МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

К ЗАЩИТЕ ДОПУСТИТЬ

заведующий каф. КИБЭВС

д-р техн. наук, проф.

_____А.А. Шелупанов

«_____»_____2017г.

«ОПРЕДЕЛЕНИЕ АВТОРСТВА ИСХОДНОГО КОДА»

Специалистская работа по направлению 10.05.03 –

Информационная безопасность автоматизированных систем

Студент гр. 722

_____ Мейта М.В.

«_____»_____2017г.

Руководитель:

канд. техн. наук, доцент каф. БИС

_____ Романов А.С.

«_____»_____2017г.

ТОМСК 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

УТВЕРЖДАЮ
Зав. кафедрой КИБЭВС

_____ А.А. Шелупанов
«_____»_____2017 г.

ЗАДАНИЕ
по дипломному проектированию студенту _____
_____ группа _____ факультет _____

1. Тема проекта (работы): _____
_____

2. Срок сдачи студентом законченного проекта

_____

3. Исходные данные к проекту

_____
_____
_____
_____
_____
_____

4. Содержание расчетно-пояснительной записи (перечень подлежащих разработке вопросов)
_____
_____
_____
_____
_____
_____

5. Перечень графического материала (с точным указанием обязательных чертежей)
_____
_____
_____
_____
_____
_____
_____

РЕФЕРАТ

Отчет содержит 34 страницу, 10 рисунков, 6 таблиц, 31 источник, 3 приложения.

СТИЛОМЕТРИЯ, ИСХОДНЫЙ КОД, ДЕАНОНИМИЗАЦИЯ АВТОРА, C/C++, КЛАССИФИКАЦИЯ, PYTHON, SKLEARN, JUPYTER NOTEBOOK, DECISION TREES, RANDOM FOREST CLASSIFIER, CROSS-VALIDATION, ADA BOOST, EXTREMLY RANDOMIZED TREES, GITHUB, LATEX.

Цель работы — разработка программного обеспечения для определения авторства исходного кода программ на на языке C/C++, основанного на методах стилометрического анализа текста.

В рамках преддипломной практики были поставленны следующие задачи:

– обзор существующих исследований, разработок, методов стилометрического анализа текста, в том числе исходного кода программ;

– построение модели процесса определения авторства исходного кода;

– разработка программного обеспечения для анализа исходного кода программ с применением стилометрии для определения авторства программного обеспечения;

– разработка программного интерфейса;

– подготовка и обработка тестового набора данных;

– исследование эффективности разработанной программы на основе модели анализа исходных кодов, анализ результатов.

Объект исследования: деанонимизация автора программного обеспечения.

Предмет исследования: стилометрия исходного кода программ на языках высокого уровня.

Достигнутые результаты: главным результатом преддипломной практики является программное обеспечение «WhoseCppCode», предназначенное для построения, тестирования и оценки модели классификации авторов исходного кода на языке C/C++, а также визуализации полученных результатов.

Отчет по преддипломной практике выполнен согласно ОС ТУСУР 01-2013 [1] при помощи системы компьютерной вёрстки LaTeX.

Содержание

Введение

Задача определения авторства является широко распространенной проблемой в рамках исследования естественных языков, однако в меньшей степени для языков программирования. Тем не менее, с распространением применения компьютерных систем и сетей возросло и количество преступлений в информационной сфере. Существует множество разновидностей кибератак — различные компьютерные вирусы, трояны, несанкционированное копирование данных с кредитных карт, DDoS-атаки и многое другое. Возможность деанонимизации авторов вредоносного программного обеспечения может внести существенный вклад в развитие компьютерной криминалистики.

Считается, что у каждого программиста есть свои специфические профессиональные приемы, привычки, методы написания программного кода, свой так называемый «стиль программирования» и иные признаки, идентифицирующие автора. При этом, как и в случае с естественными языками, на индивидуальный «почерк» программиста может оказывать влияние множество факторов, таких как образование, географическое место проживания, уровень квалификации и другие. «Почерк» также может изменяться с течением времени, развитием технологий и общепринятых норм «хорошего» стиля написания программ. Под «хорошим» стилем обычно понимается набор правил, позволяющих писать код, удобный для чтения, понимания, внедрения дальнейших изменений и рефакторинга. Крупные IT-компании и корпорации обычно вводят свои собственные стандарты кодирования, которые зачастую используются сторонними организациями и индивидуальными программистами в своей работе. Примером могут служить руководства по стилю программирования на языке C++ компаний Google [2] и Geosoft [3].

