Batch-Advantage Transformer with Hypergraph Optimized Grammar (BAT/HOG)

Egor Kraev
Mosaic Smart Data Ltd
egor.kraev@gmail.com

Mark Harley
Mosaic Smart Data Ltd
mharley.code@gmail.com

Abstract

We present a novel approach to the issue of molecular optimization. Our approach uses a hypergraph replacement grammar inferred from the ZINC database, with grammar construction optimized for molecular structure creation. We treat the optimization as a reinforcement learning problem, using a batch-advantage modification of the policy gradient algorithm - using individual rewards minus the batch average reward to weight the log probability loss. The reinforcement learning agent is tasked with building molecules using this grammar, with the goal of maximizing benchmark scores available from the literature. To do so, the agent has policies both to choose the next node in the graph to expand and to select the next grammar rule to apply. The policies are implemented using the Transformer architecture with the partially expanded graph as the input. We achieve state of the art performance on common benchmarks from the literature, such as penalized logP and QED, with only hundreds of steps (without pre-training) on a budget GPU instance. Competitive performance is obtained on more advanced GuacaMol v2 goal-oriented benchmarks. Coupled with a Transformer based discriminator, the model achieves competitive results on the GuacaMol distribution benchmarks; training is stable over a range of hyperparameter values.

1 Introduction

A major problem facing the exploration of novel molecules for the purposes of drug design is the vast array of potentially useful compounds – estimated to be in the range of 10^{24} and 10^{60} possible drug-like structures [1, 2]. While it is of course necessary to experimentally determine the usefulness, and safety, of candidate drugs in the laboratory, de novo drug design is an approach to finding candidate molecules through either exhaustive search, or through various generative and machine learning models **TODOCITE [refs in Evernote]**. This approach takes the form of an optimization procedure over given target scoring functions, giving pre-screened, promising molecules and thereby reducing drug discovery costs.

Deep learning has now been extensively investigated for encoding and generating molecular graphs [3–5] **TODOCITE [recent in Evernote]**, and remains an area of active research. Typically, the approach taken has been to generate a linear molecular representation, such as the SMILES format [6], with an encoder-decoder network architecture similar to that used in machine translation [7].

This route is, however, not optimal for this problem domain. Unlike written text, a molecule's structure is non-linear – including both cycles and branches. The model is therefore forced not only to learn to optimize molecules on the given benchmark, but also to learn to generate SMILES strings corresponding to chemically valid molecules. This task is non-trivial and robs capacity of the model from the true task at hand.

Preprint. Under review.

A recent development which partially remedied the issue was presented by Kusner, et. al. [8], in which the authors deduce a context-free grammar (CFG) for SMILES strings. This grammar guarantees that only valid SMILES strings will be produced, however not all valid SMILES strings are chemically possible molecules and so some model capacity must still be spent on learning the subset of chemically valid SMILES.

Moving away from linear representations, Kajino [9] proposed the use of a grammar defined on a hypergraph representation of molecular structure. The molecular hypergraph grammar (MHG), a special case of an hyperedge replacement grammar (HRG) [10], uses rules which can be pictured as splitting the molecular graph at non-terminal bonds, and replacing them with another subgraph, thereby constructing any desired molecular structure while guaranteeing that only chemically valid molecules are produced. In particular, this approach does not have issues of invalid atom valences or loss of stereochemistry that other approaches suffer.

In this work, we make use of an optimized form of the MHG, using an approach we term *rule-pair encoding* (RPE) after the commonly known byte-pair encoding of Gage [11]. The RPE technique creates additional grammar rules from the most commonly found subsequent rule pairs in the set of molecules used to deduce the grammar. These new rules allow the model to obtain richer molecular structure, while keeping the rule sequence length reasonable and so producing molecules that would otherwise be impossible with even a generous maximum rule length. We infer our grammar from the GuacaMol [12] training set of 1,273,104 SMILES strings, the resulting grammar is able to express any of these molecules, and with fewer rules than the unoptimized MHG.

The second innovation we present is the use of the Transformer architecture [13] in place of the more typically applied RNNs. Unlike a RNN, the Transformer's information distance between any two inputs is always one, giving the network full information about the sequence so far when selecting the next graph node and rule to expand. This will clearly have an impact on the memory performance of the algorithm, which grows as the square of the sequence length, and so it is beneficial that our optimized MHG, which we call the hypergraph optimized grammar (HOG), provides a concise representation of a given molecule.

Furthermore, in place of the encoder-decoder architecture used in previous work, we treat this problem with a reinforcement learning approach, using a batch-advantage modification of the policy gradient algorithm.

