Learning Graph Models for Template-Free Retrosynthesis

Vignesh Ram Somnath [1] **Charlotte Bunne** [1] **Connor W. Coley** [2] **Andreas Krause** [1] **Regina Barzilay** [3]

Abstract

Retrosynthesis prediction is a fundamental problem in organic synthesis, where the task is to identify precursor molecules that can be used to synthesize a target molecule. Despite advancements in neural retrosynthesis algorithms, they are unable to fully recapitulate the strategies employed by chemists and do not generalize well to infrequent reactions. In this paper, we propose a graph-based approach that capitalizes on the idea that graph topology of precursor molecules is largely unaltered during the reaction. The model first predicts the set of graph edits transforming the target into incomplete molecules called *synthons*. Next, the model learns to expand synthons into complete molecules by attaching relevant *leaving groups*. Since the model operates at the level of molecular fragments, it avoids full generation, greatly simplifying the underlying architecture and improving its ability to generalize. The model yields 11.7% absolute improvement over state of the art approaches on the USPTO-50k dataset.

1. Introduction

Retrosynthesis prediction, first formalized by E.J. Corey (Corey, 1991), is a fundamental problem in organic synthesis that attempts to identify a series of chemical transformations for synthesizing a target molecule. In the single-step formulation, the task is to identify a set of reactant molecules given a target. Beyond simple reactions, many practical tasks involving complex organic molecules are difficult even for expert chemists. This has motivated interest in computer-assisted retrosynthesis (Corey & Wipke, 1969), with a recent surge in machine learning methods (Coley et al., 2017; Dai et al., 2019; Zheng et al., 2019).

The key computational challenge is how to efficiently explore the combinatorial space of reactions that can yield the target molecule. Existing machine learning methods for retrosynthesis prediction fall into template-based (Coley et al., 2017; Dai et al., 2019; Segler & Waller, 2017) and template-free approaches (Chen et al., 2019; Zheng et al., 2019). Template-based methods match the target molecule against a large set of templates, which are molecular subgraph patterns that highlight changes during a chemical reaction. Despite their interpretability, these methods suffer from poor generalization to new and rare reactions. Template-free methods bypass templates by learning a direct mapping from the SMILES representations (Weininger, 1988) of the product to reactants. Despite their greater generalization potential, these methods generate reactant SMILES character by character, increasing generation complexity and which in turn negatively impacts their complexity.

In this paper, we propose a retrosynthesis model which provides the generalization capacity of template-free models without resorting to full generation. This is achieved by learning to maximally reuse and recombine large fragments from the target molecule. This idea is grounded in the fundamental property of chemical reactions — the molecular graph topology is largely unaltered from products to reactants. For example, in the standard retrosynthesis dataset, only 6.3% of the atoms in the product undergo any change in connectivity. Operating at the level of these preserved subgraphs greatly reduces the complexity of reactant generation, leading to improved empirical performance.

Our template-free approach called GRAPHRETRO generates reactants in two stages: (i.) deriving intermediate molecules called *synthons* (Corey, 1967) from the product, and (ii.) expanding synthons into reactants by adding specific functionalities called *leaving groups*. Synthons are derived from products by applying modifications called *edits* to bonds or hydrogen counts on atoms in the product. Leaving groups are selected from a precomputed vocabulary, and attached to synthons through chemically constrained rules. The vocabulary is constructed during preprocessing by extracting subgraphs that differ between a synthon and corresponding reactant, with a 99.7% coverage on the test set.

We evaluate GRAPHRETRO on the benchmark USPTO-50k dataset. GRAPHRETRO achieves 64.2% top-1 accuracy without the knowledge of reaction class, outperforming the state-of-the-art method by a margin of 11.7%.

[1]Department of Computer Science, ETH Zürich [2]Department of Chemical Engineering, MIT [3]Computer Science and Artificial Intelligence Lab, MIT. Correspondence to: Vignesh Ram Somnath <vsomnath@ethz.ch>.

