Chess-Ant

Release 0.0.9

Akihiro Kuroiwa

Sep 06, 2024

CONTENTS:

CHAPTER

ONE

CHESS-ANT INTRODUCTION

Simulator to solve chess problems with MCTS Solver and Genetic Programming.

`chess_ant.py` is based on the code of deap/examples/gp/ant.py.

1.1 Requirements

On Ubuntu:

```
sudo -H -s
apt install python3-pip
pip3 install -r requirements.txt
exit
```

Or:

```
pip3 install chess
pip3 install deap
pip3 install mcts
pip3 install mcts-solver
```

Or:

```
pip3 install chess-ant
```

- python-chess: a chess library for Python
- DEAP
- MCTS
- chess-ant
- chess-classification
- mcts-solver

1.2 General Usage

Sample chess problems are available in pgn/. Jerry is useful for pasting Forsyth-Edwards Notation (FEN).

```
cd chess-ant/chess-ant/
git checkout -b test-run
python3 chess_ant.py --help
python3 chess_ant.py --auto --fen "7k/1Q6/8/8/5N2/1B6/8/3K4 w - - 0 1"
python3 chess_ant.py -a -c -p "my-pgn" -l1 -n100 -g10 -f "7k/1Q6/8/8/5N2/1B6/8/3K4 w - -␣
↪0 1"
```

If you installed chess-ant from PyPI:

```
chess-ant --help
chess-ant -a -n100 -g5 -f "7r/8/8/8/7k/2q5/6P1/6NK b - - 0 1"
```

This command will output the wrong answer. It will take some time, but the following command will output correctly.

```
chess-ant -a -n1000 -g5 -f "7r/8/8/8/7k/2q5/6P1/6NK b - - 0 1"
```

1.3 Chess-Classification

Version 0.0.1 of genPgn.py contains the Walrus operator, so it only works with Python 3.8 or higher. Please install Pytorch before installing Simple Transformers.

```
sudo -H -s
pip3 install pandas
pip3 install simpletransformers
apt install stockfish
pip3 install chess-classification
exit
genPgn --help
genPgn -l 10 -t 1 -p "train-pgn" -f "3qkbnr/8/8/8/8/8/PPPPPPPP/RNBQKBNR w - - 0 1"
cat train-pgn/train-*.pgn >> train-pgn/1.pgn
rm train-pgn/train-*.pgn
genPgn -l 10 -t 1 -p "train-pgn" -f "rnbqkbnr/pppppppp/8/8/8/8/3QKBNR w - - 0 1"
cat train-pgn/train-*.pgn >> train-pgn/2.pgn
rm train-pgn/train-*.pgn
genPgn -l 10 -t 1 -p "train-pgn" -f "4k3/pppppppp/8/8/8/8/PPPPPPPP/4K3 w - - 0 1"
cat train-pgn/train-*.pgn >> train-pgn/3.pgn
rm train-pgn/train-*.pgn
importPgn -p "train-pgn"
genPgn -l 10 -t 1 -p "eval-pgn" -f "3qkbnr/8/8/8/8/8/PPPPPPPP/RNBQKBNR w - - 0 1"
cat eval-pgn/train-*.pgn >> eval-pgn/1.pgn
rm eval-pgn/train-*.pgn
genPgn -l 10 -t 1 -p "eval-pgn" -f "rnbqkbnr/pppppppp/8/8/8/8/3QKBNR w - - 0 1"
cat eval-pgn/train-*.pgn >> eval-pgn/2.pgn
rm eval-pgn/train-*.pgn
genPgn -l 10 -t 1 -p "eval-pgn" -f "4k3/pppppppp/8/8/8/8/PPPPPPPP/4K3 w - - 0 1"
cat eval-pgn/train-*.pgn >> eval-pgn/3.pgn
rm eval-pgn/train-*.pgn
importPgn -p "eval-pgn"
```

```
from chess_classification.chess_classification import ChessClassification
classification = ChessClassification()
```

Loading a local save:

```
classification = ChessClassification("local-path/your-outputs")
```

Train or retrain:

```
classification.train_and_eval("train-pgn/fen.json", "eval-pgn/fen.json")
```

Test:

```
my_fen = "7r/8/8/8/7k/2q5/6P1/6NK b - - 0 1"
classification.predict_fen(my_fen)
```

predictions	labels
1-0	2
0-1	1
1/2-1/2	0

With `chess_ant.py`:

```
chess-ant -d -n100 -g5 -f "6rk/4pppp/8/8/3Q4/8/RB2PPPP/R6K w - - 0 1"
```

Loading a local save:

```
chess-ant -d -n100 -g5 -f "6rk/4pppp/8/8/3Q4/8/RB2PPPP/R6K w - - 0 1" --model-outputs
↪"local-path/your-outputs"
```

- Simple Transformers
- Start Locally | PyTorch
- pandas
- Chess-Classification

Todo

- It's too slow.
- Low correct answer rate.
- Parallelization.
- Support for other board games like shogi.
- Support for Universal Chess Interface (UCI).
- Docstring.
- Boil spaghetti code.

CHAPTER

TWO

CHESS-ANT ARTICLE

author
Akihiro Kuroiwa

date
2021/12/25

abstract
Like AlphaFold developed by DeepMind, MCTS can be applied to bioinformatics and cheminformatics. Chess-Ant is influenced by AlphaZero, but tries a different method. I would like to show the possibility that the combination of MCTS Solver and Genetic Programming can be put to practical use.

2.1 Influence from AlphaZero

A variant of the Polynomial Upper Confidence Tree (PUCT) introduced in AlphaGo Zero and AlphaZero complements the exploration rate $C(s)$ with the prior probability $P(s, a)$. My chess-ant uses the conventional Upper Confidence Tree (UCT) and replaces the adjustment of the constant C with Genetic Programming (GP)[1]. From $1/\sqrt{1}$ to $1/\sqrt{9}$, GP chooses the value of the constant C according to the conditions. Depending on the value of the constant C, `chess_ant.py` will either actively or passively search.

