Weisfeiler Lehman, Tree Metrics and the French Railway System

2. Progress meeting

Fabrice Beaumont & Yuchi Zhang

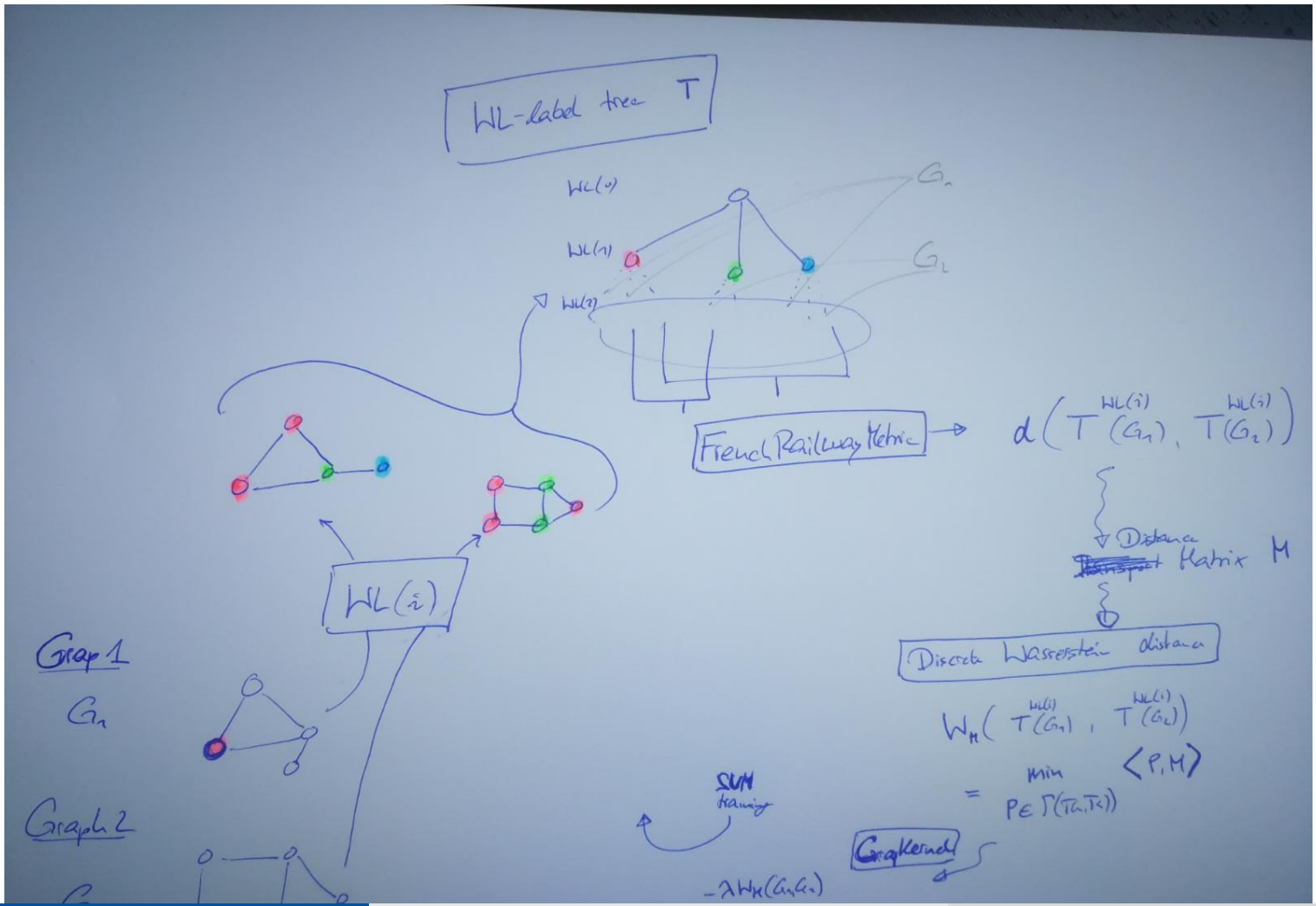
Lab Development and Application of Data Mining and Learning Systems:
Machine Learning and Data Mining
Supervisor: **Dr. Pascal Welke**