International Conference on Machine Learning (ICML) *Workshop on Graph Representation Learning and Beyond* (GRL+), Vienna, Austria, PMLR 108, 2020.

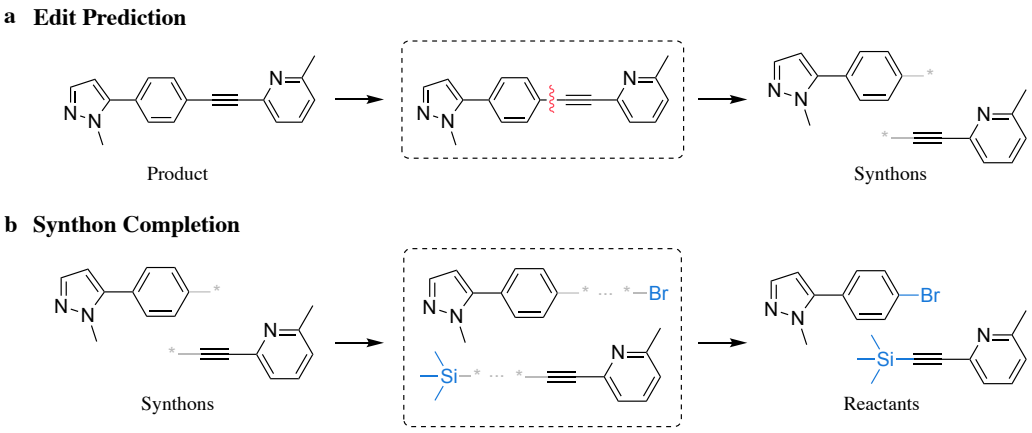


Figure 1. **Overview of Our Approach**. **a. Edit Prediction**. We train a model to learn a distribution over possible graph edits. In this case, the correct edit corresponds to breaking the bond marked in red. Applying this edit produces two synthons. **b. Synthon Completion**. Another model is trained to pick candidate leaving groups (blue) for each synthon from a discrete vocabulary, which are then attached to produce the final reactants.

2. Model Design

Molecules are represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with atoms $\mathcal{V}$ as nodes and bonds $\mathcal{E}$ as edges. A *retrosynthesis pair* R is described by a pair of molecules $(\mathcal{G}_p, \mathcal{G}_r)$, where $\mathcal{G}_p$ are the products and $\mathcal{G}_r$ the reactants. Prior work has focused on the single product case, while reactants can have multiple connected components, i.e. $\mathcal{G}_r = \{\mathcal{G}_{r_c}\}_{c=1}^{C}$. Retrosynthesis pairs are *atom-mapped* so that each product atom has a unique corresponding reactant atom. The retrosynthesis task then, is to infer $\{\mathcal{G}_{r_c}\}_{c=1}^{C}$ given $\mathcal{G}_p$.

Edits consist of atom pairs $\{(a_i, a_j)\}$ with a change in bond type from products to reactants, and atoms $\{a_i\}$ with a change in associated hydrogen count. We denote the set of edits by E. Applying edits E to the product $\mathcal{G}_p$ produces incomplete molecules called *synthons*. To generate valid reactants, specific functionalities called *leaving groups* are added to synthons. We denote synthons by $\mathcal{G}_s$ and leaving groups by $\mathcal{G}_l$, and assume they have C connected components each, i.e $\mathcal{G}_s = \{\mathcal{G}_{s_c}\}_{c=1}^{C}$ and $\mathcal{G}_l = \{\mathcal{G}_{l_c}\}_{c=1}^{C}$.