AlphaZero's PUCT[2345]:

$$a_t = arg\,max_a(Q(s_t, a) + U(s_t, a))$$
$$U(s, a) = C(s)P(s, a)\frac{\sqrt{N(s)}}{1 + N(s, a)}$$
$$C(s) = log\frac{1 + N(s) + c_{base}}{c_{base}} + c_{init}$$

A deep neural network $(p, v) = f_\theta(s)$ with parameters θ evaluates the leaf node s_L :

$$(p, v) = f_\theta(s_L)$$

[1] Cazenave, Tristan. "Evolving Monte-Carlo Tree Search Algorithms." (2007).
[2] AlphaZero: Shedding new light on chess, shogi, and Go
[3] Silver, David et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." Science 362 (2018): 1140 - 1144.
[4] Foster, David. (2017). AlphaGo Zero Explained In One Diagram
[5] Tadao Yamaoka's diary

Initialize:

$$N(s_L, a) = 0$$
$$W(s_L, a) = 0$$
$$Q(s_L, a) = 0$$
$$P(s_L, a) = p_a$$

Update:

$$t \leq L$$
$$N(s_t, a_t) = N(s_t, a_t) + 1$$
$$W(s_t, a_t) = W(s_t, a_t) + v$$
$$Q(s_t, a_t) = \frac{W(s_t, a_t)}{N(s_t, a_t)}$$

	Details
(s,a)	Each state-action pair.
N(s)	The parent visit count.
N(s,a)	The visit count.
W(s,a)	The total action-value.
Q(s,a)	The mean action-value.
P(s,a)	The prior probability of selecting a in s.
C(s)	The exploration rate.
p	A vector of Move probabilities (predicted by a deep neural network).
v	A scalar value (predicted by a deep neural network).

Chess-Ant's UCT[6][7][8][9][10]:

$$a_t = arg\,max_a(Q(s_t, a) + C_{gp}\sqrt{\frac{2lnN(s_t)}{N(s_t, a)}})$$

$$C_{gp} = \begin{cases} 1/\sqrt{1}, \\ 1/\sqrt{2}, \\ 1/\sqrt{3}, \\ 1/\sqrt{4}, \\ 1/\sqrt{5}, \\ 1/\sqrt{6}, \\ 1/\sqrt{7}, \\ 1/\sqrt{8}, \\ 1/\sqrt{9}, \end{cases} \text{If the previously selected node state is under certain conditions}$$

The judgment conditions are as follows:

[6] Auer, Peter et al. "Finite-time Analysis of the Multiarmed Bandit Problem." Machine Learning 47 (2004): 235-256.
[7] Kocsis, Levente and Csaba Szepesvari. "Bandit Based Monte-Carlo Planning." ECML (2006).
[8] Swiechowski, Maciej et al. "Monte Carlo Tree Search: A Review of Recent Modifications and Applications." ArXiv abs/2103.04931 (2021): n. pag.
[9] Wikipedia contributors. "Monte Carlo tree search." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 Oct. 2021. Web. 25 Dec. 2021.
[10] Wikipedia contributors. "." Wikipedia. Wikipedia, 8 Oct. 2021. Web. 25 Dec. 2021.

Conditions	Details
if_improvement	If UCT increases from the previous time
if_shortcut	If move_stack is shorter than last time
if_result	Whether the last time rollout was a win, a loss or a draw
if_pruning	Whether UCT was infinity, -infinity or something else
if_is_check	Is the board of the node selected last time checked?

Also, since GP can be redone in each generation, it is possible to prevent the UCT from being highly evaluated at the beginning of the search based on the number of visits. It may get the effect of this paper[11].

2.2 MCTS Solver

With the introduction of MCTS Solver[12][13], chess-ant will cut branches to speed up and reduce wrong answers.

The reason for the significant changes to the pseudocode in the paper is that the code for the package referenced by the inherited classes is written differently.

In addition, there is no `goto` in Python, so there is extra code. In the pseudo code, everything is completed inside the MCTS Solver, but in `chess_ant.py`, it is written separately in `_executeRound()`. Execute rollout in `mctsSolver()`, perform processing such as pruning, output reward like rollout, and input it to `backpropogate()`.

Like negamax, the MCTS solver is a recursive function and requires a stop condition.

2.3 Parallelization

In `mcts-solver` 0.0.5, with the help of OpenAI's ChatGPT and Google Bard, I modified the code to allow parallel processing[14][15]. Tree parallelization is more difficult than root parallelization because it requires the use of locks. I'm not sure if these changes are working correctly, so they may change significantly in the future. Context managers are useful, but they can get you into trouble if you use them incorrectly.

2.4 Change History

Since both `chess-classification` and `chem-classification` use the same algorithm, I made similar changes at the same time.

In `chess-classification` 0.0.4, fen is separated into tokens to create a dataset. The fen is separated by columns instead of letter by letter.

Corrected by adding `math.sqrt(2)` since there was an error in the UCB1 algorithm until `chess-ant` 0.0.5 manual. It seems that the author of `mcts` initially set `explorationConstant = 1 / math.sqrt(2)` in `mcts.py` to cancel out `math.sqrt(2)`. I added a new terminal called `selectNodeEphemeralConstant` to replace the ephemeral constant in `chess-ant` 0.0.6.

According to Issues #658, Deap does not work with Python 3.10, but with version 3.11.

[11] Imagawa, Takahisa and Tomoyuki Kaneko. "Improvement of State's Value Estimation for Monte Carlo Tree Search." (2017).

[12] Winands, Mark & Björnsson, Yngvi & Saito, Jahn-Takeshi. (2008). Monte-Carlo Tree Search Solver. 25-36. 10.1007/978-3-540-87608-3_3.

[13] Baier, Hendrik & Winands, Mark. (2015). MCTS-Minimax Hybrids. IEEE Transactions on Computational Intelligence and AI in Games. 7. 167-179. 10.1109/TCIAIG.2014.2366555.

[14] Chaslot, Guillaume & Winands, Mark & Herik, H.. (2008). Parallel Monte-Carlo Tree Search. 60-71. 10.1007/978-3-540-87608-3_6.

[15] Soejima, Yusuke & Kishimoto, Akihiro & Watanabe, Osamu. (2009). Root Parallelization of Monte Carlo Tree Search and Its Effectiveness in Computer Go.

In ~/.bashrc or ~/.bash_aliases:

```
# Python version
alias python3='/usr/bin/python3.11'
```

and execute **source** command:

```
source ~/.bash_aliases
```

Another method is **update-alternatives**.

```
update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.10 2
update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.11 1
update-alternatives --config python3
python3 --version
```

To revert to the original version that was set as top priority:

```
update-alternatives --auto python3
```

Please note that as of December 2022, gnome-terminal does not start with Python 3.11 on Ubuntu 22.04.1 LTS.

