Chess-Ant

リリース *0.0.9*

Akihiro Kuroiwa

2024 年 09 月 06 日

目次

第 1 章

Chess-Ant 紹介

Chess problems を MCTS Solver と Genetic Programming で解くシミュレーター

`chess_ant.py` は deap/examples/gp/ant.py のコードを改変しています。

1.1 準備

Ubuntu で：

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

或いは：

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

或いは：

```
pip3 install chess-ant
```

- python-chess: a chess library for Python
- DEAP
- MCTS

- chess-ant
- chess-classification
- mcts-solver

1.2 使用法

サンプルの chess problems は pgn/ にあります。 Jerry は Forsyth-Edwards Notation (FEN) を貼り付けるのに便利です。

```
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"
```

chess-ant を PyPI からインストールした場合：

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

このコマンドは誤答を出力するでしょう。時間はかかりますが、以下のコマンドは正しく出力するでしょう。

```
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 の genPgn.py はウォルラス演算子を含むので、Python>=3.8 でしか動作しません。Simple Transformers をインストールする前に Pytorch をインストールして下さい。

```
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"
```

(次のページに続く)

(前のページからの続き)

```
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/8/8/8/8/8/3QKBNR 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/8/8/8/8/8/3QKBNR 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()
```

保存したモデルの読み込み：

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

学習か再学習：

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

動作確認：

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

予測	ラベル
1-0	2
0-1	1
1/2-1/2	0

chess_ant.py との併用：

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

保存したモデルの読み込み：

```
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

課題

- 遅い。
- 低い正答率。
- 並列化。
- 将棋のような他のボードゲームへの対応。
- Universal Chess Interface (UCI) への対応。
- Docstring.
- スパゲティ・コードを茹でる。

第 2 章

Chess-Ant 論文

author

Akihiro Kuroiwa

date

2021/12/25

abstract

DeepMind が開発した AlphaFold のように、 MCTS は bioinformatics や cheminformatics へ応用できる。Chess-Ant では AlphaZero に影響を受けつつも、それとは異なる方法を試す。MCTS Solver と Genetic Programming の組み合わせが実用に耐えうるか、その可能性を示すのが目標だ。

2.1 AlphaZero からの影響

AlphaGo Zero や AlphaZero で導入された Polynomial Upper Confidence Tree (PUCT) の変種は事前確率 $P(s,a)$ で定数 $C(s)$ を補完している。Chess-Ant では従来の Conventional Upper Confidence Tree (UCT) を用い、定数 $C(s)$ の調整を Genetic Programming (GP) に置き換える[*1]。$1/\sqrt{1}$ から $1/\sqrt{9}$ まで GP は状況に応じて定数 C の値を選ぶ。定数 C の値により、 `chess_ant.py` は積極的に調べたり、消極的な調べ方をする。

AlphaZero's PUCT[*2][*3][*4][*5]:

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

[*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

Deep neural network $(p, v) = f_\theta(s)$ はパラメータ θ により末端のノード s_L を評価する：

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

初期化：

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

更新：

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

	詳細
(s,a)	それぞれの state-action のペア
N(s)	親（ノード）の訪問数
N(s,a)	訪問数
W(s,a)	action-value の合計
Q(s,a)	action-value の平均
P(s,a)	s での a を選択する事前確率
C(s)	Exploration（と Exploitation）の割合
p	（Deep neural network により予測された）手を選ぶ確率のベクトル
v	（Deep neural network により予測された）スカラー値

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

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

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

判断する条件は以下の通り：

条件	詳細
if_improvement	前回よりも UCT が増加したら
if_shortcut	move_stack が前回よりも短くなったら
if_result	前回の rollout が勝ちか負けか引き分けか
if_pruning	UCT が無限大か無限小か、何れでもないか
if_is_check	前回選ばれた node で check だったか

また、GP は各 generation において、やり直すことができるので、訪問回数による検索初期の UCT の高評価を防げる。この論文のような効果を得られるかも知れない[*11]。

2.2 MCTS Solver

MCTS Solver[*12][*13] の導入により、高速化と誤答を減らす為の枝切りを行う。

論文の擬似コードを大幅に変更している理由は、継承クラスが参照する パッケージ のコードの書き方が異なるからだ。

[*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. "モンテカルロ木探索." Wikipedia. Wikipedia, 8 Oct. 2021. Web. 25 Dec. 2021.
[*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.