Our model consists of two modules (Figure 1). Given a product $\mathcal{G}_p$, we first infer edits E (Section 2.1). Applying edits E to $\mathcal{G}_p$ produces synthons $\mathcal{G}_s$. Next, we infer a leaving group $\mathcal{G}_{l_c}$ for each synthon component $\mathcal{G}_{s_c}$ (Section 2.2). The model is defined as

$$P(\mathcal{G}_r|\mathcal{G}_p) = \sum_{E,\mathcal{G}_l} \underbrace{P(E|\mathcal{G}_p)}_{\text{Edit Prediction}} \underbrace{P(\mathcal{G}_l|\mathcal{G}_p, \mathcal{G}_s)}_{\text{Synthon Completion}}, \qquad (1)$$

where $\mathcal{G}_s, \mathcal{G}_r$ are deterministic given $E, \mathcal{G}_l$, and $\mathcal{G}_p$.

2.1. Edit Prediction

Edits can be identified automatically by using the atom-mapping to compare bond and hydrogen count changes between products and reactants. Our edit prediction model has variants tailored to single and multiple edit prediction. Since 95% of the training set consists of single-edit examples, the remainder of this section describes the setup for single edit prediction. A detailed description of our multiple edit prediction model can be found in Appendix B.

Each bond (u, v) in $\mathcal{G}_p$ is associated with a label $y_{uvk} \in \{0, 1\}$ indicating whether its bond type k has changed from the products to reactants. Each atom u is associated with a label $y_u \in \{0, 1\}$ indicating a change in hydrogen count. We predict edit scores using representations that are learnt using a graph encoder.

Graph Encoder To learn atom representations, we use a variant of the *message passing network* (MPN) described in (Gilmer et al., 2017). For simplicity, we denote the encoding process by $\text{MPN}(\cdot)$ and describe architectural details in Appendix A. Given features $\mathbf{x}_u$ for atom u, $\mathbf{x}_{uv}$ for bond (u, v) and neighbors $\mathcal{N}(u)$ of atom u, The MPN computes atom representations $\{\mathbf{c}_u | u \in \mathcal{G}\}$ via

$$\{\mathbf{c}_u\} = \text{MPN}(\mathcal{G}, \{\mathbf{x}_u\}, \{\mathbf{x}_{uv}\}_{v \in \mathcal{N}(u)}). \qquad (2)$$

The graph representation $\mathbf{c}_\mathcal{G}$ is an aggregation of atom representations, i.e. $\mathbf{c}_\mathcal{G} = \sum_{u \in \mathcal{V}} \mathbf{c}_u$. When $\mathcal{G}$ has connected components $\{\mathcal{G}_i\}$, we get graph representations $\{\mathbf{c}_{\mathcal{G}_i}\}$. For a bond (u, v), we define its representation $\mathbf{c}_{uv} = (\mathbf{c}_u \,||\, \mathbf{c}_v)$ as the concatentation of atom representations $\mathbf{c}_u$ and $\mathbf{c}_v$.

Using these representations to directly predict edit scores constrains predictions to the neighborhood the messages were aggregated from. We include global dependencies in the prediction input by using convolutional layers, which have been used successfully to extract globally occurring

features using locally operating filters (Krizhevsky et al., 2012). We apply P layers of convolutions to atom and bond representations to obtain embeddings $\mathbf{c}_u^P$ and $\mathbf{c}_{uv}^P$. These representations are then used to predict atom and bond edit scores using corresponding neural networks,

$$s_{uvk} = \mathbf{u_k}^T \tau(\mathbf{W_k} \mathbf{c}_{uv}^P + b_k) \quad (3)$$

$$s_u = \mathbf{u_a}^T \tau(\mathbf{W_a} \mathbf{c}_u^P + b). \quad (4)$$

Training The edit prediction model minimizes the cross entropy loss over possible bond and atom edits

$$\mathcal{L}_e = -\sum_{((u,v),k)\in E} y_{uvk} \log(s_{uvk}) + \sum_{u \in E} y_u \log(s_u) \quad (5)$$

2.2. Synthon Completion

Synthons are completed into valid reactants by adding specific functionalities called *leaving groups*. This involves two complementary tasks: (i.) selecting the appropriate leaving group, and (ii.) attaching the leaving group to the synthon. As ground truth leaving groups are not directly provided, we extract the leaving groups and construct a vocabulary $\mathcal{X}$ of unique leaving groups during preprocessing. The vocabulary has a limited size ($|\mathcal{X}| = 170$ for a standard dataset with $50,000$ examples). We thus formulate leaving group selection as a classification problem over $\mathcal{X}$.