A more common method is to use venv:

```
sudo apt install python3.11-venv
python3.11 -m venv ~/.venv3.11
source ~/.venv3.11/bin/activate
which pip3
```

To finish:

```
deactivate
```

In chess-classification 0.0.5, you can load a local save.

In order to shorten the training time,[16] the conventional model was changed to the checkpoint google/electra-small-discriminator[17] of the electra model.

My pgn files are more like tactics[18][19] than chess problems, so it's more efficient to create a dataset from tactics. More than 300 rows of pgn data are required for model training and model evaluation in order to increase the accuracy rate.

2.5 Development Plan

Todo

As another project, I will introduce deep learning[20][21] for Natural Language Processing (NLP) in FEN's win / loss evaluation.

genPgn.py automatically plays with stockfish and outputs PGN files. By the way, version 0.0.1 of genPgn.py contains the walrus operator, so it only works with Python 3.8 or higher. importPgn.py creates a dataset from

[16] Rajapakse, Thilina. (2020). Battle of the Transformers: ELECTRA, BERT, RoBERTa, or XLNet
[17] ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators
[18] Download tactics database
[19] Gorgonian's Chess Site

PGN files. `chess_classification.py` generates a trained model with simple transformers. I plan to use this trained model to replace the rollout of `chess_ant.py`, like AlphaZero.

2.6 Reference

2.7 Bibliography

- Home Page of John R. Koza
- Astro Teller | Technical Papers
- Lones, Michael. (2014). Genetic Programming: Memory, Loops and Modules. David Corne: Open Courseware.
- Alpha“Othello” Zero
- Czech, Johannes et al. “Monte-Carlo Graph Search for AlphaZero.” ArXiv abs/2012.11045 (2020): n. pag.
- Prasad, Aditya. (2018). Lessons From Implementing AlphaZero
- chess-alpha-zero
- Yao, Yao. (2018). API Python Chess: Distribution of Chess Wins based on random moves
- Stöckl, Andreas. (2018). Writing a chess program in one day
- Stöckl, Andreas. (2019). An incremental evaluation function and a test-suite for computer chess
- Stöckl, Andreas. (2019). Reconstructing chess positions
- Python Chess
- Fiekas, Niklas. (2015). An implementation of the Bratko-Kopec Test using python-chess
- Bratko-Kopec Test
- Kurt & Rolf Chess Homepage of Kurt Utzinger
- PGN Mentor
- Hart, Alex. (2011). Alpha Beta pruning on a Minimax tree in Python
- PythonChessAi
- easyAI
- Shrott, Ryan. (2017). Genetic Programming applied to AI Heuristic Optimization
- Alpha-Beta Pruning
- Hartikka, Lauri. (2017). A step-by-step guide to building a simple chess AI
- Simplified Evaluation Function
- Brynjulfsson, Erlingur. A Genetic Minimax Game-Playing Strategy
- Öberg, Viktor. “EVOLUTIONARY AI IN BOARD GAMES : An evaluation of the performance of an evolutionary algorithm in two perfect information board games with low branching factor.” (2015).
- Agapitos, Alexandros & Lucas, Simon. (2006). Learning Recursive Functions with Object Oriented Genetic Programming. 3905. 166-177. 10.1007/11729976_15.

[20] Savransky, Dmitriy. (2020). How to Use GPT-2 for Custom Data Generation. INTERSOG Inc.

[21] Noever, David et al. “The Chess Transformer: Mastering Play using Generative Language Models.” arXiv: Artificial Intelligence (2020): n. pag.

- Yu, Tina and Christopher D. Clack. “Recursion , Lambda Abstractions and Genetic Programming.” .
- YouTube channel of David Beazley
- Welcome to AntWiki

CHAPTER

THREE

PY-CHESSBOARDJS

Chess GUI using pywebview and chessboard.js. The main files are `start.py`, `js/my-script.js` and `index.html`.

Chess-Ant is currently too slow to function as a chess engine. To make matters worse, there is a glitch in the call to chess-ant. It is possible to experiment by loading pgn and having it solve the problem.

3.1 Installation

Please read the pywebview and PyGObject manuals, and install dependent packages before proceeding.

If you are Ubuntu user:

```
sudo apt install python3-venv
python3.11 -m venv ~/.venv3.11
source ~/.venv3.11/bin/activate
which pip
pip install py-chessboardjs[gtk]
```

If you want to install it on local repository:

```
cd py-chessboardjs
pip install .[gtk]
```

QT user:

```
pip install py-chessboardjs[qt]
```

CEF user:

```
pip install py-chessboardjs[cef]
```

Install your favorite UCI engine:

```
sudo apt install stockfish
```

3.2 Usage

```
py-chessboardjs-gtk
```

```
py-chessboardjs-qt
```

```
py-chessboardjs-cef
```

3.3 Related Links

- pywebview
- chessboard.js
- chess.js
- Bootstrap

CHAPTER

FOUR

CHEM-ANT INTRODUCTION

Select material candidates to output molecules similar to the target molecule with MCTS Solver and Genetic Programming.

`similarity_ant.py` is based on the code of deap/examples/gp/ant.py.

4.1 Requirements

On Ubuntu:

```
sudo -H -s
apt install python3-pip
pip3 install -r requirements.txt
exit
```

Or:

```
pip3 install deap
pip3 install mcts
pip3 install rdkit
pip3 install global_chem_extensions
pip3 install mcts-solver
```

Or:

```
pip3 install chem-ant
```

If you want to use `chem-ant` with `chem-classification`:

```
pip3 install simpletransformers
```

Or:

```
pip3 install chem-classification
```

- DEAP
- MCTS
- RDKit
- rdkit-pypi
- Global-Chem

- chem-ant
- chem-classification
- mcts-solver

Note

The new package `rdkit` supports Python versions 3.8 through 3.12, whereas `rdkit-pypi` only supports Python versions 3.7 through 3.11.