Определение авторства исходного кода представляет собой актуальную задачу в сфере информационной безопасности, лицензирования в области разработки программного обеспечения, а также может оказать существенную помощь во время судебных разбирательств, при решении вопросов об интеллектуальной собственности и плагиате.

Целью преддипломной практики является разработка программного обеспечения для определения авторства исходного кода программ на языке C/C++, основанного на методах стилометрического анализа текста.

1 Кафедра КИБЭВС

Местом прохождения практики была выбрана кафедра ТУСУРа – КИБЭВС (Кафедра комплексной информационной безопасности электронно- вычислительных систем).

Согласно информации с официального сайта [4], кафедра организована в ТУСУР в 1971 году как кафедра «Конструирования и производства электронно-вычислительной аппаратуры» (КиПЭВА) вскоре переименованной в кафедру «Конструирования электронно-вычислительной аппаратуры» (КЭВА).

21 сентября 1999 г. в связи с открытием новой актуальной специальности 090105 – «Комплексное обеспечение информационной безопасности автоматизированных систем» кафедра КЭВА была переименована в кафедру «Комплексной информационной безопасности электронно-вычислительных систем» (КИБЭВС). Заведующим кафедрой КИБЭВС, а также ректором ТУСУРа на сегодняшний день является ректор ТУСУРа, Александр Александрович Шелупанов, лауреат премии Правительства Российской Федерации, действительный член Международной Академииnаук высшей школы РФ, действительный член Международной Академии информации, Почетный работник высшего профессионального образования РФ, заместитель Председателя Сибирского регионального отделения учебно-методического объединения вузов России по образованию в области информационной безопасности, профессор, доктор технических наук.

Кадровый состав непрерывно укреплялся с момента её образования в 1971 году. В 2011 году, в год 40-летия кафедры, её коллектив состоял из 53 человек, в их числе 34 опытных высококвалифицированных специалиста и 19 аспирантов. Среди сотрудников кафедры члены Академий наук РФ; 4 профессора; 12 доцентов, кандидатов наук; старшие научные сотрудники, кандидаты наук и др.

С 2008 г. кафедра КИБЭВС входит в состав Института «Системной интеграции и безопасности».

На базе кафедры КИБЭВС ТУСУР в 2002 году организовано «Сибирское региональное отделение учебно-методического объединения Вузов России по образованию в области информационной безопасности.

2 Обзор информационных источников

На первом этапе практики необходимо было провести подробный аналитический обзор информационных источников, рассмотреть существующие методы определения авторства исходного кода и различные подходы к решению такого рода задачи.

В работе [5] представлен набор инструментов и техник, используемых для решения задач анализа авторства исходного кода, а также обзор некоторых наработок в данной предметной области. Кроме того, авторы приводят собственную классификацию проблем и подходов к их решению в рамках задачи деанонимизации авторов программного обеспечения.

Frantzeskou [5] выделяет следующие проблемы (задачи) анализа авторства исходного кода:

– идентификация автора — направлена на определение, принадлежит ли определенный фрагмент кода конкретному автору;

– характеристика автора — базируется на анализе стиля программирования;

– определение плагиата — нахождение схожестей среди множества фрагментов файлов исходного кода;

– определение намерений автора — был ли код изначально вредоносным или стал таковым в следствие программной ошибки;

– дискриминация авторов — определение, был ли код написан одним автором или несколькими.

Подходы к решению вышеперечисленных проблем (задач):

– анализ «вручную» — данный подход включает в себя исследование и анализ фрагмента исходного кода экспертом;

– вычисление схожести — базируется на измерении и сравнении различных метрик или токенов для набора файлов исходного кода;

– статистический анализ — в таком подходе используются статистические техники, такие как дискриминантный анализ и стилометрия, позволяющие определить различия между авторами;

– машинное обучение — используются методы рассуждения на основе прецедентов и нейронные сети для классификации автора на базе некоторого набора метрик.

