Определение программистов с одинаковым стилем кода

Богомолов Егор Олегович
научный руководитель: Т. А. Брыксин

JetBrains Research

20 февраля 2018 г.

Мотивация

Программист A

```
int mx=0;
for(int i=0; i<n; i++){
    mx=max(a[i],mx);
}
```

Программист B

```
int sum=0;
for(int i=0; i<n; i++){
    sum+=a[i];
}
```

Программист C

```
int inversions = 0;
for (int i = 0; i < size; i++) {
    for (int j = i + 1; j < size; j++) {
        if (array[i] > array[j]) {
            inversions++;
        }
    }
}
```

Конкретизация целей

- Формализовать понятие похожести кода
- Определить метрику для сравнения разных подходов и определения качества
- Реализовать инструмент
- Протестировать на большой базе программистов (GitHub)

Смежные задачи

- Поиск плагиата
  - J. Yasaswi, S. Purini, C. V. Jawahar (2017). Plagiarism Detection in Programming Assignments Using Deep Features. *IIIT Conference 2017*
  - *J. Kothari, M. Shevertalov, E. Stehle, and S. Mancoridis (2007). A Probabilistic Approach to Source Code Authorship Identification. ITNG Conference'07*
- Определение авторства по исходному коду
  - Alsulami B., Dauber E., Harang R., Mancoridis S., Greenstadt R. (2017) Source Code Authorship Attribution Using LSTM Based Networks. *Computer Security – ESORICS 2017*
  - *A. Caliskan-Islam, R. Harang, A. Liu, A. Narayanan (2015). De-anonymizing Programmers via Code Stylometry. 24th USENIX Security Symposium*

Структура решения

1. Поиск данных
2. Обработка данных
3. Выбор модели
4. Обучение модели
5. Анализ результатов

Данные: материал

- GitHub Java Corpus
  - 14.768 проектов
  - 33M строк кода
  - Проекты разного размера, из разных областей
- Для нашей задачи
  - Проекты с одним автором
  - Проекты, в которых достаточно много файлов
- Для экспериментов выбраны 55 проектов, 4000 файлов

Данные: признаки

- Лексические
  - Частота встречаемости ключевых слов языка, функций, макросов, вложенность кода, среднее число параметров у функций
- Внешние
  - Использование табов или пробелов, отступы, переносы строк, доля пробельных символов
- AST
  - Глубина, средняя глубина, число вершин каждого типа
- Неявные
  - Скрытый слой обученной нейросети

Как анализировать?

- Сравнение качества
  - Точность классификации
- Верификация
  - Домашние задания студентов по Java

Модели

- SVM
  - Очень быстрое обучение
  - Точность 58%
- Нейросеть
  - Использована простая архитектура
  - Точность 65%
- Случайный лес
  - Сравнительно долго обучается
  - Много весит
  - Точность 74%

Верификация на дз студентов

- Работы группы 302-1 АУ по Java в 2016-17 году
- Известные проекты, похожие задачи
- Точность 40-50%
- Ошибочно адресованные работы действительно похожи по стилю

Дальнейшая работа

- Ускорить получение неявных признаков
- Более сложная архитектура нейросети
- Сделать интерфейс для использования
- Активное обучение

Результаты

- Побочный результат: точность классификации лучше чем в большинстве статей
- Программа верифицирована на домашних заданиях по Java

	SVM	NN	RF
Точность	58%	65%	74%
Время обучения	≈ 10 сек	≈ 80 сек	≈ 200 сек
Память	< 5 MB	< 10 MB	230 MB