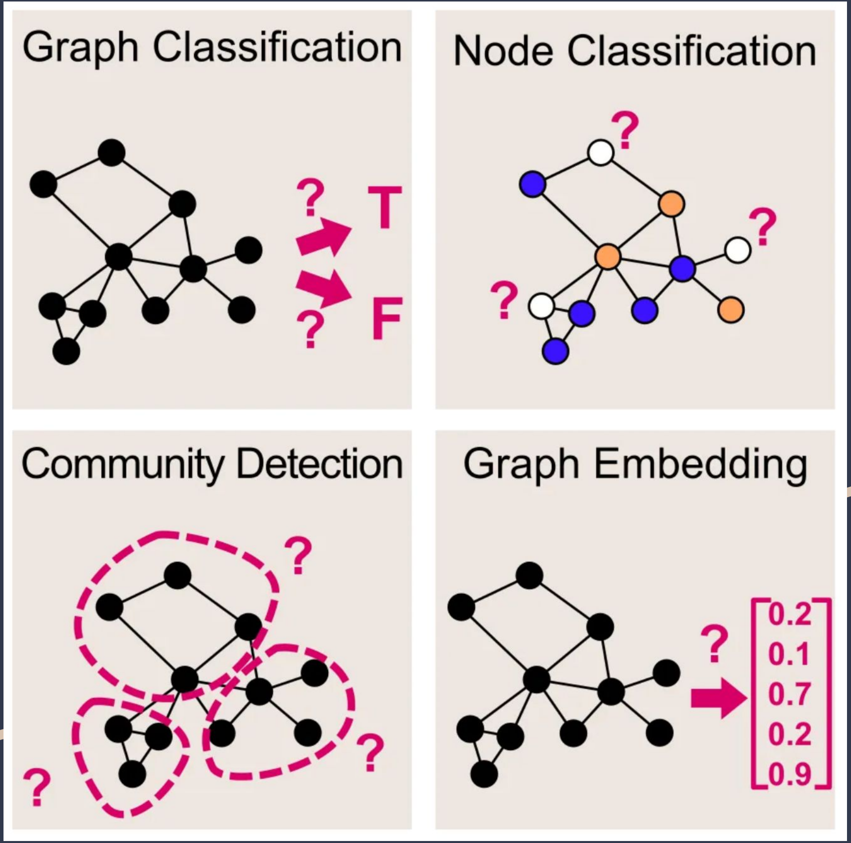
Molecular Fingerprints in Chemoinformatics and Graph Preprocessing

Piotr Ludynia
Michał Szafarczyk

Graphs in ML



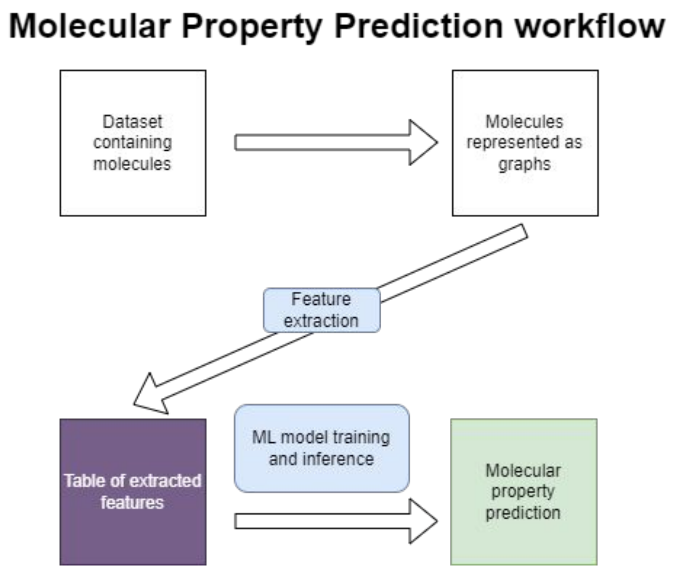
Graphs are expressive data structures. They are commonly used for modeling real-life subjects. For example:

- Representing chemical molecules
- Modelling social networks
- Holding information about traffic and roads

It is natural for us to want to use ML to process solve different tasks related to graphs. For example:

- classification of whole graphs
- classification of edges
- classification of vertices,
- detecting communities.