В работе [6] предложен способ определения авторства программного обеспечения. в основе которого лежит статистический подсчет метрик, отражающих «почерк создателя» программного обеспечения. На основе метрик составлен «профиль почерка» программистов и вычисляется отклонение от данного профиля для каждого автора. Преимуществом данного метода является его независимость от языков программирования. Метод получил название SCAP (Source Code Author Profiles).

В [7] исходный код транслировался в абстрактные синтаксические деревья, после чего разби-

вался на функции. Дерево каждой функции принималось за отдельный документ с известным автором. Выборка, состоящая из такого рода деревьев подавалась на вход SVM-классификатору, оперирующему данными типа «дерево». Классификатор обучался на файлах исходного кода двух авторов, в результате чего удалось достичь точности около 67-88%.

В работах [8], [9] и [10] рассматривался способ атрибуции исходного кода с использованием метода N-грамм. Вопрос определения авторства программ в данной работе рассматривался с точки зрения определения плагиата. В качестве выборки использовался набор из 1640 файлов исходного кода, написанных 100 авторами. Позднее удалось улучшить точность классификации данной модели до 77% за счет применения рейтинговых схем.

В [11] применялся алгоритм классификации Random Forest [12] и построение абстрактных синтаксических деревьев. Обучение и тестирование производилось для количества авторов от 250 до 1600. При этом удалось добиться высокой точности — 94-98%. Кроме того, авторы статьи выяснили в ходе работы, что сложнее определить авторов более простых примеров, нежели сложных программ, а также значительно выделяются авторы с большим опытом программирования на C/C++. В своей дальнейшей работе [13] авторы предложили применение данного подхода для анализа неполных, некомпилируемых образцов кода.

На основании проведенного исследования было решено опробовать подход, основанный на вычислении статистических метрик, характеризующих авторский стиль написания программ, и методов машинного обучения.

Сравнительная таблица с подробным описанием данных информационных источников и используемых в них методов приведена в приложении Б.

3 Программа и методика испытаний

Для тестирования аналитической модели в машинном обучении применяется процедура скользящего контроля, получившая название кросс-валидации (cross-validation) или перекрестной проверки. [14]

Процедура кросс-валидации включает в себя случайное разбиение на k подгрупп (или фолдов) примерно одинакового размера. Первый фолд служит для тестирования модели, остальные используются для обучения классификатора. Для тестовой подвыборки вычисляется среднеквадратичное отклонение. Процедура повторяется k-1 раз, при этом каждая из подгрупп выступает в роли тестовой выборки.

В данной работе тестирование производилось с применением 10-фолдовой кросс-валидации. Всего было произведено 10 вычислительных экспериментов.

4 Выбор набора признаков, характеризующих автора программы

Задача стилометрического анализа исходного кода состоит в выделении и статистическом подсчете лексических, синтаксических, структурных и или каких-либо иных признаков на основании обработки текста программы.

Перечисленные в данном разделе признаки, по которым идентифицировались авторы, являются характерными для языков C и C++, однако могут быть использованы для исследования C-подобных языков, например, D, Java, Objective C, C#, PHP, perl и другие.

4.1 Лексические признаки

Главная особенность данной группы признаков состоит в том, что они могут быть вычислены при непосредственном анализе исходного кода программы в виде текстового файла. При этом код программы может быть некомпилируемым, неполным, содержащим синтаксические или программные ошибки.

Лексические признаки, как правило, улучшают читаемость кода и включают в себя:

– Стиль комментирования — данная подгруппа определяет преобладающий в тексте вид комментариев (однострочные или многострочные), а также общее их количество.

– Стиль расстановки фигурных скобок — к наиболее известным относят «K&R», «Whitesmith», «One True Bracing Style», стиль Алмена и другие. [15]

– Стиль разметки — расстановка пробелов, табуляций, число переносов строки к общей длине файла.

Дополнительно вычисляются:

– Число макросов — использует ли программист директивы препроцессора.[16]

– Средняя длина строки — позволяет также оценить читаемость кода (слишком длинные программные файлы плохо воспринимаются человеком).

