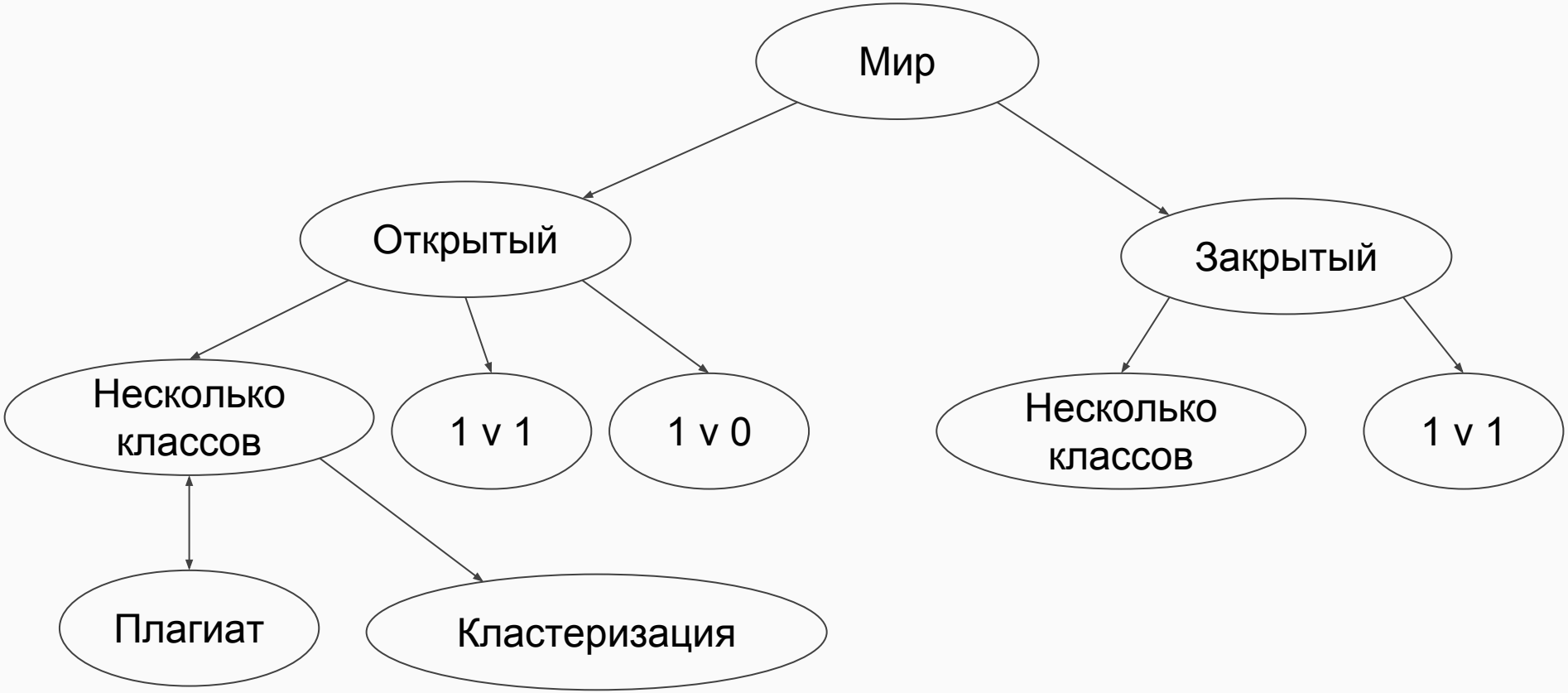
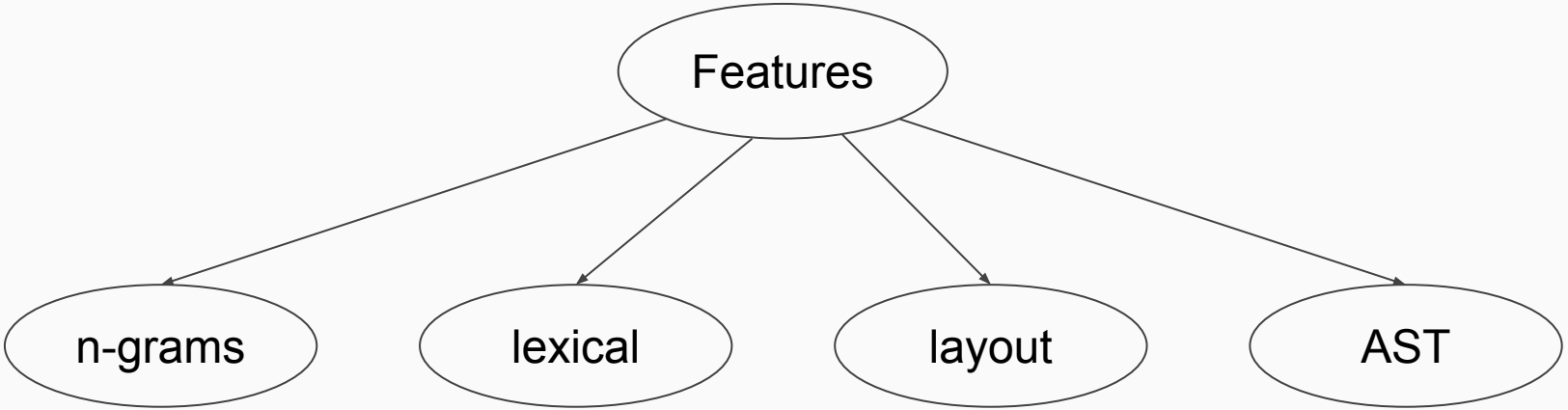
Анализ стиля написания кода

