Clustering Python Programs

Mihhail Mihhailov & Mikk Märtin

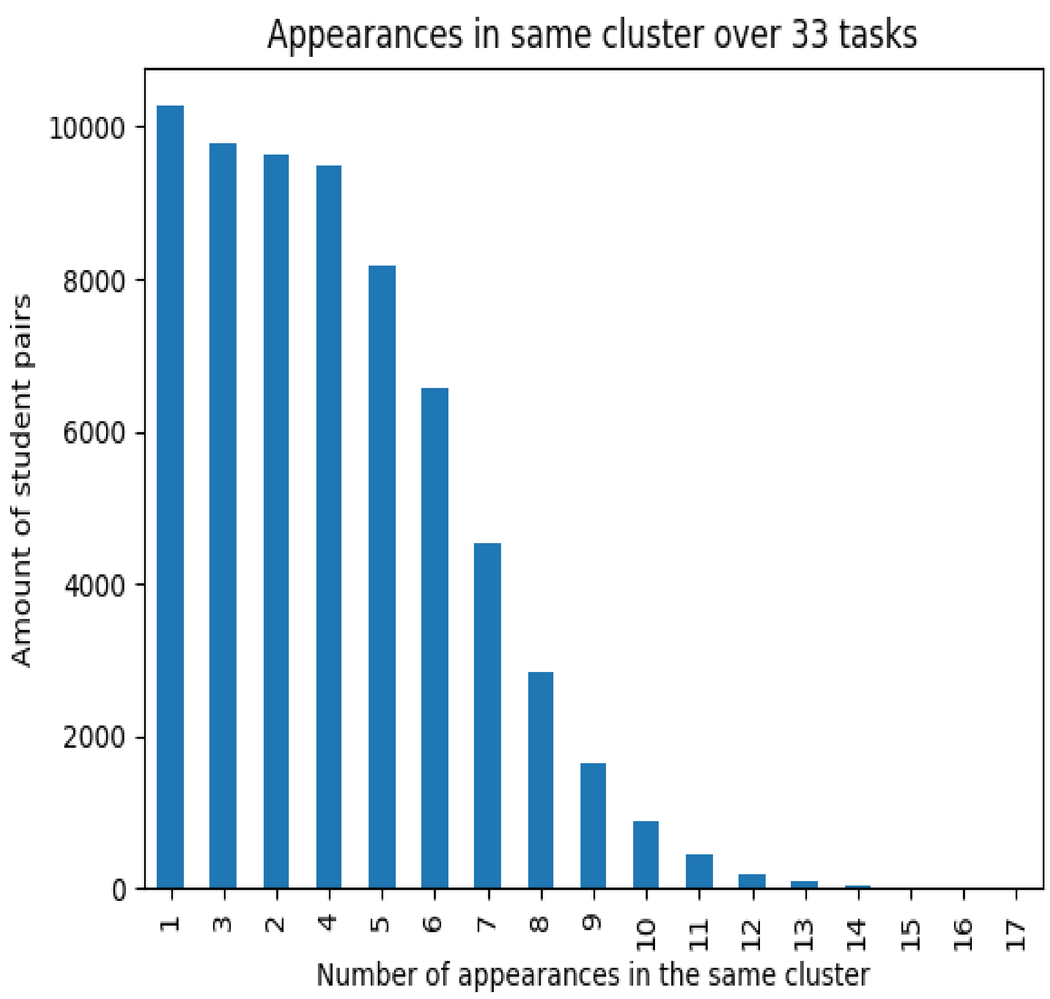
Introduction

For large university courses, a single professor can not keep up with the sheer volume of work that the students submit. Because of this, assistants, and/or automated tests are often relied upon for grading. This can cause two problems.

Firstly, dishonest students, whose works are being graded by different assistants could potentially copy each others work.

Secondly, since programming is to an extent a creative endeavor, the automated tests can not always account for every possible way to solve a given problem.

This project seeks to alleviate both issues using clustering.



Results

For both tasks, we found that the KMeans algorithm significantly outperformed HDBScan, with the latter being unable to assign a cluster to over half the datapoints.

To find outliers and isolate similar approaches to problems, the clusters by themselves proved useful. Unusual solutions would form small but semantically concise clusters(ie. heavy use of libraries, functions, etc.).

For plagiarism detection, a probabilistic approach proved more viable; for all possible student pairs, the amount of times they occurred in the same cluster across all tasks was counted. The highest number of co-occurrences was 17, achieved by 3 pairs.

Data

The dataset was provided to us by Reimo Palm, and consisted of the homeworks submitted by students who took „Programmeerimine“ last year. Before the data could be used, it needed to be read into a dataframe. This proved to be surprisingly challenging for several small, but not insignificant reasons.

Feature extraction and algorithm choice

We extracted features such as number of times a function was declared, number of lines left blank, number of libraries imported, etc. In total, close to 30 features have been extracted. Finding regular expression matches was used for this process. After that, dimensionality reduction was used, so that only features relevant to finding generally similar groups of works were used for that purpose, and features that might be helpful in finding potentially plagiarized works were used for that.

The choice of model was heavily influenced by the unsupervised nature of the task. Ultimately, KMeans and HDBScan were used.

Each homework task was clustered separately.