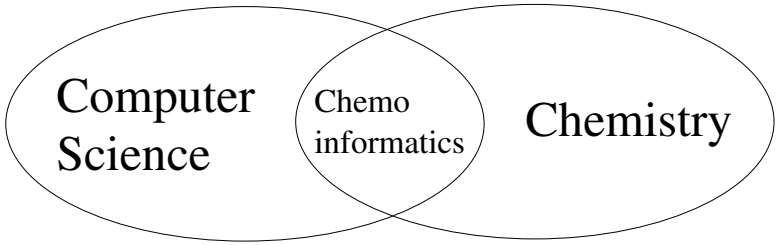
Générer 340000 molécules valides par seconde sur un seul cœur et à dos de chameau

François Bérenger
Tsuda Laboratory - The University of Tokyo

09/12/2021

Outline

Chemoinformatics 101



- Computer science for chemical data.
- Computer-Aided Drug Discovery (CADD).
- != computational chemistry (but sometimes close).
- != quantum chemistry (not at this level of details; much faster methods).
- != structural bioinformatics (but sometimes we work with the same data).
- Chemoinformatics methods process molecules at high speed.
- Good chemoinformatics impacts the real world and is a collaborative science (w/ chemists, biologists).

Molecular Generation 101

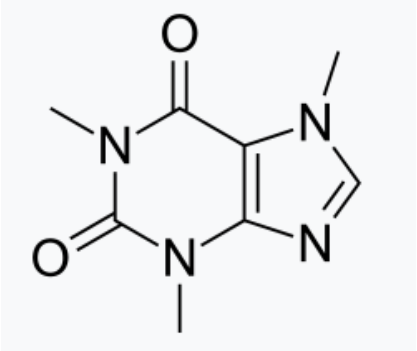


Figure 1: Caffeine: a popular molecule among programmers. Orally available drug, water soluble, classified as a central nervous system stimulant, 14 "heavy atoms".

To generate on the computer (useful) molecules, you need:

- i) a molecular generator $\leftarrow$ This talk !
- ii) a scoring function.
- iii) an optimization algorithm.

A Molecular Encoding / File Format: SMILES

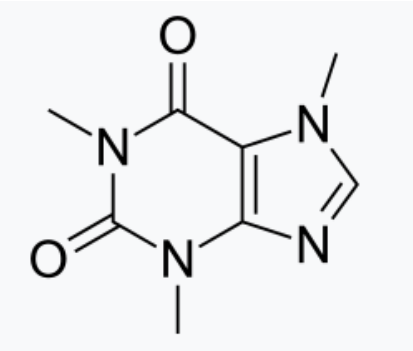


Figure 2: SMILES: 'CN1C=NC2=C1C(=O)N(C(=O)N2C)C'

- Simplified Molecular-Input Line-Entry System (SMILES).
- Linear encoding of a molecular graph.
- The most useful chemical file format?
- Not unique (possibly several SMILES for a given molecule [1]).
- Compact format (also, compress well).
- Human-readable (for small molecules).

[1] Can be useful for machine-learning.

A SMILES Tokenizer in OCaml

FASMIFRA doesn't need a complete SMILES parser.

```
let tokenize_full (s: string): input_smi_token list =
  let res =
    L.map Str.(function
        | Delim "(" -> [[[[Open_paren]]]]
        | Delim ")" -> [[[[Close_paren]]]]
        | Delim _ -> assert(false) (* parens_regexp would be wrong then *)
        | Text a ->
          L.map Str.(function
              | Delim cut_bond_str -> [[[parse_cut_bond cut_bond_str]]]
              | Text b ->
                L.map Str.(function
                    | Delim bracket_atom_str -> [[Bracket_atom bracket_atom_str]]
                    | Text c ->
                      L.map Str.(function
                          | Delim double_digit_rc ->
                            [Ring_closure (parse_double_digit_ring_closure double_digit_rc)]
                          | Text d ->
                            L.map Str.(function
                                | Delim single_digit_rc ->
                                  Ring_closure (parse_single_digit_ring_closure single_digit_rc)
                                | Text e -> Rest e
                              )
                              (Str.bounded_full_split single_digit_regexp d 1024)
                        )
                        (Str.bounded_full_split double_digits_regexp c 1024)
                  )
                  (Str.bounded_full_split bracket_atom_regexp b 1024)
            )
            (Str.bounded_full_split cut_bond_regexp a 1024)
      )
      (* WARNING: bug if more than 1024 tokens in the string ! *)
      (Str.bounded_full_split parens_regexp s 1024) in
  L.flatten (L.flatten (L.flatten (L.flatten res)))
```