Vocabulary Construction We first align connected components of synthon and reactant graphs by comparing atom mapping overlaps. Using aligned pairs $\mathcal{G}_{s_c} = (\mathcal{V}_{s_c}, \mathcal{E}_{s_c})$ and $\mathcal{G}_{r_c} = (\mathcal{V}_{r_c}, \mathcal{E}_{r_c})$, the leaving group vocabulary $\mathcal{X}$ is constructed by extracting subgraphs $\mathcal{G}_{l_c} = (\mathcal{V}_{l_c}, \mathcal{E}_{l_c})$ such that $\mathcal{V}_{l_c} = \mathcal{V}_{r_c} \setminus \mathcal{V}_{s_c}$. Atoms $\{a_i\}$ in the leaving groups that attach to synthons are marked with a special symbol.

Leaving Group Selection Treating each $x_i \in \mathcal{X}$ as a molecular subgraph, we learn representations $\mathbf{e}_{x_i}$ using the $\mathrm{MPN}(\cdot)$. We also use the same $\mathrm{MPN}(\cdot)$ to learn the product graph representation $\mathbf{c}_{\mathcal{G}_p}$ and sython representations $\{\mathbf{c}_{\mathcal{G}_{s_c}}\}_{c=1}^{C}$, where C is the number of connected components. For each step $c \leq C$, we compute leaving group probabilities via

$$\hat{q}_{l_c} = \sigma\left(\mathbf{U}\tau\left(\mathbf{W_1}\mathbf{c}_{\mathcal{G}_p} + \mathbf{W_2}\mathbf{c}_{\mathcal{G}_{s_c}} + \mathbf{W_3}\mathbf{e}_{l_{(c-1)}}\right)\right), \quad (6)$$

where $\hat{q}_{l_c}$ is distribution learnt over $\mathcal{X}$, $\mathbf{e}_{l_{(c-1)}}$ is the representation of the leaving group predicted in step $c-1$, and $\sigma(\cdot)$ denotes the softmax function.

Training For step c, given the one hot encoding of the true leaving group q_{l_c}, we minimize the cross-entropy loss

$$\mathcal{L}_s = \sum_{c=1}^{C} \mathcal{L}(\hat{q}_{l_c}, q_{l_c}). \quad (7)$$

Training utilizes teacher-forcing (Williams & Zipser, 1989) so that the model makes predictions given correct histories. During inference, we use the representation of leaving group from the previous step with the highest predicted probability.

Leaving Group Attachment Leaving group attachment is a deterministic process and not learnt during training. The task involves identification of the type of bonds to add between attaching atoms in the leaving group (marked during vocabulary construction), and the atom(s) participating in the edit. These bonds can be inferred by applying the *valency* constraint, which determines the maximum number of neighbors for each atom. Given synthons and leaving groups, the attachment process has a 100% accuracy.

2.3. Overall Training and Inference

The two modules can either be trained separately (*separate*) or jointly by sharing (*shared*) the encoder. The shared training minimizes the loss $\mathcal{L} = \lambda_e \mathcal{L}_e + \lambda_s \mathcal{L}_s$, where λ_e and λ_s weigh the influence of each term on the final loss.