`Chem-ant` depends on `global-chem-extensions`, but both depend on `rdkit-pypi`. `Chem-ant` version 0.1.0 will depend on `rdkit`, but `global-chem-extensions` will support `rdkit` in v2.0. Therefore, if you want to install `chem-ant` on Python 3.12, you must follow these steps:

1. Get the git repository of global-chem
2. Manually edit `global-chem/global_chem_extensions/requirements.txt`
3. Build and install it

```
git clone https://github.com/Global-Chem/global-chem.git
cd global_chem_extensions/
```

After editing the file `requirements.txt`:

```
sed -i 's/rdkit-pypi/rdkit/g' requirements.txt
pip install .
```

See also

- Global-Chem Pull Request #309
- Global-Chem requirements.txt

4.2 General Usage

By default, you get a list of molecules from `smiles.csv`. The target is Nirmatrelvir. From that list, the best material for the fragments is selected. The output csv file also contains molecules created during the execution of mcts. If you want to reuse the csv file as a smiles list, add **`--select`** option. If you want to run commands directly without installing the packages, execute just like **`python3 similarity_mcts.py --help`**:

```
similarity-mcts --help
similarity-mcts -i -l1 -e3 -r10 -b500 -p train_smiles
similarity-mcts -i -l1 -e3 -r10 -b500 -p eval_smiles
```

If you want to specify a target and execute:

```
similarity-mcts -i -l1 -e3 -r10 -b500 -p train_smiles -t
↪"CC(C)(C)C(NC(=O)C(F)(F)F)C(=O)N1CC2C(C1C1CCNC1=O)C2(C)C"
similarity-mcts -i -l1 -e3 -r10 -b500 -p eval_smiles -t
↪"CC(C)(C)C(NC(=O)C(F)(F)F)C(=O)N1CC2C(C1C1CCNC1=O)C2(C)C"
```

similarity-mcts selects and outputs the candidates that can be the material of the fragments from the smiles list. If you just want to output target-like molecules from the smiles list without running mcts:

```
similarity-genMols --help
similarity-genMols -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C
↪#N)C" -m "CC1=CC=CC=C1C(C)C" "Cc1ccccc1CC(C#N)NC1CCNC1=O" -f "gen2.csv"
```

The StopIteration problem has been fixed since `chem-ant` 0.1.0, so the **similarity-ant** command will run without stopping. I plan to continue improving this bug.

In addition, a new **--GlobalChem** option has been added. This gets smiles from the `global-chem` database as the material for fragments.

```
similarity-ant -n20 -g10 -b 1 -p train_smiles -e1 -c electrophilic_warheads_for_kinases
```

4.3 Chem-Classification

Output dataset in json format for `chem-classification`:

```
importSmiles -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C"␣
↪-p "train_smiles"
importSmiles -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C"␣
↪-p "eval_smiles"
```

If you want to output the dataset for regression model:

```
importSmiles -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C"␣
↪-p "train_smiles" -r
importSmiles -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C"␣
↪-p "eval_smiles" -r
```

Train the classification model and predict the similarity between Nirmatrelvir and YH-53:

```
from chem_classification.similarity_classification import SimilarityClassification
s = SimilarityClassification()
s.train_and_eval("train_smiles/smiles.json", "eval_smiles/smiles.json")
s.predict_smiles_pair([
↪"CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C",
↪"CC(C)CC(C(=O)NC(CC1CCNC1=O)C(=O)C2=NC3=CC=CC=C3S2)NC(=O)C4=CC5=C(N4)C=CC=C5OC"])
```

Loading a local save:

```
s = SimilarityClassification("local-path/your-outputs")
```

Train regression model to predict similarity between Nirmatrelvir and YH-53:

```
from chem_classification.similarity_classification import SimilarityRegression
s = SimilarityRegression()
s.train_and_eval("train_smiles/smiles.json", "eval_smiles/smiles.json")
s.predict_smiles_pair([
↪"CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C",
↪"CC(C)CC(C(=O)NC(CC1CCNC1=O)C(=O)C2=NC3=CC=CC=C3S2)NC(=O)C4=CC5=C(N4)C=CC=C5OC"])
```

Another regression model trained by json files output by **similarity-mcts** can predict the similarity with the target molecule from the material candidates and cooperate with **similarity-ant**:

```
similarity-mcts -i -l2 -e3 -r10 -b100 -p "train_smiles" -f "smiles.json" -j
similarity-mcts -i -l2 -e3 -r10 -b100 -p "eval_smiles" -f "smiles.json" -j
```

> **Note**
>
> From chem-ant 0.0.7, I changed it to create datasets with molecular fragments as tokens, so the difference between the two regression models is gone.

Cooperation between chem-classification and **similarity-ant** (currently not working):

```
similarity-ant -n20 -g5 -b 1 -p gen_smiles -d -o "local-path/your-outputs"
```

Cooperation between regression model of chem-classification and **similarity-ant**:

```
similarity-ant -n20 -g5 -b 1 -p gen_smiles -r -o "local-path/your-outputs"
```

CHAPTER

FIVE

CHEM-ANT ARTICLE

author
Akihiro Kuroiwa

date
2022/07/08

abstract
I started writing `chess-ant` in 2019, but at first I was particular about minimax and the work did not proceed slowly. With the COVID-19 outbreak of the cruise ship Diamond Princess in 2020, when the pandemic was finally beginning to attract attention, I decided to use the `chess-ant` algorithm for the development of therapeutic agents. After that, I read the paper of MCTS solver and the performance improved. At the same time, I learned how to use the cheminformatics software. Even after the SARS-CoV-2 pandemic has converged, the next pandemic is waiting. Let's contribute to society with our skills.

5.1 UCSF Chimera

On my old laptop Fujitsu LIFEBOOK AH42/C, when I try to install UCSF ChimeraX, I get the following error:

```
ERROR: ChimeraX requires OpenGL graphics version 3.3.
Your computer graphics driver provided version 2.1
Try updating your graphics driver.
```

Therefore, in this experiment, I use UCSF Chimera. The advantage is that you can specify the binding site with the mouse.

```
chmod u+x chimera-alpha-linux_x86_64.bin
./chimera-alpha-linux_x86_64.bin
```

In `~/.profile` on Ubuntu:

```
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
```

In my case, there are two installation locations:

```
~/.local/bin/
~/.local/UCSF-Chimera64-2022-05-18/
```

Run **chimera** from the terminal on the command line, or right-click the desktop icon to give execute permission and double-click to launch UCSF Chimera.