A SMILES variant: DeepSMILES

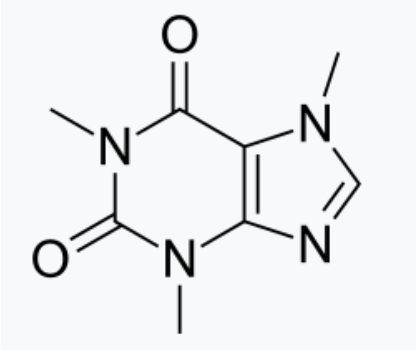


Figure 3: SMILES: 'CN1C=NC2=C1C(=O)N(C(=O)N2C)C'. DeepSMILES no-ring-opening variant: 'CNC=NC=C5C(=O)N(C(=O)N6C)C'

- The SMILES syntax is 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 refer to "deep-learning").

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

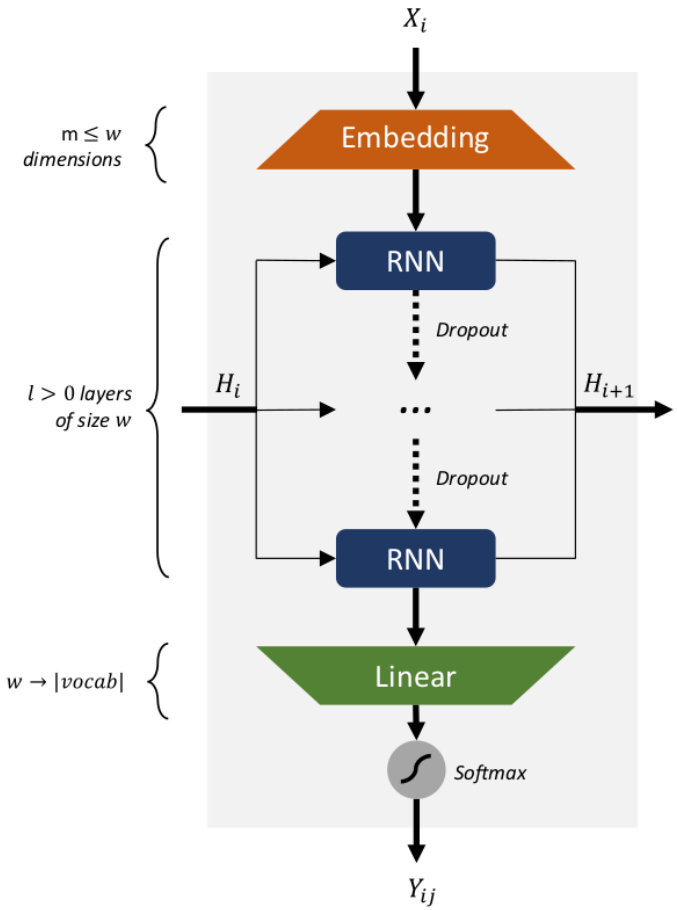


Figure 4: X_i: one-hot encoded input token. Hyper parameters: at least $(w, l, dropout)$. DNNs: slow to train, difficult to design, require ample training data. In the literature, one of the fast molecular generators.

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; this one is quite useful in chemoinformatics.

FASMIFRA 2/4: Typing Bonds (Precisely)

