CORE: Automatic Molecule Optimization Using Copy & Refine Strategy

Tianfan Fu,[1] **Cao Xiao,**[2] **Jimeng Sun**[1]
[1]College of Computing, Georgia Institute of Technology, Atlanta, USA
[2]Analytics Center of Excellence, IQVIA, Cambridge, USA
tfu42@gatech.edu, cao.xiao@iqvia.com, jsun@cc.gatech.edu

Abstract

Molecule optimization is about generating molecule Y with more desirable properties based on an input molecule X. The state-of-the-art approaches partition the molecules into a large set of substructures S and grow the new molecule structure by iteratively predicting which substructure from S to add. However, since the set of available substructures S is large, such an iterative prediction task is often inaccurate especially for substructures that are infrequent in the training data. To address this challenge, we propose a new generating strategy called "Copy&Refine" (CORE), where at each step the generator first decides whether to copy an existing substructure from input X or to generate a new substructure, then the most promising substructure will be added to the new molecule. Combining together with scaffolding tree generation and adversarial training, CORE can significantly improve several latest molecule optimization methods in various measures including drug likeness (QED), dopamine receptor (DRD2) and penalized LogP. We tested CORE and baselines using the ZINC database and CORE obtained up to 11% and 21% relatively improvement over the baselines on success rate on the complete test set and the subset with infrequent substructures, respectively.

Introduction

Designing molecules or chemical compounds with desired properties is a fundamental task in drug discovery. Since the number of drug-like molecule is large as estimated between 10^{23} and 10^{60} (Polishchuk, Madzhidov, and Varnek 2013), traditional methods such high throughput screening (HTS) is not scalable. One particular task in drug discovery is called *lead optimization*, where after a drug candidate (a *hit*) is identified via HTS, enhanced similar candidates are created and tested in order to find a lead compound with better properties than the original hit. To model lead optimization as a machine learning problem, the training data involves a set of paired molecules that map input molecule X to target molecule Y with more desirable properties. The goal is to learn a generating model that can produce target molecules with better properties from an input molecule.

Automatic molecular generation algorithms have made great progress in recent years thanks to the successful use of deep generative models (Jin, Barzilay, and Jaakkola 2018; Liu et al. 2018; Zhou et al. 2019; Jin et al. 2019). Early ones cast the molecular generation as a sequence generation problem since a molecule can be represented by a SMILES string[1] (Kusner, Paige, and Hernández-Lobato 2017; Dai et al. 2018; Gómez-Bombarelli et al. 2018). However, many of those algorithms generate many invalid SMILES strings which do not correspond to any valid molecules.

Progress: Recent years graph generation methods have been proposed to bypass the need to produce a valid SMILES string by directly generating molecular graphs(Jin, Barzilay, and Jaakkola 2018; Jin et al. 2019; Ma, Chen, and Xiao 2018). These graph based approaches reformulate the molecular generation task as a graph-to-graph translation problem, which avoid the need of generating SMILES strings. Their key strategy is to partition the input molecule graph into a scaffolding tree of substructures (e.g., rings, atoms, and bonds), and to learn to generate such a tree. All possible tree nodes lead to a large set of substructures, e.g., around 800 unique substructures in the ZINC database (Sterling and Irwin 2015).

Challenge: However, the graph generation methods still exhibit undesirable behaviors such as generating inaccurate substructures because the set of all possible substructures is large, especially for infrequent substructures. In each generating step, the model has to decide which substructure to add from a large set of possible substructures. On the other hand, from real data we observe

- **Stable principle:** Vast majority of substructures in target molecules are from the input molecule. The first row of Table 1 shows about and over 80% substructures are from the input molecule in four datasets/tasks.
- **Novelty principle:** New substructures are present in most target molecules. The second row of Table 1 shows that for 80% target molecules have new substructures compared with their corresponding input molecules.

Based on these observations, we propose a new strategy for molecular optimization called *Copy & Refine* (CORE). The key idea is at each generating step, CORE will decide whether

[1]The Simplified Molecular-Input Line-Entry System (SMILES) is a specification in the form of a line notation for describing the structure of chemical species using short ASCII strings.

Table 1: Comparison between input and target molecules on 4 datasets/tasks. **Stable principle**: Row 1 shows the percentage of original substructures in the target molecule, which is about 80% or more and indicates many original substructures are kept in the newly generated targets. **Novelty principle**: Row 2 shows the percentage of targets have any new substructures that do not belong to the input molecule, which is also high and indicates the need for including new substructures in the targets. Row 3 lists the number of all the substructure, i.e., $|S|$ and Row 4 lists the average substructures for molecules.

	DRD2	LogP04	LogP06	QED
% of original	80.42%	79.47%	88.90%	83.32%
% of novel	86.40%	84.06%	70.14%	80.84%
# substructures	967	785	780	780
Molecule size	13.85	14.30	14.65	14.99

Table 2: Important notations used in the paper.

