522–32