さらに、Python では goto が無いので、余計なコードがある。擬似コードでは MCTS Solver 内部で全て完結するが、 chess_ant.py では _executeRound() に分けて書いてある。 mctsSolver() 内で rollout を実行し、枝切りなどの処理をし、rollout と同様に reward を出力し、 backpropogate() に入力する。

Negamax と同様に MCTS solver も再帰関数であり、停止条件が必要だ。

2.3 並列処理

mcts-solver 0.0.5 から、OpenAI の ChatGPT と Google Bard の助けを得てコードを変更し、並列処理[*14][*15] を行えるようにした。Tree 並列化は root 並列化に比べて、locks を用いなければならないので作業が難しい。これらの変更が正しく動作しているか確証が持てないので、今後大幅に変更するかも知れない。Context managers は便利だけれども、使用法を誤ると大変なことになる。

2.4 変更履歴

chess-classification と chem-classification は同じアルゴリズムを用いているので、同時期に同じ変更を施している。

chess-classification 0.0.4 で fen を token に区切ってデータセットを作成するようにした。その fen は 1 文字ずつではなく、一列ごとに区切ってある。

chess-ant 0.0.5 まで UCB1 の数式に誤りがあったので math.sqrt(2) を追加し、訂正した。mcts の作者が、mcts.py 内で explorationConstant = 1 / math.sqrt(2) を初期値としたのは、math.sqrt(2) を打ち消すためと思われる。新たに selectNodeEphemeralConstant という terminal を chess-ant 0.0.6 で追加し、ephemeral constant の代替とした。

Issues #658 によると、Deap は Python 3.10 で動作せず、version 3.11 で動作する。

~/.bashrc 或いは ~/.bash_aliases で：

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

続いて **source** を実行する：

```
source ~/.bash_aliases
```

別の方法として、 **update-alternatives** がある。

[*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.

```
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
```

最優先に設定した元のヴァージョンに戻すには：

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

2022 年 12 月現在、gnome-terminal は Ubuntu 22.04.1 LTS の Python 3.11 では起動しないことを覚えておいて欲しい。

より一般的な方法は venv を用いる：

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

終了するには：

```
deactivate
```

chess-classification 0.0.5 から、保存したモデルの読み込みが出来る。

学習時間を短縮するために、[*16] checkpoint が google/electra-small-discriminator[*17] の electra model に変更した。

私が用意した pgn ファイルは chess problems と言うよりも寧ろ tactics[*18][*19] に近いので、データセットは tactics から作成する方が効率が良い。正解率を高めるためには、モデルの学習とモデルの評価にそれぞれ 300 行以上の pgn データが必要だ。

2.5 開発予定

課題

別のプロジェクトとして FEN の勝敗評価に Natural Language Processing (NLP) 用の deep learning[*20][*21] を導入する。

[*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

genPgn.py は stockfish で自動対局をし、PGN files を出力する。ちなみに、Version 0.0.1 の genPgn.py はウォルラス演算子を含むので、Python>=3.8 でしか動作しない。 importPgn.py は PGN files からデータセットを作成する。 chess_classification.py は Simple Transformers による学習済みモデルを生成する。この学習済みモデルを用いて、AlphaZero のように chess_ant.py の rollout の代用とする予定だ。

2.6 参照

2.7 参考文献

- 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

[*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.