01.06.2021

Table of contents

Task explanation - Overview

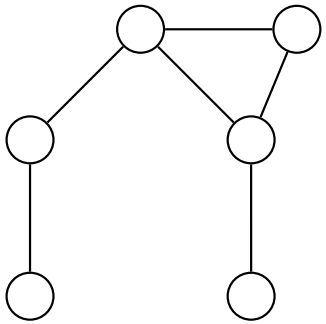
TODO: make beautiful



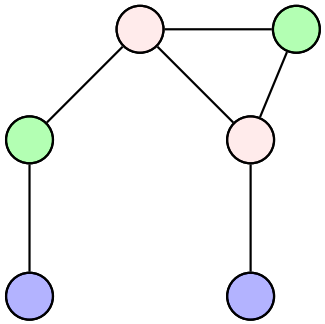
Task explanation - WL labeling

$$N = \text{sort}\big([\ell_i(u)|\ u \in \mathcal{N}(v)]\big)$$
$$\ell_{i+1}(v) := \mathcal{H}(\ell_i, N)$$

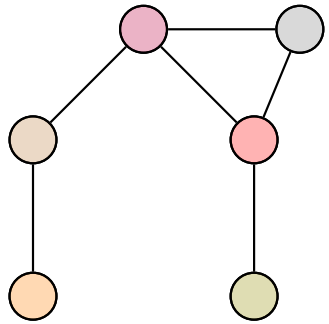
WL 1.



WL 2.



WL 3.



Task explanation - WL label tree

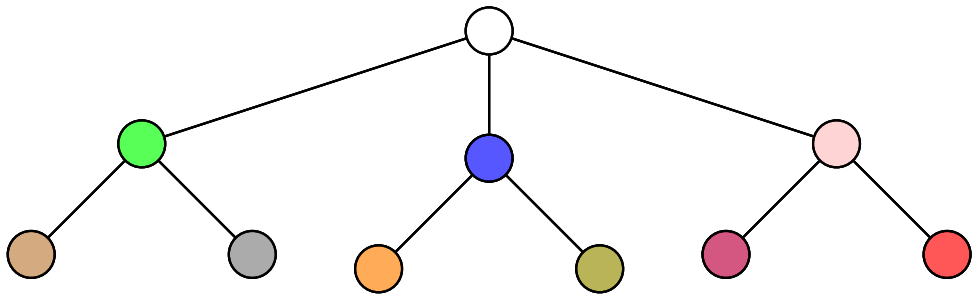
$$N = \text{sort}\big([\ell_i(u)|\ u \in \mathcal{N}(v)]\big)$$
$$\ell_{i+1}(v) := \mathcal{H}(\ell_i, N)$$

WL tree

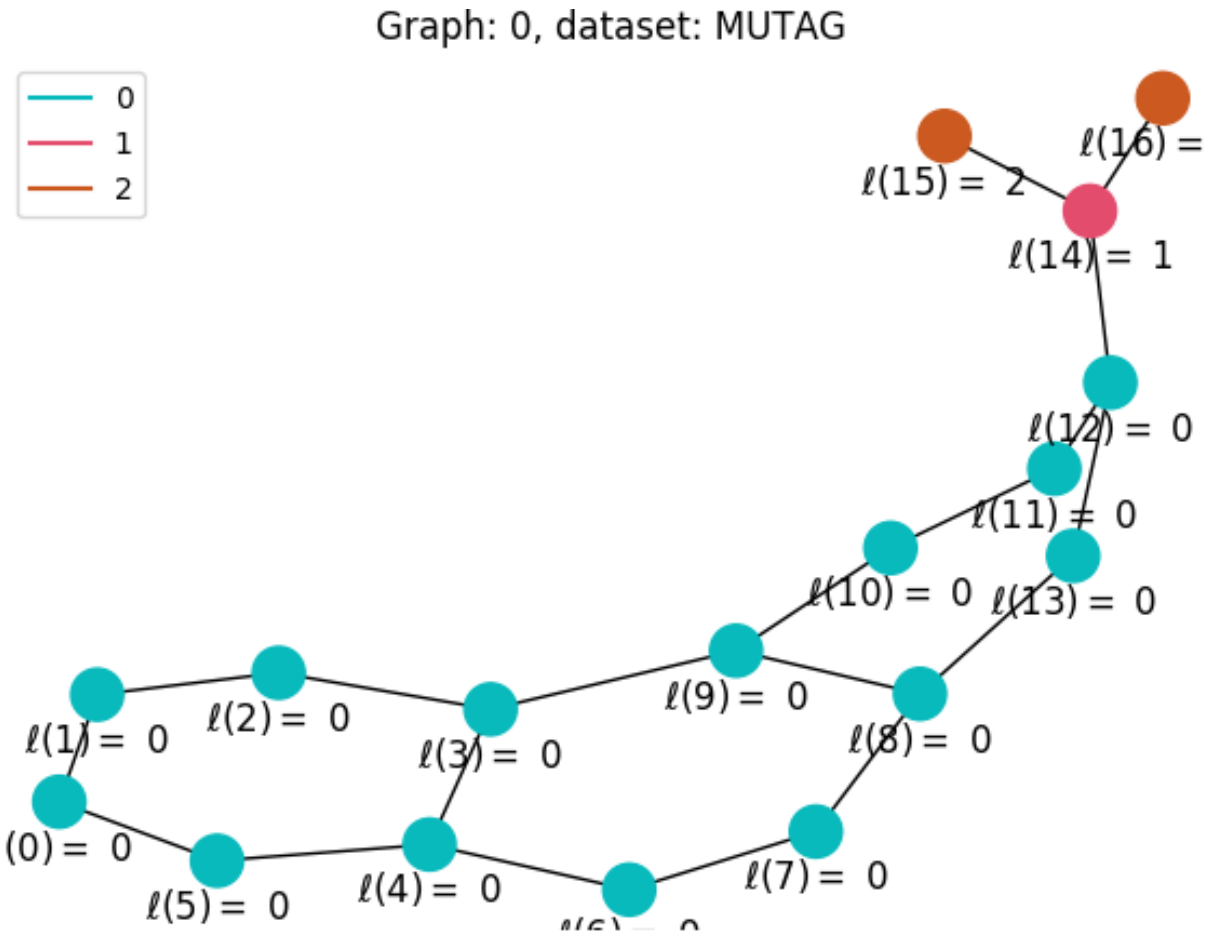
WL 1.