В приложении В приведено описание использованных при классификации авторов признаков.

4.2 Ключевые слова C++

Ключевые слова C++ представляют собой список зарезервированных последовательностей символов, используемых языком, недоступных для переопределения.

Для ключевых слов языка C++ вычислялась статистическая мера TF (term frequency), отображающая число вхождения некоторого ключевого слова к общему количеству слов в документе. Подсчет частот ключевых слов может дать представление о предпочтениях автора в определенного рода конструкциях, например, циклов «for» относитльно «while» или «do while», а также об уровне его профессиональной квалификации (определенные конструкции языка C/C++ использу-

ются крайне редко, сложны для понимания и предназначены для решения узкоспециализированных задач).

Словарь из 84 ключевых слов C++ (стандарт 11) был взят на сайте с официальной документацией [17] и представлен в таблице 4.1.

Таблица 4.1 – Ключевые слова языка C++ (стандарт 11)

Ключевые слова языка C++					
alignas	char32_t	enum	namespace	return	try
alignof	class	explicit	new	short	typedef
and	compl	export	noexcept	signed	typeid
and_eq	const	extern	not	sizeof	typename
asm	constexpr	false	not_eq	static	union
auto	const_cast	float	nullptr	static_assert	unsigned
bitand	continue	for	operator	static_cast	using
bitor	decltype	friend	or	struct	virtual
bool	default	goto	or_eq	switch	void
break	delete	if	private	template	volatile
case	do	inline	protected	this	wchar_t
catch	double	int	public	thread_local	while
char	dynamic_cast	long	register	throw	xor
char16_t	else	mutable	reinterpret_cast	true	xor_eq

5 Моделирование

Описание процесса определения авторства исходного кода программ в виде модели «черного ящика» согласно методологии IDEF0 представлено на рисунке 5.1, его декомпозиция — на рисунке 5.2.

Рисунок 5.1 – Модель «черного ящика» процесса определения авторства исходного кода по методологии IDEF0