Notations	short explanation
(X, Y)	input-target molecule pair
S	substructure set S (a.k.a. vocabulary)
V/E	set of vertex(atom) / edge(bond)
$G = (V, E)$	molecular graph
$\mathcal{T}_G = (\mathcal{V}, \mathcal{E})$	scaffolding tree of graph G
$N(v)$	set of neighbor nodes of vertex v
$\mathbf{f}_v/\mathbf{f}_{uv}$	feature vector for node v / edge (u, v)
$\mathbf{v}_{uv}^{(t)}$	message for edge (u, v) at the t-th iteration
$\mathbf{x}_i^G/\mathbf{x}_i^{\mathcal{T}}$	embedding of node i in G / $\mathcal{T}$,
$\mathbf{X}^G$	the set of node embedding $\mathbf{X}^G = \{\mathbf{x}_1^G, \cdots\}$
$\mathbf{h}_{i_t,j_t}$	message vector for edge (i_t, j_t)
$\mathbf{z}_G$	Embedding of Graph G
p_t^{topo}	topological prediction score
$\mathbf{q}_t^{\text{sub}}/\tilde{\mathbf{q}}_t^{\text{sub}}$	pred. dist. over all substructures
$g_i(\cdot),\ i = 1, \cdots, 6$	parameterized neural networks

to copy a substructure from the input molecule (*Copy*) or sample a novel substructure from the entire space of substructures (*Refine*).

We compare CORE with different baselines and demonstrate significant performance gain in several metrics including drug likeness (QED), dopamine receptor (DRD2) and penalized LogP. We tested CORE and baselines using the ZINC database and CORE achieved up to 11% and 21% relatively improvement over the baselines on success rate on the complete test set and the subset with infrequent substructures, respectively.

The rest of paper is organized as follows. We briefly review related work on molecule generation. Then we describe CORE method. Finally, we show empirical studies and conclude our paper.

Related Work

We review related works in molecule generation.

Sequence-based methods One research line is to formulate molecular generation as a sequence based problem (Dai et al. 2018; Kusner, Paige, and Hernández-Lobato 2017; Zhou et al. 2017; Gómez-Bombarelli et al. 2018; Hong et al. 2019; Huang et al. 2020). Most of these methods are based on the simplified molecular-input line-entry system (SMILES), a line notation describing the molecular structure using short ASCII strings (Weininger 1988). Character Variational Auto-Encoder (C-VAE) generates SMILES string character-by-character (Gómez-Bombarelli et al. 2018). Grammar VAE (G-VAE) generates SMILES following syntactic constraints given by a context-free grammar (Kusner, Paige, and Hernández-Lobato 2017). Syntax-directed VAE (SD-VAE) that incorporates both syntactic and semantic constraints of SMILES via attribute grammar (Dai et al. 2018). Reinforcement learning (RL) is also used to generate SMILES strings (Popova, Isayev, and Tropsha 2018; Olivecrona et al. 2017). However, many generated SMILES strings using these methods are not even syntactically valid which lead to inferior results.

Graph-based methods Another research line is to directly generate molecule graphs. Comparing sequence based method, graph-based methods bypass the need of generating valid sequences (e.g., SMILES strings) all together. As a result, all the generated molecules are valid and often with improved properties. The original idea of (Jin, Barzilay, and Jaakkola 2018) is to eliminate cycles in a molecular graph by representing the graph as a scaffolding tree where nodes are substructures such as rings and atoms. Then a two-level generating function is used to create such a tree then decode the tree into a new molecular graph. Recently another enhancement is produced called graph-to-graph translation model (Jin et al. 2019), which extends the junction variational autoencoder via attention mechanism and generative adversarial networks. It is able to translate the current molecule to a similar molecule with prespecified desired property (e.g., drug-likeness). Also a RL based method with graph convolutional policy network has been proposed to generate molecular graphs with desirable properties (You et al. 2018). However, these graph based models require to iteratively predict the best substructure to add from a large set of possible choices, which is often inaccurate especially for the rare substructures. We overcome this limitation using a hybrid strategy which involves copy from the input molecule then refine it by selectively adding new substructures.

Method

In this section, we first describe overall framework of CORE which shares the same foundation as (Jin et al. 2019) . Then we present model enhancement that CORE introduces namely the copy & refine strategy. We list some important notations and their short explanations in Table 2.

Overview

Graph-to-graph model involves two important structures:

- **molecular graph** G is the graph structure for a molecule;
- **scaffolding tree** $\mathcal{T}_G$ (also referred as scaffolding trees in (Jin et al. 2019)) is the skeleton of the molecular graph G by partitioning the original graph into substructures