Богомолов Егор, СПбАУ

Классификация задач определения авторства



Что используют для обучения?



Что это такое?

N-grams: строятся на исходном коде или байткоде

Lexical: частота встречаемости ключевых слов языка, функций, макросов, вложенность кода, среднее число параметров у функций...

Layout: число пробелов/табов/пустых строк, переносы строк перед скобками, пробелы вокруг операторов, использование табов/пробелов...

AST-based: типы вершин, средняя глубина для каждого типа, максимальная глубина, что находится в листьях, частота встречаемости для смежных пар вершин...

Связанные проблемы

N-grams: короткие для хорошего сохранения структуры

Lexical: подвержены обфускации, хоть и в меньшей степени, чем layout.

Layout: подвержены обфускации. Легко изменяется с помощью IDE.

AST-based: не всегда понятно, что именно связанное с AST интересно и полезно брать

N-grams

Effective Identification of Authorship Using Byte-Level Information , 2006

- 100% точности на 6-8 программистах, C++
- Если информации мало (студенческие работы на Java), точность падает до 88%
- При росте числа программистов падает точность и требуется больше данных/больше сохраняемых данных.
- Но не все так гладко!

Lexical / Layout

A Probabilistic Approach to Source Code Authorship Identification, 2007

- Векторизация стиля кода
- Выкидывают параметры с нулевым приростом информации
- Используют отдельно 4-grams, отдельно стилевые параметры
- Bayes classifier и VFI
- 70% правильный ответ -- ближайший сосед
- 80-90% правильный ответ -- один из трех ближайших соседей

Pure AST

Source Code Authorship Attribution Using LSTM Based Networks, 2017

- Не выбирают отдельные параметры, а скармливают AST нейросети
- Стараются сохранить древовидную структуру
- Используют BiLSTM, проходящую по детям в DFS в прямом и обратном порядке
- Последнее значение скрытого слоя -- следующий набор параметров
- 96% Python, 25 авторов
- 88% Python, 70 авторов
- 85% C++, 10 авторов

Lexical + Layout + AST

De-anonymizing Programmers via Code Stylometry, 2015

- Обучают Random Forest на трех типах параметров
- Параметры из AST выбраны руками
- Тестируют на посылках GCJ
- Отбирают параметры считая прибавку информации
- 100% C++, 35 авторов
- 98% C++, 250 авторов
- 93% C++, 1600 авторов

Clone detection

Plagiarism Detection in Programming Assignments Using Deep Features, 2017

- Используют значения на скрытом слое char-RNN как один из параметров, затем SVM
- Ищут копии в коде, но можно было бы использовать и для определения авторства

Deep Features for Software Functional Clone Detection, 2016

- Обучают LSTM на AST для определения копирований в коде
- Можно адаптировать для авторства

Что можно сделать?

1. В работе Lexical+Layout+AST можно улучшить часть при помощи pure-AST
2. Нет готовой базы работ, по которым можно быстро искать плагиат
   a. Syntax tree fingerprinting for source code similarity detection - разбирают то, как индексировать векторизованные AST
3. Инструмент для поиска не в точности плагиата, а программистов с похожим стилем кода
   a. Учитывается и непосредственный стиль, и параметры, связанные с AST
   b. Отличается от определения авторства отсечением на последнем этапе
   c. Предполагает дальнейшее обучение после создания инструмента