Molecular generation by Fast Assembly of (Deep)SMILES fragments

François Bérenger - Tsuda Laboratory

東京大学
THE UNIVERSITY OF TOKYO

OCaml

29/06/2022

Outline

1. Molecular Generation 101
2. A Chemistry File Format: SMILES
3. Simpler SMILES: DeepSMILES
4. A Recurrent Neural Network (RNN) to Generate Molecules
5. Generating Molecules: The FASMIFRA Way
6. FASMIFRA Results

Molecular Generation 101

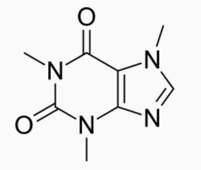


Figure 1: Caffeine: a popular molecule among programmers. Orally available drug, water soluble, classified as a central nervous system stimulant, 14 “heavy atoms”.

To generate optimized molecules on a computer, you might combine:

- i) a molecular generator ← **This talk!**
- ii) scoring function(s)
- iii) an applicability domain
- iv) an optimization algorithm

A Molecular Encoding / File Format: SMILES

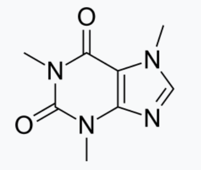


Figure 2: SMILES: 'CN1C=NC2=C1C(=O)N(C(=O)N2C)C'

- Simplified Molecular-Input Line-Entry System (Weininger, 1998).
- Linear encoding of a molecular graph.
- The most useful chemical file format?
- Not unique (possibly several SMILES for a given molecule; exploitable for data-augmentation in ML).
- Compact format, compresses well, human-readable for *small* molecules.

A SMILES variant: DeepSMILES

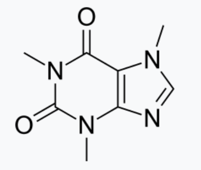


Figure 3: SMILES: 'CN1C=NC2=C1C(=O)N(C(=O)N2C)C'. DeepSMILES no-ring-opening variant: 'CNC=NC=C5C(=O)N(C(=O)N6C)C'

- SMILES are difficult to generate correctly by computers (full specification `http://opensmiles.org/`).
- Noel O'Boyle and Andrew Dalke came up with a simpler syntax called DeepSMILES ("deep" in the name probably means "for deep-learning").
- Several possible flavors of DeepSMILES (no-ring-opening, no-branch-opening).
- In this talk we only consider the no-ring-opening DeepSMILES flavor.

A RNN to Generate Molecules (J. Arus-Pous; 2019)

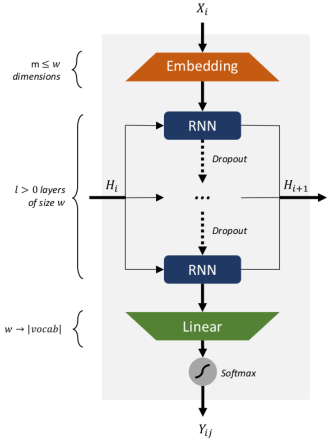


Figure 4: X_i: one-hot encoded input token. Hyper parameters: at least $(w, l, dropout)$. DNNs: slow to train, difficult to design, require ample training data. In the literature, one of the fast molecular generators on a GPU.

FASMIFRA 1/4: Typing Atoms

$type(a_i) = (\pi, e, h, f)$

- a_i: an atom of the molecule.
- π: number of pi electrons.
- e: chemical element symbol (or atomic number).
- h: number of bonded heavy atom neighbors.
- f: formal charge.

Many possible atom typing schemes; you can use atom types from your favorite force field.

