RDKit's New Fingerprint Generators

Google Summer of Code 2018

Boran Adas

19.09.2018

Google Summer of Code

RDKit

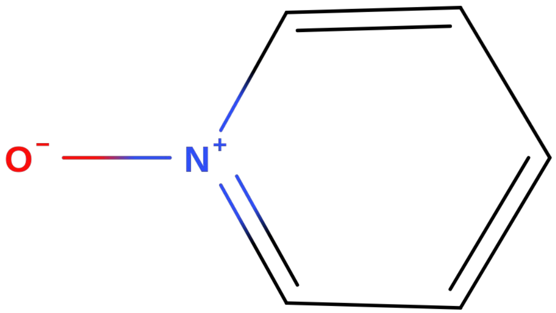
Open-Source Cheminformatics and Machine Learning

GSoC & project

- What?
- Why?
- Results
- Challenges
- Next steps

Molecular fingerprints

- Representation of structure
- Encoding of a molecule into a bit vector



→

0000 0010 0111 0000 ...

6, 9, 10, 11, ...

6: 1, 9: 2, 10: 5, 11: 1

- Enables similarity search, machine learning processes, activity prediction
- Used in drug design

RDKit fingerprints

	Morgan Fp	RDKit Fp	Atom Pairs Fp	Topological Torsion Fp	Layered Fp	Pattern Fp
Counts output	✓	✓	✓	✓	×	×
Count simulation	×	×	✓	✓	×	×
Target density	×	✓	×	×	×	×

```
ExplicitBitVect *LayeredFingerprintMol(
    const ROMol &mol,
    unsigned int layerFlags = 0xFFFFFFFF,
    unsigned int minPath = 1,
    unsigned int maxPath = 7,
    unsigned int fpSize = 2048,
    std::vector<unsigned int> *atomCounts = 0,
    ExplicitBitVect *setOnlyBits = 0,
    bool branchedPaths = true,
    const std::vector<boost::uint32_t> *fromAtoms = 0);
```

```
SparseIntVect<boost::uint32_t> *getFingerprint(
    const ROMol &mol,
    unsigned int radius,
    std::vector<boost::uint32_t> *invariants = 0,
    const std::vector<boost::uint32_t> *fromAtoms = 0,
    bool useChirality = false,
    bool useBondTypes = true,
    bool useCounts = true,
    bool onlyNonzeroInvariants = false,
    BitInfoMap *atomsSettingBits = 0);
```

What can be improved?

- Different fingerprints, different functionality
- Different inputs and output types
- Repeated and independently implemented logic

- Unified structure
- Reducing code duplication
- Template for development
- Flexibility by mixing and matching components