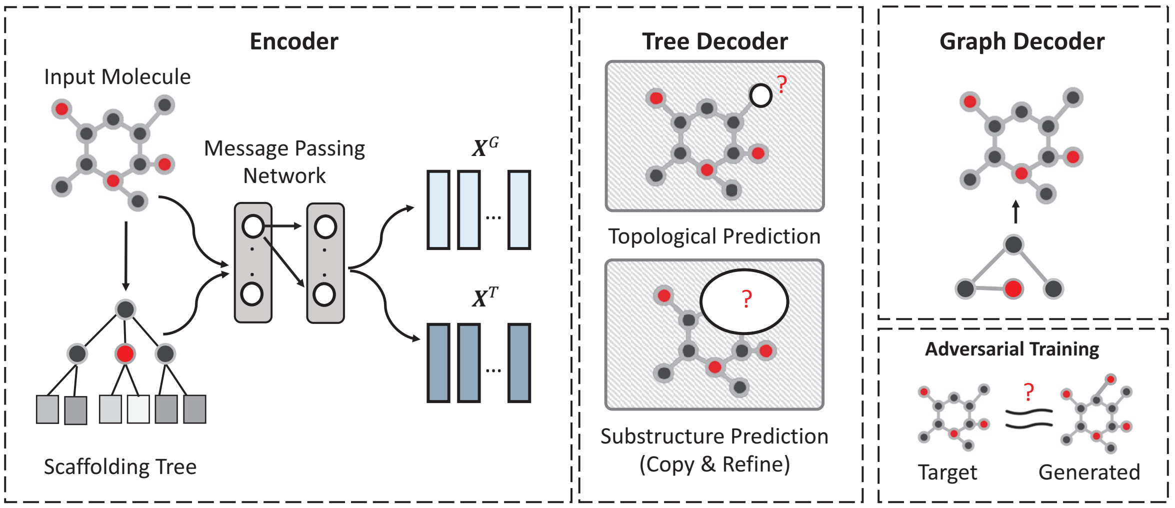


Figure 1: Pipeline for Graph-to-Graph Model. Encoder include both graph and scaffolding tree levels. Decoding mainly split two parts scaffolding tree decoder and graph decoder. scaffolding tree decoder generate molecule in greedy manner using Depth First Search with topological and substructure prediction on each node. To assemble the node of scaffolding tree into the molecule, graph decoder enumerates all possible combinations.

(subgraphs), and connecting those substructures into a tree.

Given a molecule pair (input X and target Y), we first train encoders to embed both G and $\mathcal{T}_G$ for input X into vector representations via a message passing algorithm on graphs (or trees). Finally two-level decoders are introduced to create a new scaffold tree and the corresponding molecular graph. Our main methodology contribution lies in the decoder module where we propose a copy & refine strategy to create novel but stable molecules from input molecules. The model is trained with a set of molecule pairs (X, Y), where Y is a target molecule that is optimized with better chemical properties based on the input molecule X.

Encoder

To construct cycly-free structures, scaffolding tree $\mathcal{T}_G$ is generated via contracting certain vertices of G into a single node. By viewing scaffolding tree as graph, both input molecular graphs and scaffolding trees can be encoded via graph Message Passing Networks (MPN) (Dai, Dai, and Song 2016; Jin, Barzilay, and Jaakkola 2018). The encoder yields an embedding vector for each node in either scaffolding tree or the input molecular graph. More formally, on node level $\mathbf{f}_v$ denotes the feature vector for node v. For atoms, $\mathbf{f}_v$ includes the atom type, valence, and other atomic properties. For nodes in the scaffolding tree representing substructures, $\mathbf{f}_v$ is a one-hot vector indicating its substructure index. On the other hand, on edge level, $\mathbf{f}_{uv}$ feature vector for edge $(u, v) \in E$. $N(v)$ denotes the set of neighbor nodes for node v. $\mathbf{v}_{uv}$ and $\mathbf{v}_{vu}$ are the hidden variables that represent the message from node u to v and vice versa. They are iteratively updated via a fully-connected neural network $g_1(\cdot)$:

$$\mathbf{v}_{uv}^{(t)} = g_1\big(\mathbf{f}_u, \mathbf{f}_{uv}, \sum_{w \in N(u) \backslash v} \mathbf{v}_{wu}^{(t-1)}\big), \tag{1}$$

where $\mathbf{v}_{uv}^{(t)}$ is the message vector at the t-th iteration, whose initialization is $\mathbf{v}_{uv}^{(0)} = \mathbf{0}$. After T steps of iteration, another network $g_2(\cdot)$ is used to aggregate these messages. Each vertex has a latent vector as

$$\mathbf{x}_u = g_2\big(\mathbf{f}_u, \sum_{v \in N(v)} \mathbf{v}_{vu}^{(T)}\big), \tag{2}$$

where $g_2(\cdot)$ is another fully-connected neural network.

In summary, the encoder module yield embedding vectors for nodes in graph G and scaffolding tree $\mathcal{T}_G$, denoted $\mathbf{X}^G = \{\mathbf{x}_1^G, \mathbf{x}_2^G, \cdots\}$ and $\mathbf{X}^{\mathcal{T}_G} = \{\mathbf{x}_1^{\mathcal{T}_G}, \mathbf{x}_2^{\mathcal{T}_G}, \cdots\}$, respectively.

Decoder

Once the embedding vectors are constructed, decoder can also be divided into two phases in coarse-to-fine manner: (a) tree decoder; (b) graph decoder. We firstly discuss scaffolding tree decoder. Our method improve the tree decoder in (Jin et al. 2019), so we describe the enhancement in detail.

A. Tree decoder The objective of the scaffolding tree decoder is to generate a new scaffolding tree from the embeddings. The overall idea is to generate one substructure at a time from an empty tree, and at each time we decide whether to expand the current node or backtrack to its parent (*topological prediction*) and which to add (*substructure prediction*). The generation will terminate once the condition to backtrack from the root is reached.

More specifically the tree decoder has two prediction tasks:

- **Topological prediction:** When the decoder visit the node i_t, the model has to make a binary prediction on either "expanding a new node" or "backtracking to the parent node of i_t".
- **Substructure prediction:** If the decoder decides to expand, we have to select which substructure to add by either copying from original input or selecting from the global set of substructures.