FASMIFRA 2/4: Typing Bonds Precisely

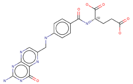
$type(b_j) = (type(a_i), BO(b_j), type(a_{i+1}))$

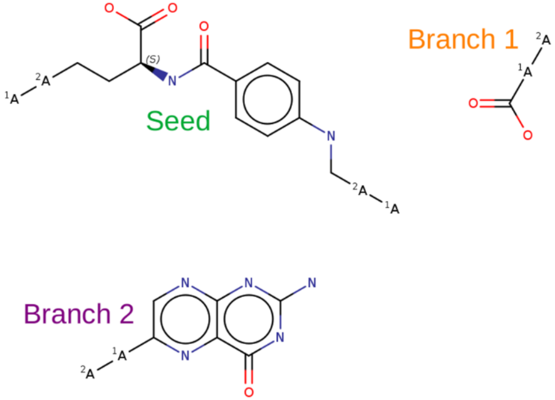
- The natural way of typing bonds would be to use the Bond Order (BO).
- But, if we want to type bonds more precisely, we can extend the bond type to also include atom types of the bonded atoms.
- b_j: a bond of the input molecule, between atoms a_i and a_{i+1}.
- This precise bond typing scheme is very important (cf. training-set distribution matching property later).

FASMIFRA 3/4: Tagging Cleaved Bonds

- We could cut a molecule into fragments.
- We can also just annotate in a valid SMILES string which bonds were selected for cleavage.
- Our prototype's fragmenting scheme only cleaves single bonds, not involved in rings and not connected to a stereo center.
- In fact, FASMIFRA is *parameterized* by a molecular fragmenting scheme $\mathbb{F}$ (constraint: $\mathbb{F}$ must not open rings).

FASMIFRA 3/4: Tagging Cleaved Bonds





```
N([C@@H](CC[2*][1*]C(O)=O)C(O)=O)C(c1ccc(NC[2*][1*]c2cnc3nc(N)[nH]c(=O)c3n2)cc1)=O
N([C@@H](CC[2*][1*]C(O)=O)C(O)=O)C(cccc(NC[2*][1*]ccncnc(N)[nH]c(=O)c6n%10)cc6)=O
```

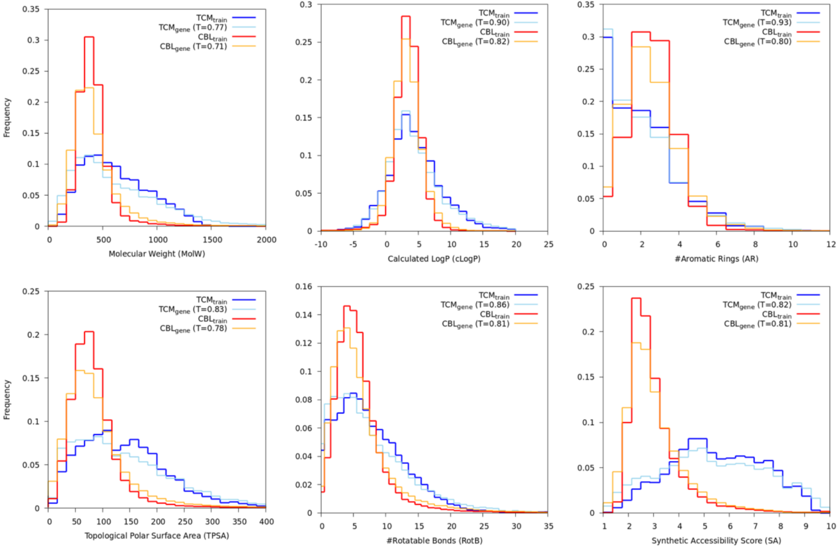
FASMIFRA 4/4: Assembling Fragments Algorithm

Property 1: FASMIFRA generates *only* valence-correct molecules

To generate one molecule:

- Uniform random draw seed fragment.
- Attach compatible (w/ correct bond type) branch fragments until no tagged cut bond is left (i.e. in the SMILES under construction, tagged cleaved bonds are replaced by molecular fragments).
- With DeepSMILES: almost only string operations; an array of strings (all possible seed fragments), a hash table of branch fragments arrays (arrays of compatible branch fragments) indexed by cleaved bond type.