Inference is performed using beam search with a log-likelihood scoring function. For a beam width n, we select n edits with highest scores and apply them to the product to obtain n synthons, where each synthon can consist of multiple connected components. The synthons form the nodes for beam search. Each node maintains a cumulative score by aggregating the log-likelihoods of the edit and predicted leaving groups. Leaving group inference starts with a connected component for each synthon, and selects n leaving groups with highest log-likelihoods. From the n^2 possibilities, we select n nodes with the highest cumulative scores. This process is repeated until all nodes have a leaving group predicted for each synthon component.

3. Experiments

Data We evaluate GRAPHRETRO on the benchmark dataset USPTO-50k (Schneider et al., 2016), which contains $50,000$ atom-mapped reactions across 10 reaction classes, and partitioned into a 80%/10%/10% train/validation/test split (Liu et al., 2017).

Baseline Models LV-TRANSFORMER (Chen et al., 2019) extends the Transformer architecture with a latent variable to improve diversity of suggested reactants. SCROP (Zheng et al., 2019) uses an additional Transformer to correct the syntax of candidates generated from the first one. RETROSIM (Coley et al., 2017) uses molecular similarities to precedent reactions for template ranking. NEURALSYM (Segler & Waller, 2017) learns a conditional distribution over templates given a molecule. The state-of-the-art method, GLN (Dai et al., 2019) models the joint distribution of templates and reactants using logic variables.

Table 1. **Overall Performance**. (sh) and (se) denote *shared* and *separate* training.

Model	Top-n Accuracy (%)							
	Reaction class known				Reaction class unknown			
$n =$	1	3	5	10	1	3	5	10
RETROSIM	52.9	73.8	81.2	88.1	37.3	54.7	63.3	74.1
NEURALSYM	55.3	76.0	81.4	85.1	44.4	65.3	72.4	78.9
GLN	64.2	79.1	85.2	90.0	52.5	69.0	75.6	83.7
SCROP	59.0	74.8	78.1	81.1	43.7	60.0	65.2	68.7
LV-TRANSFORMER	-	-	-	-	40.5	65.1	72.8	79.4
GRAPHRETRO (sh)	67.2	81.7	84.6	87.0	64.2	78.6	81.4	83.1
GRAPHRETRO (se)	67.8	82.7	85.3	87.0	63.8	80.5	84.1	85.9

Evaluation Metrics Similar to prior work, we use the top-n exact match accuracy as our evaluation metric, for $n = 1, 3, 5$ and 10. The accuracy is computed by matching canonical SMILES strings of predicted reactants with those of ground truth reactants. We evaluate the top-n accuracy with and without the knowledge of reaction class.

3.1. Overall Performance

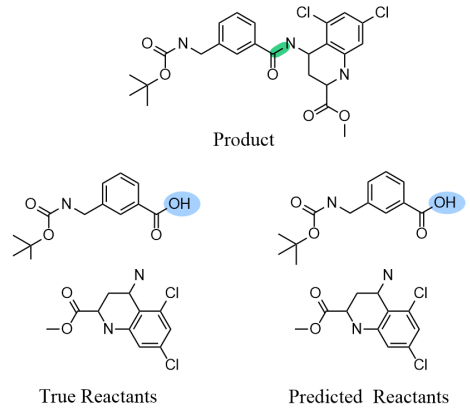
As shown in Table 1, when the reaction class is unknown, our shared and separate configurations outperform GLN by 11.7% and 11.3% in top-1 accuracy, respectively. Similar improvements are achieved for larger n, with ~84% of the true precursors in the top-5 choices. When the reaction class is known, RETROSIM and GLN models restrict prediction to template sets corresponding to the reaction class, thus improving performance. Both of our model configurations outperform the other models till $n = 5$.