Use of ML in Chemoinformatics


Molecular Property Prediction workflow
Dataset containing molecules
Molecules represented as graphs
Feature extraction
Table of extracted features
ML model training and inference
Molecular property prediction

Chemical molecules can be represented as graphs. ML is a good tool to perform such tasks through:

- Molecular property prediction
- Molecular similarity search
- Virtual screening

BUT!

There's one issue that makes those tasks difficult - graphs are permutation invariant.

The nodes and edges of one graph can be listed out in more than one way.

Usually, when we perform machine learning, we work with ordered data:

- Arrays - as vectors in vector spaces
- Images that have height, width, and depth
- Sequences - for example text

Graphs are non-euclidean data structures, and we can't represent them like that. At least we can't do that directly…

Molecular Fingerprints

Molecular fingerprints are feature extraction methods, that turn molecular graphs into feature vectors.

Such vectors can then be used as input data in machine learning.

Descriptor Fingerprints

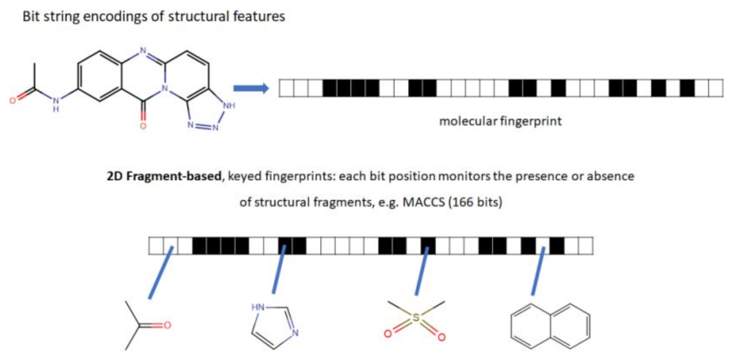
We can extract certain properties of molecules and save them as vectors. They may include the number of vertices or edges, size of the largest cycle in the molecule graph, or more complex information.

Typically, such fingerprints search for features designed by experts in the field of chemistry.

Example fingerprints:

- PubChem
- E-state
- Klekota-Roth
- CATS
- MACCS Keys

MACCS Keys fingerprint


Bit string encodings of structural features
molecular fingerprint
2D Fragment-based, keyed fingerprints: each bit position monitors the presence or absence of structural fragments, e.g. MACCS (166 bits)

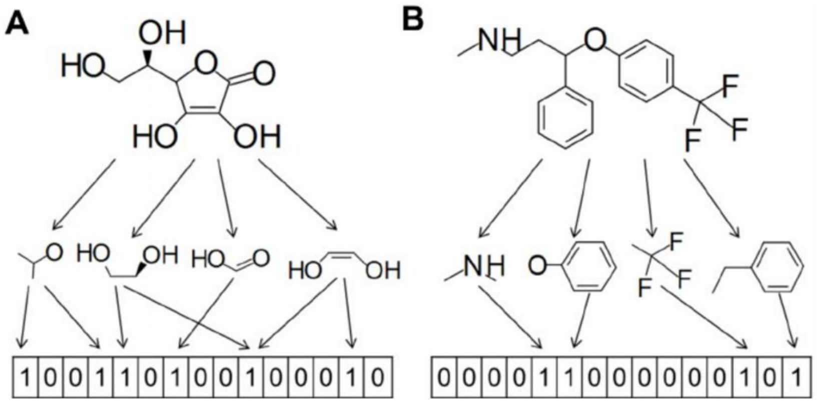
166 bits - descriptors that answer questions about the molecule.

For example, if the molecule has:

- fewer than 3 oxygen atoms
- -S-S- bond
- A ring of size 4

Those features have been designed by chemists.