Property 2: Training-Set Distribution Matching



Property 3: Molecular Generation Speed

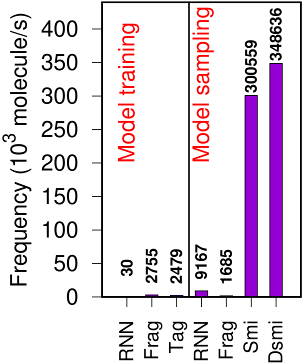


Figure 5: **Left:** model training in molecule/s. RNN: DNN on GPU; Frag: SMILES fragments; Tag: SMILES w/ tagged cut bonds. **Right:** model sampling in 10^3 molecule/s. Smi: FASMIFRA generating SMILES; Dsmi: FASMIFRA generating DeepSMILES.

Comparison with Other Methods

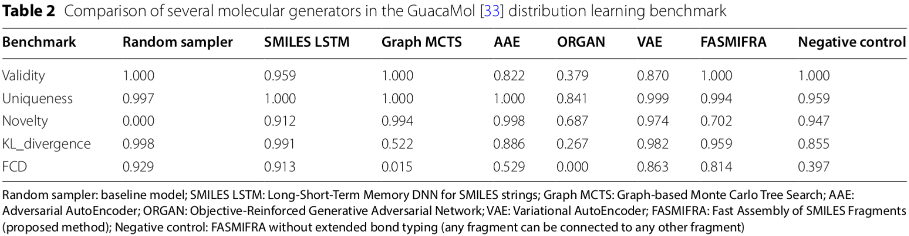


Table 2 Comparison of several molecular generators in the GuacaMol [33] distribution learning benchmark

Benchmark	Random sampler	SMILES LSTM	Graph MCTS	AAE	ORGAN	VAE	FASMIFRA	Negative control
Validity	1.000	0.959	1.000	0.822	0.379	0.870	1.000	1.000
Uniqueness	0.997	1.000	1.000	1.000	0.841	0.999	0.994	0.959
Novelty	0.000	0.912	0.994	0.998	0.687	0.974	0.702	0.947
KL_divergence	0.998	0.991	0.522	0.886	0.267	0.982	0.959	0.855
FCD	0.929	0.913	0.015	0.529	0.000	0.863	0.814	0.397

Random sampler: baseline model; SMILES LSTM: Long-Short-Term Memory DNN for SMILES strings; Graph MCTS: Graph-based Monte Carlo Tree Search; AAE: Adversarial AutoEncoder; ORGAN: Objective-Reinforced Generative Adversarial Network; VAE: Variational AutoEncoder; FASMIFRA: Fast Assembly of SMILES Fragments (proposed method); Negative control: FASMIFRA without extended bond typing (any fragment can be connected to any other fragment)

Despite its simplicity: FASMIFRA stands very well the comparison with other (much more complex and slower!) methods.

All questions are welcome!

- Thanks to prof. Koji Tsuda (Todai) for funding.
- Software: `https://github.com/UnixJunkie/FASMIFRA`.

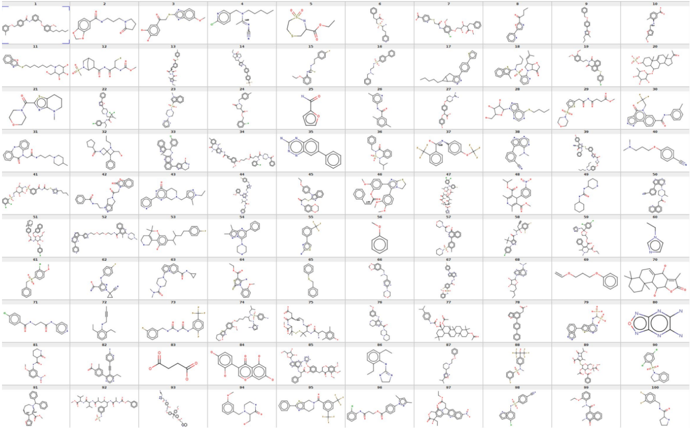


Figure 6: 100 molecules generated by FASMIFRA (ChEMBL-24 training set).