Project Report
(Homework 10)

Python Programs Clustering

Yehor Atell Krasnopolskyi
Oliver Vainikko

Business understanding (Task 2)

Identifying our project goals

The data set for our project has been collected here, at the University of Tartu, by Reimo Palm, thus our project, besides being of scientific interest, is also going to benefit the University of Tartu and its instructors by developing a tool to analyse similarities between the students' works that involve coding with Python. **The goal is, therefore, to develop a working python program classification tool.**

Assessing the situation

One of the main challenges we face here is that the data is absolutely not numerical. It is code, that is, text with a specific strict structure to it, which we need to analyse. Our approach is based on Python's abstract syntax trees (ASTs) that we first use to build sequences of tokens for each program in the data set and then train a vectoriser on top of the latter. Overall, it is not a very computationally or monetary costly approach which can be run on a laptop.

The set of libraries we will use is relatively small: we need a specific submodule of `sklearn`, and a couple of tools for processing the ASTs (i.e. `visitors`). There's still a chance that this approach is not good enough. In this case, if the method is proven insufficient for our purposes, we will need to modify it.

Our primary assumption about the data should be evident: we assume that analysing the structure of the ASTs with visitors is enough to quantify the difference between different pieces of code. In other words, we hope that there's enough information in this structure to make any further automated decisions.

Addressing the visitors' technique, these objects produce sequences of tokens out of ASTs. Visitors are algorithms that walk through the tree in a certain way and collect node types into a single sequence, which we can then analyse.

Data-mining goals

There are numerous ways we can adjust the so-called "visitors". The main difference between different visitors is which types of nodes they save. One of the crucial goals for data mining, therefore, is to determine which features (which types of nodes in an AST) contain the most relevant information for us.

A true success would be to develop an approach that:

1. Detects similarities between programs of the same structure but with different variable names
2. Detects similarities between programs of the same structure but with different constants or values
3. For any two programs that are clustered separately, they should have different structures.

Understanding the data (Task 3)

The data set is a set of Python programs grouped by purpose. It consists of Python programs submitted by the students as the solutions to the homework tasks in the introductory Computer Programming course. The programs are grouped by homework number and task number; there are 13 homework and 33 tasks. The average number of submissions per task is around 300; the average program size is around 600 B.

Python code could be turned into numerical data by calculating different code metrics. Code metrics could be:

- Cyclomatic complexity (number of decisions in a block of code)
- Maintainability index (a metric used by Visual Studio code)
- Raw metrics (number of comments, lines of code, logical lines of code etc.)
- Halstead metrics (different metrics derived from the code structure)
- And many others…

Python code is constructed by humans to communicate information from one entity to another and is technically considered a language, so the question arises: Should we turn it into numerical metrics? The answer is no.

Since Python has many similarities with natural language, a bag-of-words approach can be taken instead, treating the Python programs as sentences and calculating the similarities between those sentences. The sentences can be created by using a visitor to visit every node in the AST (abstract syntax tree - created right before the program is compiled) and adding the node names to a list. This produces a lot of so-called stop words that must be dealt with; a way of dealing with this is by using TF-IDF which is a fancier variant of bag-of-words where instead of ones and zeros, there is a float value representing the importance of the word relative to the document.

Project planning (Task 4)

1. Make the vectorisation work (this includes building the abstract syntax trees, using a visitor, and training the vectors out of `sklearn`). This includes implementing the loading of the dataset.
2. Test different visitors and training approaches. Determine whether we should pre-train a part of our “model” or train it on each task separately, meaning that we can even have a different vocabulary on each set of submissions.
3. Develop the clustering part, and choose an algorithm (or several) that would cluster programs given the similarity matrix.
4. Edit the code to make it reusable (maybe in the form of a Python module). Write the readme file and documentation.