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

第 3 章

py-chessboardjs

pywebview と chessboard.js を利用した Chess GUI です。主要ファイルは `start.py`, `js/my-script.js` と `index.html` です。

Chess-Ant は遅すぎてチェス・エンジンとして機能しません。更に、チェス・エンジンを呼び出す際の不具合もあります。実験目的として pgn ファイルを読み込み、chess problem を解かすことは出来ます。

3.1 インストール方法

まず pywebview と PyGObject のマニュアルを読み、作業前に依存パッケージをインストールして下さい。

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 使用法

```
py-chessboardjs-gtk
```

```
py-chessboardjs-qt
```

```
py-chessboardjs-cef
```

3.3 関連リンク

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

第 4 章

Chem-Ant 紹介

MCTS Solver と Genetic Programming により、目標分子に似た分子を出力する為の材料候補を選ぶ。

`similarity_ant.py` は deap/examples/gp/ant.py のコードを改変しています。

4.1 準備

Ubuntu で：

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

或いは：

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

或いは：

```
pip3 install chem-ant
```

`chem-ant` を `chem-classification` と共に使いたければ：

```
pip3 install simpletransformers
```

或いは：

```
pip3 install chem-classification
```

- DEAP
- MCTS
- RDKit
- rdkit-pypi
- Global-Chem
- chem-ant
- chem-classification
- mcts-solver

注釈

新しいパッケージ `rdkit` は Python version 3.8 から 3.12 をサポートしているが、`rdkit-pypi` は Python version 3.7 から 3.11 のみをサポートしている。

`chem-ant` は `global-chem-extensions` に依存しているが、両者とも `rdkit-pypi` に依存している。`chem-ant` version 0.1.0 は `rdkit` に依存するよう変更されるが、`global-chem-extensions` は v2.0 で `rdkit` に対応する予定だ。従って、Python 3.12 に `chess-ant` をインストールする場合は、以下の手順が必要だ：

1. `global-chem` の git リポジトリを入手する
2. `global-chem/global_chem_extensions/requirements.txt` を手作業で編集する
3. ビルドしてインストールする

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

`requirements.txt` を編集した後：

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

参考

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

4.2 使用法

初期値では、 `smiles.csv` から分子のリストを得る。ターゲットは Nirmatrelvir だ。そのリストからフラグメントの材料として最適なものが選ばれる。出力される csv ファイルには、mcts 実行途中で出来た分子も含まれる。csv ファイルを smiles list として再利用したければ、 **`--select`** オプションを追加する。インストールせずに直接実行したければ、 **`python3 similarity_mcts.py --help`** とする。

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

ターゲットを指定して実行したければ：

```
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` は、smiles のリストからフラグメントの材料と成り得る候補を選び出力する。mcts を実行せず、単にその smiles list からターゲットに似た分子を出力したければ：

```
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"
```

chem-ant 0.1.0 から StopIteration の問題を解決したので、**`similarity-ant`** コマンドが止まらず動作するようになった。この不具合は今後も改善を目指す。

更に、**`--GlobalChem`** オプションを新たに追加した。これは `global-chem` database から smiles を fragments の材料として取得する。

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

4.3 Chem-Classification

chem-classification 用に json format のデータセットを出力する：

```
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"
```

regression model 用のデータセットを出力したければ：

```
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
```

classification model を訓練し、Nirmatrelvir と YH-53 の類似性を予測する：

```
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"])
```

保存したモデルの読み込み：

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

regression model を訓練し、 Nirmatrelvir と YH-53 の類似性を予測する：

```
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"])
```

similarity-mcts により出力された json files により訓練された、もう一つの regression model は、材料候補からターゲット分子との類似性を予測する。そして、 **similarity-ant** と連携できる：

```
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
```

注釈

chem-ant 0.0.7 から、分子のフラグメントをトークンとしてデータセットを作成するように変更したので、2 つの regression models の差は無くなった。

chem-classification と **similarity-ant** の連携（現状では機能せず）：

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

Regression model の chem-classification と **similarity-ant** の連携：

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