Prior to this experiment, get the latest release of the AutoDock Vina installer from GitHub and install it according to the manual. You can check the location of the binary file with the following command:

```
cd ~/.local/bin/
ln -s vina_1.2.3_linux_x86_64 vina
which vina
```

To verify the operation, proceed with the experiment as follows:

1. Create fragments of the target molecule Nirmatrelvir and output some molecules.
2. Material candidates including the target molecule are selected by **similarity-mcts**, and some molecules are output.
3. Get the set difference.

```
similarity-genMols -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C
↪#N)C" -m "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C" -b70␣
↪-p "gen_smiles" -f "gen1-1.csv"
similarity-mcts -l2 -e3 -r10 -b100 -p "gen_smiles" -f "gen1-2.csv"
```

After running, you would see something like this:

```
Material candidates: {'CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C
↪#N)C', 'CCCC1=NC(=C(N1CC2=CC=C(C=C2)C3=CC=CC=C3C4=NNN=N4)C(=O)O)C(C)(C)O'}
```

There are some things to keep in mind when running **similarity-genMols**. Python doesn't distinguish between single and double quotes, but bash and dash do. In addition, you don't need commas on the command line:

```
similarity-genMols -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C
↪#N)C" -m "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C#N)C"
↪"CCCC1=NC(=C(N1CC2=CC=C(C=C2)C3=CC=CC=C3C4=NNN=N4)C(=O)O)C(C)(C)O" -b100 -f gen1-2.csv
similarity-genMols -t "CC1(C2C1C(N(C2)C(=O)C(C(C)(C)C)NC(=O)C(F)(F)F)C(=O)NC(CC3CCNC3=O)C
↪#N)C" -m "CCCC1=NC(=C(N1CC2=CC=C(C=C2)C3=CC=CC=C3C4=NNN=N4)C(=O)O)C(C)(C)O" -b100 -f␣
↪gen1-3.csv
```

```
import pandas as pd
df1_1 = pd.read_csv("gen_smiles/gen1-1.csv", header=0, index_col=0)
df1_2 = pd.read_csv("gen_smiles/gen1-2.csv", header=0, index_col=0)
df1_3 = pd.read_csv("gen_smiles/gen1-3.csv", header=0, index_col=0)
df1_4 = pd.concat([df1_1, df1_1, df1_2, df1_3, df1_3], axis=0)
df1_4.drop_duplicates(subset="smiles", keep=False, inplace=True)
df1_4.sort_values(["lipinski", "dice_similarity"], inplace=True, ascending=False)
df1_4.reset_index(drop=True).to_csv("gen_smiles/gen1-4.csv")
```

Create a ligand file with Open Babel. Open `gen3.csv` and specify the smiles of high-scoring molecule with **similarity-mcts**. Don't forget to add hydrogen atoms and assign partial charges. On Ubuntu:

```
sudo apt install openbabel
obabel -L
obabel -L charges
obabel -h -c -ican -:"CCCC1C2C(CN1C(=O)C1C3C(CN1C(=O)C(F)(F)F)C3(C)C)C2(C)C" -opdbqt -O␣
↪ligand.pdbqt --gen3D --partialcharge gasteiger
```

Let's go back to UCSF Chimera. Open the above file and follow the menu as follows:

1. *File → Fetch Structure by ID → PDB(mmCIF) → 7tll*
2. *File → Open → ligand.pdbqt → file type PDB*
3. *Tools → Surface/Binding Analysis → AutoDock Vina*

Our *Output file* is `all`. Specify *Receptor* and *Ligand*. Check *Resize search volume using* for your mouse. Write vina path in *Executable location*.

In my case, when I specified the binding site with the mouse, the frame was not displayed unless I switched it with the *Presets* menu. When reconfirming the experimental results, open `all.receptor.pdb` and:

1. *Tools → Surface/Binding Analysis → ViewDock → all.pdbqt*
2. *Move → Play*

5.2 AutoDock Vina

Reuse the receptor file output by UCSF Chimera and experiment on the command line. You will prepare your own ligand file. The contents of `conf.txt` are as follows:

```
receptor = all.receptor.pdbqt
ligand = ligand.pdbqt

out = all.pdbqt

center_x = -2.68714
center_y = -1.23572
center_z = 13.8821

size_x = 25.747
size_y = 22.6627
size_z = 22.1881
```

`similarity-mcts` now chose Catechin and the mysterious molecule Gnididin[1]:

```
similarity-mcts -l2 -e3 -r10 -b100 -p "gen_smiles" -f "gen2-2.csv"
```

```
Material candidates: {'C1C(C(OC2=CC(=CC(=C21)O)O)C3=CC(=C(C=C3)O)O)O',
↪'CCCCCC=CC=CC(=O)OC1C(C23C4C=C(C(=O)C4(C(C5(C(C2C6C1(OC(O6)(O3)C7=CC=CC=C7)C(=C)C)O5)CO)O)O)C)C
↪'}
```

`gen2-2.csv`:

```
,smiles,dice_similarity,lipinski
0,C=C(C)C12OC3(CO)OC1C1C4OC4(CO)C(O)C4(O)C(=O)C(C)=CC4C1(O3)C(C)C2CO,0.19672131147540983,
↪1.0
```

Unfortunately, this molecule is made up of fragments produced solely by Gnididin:

```
obabel -h -c -ican -:"C=C(C)C12OC3(CO)OC1C1C4OC4(CO)C(O)C4(O)C(=O)C(C)=CC4C1(O3)C(C)C2CO
↪" -opdbqt -O ligand.pdbqt --gen3D --partialcharge gasteiger
```

[1] Sisakht, M., Mahmoodzadeh, A., & Darabian, M. (2021). Plant-derived chemicals as potential inhibitors of SARS-CoV-2 main protease (6LU7), a virtual screening study. Phytotherapy research : PTR, 35(6), 3262–3274. https://doi.org/10.1002/ptr.7041

Execute AutoDock Vina:

```
vina --config conf.txt
```

```
mode |   affinity | dist from best mode
     | (kcal/mol) | rmsd l.b.| rmsd u.b.
-----+------------+----------+----------
   1       -8.765          0          0
   2        -8.31       1.82      6.828
   3       -8.252      2.441      4.152
   4       -8.086      1.664      7.041
   5        -7.85      2.301      7.148
   6       -7.825      1.726      6.693
   7       -7.797      3.008      6.184
   8       -7.412      2.183      7.011
   9       -7.339      2.426      4.168
```

5.3 Mold for Smiles Casting

The amino acid interaction described in this paper[2] is based on PDB ID: 6LU7, while our experiments are based on the SARS-CoV-2 Mpro Omicron P132H contained in PDB ID: 7TLL. The first three letters of active site amino acid are abbreviations for amino acids, and the rest represent the positions of sequences. Let's check with UCSF Chimera:

1. *Presets* → *Interactive 1 (ribbons)* with **chimera**.
2. Hover your cursor over the receptor's active site amino acid on the binding site to see its location.
3. Display a nucleotide or amino acid sequence alignment with **chimera** from *Tools* → *Sequence* → *Sequence* and save it in fast format.
4. If you want to check the Active site amino acid, right-click on the relevant part of the sequence.