WL 2.

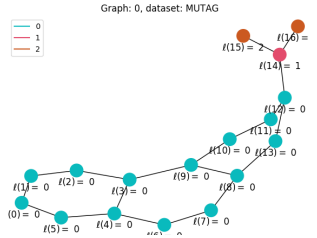
WL 3.



Task explanation - WL label tree



[7, 7, 7, 8, 8, 7, 7, 7, 8, 8, 7, 7, 9, 7, 10, 11, 11]



[7, 7, 7, 8, 8, 7, 7, 7, 8, 8, 7, 7, 9, 7, 10, 11, 11]

- This was to give an idea what we have to do and already did
- Lets look at our shedule, to see where were
- NEXT: How does this look as an algorithm?
  Replace color by integers (labels).

1. Progress meeting goals - OLD

Research:

1. Familiarize with related literature and understand the concepts
2. Familiarize with the databases
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

Evaluation:

- Traditional Weisfeiler Lehman kernels
- Wasserstein WL kernels
- ...

1. Progress meeting goals - OLD

Research:

1. Familiarize with related literature and understand the concepts
2. Familiarize with the databases
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

1. Progress meeting goals - OLD

Research:

1. Familiarize with related literature and understand the concepts
2. Familiarize with the databases
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

- Implementation amost sequential order
-

1. Progress meeting goals - OLD –> NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. Familiarize with the databases
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

1. Progress meeting goals - OLD –> NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

1. Progress meeting goals - OLD –> NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

- GraKel package
- NetworkX package -
  this also includes graph kernels

- NOT JET: Graph kernels BUT
  GraKel provides many graph kernels and examples show to use them when training an SVM.

1. Progress meeting goals - OLD –> NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**:
   - 1.1 *Computation of the WL labeling procedure*
   - 1.2 *Construct a tree structure while updating the WL labels*
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

1. Progress meeting goals - OLD –> NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**:
   - 1.1 ~~Computation of the WL labeling procedure~~
   - 1.2 Construct a tree structure while updating the WL labels
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

1. Progress meeting goals - OLD –> NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

1. Computation of **WL labeling trees**:
   - 1.1 ~~Computation of the WL labeling procedure~~
   - 1.2 Construct a tree structure while updating the WL labels
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

- Tree structure is almost complete
- Testing will being shortly

1. Progress meeting goals - NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

0. *Computation of unfolding trees [4]*
1. Computation of **WL labeling trees**:
   - 1.1 ~~Computation of the WL labeling procedure~~
   - 1.2 Construct a tree structure while updating the WL labels
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