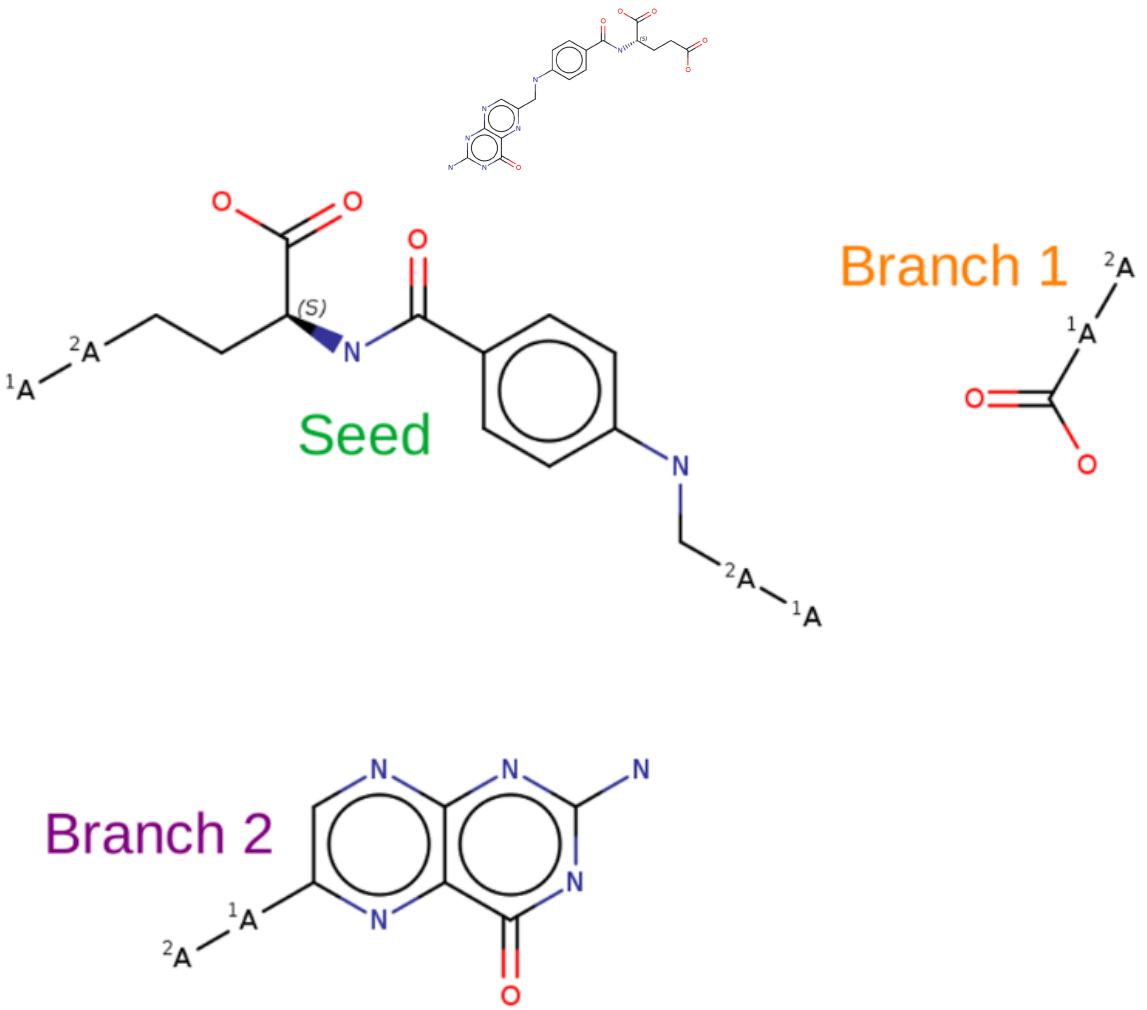
$type(b_j) = (type(a_i), BO(b_j), type(a_{i+1}))$

- 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: Fragmenting/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.
- For several reasons, 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: Fragmenting/Tagging Cleaved Bonds



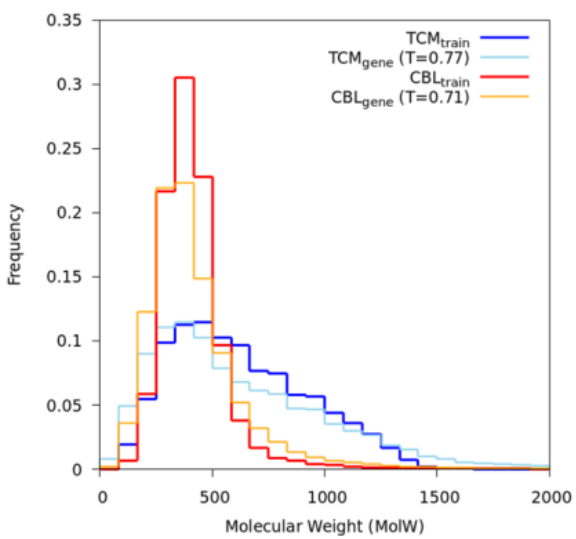
N([C@@H](CC[2*][1*]C(O)=O)C(O)=O)C(c1ccc(NC[2*][1*]c2cnc3nc(N)[nH]c(=O)c3n2)cc1)=O
N([C@@H](CC[2*][1*]C(O)=O)C(O)=O)C(cccc(NC[2*][1*]ccncnc(N)[nH]c(=O)c6n%10)cc6)=O