3 Months

Concept

Implement for atom pairs fp

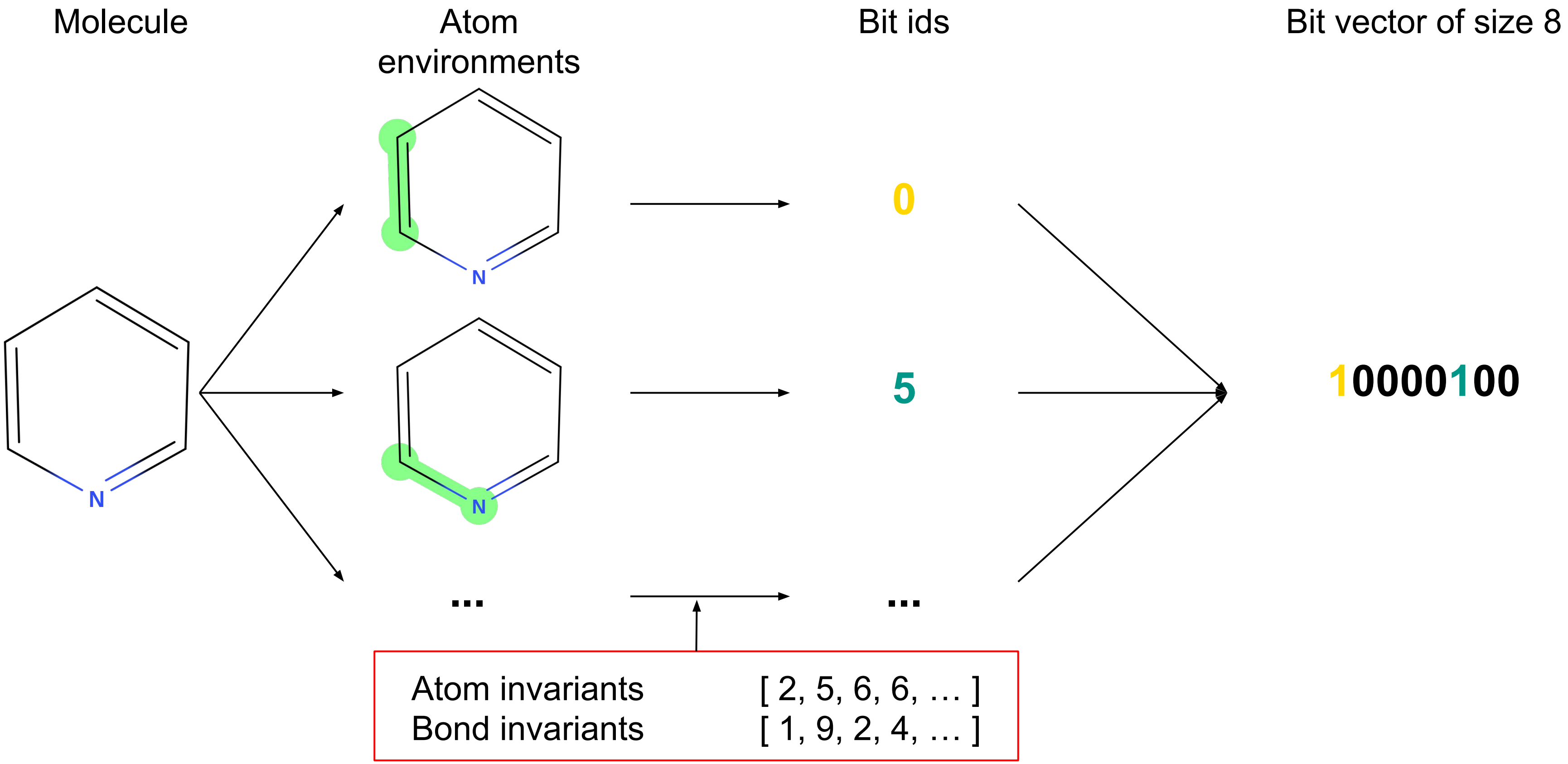
Morgan fingerprint

RDKit fingerprint

Topological torsion fingerprint

Convenience functions & wrapping up

Concept



Molecule

Atom environments

Bit ids

Bit vector of size 8

0

5

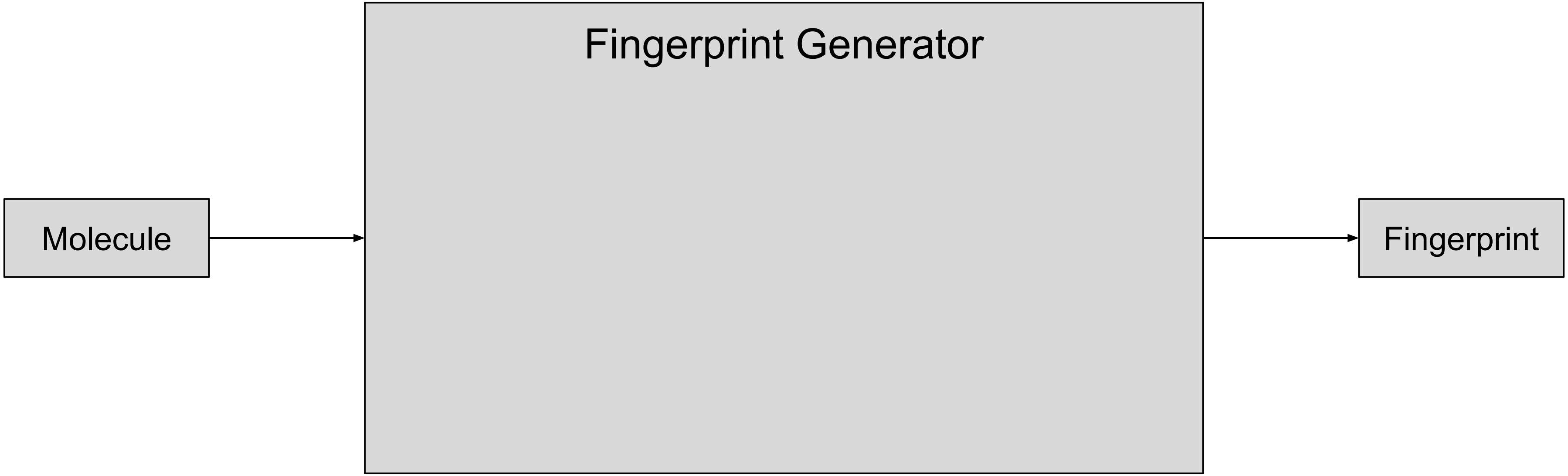
...

...