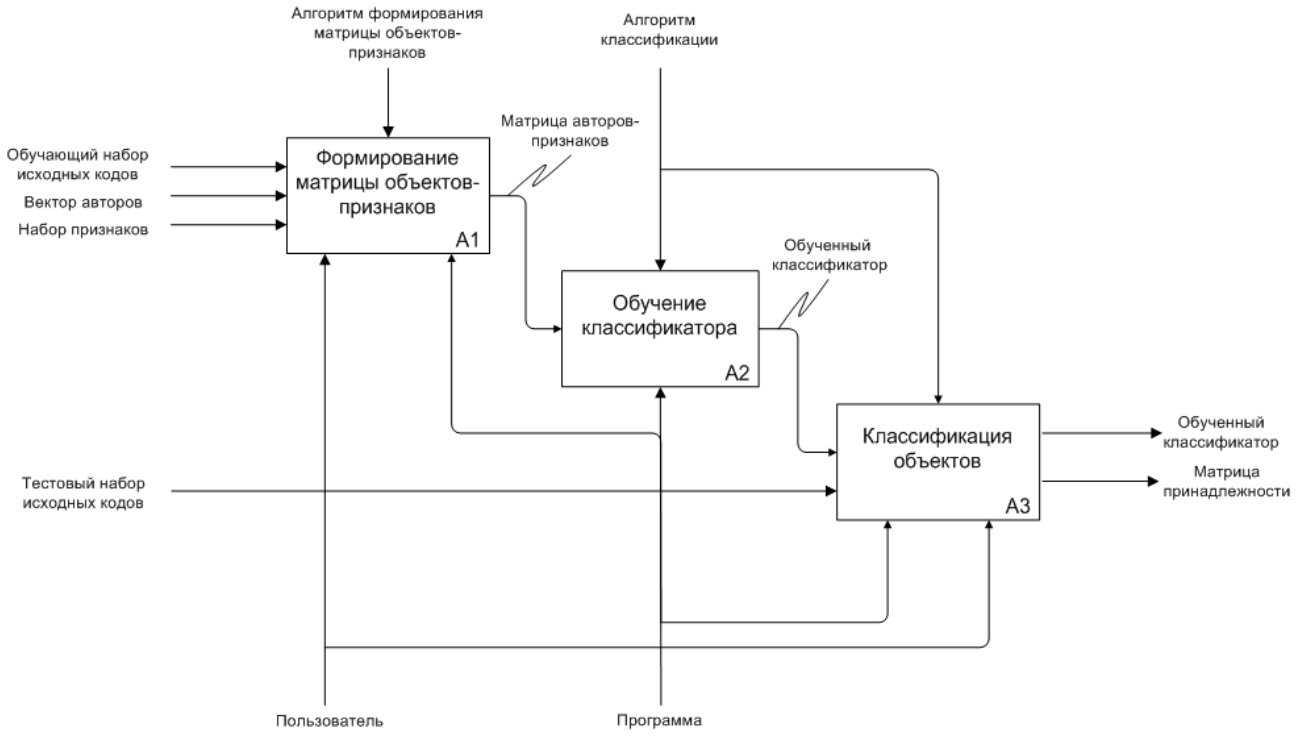

Рисунок 5.2 – Декомпозиция «черного ящика» процесса определения авторства исходного кода по методологии IDEF0

6 Классификация

В данной работе в качестве базового алгоритма для всех классификаторов (см. разделы 6.1, 6.2, 6.3) были выбраны деревья решений (Decision Trees), тестирование и оценка модели производилась на основе 10-фолдовой кросс-валидации (подробнее о тестировании модели в разделе 7).

Деревья решений (Decision Trees) [18] или деревья принятия решений являются одним из наиболее популярных методов решения задач классификации, регрессии и прогнозирования. Впервые деревья решений были предложены Ховилендом и Хантом (Hoveland, Hunt) в конце 50-х годов прошлого века и в наиболее простом виде представляют собой совокупность правил в иерархической структуре. Основа такой структуры — это ветвление при проверке условий («Да» — «Нет»).

6.1 Random Forest

Алгоритм классификации Random Forest Classifier строится на двух базовых принципах:

– bagging — мета-алгоритм в машинном обучении, при котором на основе большого числа «слабых» классификаторов (в данном случае деревьев решений) строится один «сильный» классификатор (рис. 6.1);

– метод случайных подпространств.

Преимущества данного алгоритма классификации:

– способность эффективно обрабатывать данные с большим числом признаков и классов;

– нечувствительность к масштабированию (к любым монотонным преобразованиям) значений признаков;

– существует методы оценивания значимости отдельных признаков в модели;

– внутренняя оценка способности модели к обобщению (тест out-of-bag);

– высокая параллелизуемость и масштабируемость.

Недостатки алгоритма Random Forest Classifier:

– алгоритм склонен к переобучению на некоторых задачах, особенно на зашумленных, однако для избежания переобучения используется энтропия Шеннона или коэффициент прироста информации (англ. Gain);

– большой размер получаемых моделей приводит к существенным затратам памяти на хранение деревьев, однако данный недостаток решается повышением вычислительных мощностей и распараллеливанием вычислений. [12]

6.2 AdaBoost

Алгоритм AdaBoost (сокр. от adaptive boosting) [19] является мета-алгоритмом, в процессе обучения строит композицию из базовых алгоритмов обучения для улучшения их эффективности.

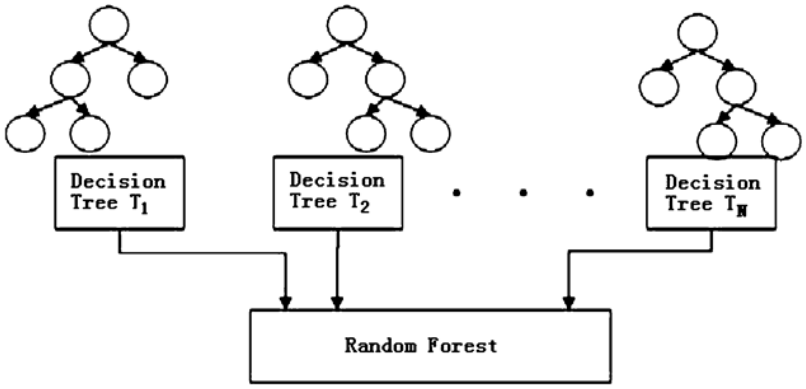

Рисунок 6.1 – Random Forest Classifier

Достоинства:

– Хорошая обобщающая способность. В реальных задачах (не всегда, но часто) удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться (в некоторых задачах) по мере увеличения числа базовых алгоритмов.