3.2. Example Predictions

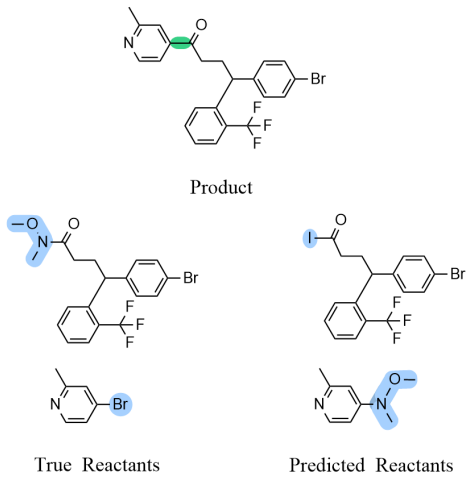
Figure 2a shows an example where the edit and leaving groups are identified correctly. In Figure 2b, the correct edit is identified but the predicted leaving groups are incorrect. In the training set, leaving groups attaching to the carbonyl carbon (C=O) are small (e.g. -OH, -NH_2, halides). The true leaving group in this example, however, is large. GRAPHRETRO is unable to reason about this and predicts the small leaving group -I.

4. Conclusion

Previous methods for single-step retrosynthesis restrict prediction to a template vocabulary or are insensitive to molecular graph structure. We address these shortcomings by introducing a graph-based template-free model inspired by a chemist's workflow. Given a target molecule, we identify synthetic building blocks (*synthons*) which are then realized into valid reactants. Our model outperforms previous methods by a significant margin on the benchmark dataset. Future work aims to extend the model to realize a single reactant from multiple synthons.



a. Correctly predicted example by the model.



b. Correctly predicted edit; incorrectly predicted leaving groups.

Figure 2. **Example Predictions**. Edits are highlighted in green. The true and predicted leaving groups are marked in blue.

References

Bradshaw, J., Kusner, M. J., Paige, B., Segler, M. H. S., and Hernández-Lobato, J. M. A Generative Model For Electron Paths. In *International Conference on Learning Representations (ICLR)*, 2019.

Chen, B., Shen, T., Jaakkola, T. S., and Barzilay, R. Learning to Make Generalizable and Diverse Predictions for Retrosynthesis. In Submission, 2019.

Coley, C. W., Rogers, L., Green, W. H., and Jensen, K. F. Computer-Assisted Retrosynthesis Based on Molecular Similarity. *ACS Central Science*, 3, 2017.

Corey, E. and Wipke, W. T. Computer-Assisted Design of Complex Organic Syntheses. *Science*, 166, 1969.

Corey, E. J. General methods for the construction of complex molecules. *Pure and Applied chemistry*, 14(1):19–38, 1967.

Corey, E. J. The Logic of Chemical Synthesis: Multistep Synthesis of Complex Carbogenic Molecules (Nobel Lecture). *Angewandte Chemie International Edition*, 30, 1991.

Dai, H., Li, C., Coley, C., Dai, B., and Song, L. Retrosynthesis Prediction with Conditional Graph Logic Network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural Message Passing for Quantum Chemistry. In *International Conference on Machine Learning (ICML)*, volume 70, 2017.

Jin, W., Coley, C., Barzilay, R., and Jaakkola, T. Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Jin, W., Barzilay, R., and Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *International Conference on Machine Learning (ICML)*, volume 32, 2018.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Landrum, G. RDKit: Open-Source Cheminformatics Software. 2016.

Liu, B., Ramsundar, B., Kawthekar, P., Shi, J., Gomes, J., Luu Nguyen, Q., Ho, S., Sloane, J., Wender, P., and Pande, V. Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models. In *ACS Central Science*, volume 3. ACS Publications, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. 2019.

Schneider, N., Stiefl, N., and Landrum, G. A. What's What: The (Nearly) Definitive Guide to Reaction Role Assignment. In *Journal of Chemical Information and Modeling*, volume 56. ACS Publications, 2016.

Segler, M. H. and Waller, M. P. Neural-Symbolic Machine Learning for Retrosynthesis and Reaction Prediction. *Chemistry–A European Journal*, 23, 2017.

Weininger, D. SMILES, a Chemical Language and Information System. *Journal of Chemical Information and Computer Sciences*, 28, 1988.