10000100

Atom invariants [2, 5, 6, 6, …]
Bond invariants [1, 9, 2, 4, …]

Fingerprint generator



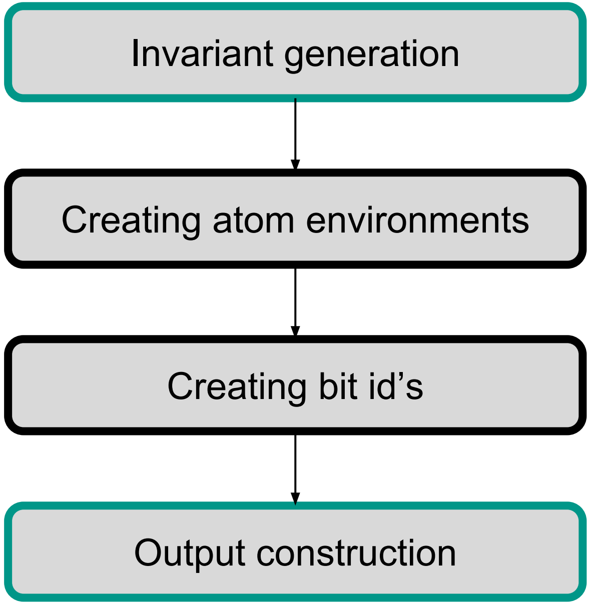
Molecule

Fingerprint Generator

Fingerprint

Fingerprint generation steps

- Atom and bond invariant generation
  - Can be same or different for different types
  - Possible to customise using invariant generators
- Atom environments from molecule
  - Varies for different types
- Bit id’s from atom environments
  - Varies for different types
  - Additional information output is formed
- Output construction from bit id’s
  - Common for all types



Common for all types
Different for different types

Using fingerprint generators

- Fingerprint generator initialisation
  - Different arguments for different types
  - Only configuration parameters related to fingerprint type
- Fingerprint calculation
  - Same arguments for all types
  - Only molecule dependent parameters
  - No fingerprint type related parameters

Initialisation

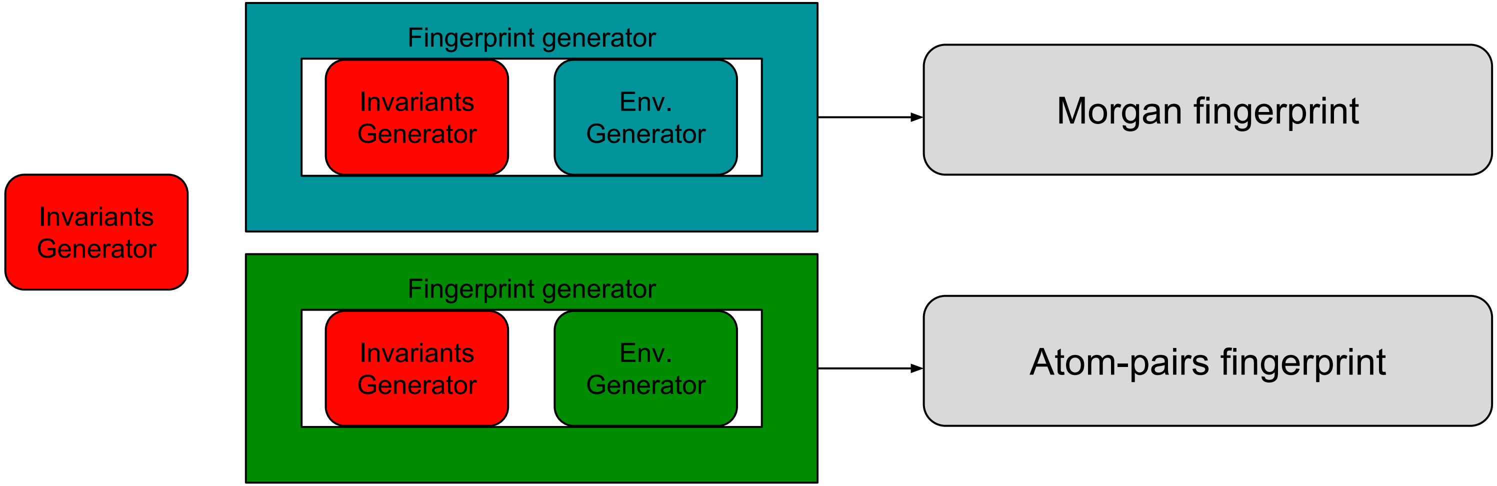
Type specific parameter set

Fingerprint calculation

Identical parameter set

Invariant generators

- Atom and bond invariants from given molecule
- Any existing invariant generator can be used
- Flexibility
- User defined invariant generators



Outcome

- Fingerprint generators
  - Tested with existing cases in RDKit
- 4 fingerprint types implemented
  - Morgan fingerprint
  - Atom-pairs fingerprint
  - RDKit fingerprint
  - Topological torsion fingerprint
- Customisation with invariant generators
- Consistency of supported features

Examples

Challenges

- Coming up with the right structure and plan
- What output types to support
- Backwards compatibility

What’s next?

- Missing planned features
- Not implemented fingerprint types
- Possible improvements
  - Computation in parallel
  - Naming standardisation
  - Invariant generators written in Python
- More ideas from the community

- PR #2005
- https://github.com/rdkit/rdkit/pull/2005

Thank you

Nadine Schneider, Andrea Volkamer & Greg Landrum

Andrew Dalke, Peter Gedeck & the RDKit community