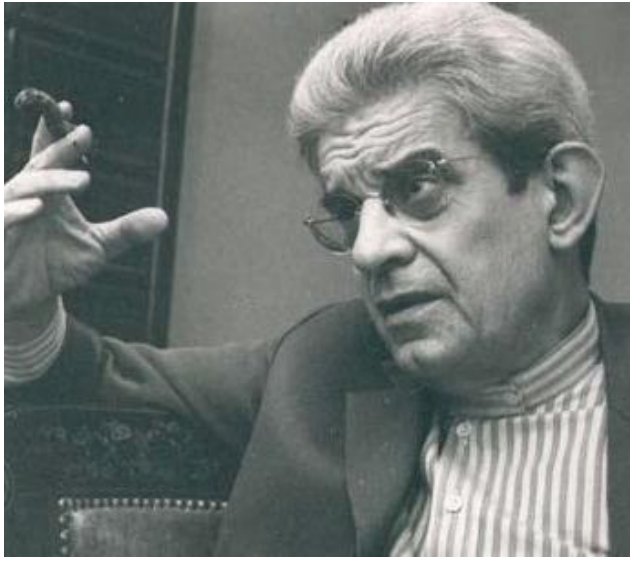
LACAN filter
Leveraging **a**djacent **c**o-occurrence of **a**tomic **n**eighborhoods for molecular filtering



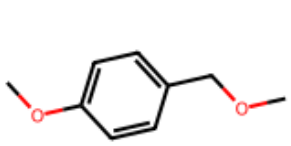
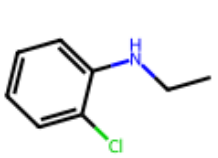
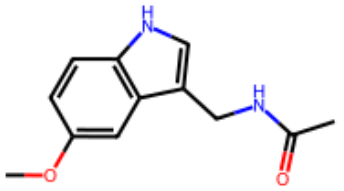
“All sorts of things in this world behave like mirrors. “

-Jacques Lacan

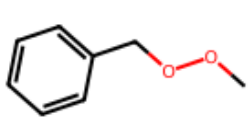
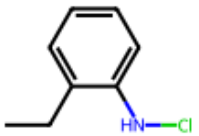
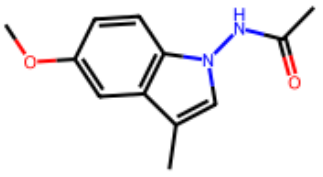
UNIVERSITY OF CHEMISTRY AND TECHNOLOGY PRAGUE

RDKit UGM 2024
Wim Dehaen

Adjacent co-occurrence

Good molecules

Bad molecules

Some fragments that co-occur a lot cannot co-occur at the bond interface !