Using this approach, we obtain state-of-the-art performance on both common benchmarks from the literature, such as penalized logP and QED, and on more advanced GuacaMol v2 goal-oriented benchmarks. Coupled with a Transformer based discriminator, the model achieves competitive results on the GuacaMol distribution benchmarks with stable training over a range of hyperparameter values. This is accomplished with only hundreds of steps (without pre-training) on a budget GPU instance.[1]

2 Hypergraph grammar

In order to address the issue highlighted in Sec. 1 relating to SMILES string grammars, we choose to use a *molecular hypergraph grammar* (MHG) as derived by Kajino in [9]. This works by first representing the molecular graph atoms as hyperedges and bonds as hypernodes of a molecular hypergraph. This will produce hypergraphs which are (i) 2-regular and (ii) have constrained cardinality of its hyperedges. (i) ensures that the hypergraph can be decoded back to a molecular graph and the cardinality constraint preserves the valence of each atom.

The MHG is defined over these hypergraphs as a hyperedge replacement grammar (HRG) [10], a context free grammar generating a hypergraph by replacement of hyperedges with other hypergraphs. This approach has a number of desirable properties, such as preserving the number of hypernodes belonging to each hyperedge, thereby preserving an atom's valence – satisfying (i) above. Furthermore, stereochemisty can be encoded directly into the hyperedge replacement rules. MHG is thus guaranteed to produce only chemically valid molecules, allowing our model below to focus on optimizing the target benchmarks, without wasting network capacity on learning to generate valid outputs.

[1] We used an Amazon Web Services p2.xlarge for all training runs

2.1 Parsing algorithm

In order to build the grammar which will be used by the RL agent outlined in 3, we use the approach of [9] to construct a parse tree for each molecule in a training set. By doing so for each molecule, we can identify the set of unique hyperedge replacement rules defining the grammar. Given an input molecular graph, the algorithm to deduce the MHG rules is as follows

1. Find a node, n, at which the graph can be subdivided by cutting a single connected hyperedge
2. Split the graph at n producing a now reduced parent graph and a new child graph containing a new non-terminal node on the cut hyperedge, which we call the *parent node*
3. Iterate steps above until the parent graph can no longer be subdivided in this manner
4. Apply all of the above recursively to the new child graphs

After this, we have a parse tree where the original graph can be reconstructed by replacing parent graphs with children at their respective parent nodes. A tree constructed in this manner will contain only graphs containing a single hyperedge or graphs composed of cycles.

2.1.1 Extraction of Hypergraph Cliques

Though the above procedure will produce a MHG able to represent any molecule, it is not the most efficient representation given the combinatorial complexity in the construction of molecular graph cycles leading to an unnecessarily large set of rules. We now continue to reduce the cycle containing graphs by indentifying any cliques of the remaining graph of length greater than or equal to three, but do not contain the parent node. This is achieved by replacing the entire clique with a new non-terminal graph node in the parent. The child consists of the clique and all edges exiting the clique connected to the new non-terminal parent node.

After removing all such cliques, we continue to remove 2-cliques from any remaining cycles of length greater than five. This greatly reduces the complexity of remaining cycles, and so the number of resultant rules, but discourages the production of triangles and boxes which are undesirable in output molecules from the model.

2.2 Rule-pair compression

Later we shall see that a key constraint in the ability of the model to produce interesting new molecules is the length of the representation of such a molecule in terms of the MHG rules. In order to ensure that the generated molecules terminate in reasonable time, we must constrain the length of allowed rule sequences generated by the model. In order to assist the model in producing more interesting results, we apply an approach we call *rule-pair encoding* (RPE) after Gage's commonly known byte-pair encoding [11].

After having produced parse trees from the grammar inference data set, as described in 2.1 and 2.1.1, we pass through the trees counting occurences of each (parent graph, child graph) pair. Selecting the most common rule pair, we apply expand the parent graph by applying the child producing a new expanded rule which is added to the grammar. The grandchild graphs (children of the old child graph) are then appended to the old parent graph's children to produce the new children of this rule in the parse trees. This entire procedure can then be iterated to extract as many new rules as desired, including occurrences of the new RPE rules. With this new RPE enhanced grammar, the model is capable of producing rich molecular structure, but without resulting in unreasonably long sequences of rules.

2.3 Using the grammar to create new molecules

After following the procedures outlined above, we can collect all unique graphs, taking into account also the position of the parent node. This set of unique graphs are the rules of our MHG. In the sense of a CFG, a rule maps a non-terminal node to a hypergraph containing further terminal and non-terminal nodes. To generate a new molecule we first select a rule which by definition will contain one or more non-terminal nodes. Next, for each non-terminal node in this first graph, we select a rule whose parent node will expand the non-terminal in the parent rule. We recursively apply the same

procedure now over the selected child rules until all non-terminals are replaced with terminals. This is guaranteed to produce a valid molecular hypergraph.

