Retrosynthetic Planning Project Report

Yinwei Huang, Yuanqi Yao, Bryan Lee Teng

June 2023

1 Task 1

1.1 Method Selection

Task 1 is to build a model for single-step retrosynthesis. Since reaction template is given in the dataset, this task can be simply treated as a classification task. There are 2 types of commonly used methods, string based methods and graph based methods. The former treat the molecule as a string, represented by the Morgan Fingerprint. The latter treat the molecule as a graph, each atom being a vertex and each bond being an edge.

For graph based methods, this task can be taken as a graph classification task, which is the method we implemented. From the perspective of chemical reactions (some bonds break and some new bonds form), it is also possible to predict the probability of whether a bond would break, meanwhile classifying the template using the two feature vectors of the atoms connected by the breaking bond. We tried on edge classification but failed due to imbalance of edge labels (most bonds are labelled not breaking instead of a potential template).

However, there are about 1/5 of reactions in the test dataset follow unseen templates. There are generative models which generates the model instead of simple classification.

For simplicity, we choose the simplest graph based method, graph classification, for implementation.

1.2 Model Details

1.2.1 Assumptions

We expect the model to calculate feature vectors of each atom, assuming it contains enough information of the chemical properties of the atom. For simplicity, we assume the feature of the molecule is the weighted average of atom features. Calculating feature vectors can be achieved by multiple backbones: GCN, GraphSAGE, GAT, etc.

1.2.2 Implementation

We tested a GCN model and a GraphSAGE model. The model has 4 GraphConv layers (GCN) or 4 SAGEConv layers (GraphSAGE), a single-layer MLP for learning vertex weight based on learned features to aggregate the graph feature, and another single-layer MLP for classification.

They both calculate the feature of a vertex based on its adjacent vertices, while iterating multiple times enables the feature vector to be influenced by nearby but not neighbouring vertices. Compared to GCN model, GraphSAGE has a learnable aggregator for aggregating information from the adjacent vertices.

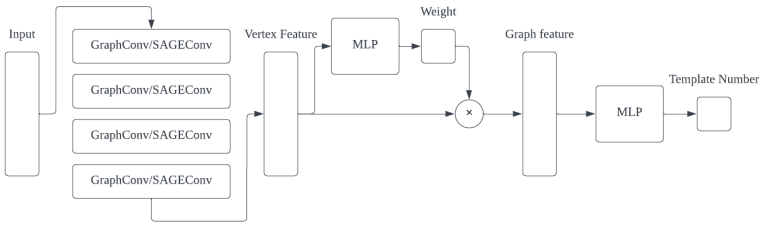

Figure 1: Network structure

1.2.3 Performance

The performances are listed in table 1. GCN performs better than GraphSAGE. It is possible that overfitting happened during the training of GraphSAGE.

Table 1: Performance of GraphSAGE and GCN on test dataset

	GraphSAGE	**GCN**
Top 1	13.86%	22.73%
Top 3	25.44%	37.19%
Top 5	30.50%	42.94%
Top 10	37.83%	49.39%

The overall accuracy of the two models are not very well. Our implementation does not fully utilize the structural information provided by the templates, it simply considers templates as a series of numbers. Furthermore, using weighted average of vertex features as graph feature also ignores the structural information of the molecule.

The advantage is that the calculation is very fast. It can improve the speed for the following tasks. Compared to string based methods, graph based methods are more explainable.

2 Task 2

2.1 Separative Model in Molecule Evaluation

2.1.1 Experimental Background

In task 2, we now aimed at minimizing the error in predicting the synthesis cost of multiple molecules simultaneously. One way to implement this is to predict each molecule separately and then sum them up, with formulation in the form of:

$$cost = f(m_1) + f(m_2) + ... + f(m_k) \tag{1}$$

2.1.2 Algorithm

To achieve this equation, we choose MLP method to fit the input and output training data. Specially, a 3-hidden layers artificial neural network is used in order to better fitting data.

Besides, using dimension reduction tools is also a possible effective way to better fitting the cost and it also fits the formulation of above. With the help of PCA method, MLP module can further decrease its loss and may increase its performance. We also provide test for this idea.

What's more, a convolution neural network is also built to compare the difference and manifest optimal parameters and models. Since convolution network collapses due to the high dimension of the input feature, PCA method is first taken to better fit the input data.

2.1.3 Performance and results

The detail performance are listed in table 2. To quantify the performance of each model, we use average deviation about 100 thousand sample of the testing data as evaluation criterion. Different parameters are applied to them to make sure their best performance and only best performance will be recorded. Error on training set are also listed as a reference.

Table 2: Performance of MLP and CNN on test dataset

Module Name	Average Error	Error on Training set
Normal MLP	1.7882131	0.4283
MLP with PCA	1.8185877	0.5244
CNN with PCA	1.8902924	0.4427

2.2 Integrative Model in Molecule Evaluation

2.2.1 Experimental background

Although each cost of molecules are independent, neural network doesn't show its accuracy since the data volume reaches 400 thousand molecules and their

feature dimension falls at 2048. Another way is to design a function to predict the cost of multiple molecules,with formulation in the form of:

$$cost = f(m_1, m_2, ..., m_k) \tag{2}$$

2.2.2 Algorithm

In this part, data are packed into small data-sets and their cost will be calculate simultaneously.