Williams, R. J. and Zipser, D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. In *Neural Computation*, volume 1. MIT Press, 1989.

Zheng, S., Rao, J., Zhang, Z., Xu, J., and Yang, Y. Predicting Retrosynthetic Reactions using Self-Corrected Transformer Neural Networks. In *Journal of Chemical Information and Modeling*. ACS Publications, 2019.

A. Message Passing Network

At message passing step t, each bond $(u, v) \in \mathcal{E}$ is associated with two messages $\mathbf{m}_{uv}^{(t)}$ and $\mathbf{m}_{vu}^{(t)}$. Message $\mathbf{m}_{uv}^{(t)}$ is updated using

$$\mathbf{m}_{uv}^{(t+1)} = \text{GRU}\left(\mathbf{x}_u, \mathbf{x}_{uv}, \{\mathbf{m}_{wu}^{(t)}\}_{w \in N(u) \setminus v}\right), \quad (8)$$

where GRU denotes the Gated Recurrent Unit, adapted for message passing (Jin et al., 2018)

$$\mathbf{s}_{uv} = \sum_{k \in N(u) \setminus v} \mathbf{m}_{ku}^{(t)} \quad (9)$$

$$\mathbf{z}_{uv} = \sigma\left(\mathbf{W_z}\left[\mathbf{x}_u, \mathbf{x}_{uv}, \mathbf{s}_{uv}\right] + b_z\right) \quad (10)$$

$$\mathbf{r}_{ku} = \sigma\left(\mathbf{W_r}\left[\mathbf{x}_u, \mathbf{x}_{uv}, \mathbf{m}_{ku}^{(t)}\right] + b_r\right) \quad (11)$$

$$\tilde{\mathbf{r}}_{uv} = \sum_{k \in N(u) \setminus v} \mathbf{r}_{ku} \odot \mathbf{m}_{ku}^{(t)} \quad (12)$$

$$\tilde{\mathbf{m}}_{uv} = \tanh\left(\mathbf{W}\left[\mathbf{x}_u, \mathbf{x}_{uv}\right] + \mathbf{U}\tilde{\mathbf{r}}_{uv} + b\right) \quad (13)$$

$$\mathbf{m}_{uv}^{(t+1)} = (1 - \mathbf{z}_{uv}) \odot \mathbf{s}_{uv} + \mathbf{z}_{uv} \odot \tilde{\mathbf{m}}_{uv}. \quad (14)$$

After T steps of iteration, we aggregate the messages with a neural network $g(\cdot)$ to derive the representation for each atom

$$\mathbf{c}_u = \text{g}\left(\mathbf{x}_u, \sum_{k \in N(u)} \mathbf{m}_{vu}^{(T)}\right). \quad (15)$$

B. Multiple Edit Prediction

We also propose an autoregressive model for multiple edit prediction that allows us to represent arbitrary length edit sets. The model makes no assumption on the connectivity of the reaction centers or the electron flow topology, addressing the drawbacks mentioned in (Bradshaw et al., 2019; Jin et al., 2017).

Each edit step t uses the intermediate graph $\mathcal{G}_s^{(t)}$ as input, obtained by applying the edits until t to $\mathcal{G}_p$. Atom and bond labels are now indexed by the edit step, and a new termination symbol $y_d^{(t)}$ is introduced such that $\sum_{(u,v),k} y_{uvk}^{(t)} + \sum_u y_u{}^{(t)} + y_d^{(t)} = 1$. The number of atoms remain unchanged during the edit prediction, allowing us to associate a hidden state $\mathbf{h}_u^{(t)}$ with every atom u. Given representations $\mathbf{c}_u^{(t)}$ returned by the $\text{MPN}(\cdot)$ for $\mathcal{G}_s^{(t)}$, we update the atom hidden states

$$\mathbf{h}_u^{(t)} = \tau\left(\mathbf{W_h}\mathbf{h}_u^{(t-1)} + \mathbf{W_c}\mathbf{c}_u^{(t)} + b\right). \quad (16)$$

