UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Pattern Recognition

Exercise Session 9

Last Group Project(s)

Linda Studer (linda.studer@unifr.ch)

Paul Märgner (paul.maergner@unifr.ch)

Exam

- Don’t forget to sign up for the exam until 14. May 2021 through the Academia platform
- Written, in-person exam: 21.06.2021 at 10:00 - 11:00h (Joseph Deiss lecture hall)
- If you are unable to attend the in-person exam because of medical reasons, travel restrictions, etc. send an email to Prof. Fischer ASAP!

Task 3 – Keyword Spotting

Deadline: **Today (end of day)!**

If you made a new GitHub repor, please send me the link to it / add me (LindaSt).

Last Group Projects

Deadline: **Sunday, May 30th, 2021, end of the day**

Task 4: Signature Verification

Can be solved with DTW

Task 5: Molecules

Use approximate Graph Edit Distance

You need to solve only one,
but you can solve both!

Final Report

Please, create the final group report until **Sunday, May 30th, 2021**

It should be between 2-4 pages (A4)
The report should describe the “lessons learned”

You should describe:

- How you organized your group
- What worked, what did not work
- For each task:
  - What is special about your solution
  - What was your approach
  - What worked, what did not work
- General thoughts about the group exercise

Task 4: Signature Verification

30 writers (listed in `users.txt`)

Enrollment: 5 genuine signatures each

Verification: 45 signatures each (20 genuine, 25 forgeries)
Ground truth in `gt.txt`

Task

Compute dissimilarity for each verification signature with the 5 genuine ones

Task 4: Signature Verification

Input

`t x y pressure penup azimuth inclination`

`Penup` 1 if change between pen-up and pen-down

`Azimuth` / `inclination` angles of the pen

Recommendation: **DTW**

Features: `x, y, vx, vy, pressure`

`vx, vy` velocity in x and y with respect to Δt

Normalize for each signature individually

Sakoe-Chiba band can be helpful

Evaluation: mean average-precision

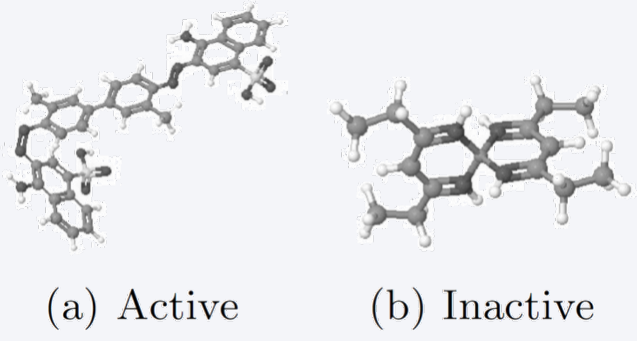
Task 4: Signature Verification

Further info about the database (MCYT):

See ortega03mcyt.pdf on ILIAS → Section 4 (there are also images of signatures)

Task 5: Molecules

AIDS Antiviral Screen Database of Active Compounds



(a) Active

(b) Inactive

250 training, 250 validation molecules

- Two classes active ‘`a`’ and inactive ‘`i`’
- Annotated in `train.txt` and `valid.txt`

Task

- Classify the molecules of the validation set using kNN
- Distance: approximate Graph Edit Distance (GED)

Input: Graph xml (gxl files)

XMLs with a lot of information

→ Only use the chemical `symbol` node label and the unlabeled, undirected edges:

```
<node id="_1"><attr name="symbol"><string>C</string></attr>
…
<edge from="_1" to="_2">
…
```

Further info about the database (AIDS):

See riesen08graphdb.pdf → section 2.8 (there are also images of molecules)

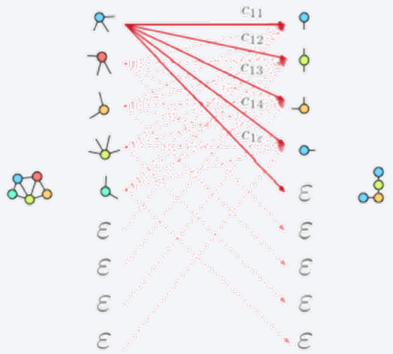
Hint: there are python libraries to parse XML files

Task 5: Molecules

Compute approximate GED between pairs of molecules with **bipartite graph matching**
(lecture 10, slide 21)

Build cost matrix (*Dirac*)

$$\mathbf{C} = \left[\begin{array}{cccc|cccc} c_{11} & c_{12} & \cdots & c_{1m} & c_{1\varepsilon} & \infty & \cdots & \infty \\ c_{21} & c_{22} & \cdots & c_{2m} & \infty & c_{2\varepsilon} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ c_{n1} & c_{n2} & \cdots & c_{nm} & \infty & \cdots & \infty & c_{n\varepsilon} \\ \hline c_{\varepsilon 1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & c_{\varepsilon 2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \cdots & \infty & c_{\varepsilon m} & 0 & \cdots & 0 & 0 \end{array}\right]$$



Hungarian Algorithm
To find optimal assignment

Derive edit path costs from the result
(distance for classification)

kNN for classification (optimize for k)

Task 5: Molecules

Recommendation

Use *Dirac* cost function for GED (optimize c_n and c_e)
(lecture 9, slide 36)

Node substitution: $2 * c_n$ if symbols ≠, 0 otherwise
Node deletion/insertion: c_n
Edge deletion/insertion: c_e

Use an existing framework for the Hungarian algorithm

Questions?