Topological prediction: The idea is to first enhance the embedding for node i_t via a tree-based RNN (Jin, Barzilay, and Jaakkola 2018), then use the enhanced embedding to predict whether to expand or backtrack. Given scaffolding tree $\mathcal{T}_G = (\mathcal{V}, \mathcal{E})$, the tree decoder uses the tree based RNN with attention mechanism to further improve embedding information learned from the original message-passing embeddings $X^{\mathcal{T}}$. Since RNN works on a sequence, the tree converts into a sequence of nodes and edges via depth-first search. Specifically, let $\tilde{\mathcal{E}} = \{(i_1, j_1), (i_2, j_2), \cdots, (i_m, j_m)\}$ be the edges traversed in depth first search, each edge is visited twice in both directions, so we have $m = |\tilde{\mathcal{E}}| = 2|\mathcal{E}|$. Suppose $\tilde{\mathcal{E}}_t$ is the first t edges in $\tilde{\mathcal{E}}$, message vector $\mathbf{h}_{i_t,j_t}$ is updated as:

$$\mathbf{h}_{i_t,j_t} = \text{GRU}(\mathbf{f}_{i_t}, \{\mathbf{h}_{k,i_t}\}_{(k,i_t)\in\tilde{\mathcal{E}}_t, k\neq j_t}). \tag{3}$$

The probability whether to expand or backtrack at node i_t is computed via aggregating the embeddings $\mathbf{X}^{\mathcal{T}}$, $\mathbf{X}^G$ and the current state $\mathbf{f}_{i_t}$, $\sum_{(k,i_t)\in\tilde{\mathcal{E}}_t} \mathbf{h}_{k,i_t}$ using a neural network $g_3(\cdot)$:

$$\begin{aligned} p_t^{\text{topo}} &= g_3(\mathbf{f}_{i_t}, \sum_{(k,i_t)\in\tilde{\mathcal{E}}_t} \mathbf{h}_{k,i_t}, \mathbf{X}^{\mathcal{T}}, \mathbf{X}^G), \\ &\text{where } t = 1, \cdots, m. \end{aligned} \tag{4}$$

Concretely, firstly compute context vector $\mathbf{c}_t^{\text{topo}}$ using attention mechanism[2], then concatenate $\mathbf{c}_t^{\text{topo}}$ and $\mathbf{f}_{i_t}$, followed by a fully connected network with sigmoid activation.

Substructure prediction: Once the node expansion is decided, we have to find what substructures to add. Empirically we observe this step is most challenging one as it leads to largest error rate. For example, during training procedure of QED dataset, topological prediction and graph decoding (mentioned later) can achieve 99%+ and 98%+ classification accuracy, respectively. In contrast, substructure prediction can achieve at most 90% accuracy, which is much lower. We design `CORE` strategy to enhance this part. The idea is every time after expanding a new node, the model have to predict its substructure from all the substructures in vocabulary. First we use attention mechanism to compute context vector based on current message vector $\mathbf{h}_{i_t,j_t}$ and node embedding $\mathbf{X}^{\mathcal{T}}, \mathbf{X}^G$:

$$\mathbf{c}_t^{\text{sub}} = \text{Attention}(\mathbf{h}_{i_t,j_t}, \mathbf{X}^{\mathcal{T}}, \mathbf{X}^G), \tag{5}$$

Specifically, we firstly compute attention weight via

$$\begin{aligned} \boldsymbol{\alpha}_j^{\mathcal{T}} = g_4(\mathbf{h}_{k,i_t}, \mathbf{x}_j^{\mathcal{T}}), \ \ \boldsymbol{\alpha}_i^{\mathcal{T}} \in \mathbb{R}, \\ [\boldsymbol{\alpha}_1^{\mathcal{T}}, \boldsymbol{\alpha}_2^{\mathcal{T}}, \cdots] = \text{Softmax}([\boldsymbol{\alpha}_1^{\mathcal{T}}, \boldsymbol{\alpha}_2^{\mathcal{T}}, \cdots]), \end{aligned} \tag{6}$$

where $g_4(\cdot)$ is dot-product function (Vaswani et al. 2017). $\{\boldsymbol{\alpha}^G\}$ are generated in same way. Then context vector is generated via concatenating tree-level and graph-level context vector

$$\mathbf{c}_t^{\text{sub}} = \Big[\sum_i \boldsymbol{\alpha}_i^{\mathcal{T}} \mathbf{x}_i^{\mathcal{T}}, \sum_j \boldsymbol{\alpha}_j^G \mathbf{x}_j^G\Big]. \tag{7}$$

Then based on attention vector $\mathbf{c}_t^{\text{sub}}$ and message vector $\mathbf{h}_{i_t,j_t}$, $g_5(\cdot)$, a fully-connected neural network with softmax activation, is added to predict the substructure:

$$\mathbf{q}_t^{\text{sub}} = g_5(\mathbf{h}_{i_t,j_t}, \mathbf{c}_t^{\text{sub}}), \tag{8}$$

where $\mathbf{q}_t^{\text{sub}}$ is a distribution over all substructures.