To achieve this equation, MLP method can also be used as a baseline experiment. Another method to fit this cost is to generate a distance metric depending on each molecules' cosine-similarity. Therefore, a data-set of samples can be filled into a graph using the similarity metric as its adjacency matrix.Then we can use GCN module to fit each data on graph and find whether it will behave better.

2.2.3 Performance and results

In table 3 and table 4, details about MLP and GCN models are presented in list. MLP model successfully reduces error but GCN model only decreases the error on training set but doesn't decrease the average error on testing set.

Table 3: Performance of MLP on test dataset

Group size	Average Error	Error on Training set
5	1.5407702	0.117329
10	1.4883298	0.040953
30	1.4730948	0.017272
100	1.4798063	0.043655

Table 4: Performance of GCN on test dataset

Group size	Average Error	Error on Training set
5	1.8956462	0.2737
10	1.8367051	0.2164
30	1.8572345	0.4359

2.3 Conclusion

It will be easy to draw conclusion from above that integrative model behaves kind of better than separative model.

In integrative model MLP, best group size appears when it become about 30, where error on training set and average error reaches the lowest. Increasing or decreasing group size will enhance error.

However, normal GCN doesn't show a better result while Normal MLP does decrease the average error and also improve the convergence ability. Its best

group size appears when it become about 10, while which is even lower than separative MLP model. That will mainly give the credit to that the summation of the data shows a better insight of the molecules' cost and reduce the complexity of fitting.

Apart from that, MLP using PCA for dimension reduction slightly increases its error.It's error becomes even higher when it comes to CNN model. Since GCN and CNN both performs ineffective, we can conclude that convolution network is not suitable for this regression problem.

In one word, integrative model have certain advantages when it comes to multiple molecules cost regression even if their synthetic cost is independent with each others.

3 Task 3: Multi-step retrosynthesis planning

3.1 Background

In this task, given a target product, the goal is to find the best synthetic path with the lowest cost from a series of backward reactions. In this experiment, I've used different algorithms to benchmark and compare. Specifically, **BFS, DPFS, Monte Carlo Tree Search (MCTS), Retro ***.

In order to effectively compare the search algorithm on retrosynthetic planning, we need to have some precondition fixed. We use the same dataset and cost function for a template reaction throughout the experiment. We used USPTO-190 dataset introduced in the A* algorithm paper. We used a fixed cost model for the one step template reaction with either pre-trained uspto or ringbreaker model.

3.2 Algorithms

DFS and BFS is quite common, they serve as a baseline for our other method in route finding.

MCTS is interesting, it combines the standards of Monte Carlo strategies, which rely upon random sampling and statistical evaluation, with tree-primarily based search techniques. Unlike traditional search algorithms that rely upon exhaustive exploration of the entire seek area, MCTS specializes in sampling and exploring only promising areas of the hunt area. It is extensively used in retrosynthetic planning.

Retro * has another forward cost model to estimate the future path cost (value function). Training the value function involves using retrosynthesis routes constructed based on the USPTO reaction dataset. The forward path model learns the relationships between the Morgan fingerprints of molecules and their corresponding synthesis costs. By leveraging existing reactions in the training set, the Retro* algorithm can estimate the value function offline and act as a static function during path evaluation.

Throughout all experiments, we take the top-50 templates predicted by MLP single step model and apply them on each product to get corresponding reactant lists during single node expansion. The evaluation dataset came from Kaggle Big DRUG (SMILES) DataSet (https://www.kaggle.com/datasets/yanmaksi/big-molecules-smiles-dataset).

3.3 Experiment and Results

For each run, I randomly selected 2 target based on molecule complexity and run them for each algorithm and record the performance. This experimented is repeated for 20 times and take the average as final result. I limit the algorithm to run at most 120 seconds and 100 iterations, the result is recorded in the following table:

Complexity	Algorithm	Node count	Solved route	Time (s)
Simple	BFS	-	-	-
Simple	DPFS	167	21	6.5
Simple	MCTS	77	23	2.2
Simple	Retro Star	1251	26	15
Medium	BFS	-	-	-
Medium	DPFS	150	2	8
Medium	MCTS	480	20	10.8
Medium	Retro Star	2134	24	15

BFS is almost unusable and result in time out even for simple cases. DPFS quickly finds solution but it cannot find all of the good route. Although retro star took longer time to run, it finds many viable path and was able to compare them. Retro star has the most expanded nodes.

When I tested the algorithm against the hard dataset provided by the retro star paper, all of the algorithm except Retro Star struggle to find a solution and stuck in local minima.

3.4 Conclusion

In conclusion, Retro * performs best when the dataset is complex. Greedy DFS always prioritizes the reaction with the highest likelihood thus was able to find first solution musch quicker. MCTS evaluate the reaction probability provided by the one-step model as the prior to bias the search. MCTS is in the middle between Retro Star and DPFS in terms of viable solution and calculation time.

For our future experiment, we might experiment with RetroGraph which is a new graph based neural network search algorithm which result in higher accuracy.

4 Contribution

Name	Student ID	Score	Work
Yinwei Huang	520021911421	34%	Task 1 report and experiment
Yuanqi Yao	520021910344	33%	Task 2 report and experiment
Bryan Lee Teng	520030990017	33%	Task 3 report and experiment

All the result and source-code can also be seen at: Here