Measuring Item Similarity in Introductory Programming

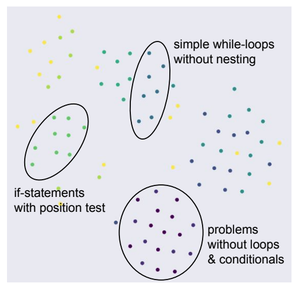
Radek Pelánek Tomáš Effenberger Matěj Vaněk Vojtěch Sassmann
Dominik Gmiterko
Masaryk University Brno, Czech Republic

Why study similarity?

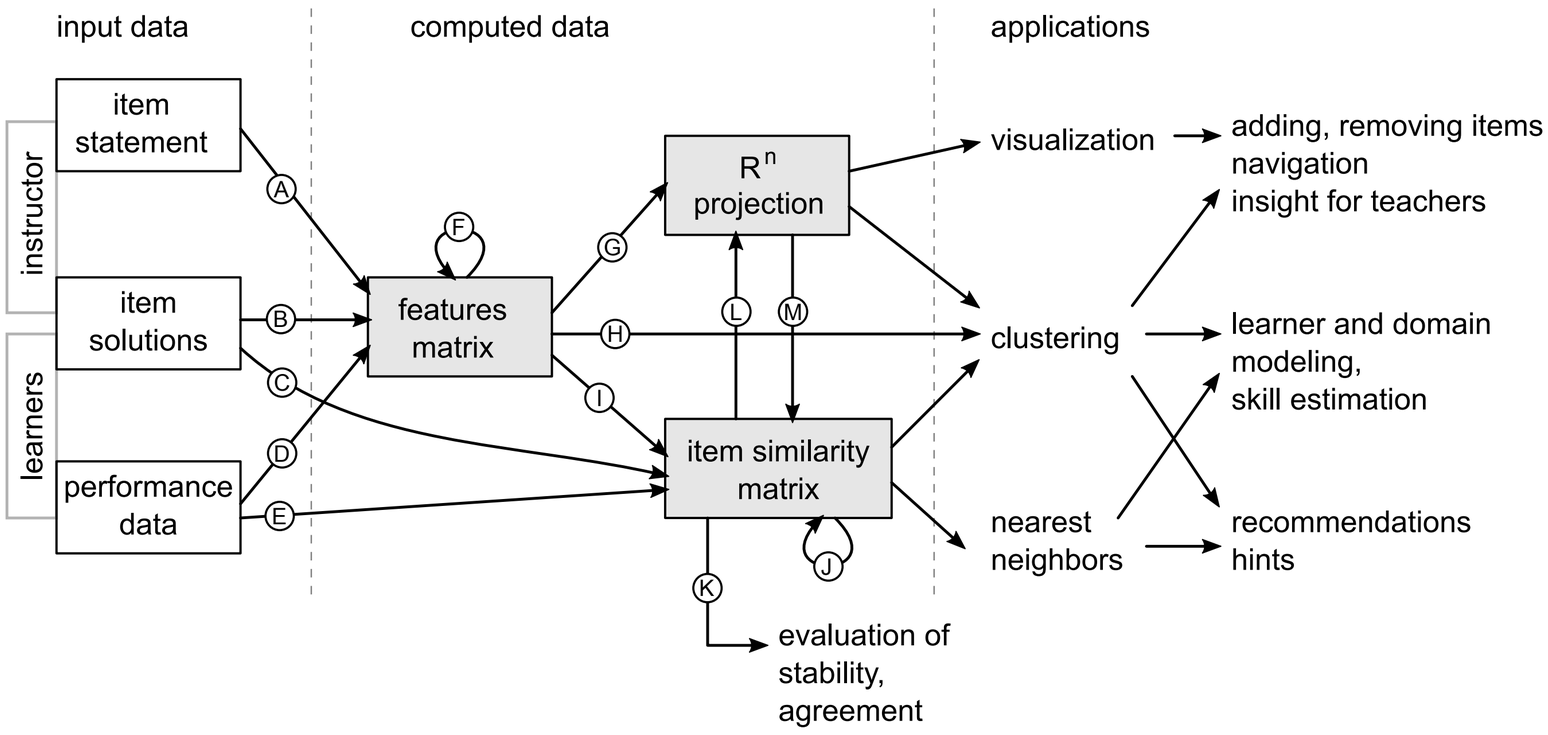
- recommendation of activities
- hints, worked-out examples
- learner and domain modeling
- user interface, navigation
- feedback for content authors

Example

t-SNE projection of RoboMission problems based on a similarity measure



Computing similarity



In each steps there are many choices available.

- choice of features
- normalization: binary, log, TF-IDF, ...
- edit distance: Levensthein, tree, ...
- cosine similarity, Pearson, Euclid, ...

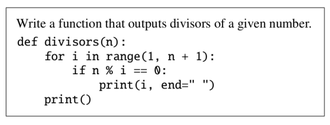
Which choices are the most important?

We explore item similarity for problems in introductory programming.

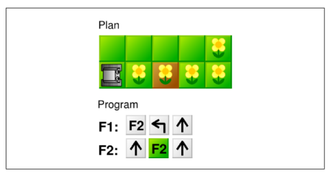
Evaluation

We use data from **three different** introductory programming activities.

Python



Robotanist



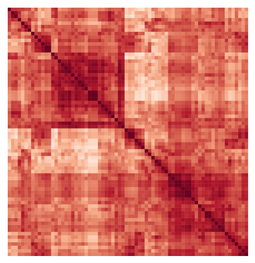
RoboMission

correlations of 72 items

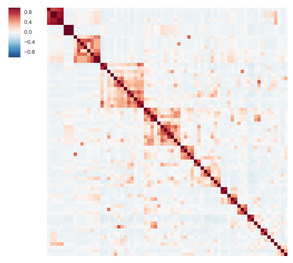
Python programming problems

very different similarity measures

item solutions
prog. keywords



item statement
natural language text

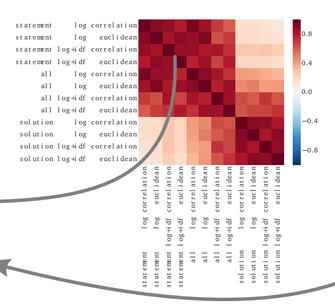


correlations among measures

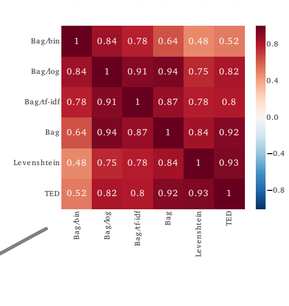
RoboMission

once we fix the type of input data similarity measures are correlated

item statements and solutions



item solutions



Summary

The most important choice in computing similarity is the choice of **input data**.

Recommendations

1. Use item solutions as input data
2. Compute feature matrix: bag-of-words, keyword occurrences
3. Normalize the feature matrix (TF-IDF)
4. Compute Euclidean distance over normalized feature matrix

Adaptive Learning
Research group, Masaryk university Brno

www.fi.muni.cz/adaptivelearning/