However, the number of all possible substructures is usually quite large, for example, in Table 1, the vocabulary size of DRD2 dataset is 967, which means the number of categories is 967, which makes prediction more challenging especially for the rare substructures.

Inspired by pointer network (Vinyals, Fortunato, and Jaitly 2015; See, Liu, and Manning 2017) we design a similar strategy to copy some of the input sequence to the output. However, pointer network does not handle the case where the target molecule contains **Out-of-Input** (**OOI**) substructures, i.e., a novel substructure is not part of the input molecule. Borrowing the idea from sequence-to-sequence model (Gu et al. 2016; See, Liu, and Manning 2017), we design a method to predict the weight of generating novel OOI substructures.

Refine with novel substructures First, we use context vector $\mathbf{c}_t^{\text{sub}}$ and embeddings of input molecule graph and its scaffolding tree to determine the weight of generating novel substructures in current step,

$$w_t^{\text{OOI}} = g_6(\mathbf{c}_t^{\text{sub}}, \mathbf{z}), \tag{9}$$

where $g_6(\cdot)$ is a fully-connected neural network with sigmoid activation. Thus, the weight ranges from 0 to 1. w_t^{OOI} represents the probability that the model generate OOI substructure at t-th step. We assume that the weight depends on not only the input molecule (global information) and the current position in the decoder (local information). We use a representation $\mathbf{z}$ to express the global information of input molecule,

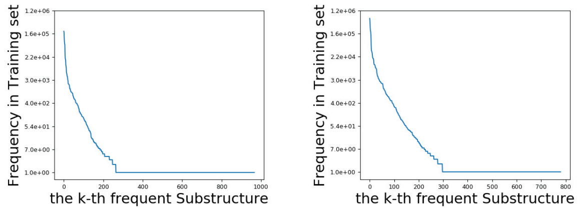
$$\mathbf{z} = \left[\frac{1}{|\{\mathbf{x}_i^{\mathcal{T}}\}|} \sum_i \mathbf{x}_i^{\mathcal{T}}, \frac{1}{|\{\mathbf{x}_j^G\}|} \sum_j \mathbf{x}_j^G\right] \tag{10}$$

where $\mathbf{z}$ is the concatenation of average embedding of all the scaffolding tree node and average embedding of all the graph node. Local information is represented by context vector $\mathbf{c}_t^{\text{sub}}$ computed via attention mechanism.

[2]same procedure as Equation (5) and (6), but different parameters

Table 3: Statistics of 4 datasets, DRD2, QED, LogP04 and LogP06. "Avg # Substructures" is the average number of substructures per molecule. For each test sample, we generate 20 molecules using different random seeds. "# Infreq Test" is the number of test samples that contain at least one infrequent substructure.

Dataset	# Training Pairs	# Valid Pairs	# Test ($\times 20$)	# Infreq Test ($\times 20$)	Vocab Size, $\|S\|$	Avg # Substructures
DRD2	32,404	2,000	1,000	405	967	13.85
QED	84,306	4,000	800	294	780	14.99
LogP04	94909	5,000	800	287	785	14.30
LogP06	71,284	4,000	800	250	780	14.65



(a) DRD2 (b) QED

Figure 2: The frequency of different substructures on DRD2(a) and QED datasets(b). It indicates that the distribution of various substructures is highly imbalanced.

Copy existing substructures After obtaining the weight of OOI substructure, we consider if and what substructures to copy from input molecule. Each substructure in input molecule has an attention weight (normalized, so the sum is 1), which measure the contribution of the substructure to decoder. Then we use it to represent the selection probability for each substructure. Specifically, we define a sparse vector $\mathbf{a}$ as

$$\{\mathbf{a}\}_i = \begin{cases} \sum_{j\in\mathcal{C}} \boldsymbol{\alpha}_j^{\mathcal{T}}, & \mathcal{C} = \{j \mid j\text{-th node is i-th substruct}\}, \\ 0, & \text{i-th substructure not in } \mathcal{T}, \end{cases} \tag{11}$$

where $\mathbf{a} \in \mathbb{R}^{|S|}$, $|S|$ is size of $\{\mathbf{a}\}_i$ represent i-th element of $\mathbf{a}$. Thanks to the normalization of attention weight (Equation 6) $\{\boldsymbol{\alpha}_1^{\mathcal{T}}, \boldsymbol{\alpha}_2^{\mathcal{T}}, \cdots\}$, $\mathbf{a}$ is also normalized.

The prediction at t-step is formulated as a hybrid of

$$\tilde{\mathbf{q}}_t^{\text{sub}} = w_t^{\text{OOI}} \mathbf{q}_t^{\text{sub}} + (1 - w_t^{\text{OOI}}) \mathbf{a}_t. \tag{12}$$

where w_t^{OOI} balances the contributions of two distributions at t-th step. If novel substructures are generated, we select the substructure from all substructures according to distribution $\mathbf{q}_t^{\text{sub}}$). Otherwise, we use the pointer network (Vinyals, Fortunato, and Jaitly 2015) to copy a certain substructure from the input molecule. The selection criteria for substructures in input molecule is proportional to the attention weight $\{\boldsymbol{\alpha}^{\mathcal{T}}\}$.