The bond hidden state $\mathbf{h}_{uv}^{(t)} = (\mathbf{h}_u^{(t)} \,||\, \mathbf{h}_v^{(t)})$ is defined similar to the single edit case. We also compute the termination score using a molecule hidden state $\mathbf{h}_m^{(t)} = \sum_{u \in \mathcal{G}_s^{(t)}} \mathbf{h}_u^{(t)}$. The edit logits are predicted by

$$s_{uvk}^{(t)} = \mathbf{u_k}^T \tau\left(\mathbf{W_k}\mathbf{h}_{uv}^{(t)} + b_k\right) \quad (17)$$

$$s_u^{(t)} = \mathbf{u_a}^T \tau\left(\mathbf{W_a}\mathbf{h}_u^{(t)} + b_a\right) \quad (18)$$

$$s_d^{(t)} = \mathbf{u_d}^T \tau\left(\mathbf{W_d}\mathbf{h}_m^{(t)} + b_d\right). \quad (19)$$

Training: The model is trained to minimize the following loss function

$$\mathcal{L}_e(\mathcal{T}) = -\sum_{(\mathcal{G}_p,E) \in \mathcal{T}} \sum_{t=1}^{|E|} \sum_{(j,k) \in E[t]} y_{uvk}^{(t)} \log(s_{uvk}^{(t)}) + \sum_{u \in E[t]} y_u^{(t)} \log(s_u^{(t)}) + y_d^{(t)} \log(s_d^{(t)}). \quad (20)$$

During training, we perform *teacher-forcing* (Williams & Zipser, 1989): after edit prediction at every stage, we replace them with the label of the true edit so that the model can make predictions given correct histories.

C. Experimental Details

Our model is implemented in PyTorch (Paszke et al., 2019). We also use the open-source software RDKit (Landrum, 2016) to process molecules for our training set, for attaching leaving groups to synthons and generating reactant SMILES.

C.1. Input Features

Atom Features We use the following atom features:

- One hot encoding of the atom symbol (65)
- One hot encoding of the degree of the atom (10)
- Explicit valency of the atom (6)
- Implicit valency of the atom (6)
- Whether the atom is part of an aromatic ring (1)

Bond Features We use the following bond features:

- One hot encoding of bond type (4)
- Whether the bond is conjugated (1)
- Whether bond is part of ring (1)

C.2. Retrosynthesis Benchmarks

The hidden layer dimension of the GRU based Message Passing Network is set to 300. We run $T = 10$ iterations of message passing in the encoder. All models are trained with the Adam optimizer and an initial learning rate of 0.001. Gradients are clipped to have a maximum norm of 20.0.

Edit Prediction The global dependency module comprises three convolutional layers with 600, 300 and 150 filters respectively, and a kernel size of 5. The atom and bond edits scoring network has a hidden layer dimension of 300. The model is trained for 100 epochs, and the learning rate is reduced by a factor of 0.9 when a plateau, as measured by the accuracy of predicted edits on the validation set, is observed. The edit prediction model has 2M parameters.

Synthon Completion The embedding dimension of leaving groups is set to 200, and graph representations are projected to the embedding dimension with a learnable projection matrix. The classifier over leaving groups also has a hidden layer dimension of 300, and a dropout probability of 0.2. The synthon completion model has 0.8M parameters.

Shared Encoder Model Trainable parameters of the associated edit prediction and synthon completion modules have the same dimensions as above. We set λ_e to 1.0 and λ_s to 2.0. The model is trained for 100 epochs, and the learning rate is reduced by a factor of 0.9 when a plateau, as measured by the accuracy of predicted edits and leaving groups on the validation set, is observed. The shared encoder model has 2.3M parameters.