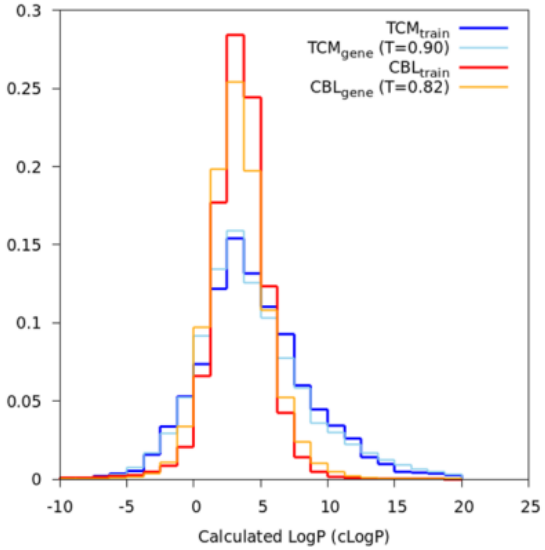
FASMIFRA 4/4: Assembling Fragments Algorithm

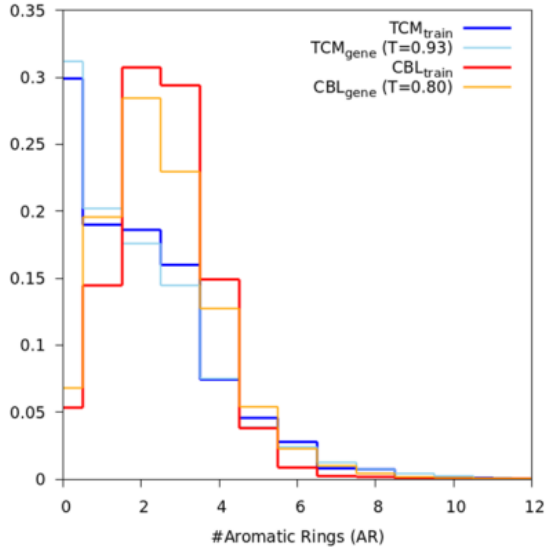
Property 1: this algorithm generates *only* valence-correct molecules.
To generate one molecule:

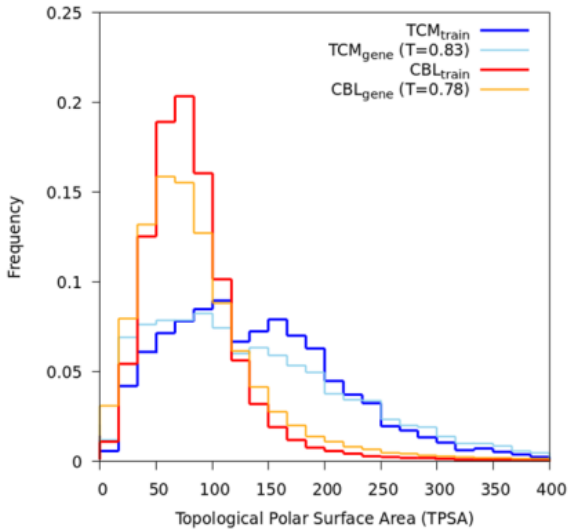
- Uniform random draw seed fragment.
- Attach compatible (w/ correct bond type) branch fragments until no tagged cut bonds are left.
- I.e. in the SMILES under construction, tagged cleaved bonds are replaced by molecular fragments.
- With DeepSMILES: almost only string operations; an array of strings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