B. Graph decoder (Finetuning) Finally, we describe graph decoder. The goal is to assemble nodes in scaffolding tree together into the correct molecular graph (Jin, Barzilay, and Jaakkola 2018), as shown in Figure 1. During learning procedure, all candidate molecular structures $\{G_i\}$ are enumerated and it is cast as a classification problem whose target is to maximize the scoring function of the right structure G_o

$$\mathcal{L}_g = f^a(G_o) - \log \sum_{G_i} \exp(f^a(G_i)), \tag{13}$$

where $f^a(G_i) = \mathbf{h}_{G_i} \cdot \mathbf{z}_{G_o}$ is a scoring function that measure the likelihood of the current structure G_i, $\mathbf{z}_{G_o}$ is an embedding of graph G_o. On the other hand, when sampling, graph decoder is to enumerate all possible combinations, and pick the most likely structure in a greedy manner.

Adversarial Learning

Adversarial training is used to further improve the performance, where the entire encoder-decoder architecture is regarded as the generator $G(\cdot)$, the target molecule Y is regarded as the real sample, discriminator $D(\cdot)$ is used to distinguish real molecules from generated molecules by our decoder (Jin et al. 2019). Following (Jin et al. 2019), $D(\cdot)$ is a multi-layer feedforward network.

Experiment

In the experiment section, we want to answer the following questions:

- **Q1:** Can `CORE` generate better molecules than other graph-based methods?
- **Q2:** How well does `CORE` handle input molecules with rare substructures?

Molecular Data

First, we introduce the molecule data that we are using. ZINC contains 250K drug molecules extracted from the ZINC database (Sterling and Irwin 2015). We extract data pairs from ZINC, which will be described later. We list the basic statistics for the datasets in Table 3.

Molecular Properties

In drug discovery, some properties are crucial in evaluating the effectiveness of generated drugs. In this paper, following (Jin et al. 2019), we mainly focus on following three properties.

- **Dopamine Receptor (DRD2)**. DRD2 score is to measure a molecule's biological activity against a biological target named the dopamine type 2 receptor (DRD2). DRD2 score ranges from 0 to 1.

Table 4: Empirical results measured by **Similarity** for various methods on different datasets.

Method	Test Set				Test subset with infrequent substructures			
	QED	DRD2	LogP04	LogP06	QED	DRD2	LogP04	LogP06
JTVAE	.2988	.2997	.2853	.4643	.2519	.2634	.2732	.4238
GCPN	.3081	.3092	.3602	.4282	.2691	.2759	.2973	.3709
Graph-to-Graph	.3145	.3164	.3579	.5256	.2723	.2760	.2901	.4744
CORE	**.3211**	**.3334**	**.3695**	**.6386**	**.2982**	**.3021**	**.3234**	**.5839**

Table 5: Empirical results measured by **Property(Y)** for various methods on different datasets.

Method	Test Set				Test subset with infrequent substructures			
	QED	DRD2	LogP04	LogP06	QED	DRD2	LogP04	LogP06
JTVAE	.8041	.7077	2.5328	1.0323	.7292	.6292	2.0219	.7832
GCPN	.8772	.7512	3.0483	**2.148**	.7627	.6743	2.5413	1.813
Graph-to-Graph	.8803	.7641	2.9728	1.983	.7632	.6843	2.4083	1.778
CORE	**.8952**	**.7694**	**3.1053**	2.021	**.7899**	**.7193**	**2.7391**	**1.820**

- **Drug likeness (QED)** (Bickerton et al. 2012). QED score is an indicator of drug-likeness ranging from 0 to 1.
- **Penalized LogP**. Penalized logP is a logP score that accounts for ring size and synthetic accessibility (Ertl and Schuffenhauer 2009).

For each SMILES string in ZINC, we generate the QED, DRD2 and LogP scores using Rdkit package (Landrum 2013). For all these three scores, higher is better. Thus, for the training data pairs (X, Y), X is the input molecule with lower scores while Y is a generated molecule with higher scores based on X.

Generation of Training Pairs

To construct training data set, we find the molecule pair (X, Y) following (Jin et al. 2019), where X is the input molecule and Y is the target molecule with desired property. Both X and Y are from the whole dataset and have to satisfy two rules: (1) they are similar enough, i.e., $\text{sim}(X, Y) \geq \eta_1$; (2) Y has significant property improvement over X, i.e., $\text{property}(Y) - \text{property}(X) \geq \eta_2$, $\text{property}(\cdot)$ can be DRD2, QED, LogP score as mentioned above. $\eta_1 = 0.4$ for LogP04 while $\eta_1 = 0.6$ for LogP06. We use the public dataset in (Jin et al. 2019) with paired data.

Infrequent Substructures We pay special attention on the infrequent substructure. Based on observation from empirical studies (for example in Figure 2), regarding a substructure if it occurs less than 2,000 times in training set, we call it "infrequent substructures". Otherwise, we call it "frequent substructure".

Baseline Methods

We compare our method with some important baseline methods, which represents state-of-the-art methods on this task.

- **JTVAE** (Jin, Barzilay, and Jaakkola 2018). scaffolding tree variational autoencoder (JTVAE) is a deep generative model that learns latent space to generate desired molecule. It also uses encoder-decoder architecture on both scaffolding tree and graph levels.
- **Graph-to-Graph** (Jin et al. 2019). It is the most important benchmark method as described above.
- **GCPN** (You et al. 2018) uses graph convolutional policy networks to generate molecular structures with specific properties. It exhibits state-of-the-art performance in RL-based method.