Hashed Fingerprints



- Find a graph substructure
- Convert it to the text format (SMILES)
- Process it with a hashing function
- Folding the hash to reduce its size

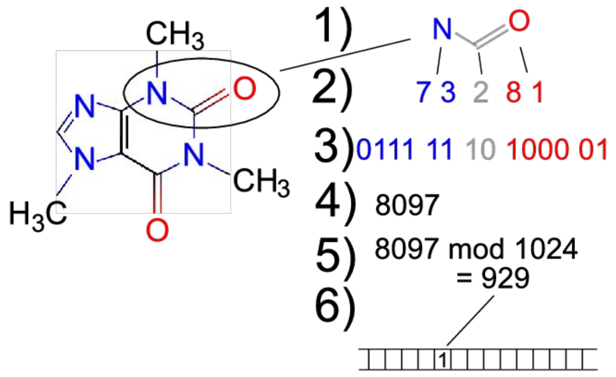
Examples of Hashed Fingerprints

- ECFP - Extended Connectivity fingerprint
- RDKit fingerprint
- Topological Torsion fingerprint
- Avalon fingerprint
- Many more…

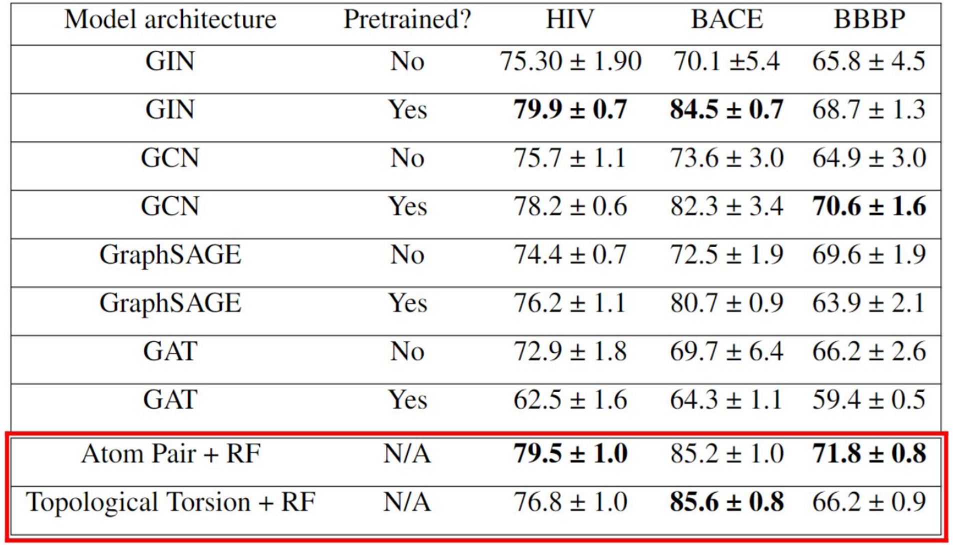
Atom Pair Fingerprint

Atom Pair fingerprint for hashing uses triplets - (atom 1, atom 2, length of the shortest path between them)

Such triplets are computed as strings and hashed for all pairs in the molecule



Comparison with GNNs



Model architecture	Pretrained?	HIV	BACE	BBBP
GIN	No	75.30 ± 1.90	70.1 ±5.4	65.8 ± 4.5
GIN	Yes	**79.9 ± 0.7**	**84.5 ± 0.7**	68.7 ± 1.3
GCN	No	75.7 ± 1.1	73.6 ± 3.0	64.9 ± 3.0
GCN	Yes	78.2 ± 0.6	82.3 ± 3.4	**70.6 ± 1.6**
GraphSAGE	No	74.4 ± 0.7	72.5 ± 1.9	69.6 ± 1.9
GraphSAGE	Yes	76.2 ± 1.1	80.7 ± 0.9	63.9 ± 2.1
GAT	No	72.9 ± 1.8	69.7 ± 6.4	66.2 ± 2.6
GAT	Yes	62.5 ± 1.6	64.3 ± 1.1	59.4 ± 0.5
Atom Pair + RF	N/A	**79.5 ± 1.0**	85.2 ± 1.0	**71.8 ± 0.8**
Topological Torsion + RF	N/A	76.8 ± 1.0	**85.6 ± 0.8**	66.2 ± 0.9