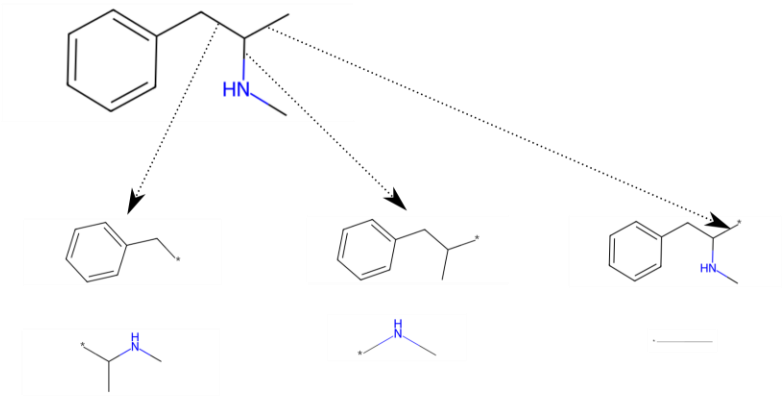
Our solution: LACAN

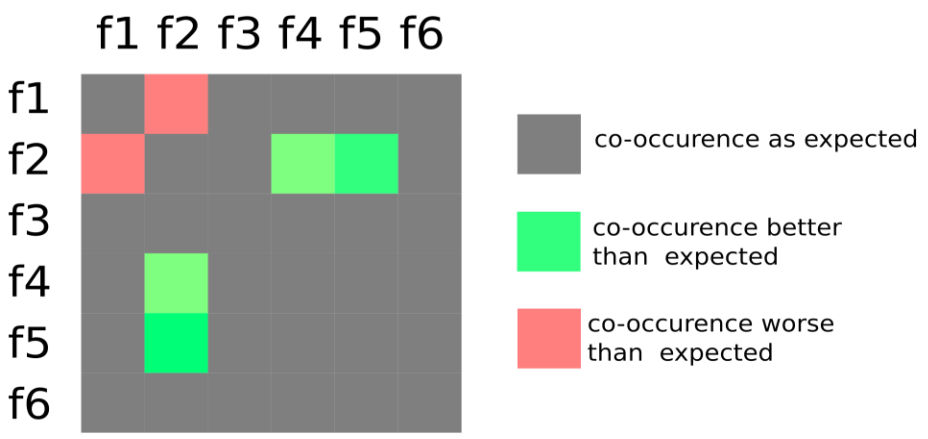
- Fragment every bond in the molecule
- Calculate ECFP2 for the atoms on the bond interface
- Use the occurrences of ECFP2 features in a set to calculate expected co-occurrences
- Lower than expected actual co-occurrences in the set: immediate reject
- Cool advantage: tells which part of the molecule is rejected

Fragment every bond

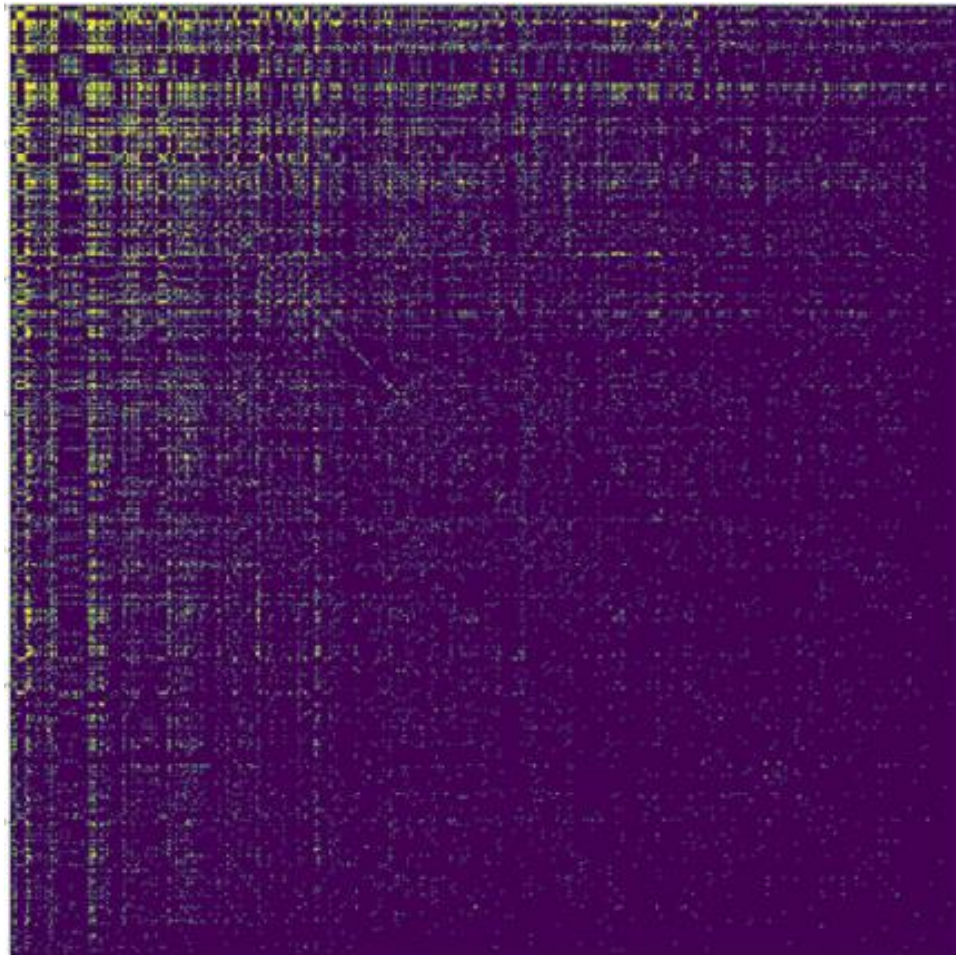
look at the 2 ECFP2 frags at bond interface

repeat for large set and use counts to populate co-occurence matrix:



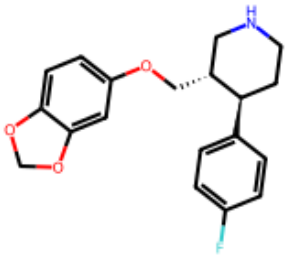


This is how the matrix actually looks

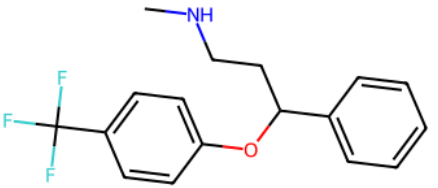


Leveraging LACAN: Median Molecules

- Cut molecules n times and recombine them (with size restriction)
- Apply LACAN filter so only sensible recombinations are made
- Filter + size filter is harsh: 95% get rejected
- Useful for crossover stage of GA

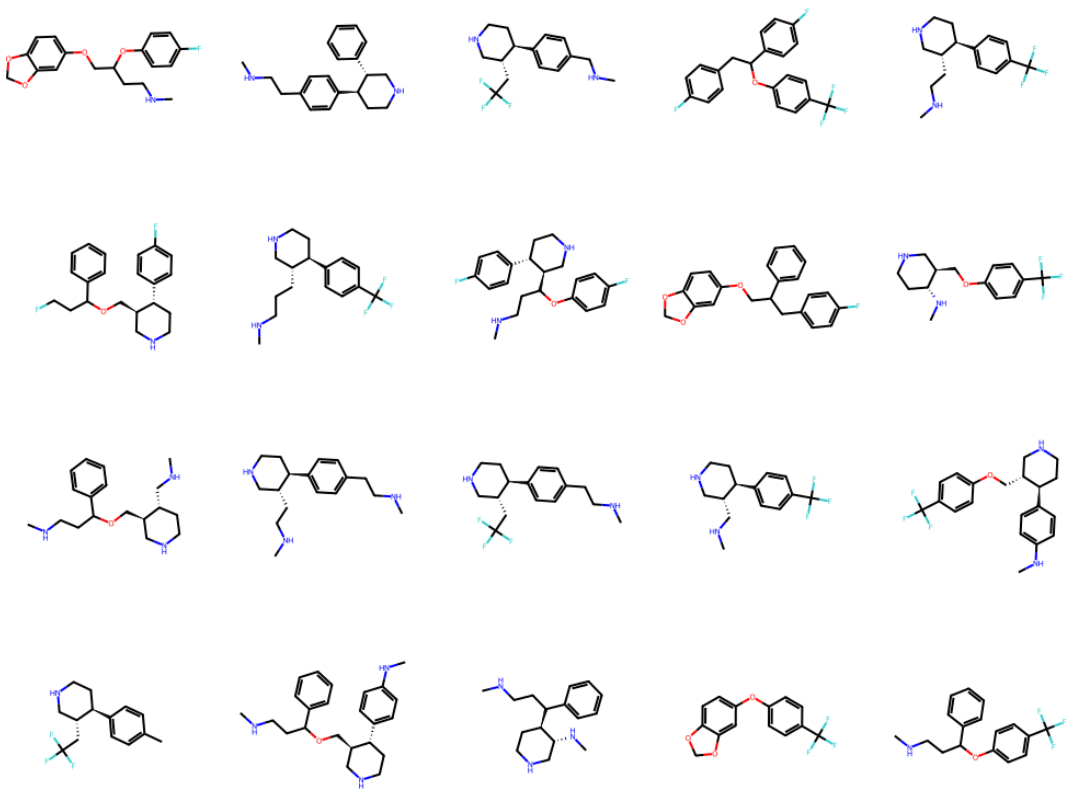


mother



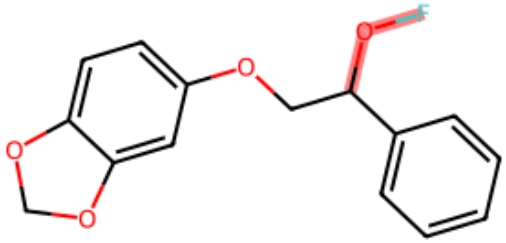
father

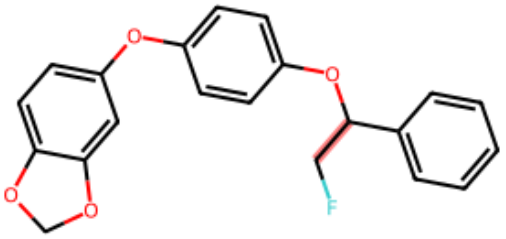
children:

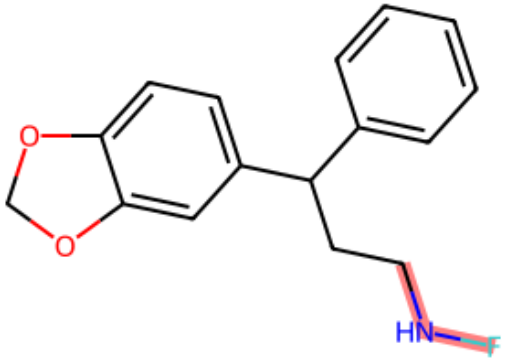


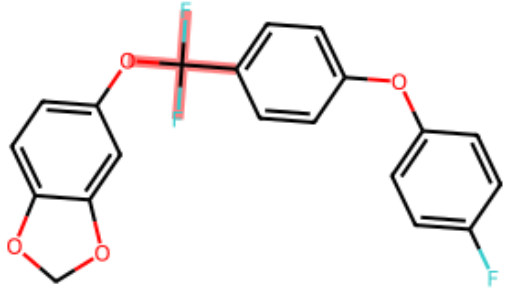
What do the rejects look like

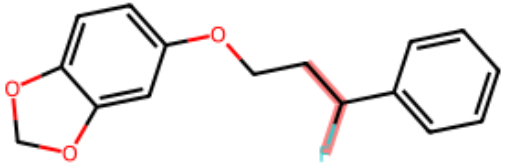
LACAN highlights the reason for rejection

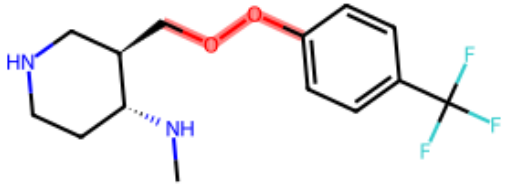








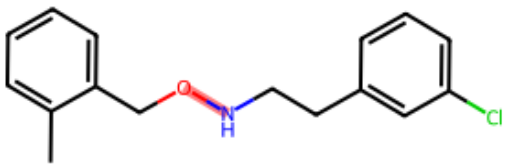




Easy to use

```
from lacan import lacan,breed
from rdkit import Chem
from rdkit.Chem import Draw
p = lacan.load_profile("chembl")
```

```
m = Chem.MolFromSmiles("Cc2ccccc2CONCCc2cc(Cl)ccc2")
score,info = lacan.score_mol(m,p)
Draw.MolToImage(m,highlightBonds=info["bad_bonds"])
```



Pip installable
Only dependency is rdkit (and its downstream dependencies)

```
m1 = Chem.MolFromSmiles("CNCCC(C1=CC=CC=C1)OC2=CC=C(C=C2)C(F)(F)F")
m2 = Chem.MolFromSmiles("CNCCC(C1=CC=CC=C1)OC2=CC=C(C=C2)C(F)(F)F")
mols = breed.breed(m1,m2,p)
print([Chem.MolToSmiles(m) for m in mols])
```

```
0.0125
['CCC(Oc1ccc(CNC)cc1)C(CCNC)c1ccccc1', 'CNCCCC(Oc1ccc(CCNC)cc1)c1ccccc1', 'CNCCC(Oc1ccc(C(F)(F)F)cc1)C(F)
(F)F', 'CC(F)(c1ccc(F)cc1)c1ccc(C(F)(F)F)cc1', 'CNCCC(CCc1ccccc1)Oc1ccc(F)cc1', 'CNCC(Oc1ccc(C(F)(F)F)cc1)c
1ccc(C(F)(F)F)cc1', 'CNCCc1ccc(OC(CCNC)CNC)cc1', 'CC(Oc1ccc(C(F)(F)C(F)(F)F)cc1)c1ccccc1', 'CNCCC(CCNC)Oc1c
cc(C(F)(F)F)cc1', 'FC(F)(F)C(F)(F)c1ccc(Oc2ccccc2)cc1']
```

Final slide

- v0.0.1alpha released on github.com/dehaenw/lacan
- Works surprisingly well for being so simple
- Thanks to Ivan Čmelo for brainstorming and inspiration