Note that we also tried Sequence-to-Sequence model (Sutskever, Vinyals, and Le 2014) on SMILES strings, but the resulting model generates too many invalid SMILES strings to compare with all the other graph-based methods. This further confirmed that graph generation is a more effective strategy for molecular optimization.

Evaluation

During evaluation procedure, we mainly focus on several evaluation metrics, where X is the input molecule in test set, Y is the generated molecule.

- **Similarity**. We evaluate the molecular similarity between the input molecule and the generated molecule, measured by Tanimoto similarity over Morgan fingerprints (Rogers and Hahn 2010). The similarity between molecule X and Y is denoted $\text{sim}(X, Y)$, ranging from 0 to 1.
- **Property of generated Molecules**. The second metric is the property scores of generated molecules. It is defined as Property(Y), where property could be including QED-score, DRD2-score and LogP-score, evaluated using Rdkit package (Landrum 2013).
- **Success Rate (SR)**. Success Rate is a metrics that consider both similarity and property improvement. Since the task is to generate a molecule that (i) is similar to input molecule and (ii) have property improvement at the same time. We design a criteria to judge whether it satisfied these two requirement: (a) Input and generated molecules are similar enough, $\text{sim}(X, Y) \geq \lambda_1$; (b) improvement are

Table 6: Empirical results measured by **SR1 (Success Rate)** for various methods on different dataset. For QED and DRD2, regarding SR1, when the similarity between input and generated molecule (λ_1) is greater than 0.3 and property of generated molecule (λ_2) is greater than 0.6, we regard it "success". For LogP04 and LogP06, $\lambda_1 = 0.4$ and $\lambda_2 = 0.8$

Method	Test Set				Test subset with infrequent substructures			
	QED	DRD2	LogP04	LogP06	QED	DRD2	LogP04	LogP06
JTVAE	43.32%	34.83%	38.43%	43.54%	38.91%	29.32%	35.32%	40.43%
GCPN	47.71%	44.05%	56.43%	52.82%	42.80%	37.82%	42.81%	43.29%
Graph-to-Graph	48.16%	45.73%	56.24%	55.15%	43.43%	38.39%	42.83%	47.02%
CORE	**50.26%**	**47.91%**	**56.47%**	**57.64%**	**47.82%**	**42.72%**	**45.01%**	**50.05%**

Table 7: Empirical results measured by **SR2 (Success Rate)** for various methods on different dataset. For QED and DRD2, regarding SR2, when the similarity between input and generated molecule is greater than 0.4 (λ_1) and property of generated molecule is greater than 0.8 (λ_2), we regard it "success". For LogP04 and LogP06, $\lambda_1 = 0.4$ and $\lambda_2 = 1.2$.

Method	Test Set				Test subset with infrequent substructures			
	QED	DRD2	LogP04	LogP06	QED	DRD2	LogP04	LogP06
JTVAE	20.72%	9.13%	21.32%	18.32%	17.53%	7.18%	19.93%	17.16%
GCPN	23.08%	14.94%	27.01%	25.29%	18.98%	12.64%	23.81%	23.02%
Graph-to-Graph	24.34%	15.31%	26.95%	25.30%	20.82%	12.88%	23.53%	23.82%
CORE	**27.23%**	**17.31%**	**27.88%**	**26.58%**	**25.32%**	**15.26%**	**25.62%**	**25.91%**

big enough, i.e., property$(Y) \geq \lambda_2$. The selection of λ_1 and λ_2 depend on datasets.

Among these metrics, similarity and property improvement are the most basic metrics. For all these metrics except run time and model size, higher values are better.

Implementation Details

In this section, we provide the implementation details for reproducibility, especially the setting of hyperparameters. We follow most of the hyperparameter setting of (Jin et al. 2019). For all these baseline methods and datasets, maximal epoch number is set to 10, batch size is set to 32. During encoder module, embedding size is set to 300. The depth of message passing network are set to 6 and 3 for tree and graph, respectively. The initial learning rate is set to $1e^{-3}$ with the Adam optimizer. Every epoch learning rate is annealed by 0.8. We save the checkpoint every epoch during training procedure. When evaluating, from all the checkpoints, we choose the one that achieves highest success rate (SR1) on validation set as the best model and use it to evaluate the test set. During adversarial training, discriminator $D(\cdot)$ is a three-layer feed-forward network with hidden layer dimension 300 and LeakyReLU activation function. For all these datasets, model size of CORE is around 4M, nearly same as Graph-to-Graph model. The code is available[3].

Results

The results for various metrics (including similarity, property improvement, success rate) are shown in Table 4, 5, 6, 7, respectively. Now we can answer three questions proposed in the beginning of the section.