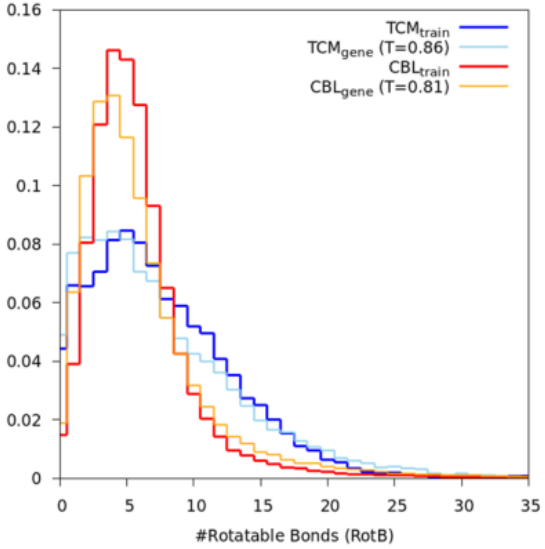
Property 2: Training-Set Distribution Matching

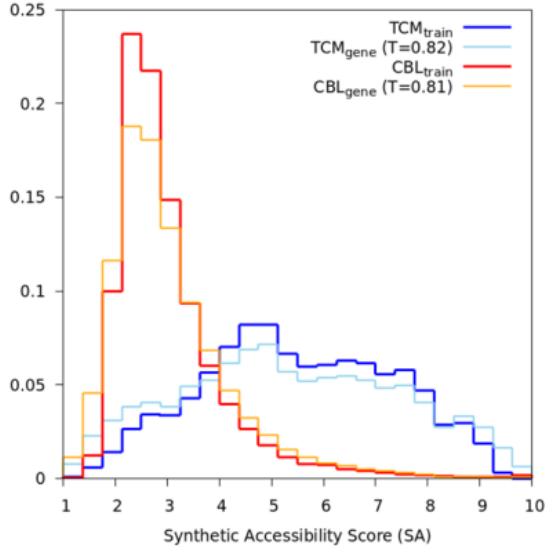












Property 3: Speed

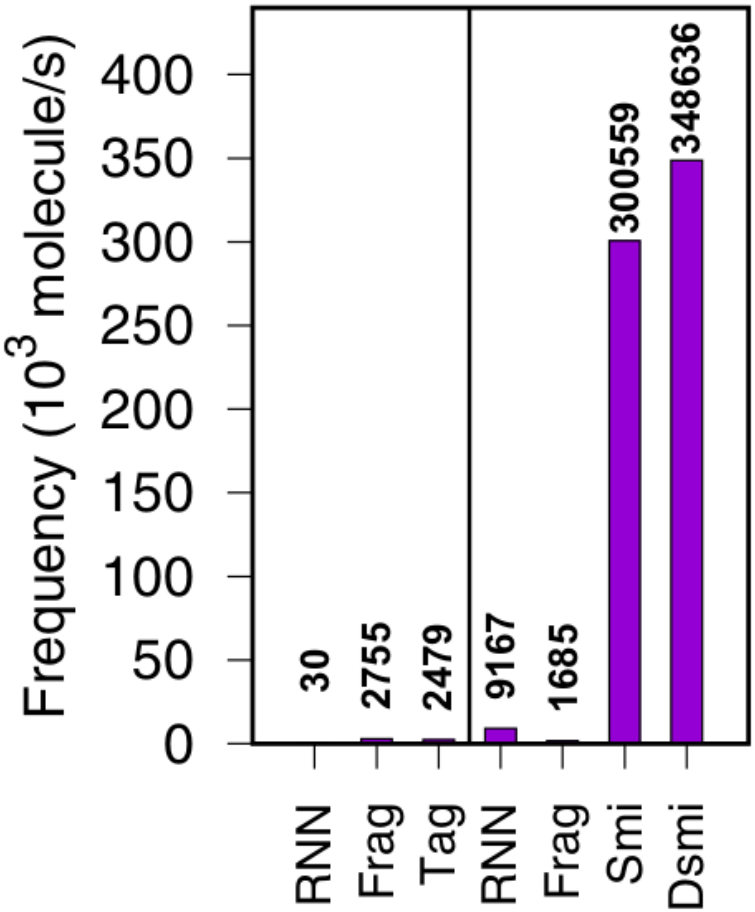


Figure 5: **Left:** model training. RNN: DNN on GPU; Frag: SMILES fragments; Tag: SMILES w/ tagged cut bonds. **Right:** model sampling. Smi: FASMIFRA generating SMILES; Dsmi: FASMIFRA generating DeepSMILES.