If you cast from a mold, the casting should fit the original mold. That's why I added amino acids and nucleotides to the file `smiles.csv`[3]. Whether the relationship between the binding site and the ligand in docking simulation can be said to be the same, let's experiment with the following method:

1. Convert the relevant part to smiles with `rdkit`. The range is from Phe140 to Glu166 in sequence.
2. The smiles string is so long, let's break it down into fragments and outputs them to some molecules.

```
from rdkit import Chem
from rdkit.Chem import BRICS
Chem.MolToSmiles(Chem.MolFromFASTA("FLNGSCGSVGFNIDYDCVSFCYMHHME"))
smiles =
↪'CC[C@H](C)[C@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](Cc1ccccc1)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)C
↪'
allfrags = set()
allfrags.update(BRICS.BRICSDecompose(Chem.MolFromSmiles(smiles), returnMols=True))
builder = BRICS.BRICSBuild(allfrags)
generated_smiles = []
for i in range(30):
```

(continues on next page)

[2] SAMANT, L., & Javle, V. (2020). Comparative Docking Analysis of Rational Drugs Against COVID-19 Main Protease. ChemRxiv. doi:10.26434/chemrxiv.12136002.v1 This content is a preprint and has not been peer-reviewed.

[3] PubChem

(continued from previous page)

```
    mol = next(builder)
    mol.UpdatePropertyCache(strict=True)
    generated_smiles.append(Chem.MolToSmiles(mol))
generated_smiles
['CSCC[C@H](SC)C(=O)N[C@@H](CCC(=O)O)C(=O)O', 'CSCC[C@H](SC)C(=O)Nc1c[nH]cn1',
↪'CSCC[C@H](SC)C(=O)Nc1ccc(O)cc1', 'CSCC[C@H](SC)C(=O)Nc1ccccc1',
↪'CS[C@@H](CC(C)C)C(=O)Nc1ccc(O)cc1',
↪'CC(C)C[C@H](N[C@@H](CCC(=O)O)C(=O)O)C(=O)Nc1ccc(O)cc1',
↪'CC(C)C[C@H](Nc1ccc(O)cc1)C(=O)Nc1ccc(O)cc1', 'CC(C)C[C@H](Nc1ccccc1)C(=O)Nc1ccc(O)cc1
↪', 'CC(C)C[C@H](Nc1c[nH]cn1)C(=O)Nc1ccc(O)cc1', 'CS[C@@H](CC(C)C)C(=O)Nc1c[nH]cn1',
↪'CS[C@@H](CC(C)C)C(=O)Nc1ccccc1', 'CS[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)O',
↪'CS[C@@H](CC(=O)O)C(=O)NC(=O)[C@H](CC(C)C)SC',
↪'CS[C@@H](CC(C)C)C(=O)NC(=O)[C@@H](SC)C(C)C',
↪'CS[C@@H](CC(N)=O)C(=O)NC(=O)[C@H](CC(C)C)SC', 'CS[C@@H](CC(C)C)C(=O)NC(=O)[C@H](CS)SC
↪', 'CS[C@@H](CC(C)C)C(=O)NC(=O)[C@@H](N)Cc1c[nH]cn1',
↪'CS[C@@H](CC(C)C)C(=O)NC(=O)[C@@H](N)Cc1ccc(O)cc1',
↪'CS[C@@H](CC(C)C)C(=O)NC(=O)[C@@H](N)Cc1ccccc1',
↪'CS[C@@H](CO)C(=O)NC(=O)[C@H](CC(C)C)SC', 'CS[C@@H](CC(C)C)C(=O)NC(=O)[C@H](CC(C)C)SC',
↪ 'CC[C@H](C)[C@H](SC)C(=O)NC(=O)[C@H](CC(C)C)SC', 'CSCC(=O)NC(=O)[C@H](CC(C)C)SC',
↪'CC(C)C[C@H](N[C@@H](CCC(=O)O)C(=O)O)C(=O)Nc1ccccc1',
↪'CC(C)C[C@H](Nc1c[nH]cn1)C(=O)Nc1ccccc1', 'CC(C)C[C@H](Nc1ccc(O)cc1)C(=O)Nc1ccccc1',
↪'CC(C)C[C@H](Nc1ccccc1)C(=O)Nc1ccccc1',
↪'CC(C)C[C@H](Nc1ccc(O)cc1)C(=O)N[C@@H](CCC(=O)O)C(=O)O',
↪'CC(C)C[C@H](Nc1ccccc1)C(=O)N[C@@H](CCC(=O)O)C(=O)O',
↪'CC(C)C[C@H](N[C@@H](CCC(=O)O)C(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)O']
```