[3]https://github.com/futianfan/CORE

- **Q1 Performance on all molecules**: We compare CORE with baseline methods on complete test sets. CORE outperforms baselines in all the measures. Specifically, when measured by success rates SR (both SR1 and SR2, Table 6 and 7), CORE obtained about 2% absolutely improvement over the best baseline. When measured by SR2, it can achieve over 10% relatively improvement on QED and DRD2.
- **Q2 Performance on molecules with infrequent substructures**: Test subset with infrequent substructures is more challenging, because for all the methods the performance would degrade on infrequent subset. Thus, it is worth to pay special attention to the molecule with infrequent substructure. When measured on the test subset with infrequent substructure, CORE achieves more significant improvement compared with the complete test set. Concretely, CORE achieves 21% and 18% relatively improvement in success rate (SR2) in QED and DRD2, and more than 3% absolutely improvement in SR (both SR1 and SR2). In a word, CORE gain more improvement in rare substructure compared with the whole test set.

Conclusion

In this paper, we propose a deep generative model for creating molecules with more desirable properties than the input molecules. The state-of-the-art Graph-to-Graph methods grow the new molecule structure by iteratively predicting substructures from a large set of substructures, which is challenging especially for infrequent substructures. To address this challenge, we have propose a new generating strategy called "Copy&Refine" (CORE), where at each step the generator first decides whether to copy an existing substructure from input X or to generate a new substructure from the large set. The resulting CORE mechanism can significantly outperform several latest molecule optimization baselines in various measures, especially on rare substructure.

Acknowledgement

This work was in part supported by the National Science Foundation award IIS-1418511, CCF-1533768 and IIS-1838042, the National Institute of Health award NIH R01 1R01NS107291-01 and R56HL138415.

References

Bickerton, G. R.; Paolini, G. V.; Besnard, J.; Muresan, S.; and Hopkins, A. L. 2012. Quantifying the Chemical Beauty of Drugs. *Nature Chemistry* 4(2):90.

Dai, H.; Tian, Y.; Dai, B.; Skiena, S.; and Song, L. 2018. Syntax-directed Variational Autoencoder for Structured Data. *6th International Conference on Learning Representations, ICLR*.

Dai, H.; Dai, B.; and Song, L. 2016. Discriminative Embeddings of Latent Variable Models for Structured Data. In *International Conference on Machine Learning*, 2702–2711.

Ertl, P., and Schuffenhauer, A. 2009. Estimation of Synthetic Accessibility Score of Drug-like Molecules based on Molecular Complexity and Fragment Contributions. *Journal of Cheminformatics* 1(1):8.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic Chemical Design Using a Data-driven Continuous Representation of Molecules. *ACS Central Science* 4(2):268–276.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Hong, S.; Xiao, C.; Ma, T.; Li, H.; and Sun, J. 2019. MINA: Multilevel Knowledge-guided Attention for Modeling Electrocardiography Signals. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 5888–5894.

Huang, K.; Xiao, C.; Hoang, N.; Glass, L.; and Sun, J. 2020. CASTER: Predicting Drug Interactions with Chemical Substructure Representation. *AAAI*.

Jin, W.; Barzilay, R.; and Jaakkola, T. S. 2018. Junction Tree Variational Autoencoder for Molecular Graph Generation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2328–2337.

Jin, W.; Yang, K.; Barzilay, R.; and Jaakkola, T. 2019. Learning Multimodal Graph-to-Graph Translation for Molecular Optimization. *ICLR*.

Kusner, M. J.; Paige, B.; and Hernández-Lobato, J. M. 2017. Grammar Variational Autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1945–1954. JMLR.org.

Landrum, G. 2013. Rdkit documentation. *Release* 1:1–79.

Liu, Q.; Allamanis, M.; Brockschmidt, M.; and Gaunt, A. 2018. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems*, 7795–7804.

Ma, T.; Chen, J.; and Xiao, C. 2018. Constrained Generation of Semantically Valid Graphs via Regularizing Variational Autoencoders. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 7113–7124.

Olivecrona, M.; Blaschke, T.; Engkvist, O.; and Chen, H. 2017. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics* 9(1):48.

Polishchuk, P. G.; Madzhidov, T. I.; and Varnek, A. 2013. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of Computer-Aided Molecular Design* 27(8):675–679.

Popova, M.; Isayev, O.; and Tropsha, A. 2018. Deep reinforcement learning for de novo drug design. *Sci Adv* 4(7):eaap7885.

Rogers, D., and Hahn, M. 2010. Extended-connectivity Fingerprints. *Journal of Chemical information and Modeling* 50(5):742–754.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*.

Sterling, T., and Irwin, J. J. 2015. ZINC 15–ligand Discovery for Everyone. *Journal of Chemical information and Modeling* 55(11):2324–2337.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104–3112.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, 2692–2700.

Weininger, D. 1988. SMILES, A Chemical Language and Information System. *Journal of Chemical Information and Computer Sciences* 28(1):31–36.

You, J.; Liu, B.; Ying, R.; Pande, V.; and Leskovec, J. 2018. Graph Convolutional Policy Network for Goal-directed Molecular Graph Generation. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, 6412–6422. USA: Curran Associates Inc.

Zhou, P.; Wei, W.; Bian, K.; Wu, D. O.; Hu, Y.; and Wang, Q. 2017. Private and Truthful Aggregative Game for Large-scale Spectrum Sharing. *IEEE Journal on Selected Areas in Communications* 35(2):463–477.

Zhou, Z.; Kearnes, S.; Li, L.; Zare, R. N.; and Riley, P. 2019. Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports* 9(1):10752.