– Простота реализации.

– Время построения композиции практически полностью определяется временем обучения базовых алгоритмов.

Недостатки алгоритма классификаций AdaBoost:

– Склонен к переобучению при наличии значительного уровня шума в данных.

– Требует достаточно длинных обучающих выборок.

– Бустинг может приводить к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти для хранения базовых алгоритмов и существенных затрат времени на вычисление классификаций.

6.3 ExtraTrees

Алгоритм ExtraTrees (Extremly Randomized Trees) [20] является модификацией алгоритма Random Forest Classifier (см. раздел 6.1), но отличается еще более рандомизированным разделением входного набора данных на подвыборки. Как правило, результаты работы данного алгоритма схожи с результатами Random Forest Classifier, однако в определенных случаях могут давать улучшение точности классификации.

7 Тестирование модели

Для тестирования аналитической модели в машинном обучении применяется процедура скользящего контроля, получившая название кросс-валидации (cross-validation) или перекрестной проверки. [14]

Процедура кросс-валидации включает в себя случайное разбиение на k подгрупп (или фолдов) примерно одинакового размера. Первый фолд служит для тестирования модели, остальные используются для обучения классификатора. Для тестовой подвыборки вычисляется среднеквадратичное отклонение. Процедура повторяется k-1 раз, при этом каждая из подгрупп выступает в роли тестовой выборки.

В данной работе тестирование производилось с применением 10-фолдовой кросс-валидации. Всего было произведено 10 вычислительных экспериментов.

8 Описание тестового набора данных

Burrows в работе [9] выделяет следующие ключевые параметры тестовых данных, которые могут влиять на точность классификации:

– Число авторов — с увеличением числа авторов сложность классификации увеличивается, точность — снижается.

– Число экземпляров выборки для каждого автора — желательно соблюдать одинаковым для всех авторов во избежание отклонения в сторону наиболее точно описанных авторов, а также иметь больше экземпляров для увеличения размера тестовой выборки.

– Средняя длина образца кода (количество непустых строк кода) — чем длиннее, тем выше точность классификации. Изменение длины экземпляров выборки может влиять на отклонение в сторону наиболее точно описанных авторов, однако не представляется возможным соблюдать длину экземпляра выборки постоянной.

– «Стилистическая зрелость» (stylistic maturity) авторов — уровень квалификации, личные и профессиональные предпочтения в стиле написания программ.

– Временные метки образцов кода (подразумевается, что со временем программы устаревают, технологии и методы программирования меняются и, как следствие, изменяется стиль программирования).

– Репрезентативность выборки — демографические, социальные и другие факторы.

– Типы авторов — студент, фрилансер, профессиональный разработчик. В идеале система должна включать в себя разные типы.

– Языки программирования — если тестировать несколько языков одновременно, результат будет зависеть от характерных признаков языка.

– Авторство в одном лице — большинство проектов выполняются в сотрудничестве с другими разработчиками.

– Корректное авторство — без плагиата, копирования и т.п.

Burrows упоминает также от том, что характерный стиль программирования нестабилен в начале карьеры программиста, что может существенно отличать начинающего специалиста и профессионала разработки.

Программное обеспечение «WhoseCppCode» тестировалось на трех наборах данных:

1) «Students» — выборка представляет собой работы студентов первого курса обучения по предмету «Основы программирования». Все программы реализуют решения однотипных задач в рамках учебной дисциплины, что исключет их разделение при классификации по функциональному назначению вместо стилистических особенностей и снижение точности классификации.

2) «Google Code Jam» — общедоступные данные ежегодной международной олимпиады по

программированию Google Code Jam 2016. [21] Так же, как и в первой выборке, авторы решали схожие задачи, используя различные подходы и алгоритмы.