1. From the output molecules, select the molecules with good results by docking simulation. Of course, it's a good result among the options.
2. Run **similarity-mcts** targeting that molecule.

```
obabel -h -c -ican -:"CC(C)C[C@H](Nc1ccc(O)cc1)C(=O)N[C@@H](CCC(=O)O)C(=O)O" -opdbqt -O␣
↪ligand.pdbqt --gen3D --partialcharge gasteiger
vina --config conf.txt
```

```
mode |   affinity | dist from best mode
     | (kcal/mol) | rmsd l.b.| rmsd u.b.
-----+------------+----------+----------
   1       -7.192          0          0
   2       -7.014      2.837      4.843
   3       -7.002      1.339      2.292
   4       -7.001      2.143      4.417
   5       -6.894      1.303       2.67
   6       -6.759      2.539       6.62
   7       -6.578      2.329      7.121
   8       -6.547      3.004      7.767
   9        -6.51        1.3      2.908
```

```
similarity-mcts -i -l2 -e3 -r10 -b100 -p "gen_smiles" -f "gen3-2.csv" -t
↪"CC(C)C[C@H](Nc1ccc(O)cc1)C(=O)N[C@@H](CCC(=O)O)C(=O)O"
```

```
Material candidates: {'CC1CCC2C(C(OC3C24C1CCC(O3)(OO4)C)OC)C', 'C(CCN)CC(C(=O)O)N',
↪'CC(C)C[C@H](Nc1ccc(O)cc1)C(=O)N[C@@H](CCC(=O)O)C(=O)O'}
```

```
,smiles,dice_similarity,lipinski
0,CCNC1CCNC1=O,0.5454545454545454,1.0
1,CC(Nc1ccccc1)C(=O)Oc1ccccc1,0.5269607843137255,1.0
2,COC(=O)[C@H](CC(C)C)Nc1ccc(O)cc1,0.5084427767354597,1.0
3,CC(C)C(=O)OC(=O)C(C)C,0.5078125,1.0
4,O=C1NCCC1Nc1ccc(F)cc1,0.5066162570888468,1.0
5,COC(=O)[C@H](CC(C)C)OC,0.5040983606557377,1.0
6,C(c1nn[nH]n1)c1nn[nH]n1,0.4921875,1.0
7,CCOc1nn[nH]n1,0.4765625,1.0
8,COc1ccc(O)cc1,0.46875,1.0
9,CO[C@@H](CCC(=O)O)C(=O)O,0.4609053497942387,1.0
10,CC(C(=O)N1Cc2ccccc2CC1C(=O)O)N1Cc2ccccc2CC1c1ccccc1,0.4494649227110582,1.0
11,CC(C(=O)N1Cc2ccccc2CC1C(=O)O)N1Cc2ccccc2CC1C(=O)O,0.4436183395291202,1.0
12,c1ccc(C2CC3CCCC3N2c2ccccc2)cc1,0.4426666666666666,1.0
13,COC(=O)[C@H](CC(C)C)NC1OC2OC3(C)CCC4C(C)CCC(C1C)C24OO3,0.43861607142857145,1.0
14,O=C(O)C1Cc2ccccc2CN1c1ccccc1,0.4348387096774193,1.0
15,CC1CCC2C(C)C(Nc3ccc(O)cc3)OC3OC4(C)CCC1C32OO4,0.4311717861205916,1.0
16,CO[C@@H](CC(C)C)C(=O)NC1OC2OC3(C)CCC4C(C)CCC(C1C)C24OO3,0.4242761692650334,1.0
17,CC(C(=O)Oc1ccccc1)N1C(c2ccccc2)CC2CCCC21,0.4230287859824781,1.0
18,CCOc1ccccc1C(=O)O,0.4228971962616822,1.0
19,Cc1cc(NC2CCNC2=O)no1,0.4113924050632911,1.0
```

Ignore short smiles:

```
obabel -h -c -ican -:"CC(C(=O)N1Cc2ccccc2CC1C(=O)O)N1Cc2ccccc2CC1c1ccccc1" -opdbqt -O␣
↪ligand.pdbqt --gen3D --partialcharge gasteiger
vina --config conf.txt
```

```
mode |   affinity | dist from best mode
     | (kcal/mol) | rmsd l.b.| rmsd u.b.
-----+------------+----------+----------
   1       -8.417          0          0
   2       -8.302      2.467      6.754
   3       -8.003       4.96      7.503
   4       -7.624      3.907      6.517
   5       -7.506      5.171      7.983
   6       -7.485      2.979      5.385
   7       -7.433      5.416      9.353
   8       -7.375      2.857      5.204
   9       -7.296      2.998      5.526
```

If you need the target molecule itself, the above method may be useful.

Todo

- Separate MCTS solver as another package.
- Is it possible to get a high score by docking simulation without the target molecule?
- Type bool is output as 1.0 or 0.0 in a csv file.

- **similarity-ant** is so slow that it is far from practical.
- Is **similarity-mcts** working properly in the first place?
- Version 0.0.3 of **similarity-mcts** now imports MCTS solver, so the output is slightly different from this document.
- Unravel the entangled spaghetti code.

5.4 Reference

5.5 Bibliography

-
- Python for chemoinformatics
- English version of Python for Chemoinformatics (pdf)
- Sharif, Suliman. Understanding drug-likeness filters with RDKit and exploring the WITHDRAWN database. (2020).
- Panikar, S., Shoba, G., Arun, M., Sahayarayan, J. J., Usha Raja Nanthini, A., Chinnathambi, A., Alharbi, S. A., Nasif, O., & Kim, H. J. (2021). Essential oils as an effective alternative for the treatment of COVID-19: Molecular interaction analysis of protease (Mpro) with pharmacokinetics and toxicological properties. Journal of infection and public health, 14(5), 601–610. https://doi.org/10.1016/j.jiph.2020.12.037
- @cat_lover. . (2021).
- GB-GM
- Jensen, J. (2019). Graph-based Genetic Algorithm and Generative Model/Monte Carlo Tree Search for the Exploration of Chemical Space. ChemRxiv. doi:10.26434/chemrxiv.7240751.v2 This content is a preprint and has not been peer-reviewed.

CHAPTER

SIX

SCRIPTS MANUAL

author
Akihiro Kuroiwa, ChatGPT of OpenAI, Perplexity AI

date
2024/09/05

6.1 `create_vina_config.py` Script Manual

6.1.1 Overview

The `create_vina_config.py` script generates an AutoDock Vina configuration file (`config.txt`) from a specified PDB or mmCIF structure file. This script allows you to set a docking box focused on specific residues.

6.1.2 Usage

```
python create_vina_config.py input_file [-o OUTPUT] [-l LIGAND] [-r RESIDUES] [-p PATH]
```

6.1.3 Arguments

- **input_file** (required): Specifies the input file in PDB or mmCIF format.
- **-o OUTPUT, -–output OUTPUT** (optional): Specifies the name of the output configuration file. The default is `config.txt`.
- **-l LIGAND, –-ligand LIGAND** (optional): Specifies the name of the ligand file. The default is `ligand.pdbqt`.
- **-r RESIDUES, –-residues RESIDUES** (optional): Specifies the residue numbers for setting the docking box. Use a hyphen (`-`) for ranges and commas (`,`) for multiple residues (e.g., `100-105,110,115-120`).
- **-p PATH, –-path PATH** (optional): Specifies the output directory. The default is the current directory.
- **-c CHAIN, –chain CHAIN** (optional): Chain ID to select (e.g., `A`).

6.1.4 Output

- **config.txt**: The generated AutoDock Vina configuration file includes the following details:
  - Receptor and ligand file names.
  - Docking box center coordinates and size.
  - Calculation settings (`exhaustiveness`, `num_modes`, `energy_range`, `cpu`).
  - If residue numbers are specified, they are included as a comment at the beginning of the file.

```
# Residue selection: 100-105,110,115-120

receptor = protein.pdbqt
ligand = ligand.pdbqt
center_x = 25.0
center_y = 20.0
center_z = 30.0
size_x = 10.0
size_y = 10.0
size_z = 10.0
out = out.pdbqt
log = log.txt
exhaustiveness = 8
num_modes = 9
energy_range = 3
cpu = 8
```

6.1.5 Notes

- The `config.txt` file can be edited with a text editor, which is useful for fine-tuning the docking box size, position, and calculation settings.
- If no residue numbers are specified, the docking box is set for the entire structure.