2.3.1 Conditional and unconditional rule frequencies

To assist the model in selecting these rules, we collect the unconditional and conditional rule frequencies as they appear in the inference data set after having inferred the final set of grammar rules. Unconditional frequencies are simply the total count of appearances of each unique rule in the inference set parse trees, whereas conditional frequencies count the occurrence of each rule conditional on the given parent it will expand in the parse tree. These are used as priors for rule selection by the model before learning.

2.3.2 Ensuring expansions terminate

The final challenge we address is making sure the rule expansion terminates before the maximum allowed number of expansion rules has been reached by the agent. To do this, we define the concept of *terminal distance* of a hypergraph node, defined as the length of the shortest sequence of rules needed to expend said node into a sub-hypergraph consisting only of terminal nodes.

We calculate this distance for each hypergraph in our set of grammar rules by means of the following algorithm:

1. Define a set R of all rule hypergraphs observed so far, and seed it with the root token `molecule`.
2. Initialize all parent node terminal distances to ∞.
3. Iterate over all parent nodes n_p of rules r in R, defining R_{n_p} as the set of rules with equivalent parent node n_p.
   (a) For each n_p, the terminal distance is defined as one plus the minimum of terminal distances for all possible child rules of this graph. Defining C(r) as the set of child nodes of the rule r,

$$\mathrm{TD}_n(n_p) = 1 + \min_{r \in R_{n_p}} \left(\sum_{c \in C(r)} \mathrm{TD}_r(c) \right) , \tag{1}$$

   where $\mathrm{TD}_n, \mathrm{TD}_r$ are the node and rule terminal distances respectively. If the child rule, c, is terminal it is assigned terminal distance $\mathrm{TD}_r(c) = 0$.

We repeat step 3. until convergence, that is until the computed terminal distances do not change between iterations.

Finally, we define the terminal distance of a rule hypergraph as the sum of terminal distances of the child nodes,

$$\mathrm{TD}_r(r) = \sum_{c \in C(r)} \mathrm{TD}_r(c) \, . \tag{2}$$

We use the terminal distance concept to make sure the rule expansion terminates before the maximum number of steps, in the following manner: at each step, we consider the number s of steps left and the terminal distance, td, of the sequence generated so far.

At each step, we make sure $\mathrm{td} \leq s$, using induction. First, we choose the maximum rule sequence length to be larger than the terminal distance of the root graph. Second, at each rule selection step we consider all child rules, r, who can expand the next nonterminal in the graph, and only allow those where $\Delta\mathrm{TD}_r(r) + \mathrm{td} \leq s - 1$. This must be a nonempty set because $\Delta\mathrm{TD}_r(r') = -1$ for at least one applicable rule r'. Thus, by the time we run out of steps, that is, $s = 0$, we know $\mathrm{td} = 0$, that is our token sequence consists only of nonterminals.

References

[1] W. P. Walters, Journal of Medicinal Chemistry **62**, 1116 (2019), https://doi.org/10.1021/acs.jmedchem.8b01048, PMID: 30148631.

[2] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, Journal of Chemical Information and Modeling **52**, 2864 (2012), https://doi.org/10.1021/ci300415d, PMID: 23088335.

[3] D. K. Duvenaud *et al.*, Convolutional networks on graphs for learning molecular fingerprints, in *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, pp. 2224–2232, Curran Associates, Inc., 2015.

[4] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, Journal of Computer-Aided Molecular Design **30**, 595 (2016), 1603.00856.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, CoRR **abs/1704.01212** (2017), 1704.01212.

[6] D. Weininger, J. Chem. Inf. Comput. Sci. **28**, 31 (1988).

[7] R. Gómez-Bombarelli *et al.*, CoRR **abs/1610.02415** (2016), 1610.02415.

[8] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato, ArXiv e-prints (2017), 1703.01925.

[9] H. Kajino, CoRR **abs/1809.02745** (2018), 1809.02745.

[10] F. Drewes, H.-J. Kreowski, and A. Habel, Handbook of graph grammars and computing by graph transformation, chap. Hyperedge Replacement Graph Grammars, pp. 95–162, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1997.

[11] P. Gage, The C Users Journal **12**, 23 (1994).

[12] P. Pogány, N. Arad, S. Genway, and S. D. Pickett, Journal of Chemical Information and Modeling **59**, 1136 (2019), https://doi.org/10.1021/acs.jcim.8b00626.

[13] A. Vaswani *et al.*, CoRR **abs/1706.03762** (2017), 1706.03762.