Performance Comparison with Other Methods

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

All questions are very welcome!

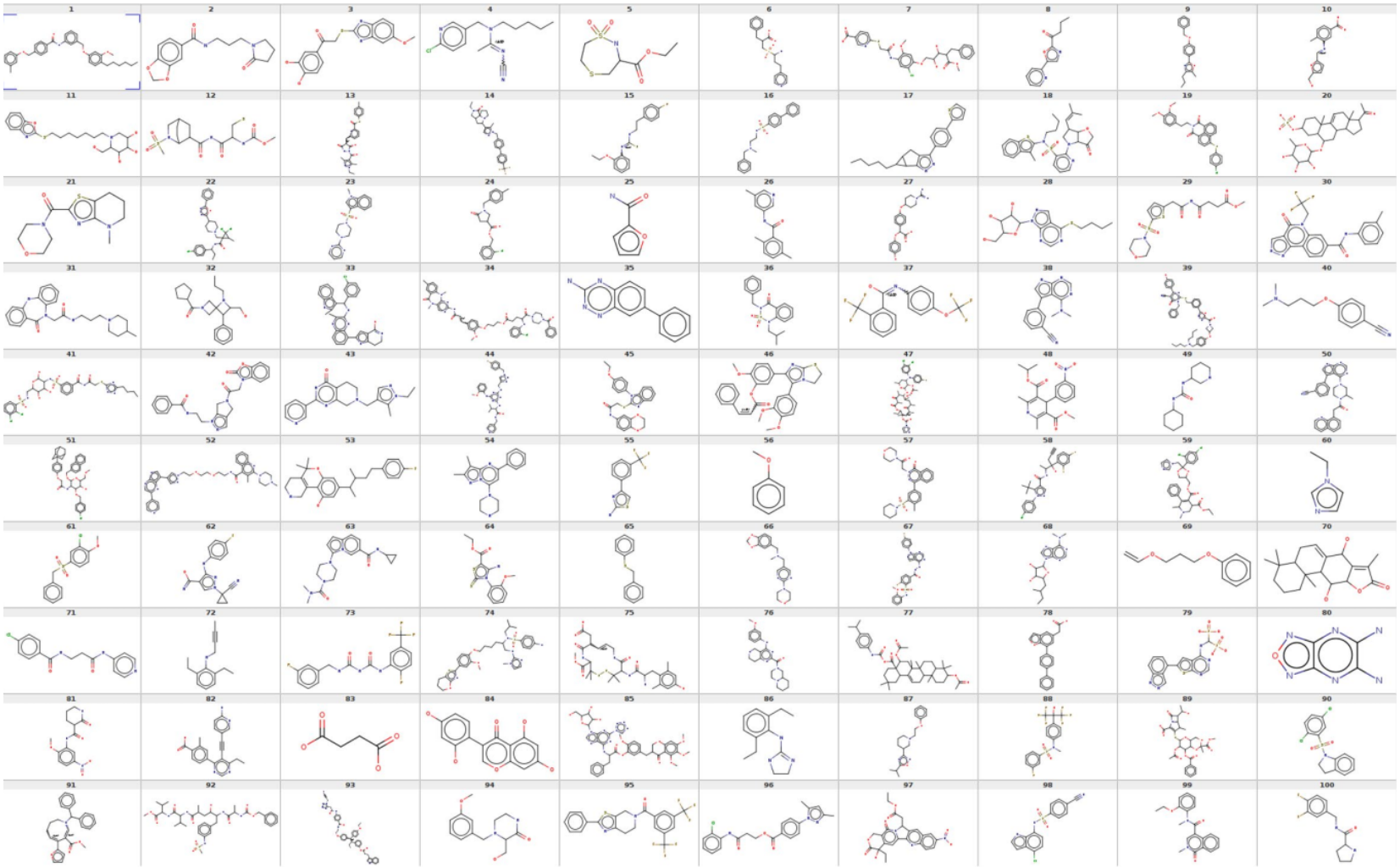


Figure 6: 100 molecules generated by FASMIFRA (ChEMBL-24 training set).