6.2 `prepare_experiment.py` Script Manual

6.2.1 Overview

The `prepare_experiment.py` script automates the preparation of experiments based on a specified PDB ID. It generates an AutoDock Vina configuration file (`config.txt`), a fragment file (`fragments.csv`), and a configuration file (`config.ini`) for `similarity-ant` and `similarity-mcts`. The script creates fragments based on specified residue numbers and outputs these fragments or molecules generated from them in SMILES format to a CSV file.

6.2.2 Usage

```
python prepare_experiment.py pdb_id [-f FORMAT] [-o OUTPUT] [--output_fragments] [--num_
↪smiles NUM]
```

6.2.3 Arguments

- **pdb_id** (required): Specifies the PDB ID for the experiment. The script downloads the PDB file based on this ID.
- **-f {pdb,cif}, –-format {pdb,cif}** (optional): Specifies the input file format. You can choose between `pdb` and `cif`. The default is `pdb`.
- **-o OUTPUT, –-output OUTPUT** (optional): Specifies the name of the output directory. The default is `test-{pdb_id}`.
- **–-output_fragments** (optional): Outputs the fragments themselves in SMILES format. If not specified, the script generates molecules from the fragments and outputs their SMILES.
- **-n NUM_SMILES, –-num_smiles NUM_SMILES** (optional): Specifies the number of SMILES to generate. The default is `10`.
- **-r RESIDUES, –-residues RESIDUES** (optional): Residue selection string (e.g., `100-105,110,115-120`).
- **-c CHAIN, –chain CHAIN** (optional): Chain ID to select (e.g., `A`).

6.2.4 Output

- **config.txt**: The AutoDock Vina configuration file. It includes receptor and ligand file names, docking box center coordinates and size, and calculation settings. This is a text file that can be edited. If residue numbers are specified, this information is included as a comment.
- **fragments.csv**: A CSV file containing the SMILES of fragments or molecules generated from the specified residue numbers.
- **config.ini**: An INI file containing the experimental settings for `similarity-ant` and `similarity-mcts`. Common options are listed under the `DEFAULT` section, while experiment-specific options are listed under each respective section. The configuration file can be edited with a text editor.

6.2.5 Notes

- In `config.ini`, do not use single or double quotes. If multiple values are required, separate them with spaces.
- Boolean, integer, or string values are automatically identified by `run_experiment.py`, but boolean options must be explicitly set with values like `Yes`.
- Options not specified in `config.ini` will be ignored.
- When using `select_ligands.py` to create a ligand file, it is preferable to use the SMILES of generated molecules rather than fragment SMILES, as the latter may cause issues.

6.3 run_experiment.py Script Manual

6.3.1 Overview

The run_experiment.py script reads settings from a config.ini file and executes either a similarity-ant or similarity-mcts experiment. The script outputs the generated molecules' SMILES to a CSV file, which can then be used with select_ligands.py to create a ligand file.

6.3.2 Usage

```
python run_experiment.py config_file [-a | --ant] [-m | --mcts]
```

6.3.3 Arguments

- **config_file** (required): Specifies the config.ini file containing the experiment settings.
- **-a, –ant** (optional): Executes the similarity-ant experiment.
- **-m, –mcts** (optional): Executes the similarity-mcts experiment.

6.3.4 Output

- **generated_smiles.csv**: A CSV file containing the SMILES of the generated molecules from the experiment. This file can be used with select_ligands.py to create a ligand file.

6.3.5 Notes

- The config.ini file must include settings for either similarity-ant or similarity-mcts. run_experiment.py automatically identifies the necessary options and executes the experiment based on the provided settings.
- The resulting SMILES in the CSV file can be used with select_ligands.py for ligand file creation.

6.4 select_ligands.py Script Manual

select_ligands.py is a Python script that selects top ligands from a CSV file and converts them to PDBQT format.

6.4.1 Usage

```
python select_ligands.py <csv_file> [-o OUTPUT_DIR] [-n TOP_N]
```

6.4.2 Arguments

- `csv_file`: Input CSV file containing generated SMILES (required)
- `-o OUTPUT`, `--output OUTPUT`: Output directory for ligand files (default: "ligands")
- `-n TOP_N`, `--top_n TOP_N`: Number of top ligands to select (default: 10)

6.4.3 Important Notes

1. **Fragment Processing**: This script is designed for complete molecule SMILES. Processing fragment SMILES (e.g., `[1*]C(=O)[C@@H]([4*])CCCCN`) directly may result in errors or inaccurate ligand files.
2. **Input Data Verification**: Ensure that the SMILES in your CSV file represent complete molecules. It is recommended to use SMILES of complete molecules generated by combining fragments.
3. **Error Handling**: The script may produce errors if it encounters invalid SMILES or unprocessable molecular structures. Check error messages and modify input data as necessary.
4. **Output Verification**: Always verify that the generated PDBQT files have the expected structure.

6.4.4 Recommended Usage

1. Generate complete molecule SMILES by combining fragments.
2. Save the generated SMILES in a CSV file.
3. Use this script to create ligand files from the complete molecule SMILES.

CHAPTER

SEVEN

CHEM-ANT PROJECT - CO-FOUNDING MEMBERS WANTED

7.1 Overview:

The Chem-Ant Project is actively seeking co-founding members to join in the pursuit of innovative approaches to drug candidate exploration. Currently, the project is in search of individuals to join as co-founders. This project involves simulations utilizing rdkit, deap's genetic programming, and an MCTS solver. The company has not yet been established, and at present, I am the sole project leader.

7.2 Open Positions:

- Software Developers
- Data Scientists
- Cheminformatics Experts
- Marketing Specialist
- Others interested in the project

7.3 Desired Skills:

- Knowledge in Cheminformatics, Molecular Biology, or related fields
- Programming skills (Python, etc.)
- Teamwork and communication abilities

7.4 How to Apply:

If you are interested, please reach out to the email address listed in the git log. Additionally, providing a brief introduction and your skill set would be appreciated.

7.5 Note:

Chem-Ant has not been formally established and is currently progressing as a project. This is a pioneering initiative towards future company formation, welcoming individuals who want to grow together and venture into the forefront of novel drug development.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search