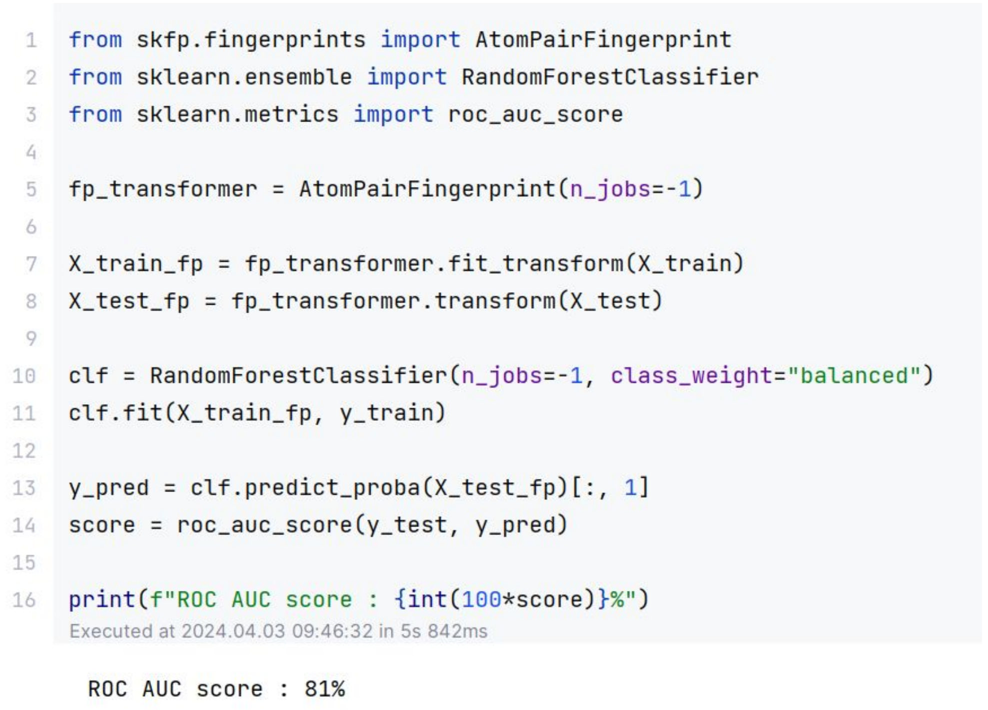
Our project: scikit-fingerprints

We created our own, library for computation of molecular fingerprints.

It provides:

- scikit-learn compatibility
- Comprehensive interface
- Parallelism - time performance improvement
- Wide variety of fingerprints

Code Example



```python
from skfp.fingerprints import AtomPairFingerprint
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score

fp_transformer = AtomPairFingerprint(n_jobs=-1)

X_train_fp = fp_transformer.fit_transform(X_train)
X_test_fp = fp_transformer.transform(X_test)

clf = RandomForestClassifier(n_jobs=-1, class_weight="balanced")
clf.fit(X_train_fp, y_train)

y_pred = clf.predict_proba(X_test_fp)[:, 1]
score = roc_auc_score(y_test, y_pred)

print(f"ROC AUC score : {int(100*score)}%")
```

Executed at 2024.04.03 09:46:32 in 5s 842ms

```
ROC AUC score : 81%
```

Thank you!

Sources and materials

- https://doi.org/10.48550/arXiv.2403.19718
- https://chemicbook.com/2021/03/25/a-beginners-guide-for-understanding-extended-connectivity-fingerprints.html
- https://drzinph.com/maccs-fingerprints-in-python-part-2/
- https://www.blopig.com/blog/2022/06/exploring-topological-fingerprints-in-rdkit/
- https://pubs.acs.org/doi/abs/10.1021/ci00054a008
- https://www.researchgate.net/PROFILE/DAVID-HOKSZA/PUBLICATION/321641342
- https://jcheminf.biomedcentral.com/articles/10.1186/s13321-020-00445-4
- https://www.ncbi.nlm.nih.gov/PMC/ARTICLES/PMC6075869/
- https://www.researchgate.net/publication/321641342_Utilizing_knowledge_base_of_amino_acids_structural_neighborhoods_to_predict_protein-protein_interaction_sites
- https://jcheminf.biomedcentral.com/articles/10.1186/s13321-022-00650-3/figures/1