1. Progress meeting goals - NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

0. *Computation of unfolding trees [4]*
1. Computation of **WL labeling trees**:
   - 1.1 ~~Computation of the WL labeling procedure~~
   - 1.2 Construct a tree structure while updating the WL labels
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

- Little distraction: Unfolding trees
- Side step

1. Progress meeting goals - NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

0. ~~Computation of unfolding trees ([4])~~
1. Computation of **WL labeling trees**:
   - 1.1 ~~Computation of the WL labeling procedure~~
   - 1.2 Construct a tree structure while updating the WL labels
2. Computation of edge weights (**French Railway Metric**)
3. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
4. Train the model (**SVM**)

1. Progress meeting goals - NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

0. ~~Computation of unfolding trees ([4])~~
1. Computation of **WL labeling trees**:
   - 1.1 ~~Computation of the WL labeling procedure~~
   - 1.2 Construct a tree structure while updating the WL labels
   - 1.3 *Computation of edge weights (**French Railway Metric**)*
2. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
3. Train the model (**SVM**)

1. Progress meeting goals - NOW

Research:

1. ~~Familiarize with related literature and understand the concepts~~
2. ~~Familiarize with the databases~~
3. Familiarize with different graph kernels

Implementation:

0. ~~Computation of unfolding trees ([4])~~
1. Computation of **WL labeling trees**:
   - 1.1 ~~Computation of the WL labeling procedure~~
   - 1.2 Construct a tree structure while updating the WL labels
   - 1.3 *Computation of edge weights* ***(French Railway Metric)***
2. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
3. Train the model (**SVM**)

- Example: Getting to know the task better
- Re-Prioritizing/weighting intermediate goals

- NEXT SLIDE: Status report

Strategy and workflow

Research:

1. **LaTeX script**
2. Trello
3. **Telegram**, **Zoom**

Implementation:

1. **Google Colab**
2. **PyCharm**
   `https://github.com/D34dP0oL/MA-INF-4306_Lab`

Research:
1. **LaTeX script**
2. Trello
3. **Telegram**, **Zoom**

Implementation:
1. **Google Colab**
2. **PyCharm**
   `https://github.com/D34dP0oL/MA-INF-4306_Lab`

- Why did Trello not work for us?
- Project to small?
- A lot of research, without intermediate goals?

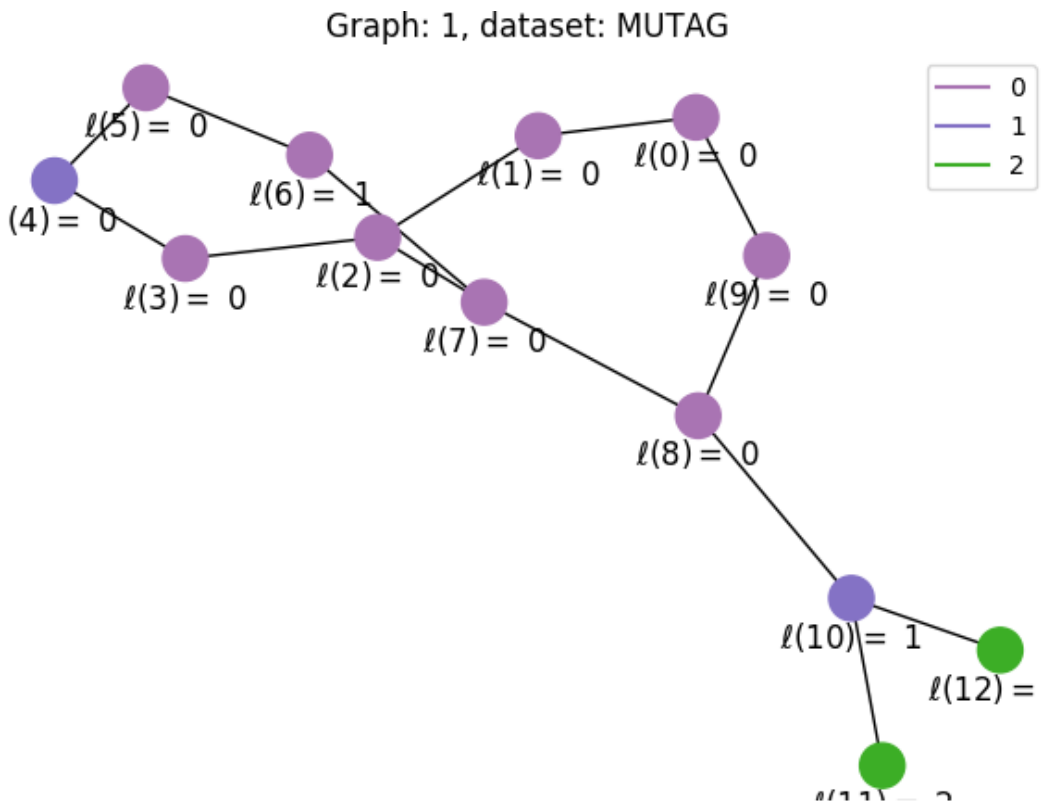
- Colab
- For now: Not to share ideas
- But fast experiments, method sketching
- Almost better than sketching pseudocode before implementing

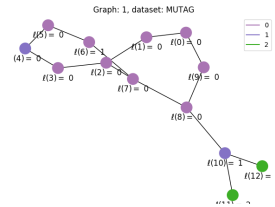
- PyCharm: Todos, OptimizeFlags, UnitTests, CleanCode
- VCS - rather as a backup for now

Live demo

- Database Reading
- Grab a graph - display it with its labels
- Display the WL-labels
- Display a unfolding tree

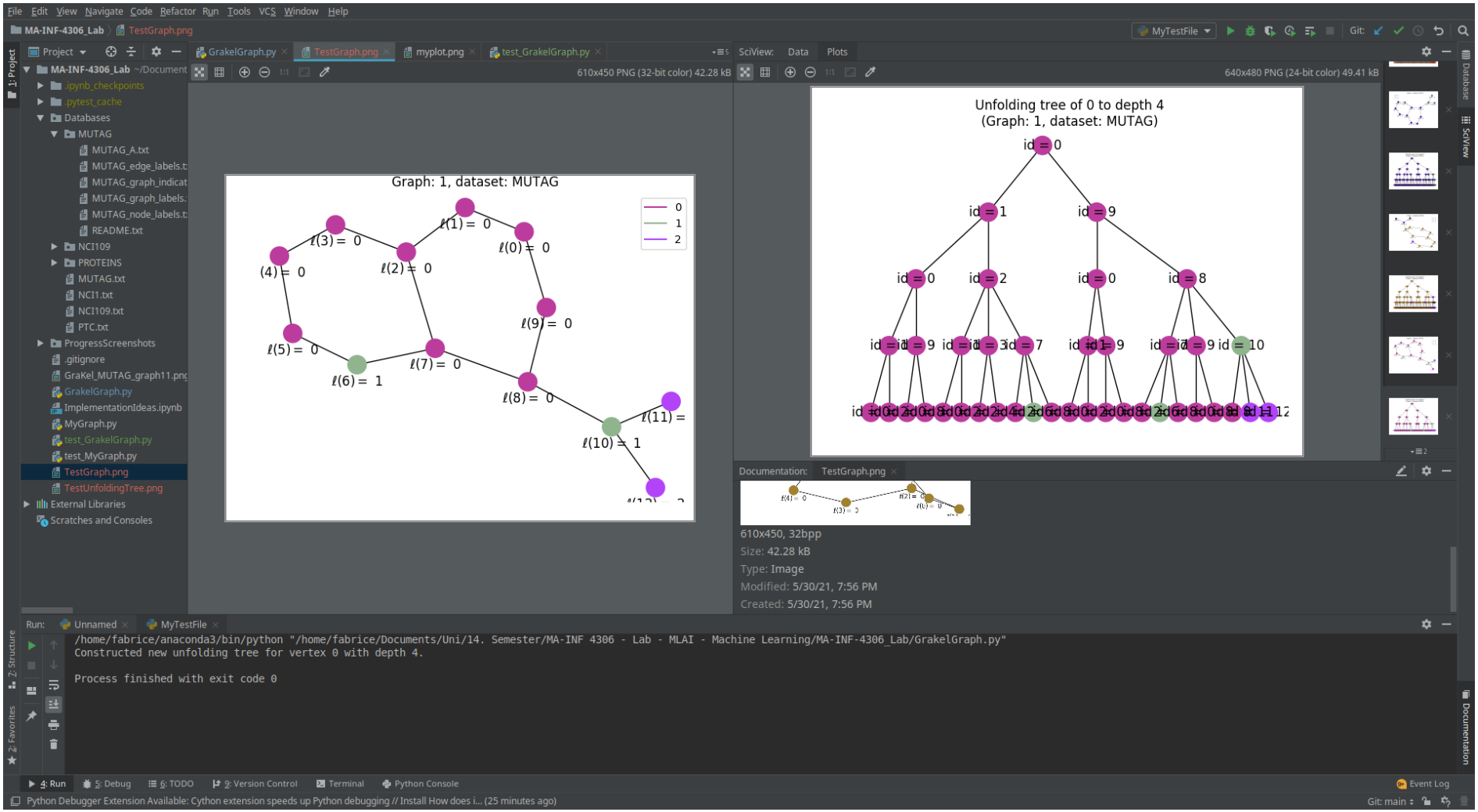
Live demo - Not so live backup





- Database Reading
- Grab a graph - display it with its labels

Live demo - Not so live backup - Unfolding tree



Everyday coding wisdom

1. **Never trust foreign code** to do what you think, one may expect . . .
2. Before going big, **try small**
3. **Eyes wide open**:
   Just because the concept is cool, it may not help your course . . .
   But with clean code, learning sth. from a related field, may benefit you later on.

4. Did you know there is a for-else-statement?

Everyday coding wisdom

1. **Never trust foreign code** to do what you think, one may expect ...
2. Before going big, **try small**
3. **Eyes wide open**:
   Just because the concept is cool, it may not help your course ...
   But with clean code, learning sth. from a related field, may benefit you later on.
4. Did you know there is a for-else-statement?

1. NetworkX initialization:
   The off-label. Correct labels, correct indices. Internal shuffeling that has nothing to do with the labels. The vertices are only adressed by their index in the list!
2. Failures with the nested dictionary-hash.
   Ideas of maintaining a dictionary of lists. At first: last dictionary shall have the new label as value. That makes dynamic adding of different values extremely difficult.
   Simple solution: None: newLabel
   Obvious with a simple toy example. Debugging the whole thing was like financing the new Berlin airport
3. Implementation of the unfolding tree > Tree structure, package knowledge (GraKel) and displaying graphs.

Next goals

Research:

1. Cai, Fürer, and Immerman [2]
2. Babai and Kucera [1]
3. Familiarize with different graph kernels

Implementation:

0. Computation of **WL labeling trees**
1. Computation of a distance/**kernel** matrix (**Wasserstein distance**)
2. Train the model (**SVM**)

Task definition

Write a program, that is able to compute a **similarity measure between graphs**. It shall use the

- Wasserstein distance over a tree metric, which is defined as a
- French Railway Metric on the
- Weisfeiler Lehman (WL) labeling tree.

The program shall be applicable as a **graph kernel** (using a SVM to learn over graph databases).

Task explanation - WL labeling

Input: a labelled graphs $G = (V, E, \ell_0)$,
a list of alphabets Σ_i (for $i = 1, \ldots, h$) and
a perfect (injective) hash function $\mathcal{H} : \Sigma_i \times \Sigma_i^* \to \Sigma_{i+1}$.
Output: a re-labeled graph $G = (V, E, \ell_h)$.

1: **for** $i = 1, \ldots, h$ **do**
2: **for** $v \in V$ **do**
3: $N = \text{sort}\big([\ell_i(u) |\ u \in \mathcal{N}(v)]\big)$
4: $\ell_{i+1}(v) := \mathcal{H}(\ell_i, N) \ \in \Sigma_{i+1}$
5: **end for**
6: **end for**

WL, Tree Metrics & FRS

Task explanation - WL labeling

Task explanation - WL labeling

Input: a labelled graphs $G = (V, E, \ell_0)$,
a list of alphabets Σ_i (for $i = 1, \ldots, h$) and
a perfect (injective) hash function $\mathcal{H} : \Sigma_i \times \Sigma_i^* \to \Sigma_{i+1}$.
Output: a re-labeled graph $G = (V, E, \ell_h)$.

1: **for** $i = 1, \ldots, h$ **do**
2: **for** $v \in V$ **do**
3: $N = \text{sort}([\ell_i(u) | \ u \in \mathcal{N}(v)])$
4: $\ell_{i+1}(v) := \mathcal{H}(\ell_i, N) \ \in \Sigma_{i+1}$
5: **end for**
6: **end for**

- INPUT:
  - a number of iterations $h \in \mathbb{N}$
  - the set of all possible labels Σ_0 (finite and with a total order $>$).
- Sort the list of neighbor labels