3) «GitHub» — данные, собранные с сайта GitHub [22], крупнейшего [23] веб-сервиса для хостинга IT-проектов и их совместной разработки.

Сбор данных с веб-хостинга GitHub производился по следующему принципу:

1) Выбирались крупные open-source репозитории (удаленные хранилища программного кода и данных), посвященные разработке проектов на C/C++.

2) Просматривался список контрибьюторов (пользователей, вносивших изменения в проект).

3) В качестве авторов выбирались те контрибьюторы, у которых имеются личные проекты, написанные на C/C++.

4) На основе списка пользователей автоматически, средствами программы «WhoseCppCode», производился сбор и сохранение файлов исходного кода для каждого автора.

В таблице 8.1 приводится описание некоторых характеристик каждого набора данных. В данном случае под смешанным типом авторов подразумевается, что разработчики могли быть совершенно разного уровня квалификации и рода деятельности (студенты, фрилансеры, начинающие и профессиональные разработчики, программисты-любители и т.д.).

Таблица 8.1 – Тестовые данные

Набор данных	«Students»	«Google Code Jam»	«GitHub»
Число авторов	3	30	30
исло файлов исходного кода на одного автора	14	9	78
Всего файлов исходного кода	42	278	2334
Минимальное число строк кода	33	36	26
Максимальное число строк кода	160	461	16348
Среднее число строк кода на один файл исходного кода	45	87	234
Тип авторов	Студенты	Смешанный	Смешанный

Каждая выборка представляет собой совокупность файлов исходного кода программ на языке C/C++ с расширениями *.cpp, *.c, *.h, *.hpp, *.cxx, *.cc, *.ii, *.ixx, *.ipp, *.inl, *.txx, *.tpp, *.tpl.

9 Описание программного обеспечения «WhoseCppCode»

Программа «WhoseCppCode», разработанная в ходе работы, состоит из двух основных частей:

– программного модуля, реализующего все необходимые функции для сбора, анализа и обработки данных, а также построения модели классификации авторов исходного кода, описанной в разделе 5;

– программного интерфеса на основе технологии Jupyter Notebook [24], предназначенного для визуализации полученных в ходе классификации результатов, сбора необходимых данных с ресурса GitHub [22], построения матрицы объектов-признаков на основе входных данных, проведения вычислительных экспериментов.

Программный модуль реализован на языке прораммирования высокого уровня Python с использованием следующих программных библиотек:

– Scikit-Learn [25] — open-source библиотека для машинного обучения: классификации, регрессии, кластеризации и т.д.

– Plotly [26] — графическая Python-библиотека для построения интерактивных графиков, таблиц, диаграмм.

– Numpy [27] — библиотека для научных вычислений, предоставляющая методы работы с большими массивами данных.

– Scipy [28] — предоставляет среду для проведения математических и научных вычислений.

– Pandas [29] — open-source библиотека, предназначенная для анализа данных.

– Ipywidgets [30] — интерактивные HTML виджеты для Jupyter Notebook.

Интерфейс основан на веб-технологиях, может использоваться для демонстрации возможностей программ на языке Python. Библиотека Jupyter Notebook, с помощью которой был реализован данный интерфейс, была выбрана за счет ряда преимуществ:

– является свободным ПО;

– поддерживает множество языков программирования;

– позволяет хранить вместе код, изображения, комментарии, формулы и графики;

– не требует знаний и применения веб-технологий, таких как CSS, HTML, JavaScript;

– может быть запущен на любом сервере, необходим только доступ по ssh/http;

– позволяет экспортировать код и сам блокнот в любом формате;

– предназначена для демонстрации разработок на языке Python (в основном в машинном обучении).

Основной модуль программы «WhoseCppCode» может быть использован отдельно от Jupyter Notebook при разработке различного рода программ, систем и интерфейсов лицами, заинтересован-

ными в задаче классификации программистов.

Диаграмма действий в нотации UML, описывающая основной алгоритм работы программы «WhoseCppCode» представлена на рисунке 9.1, примеры ввода и вывода данных в интерфейсе Jupyter Notebook — на рисунках 9.2, 9.3 и 9.4.

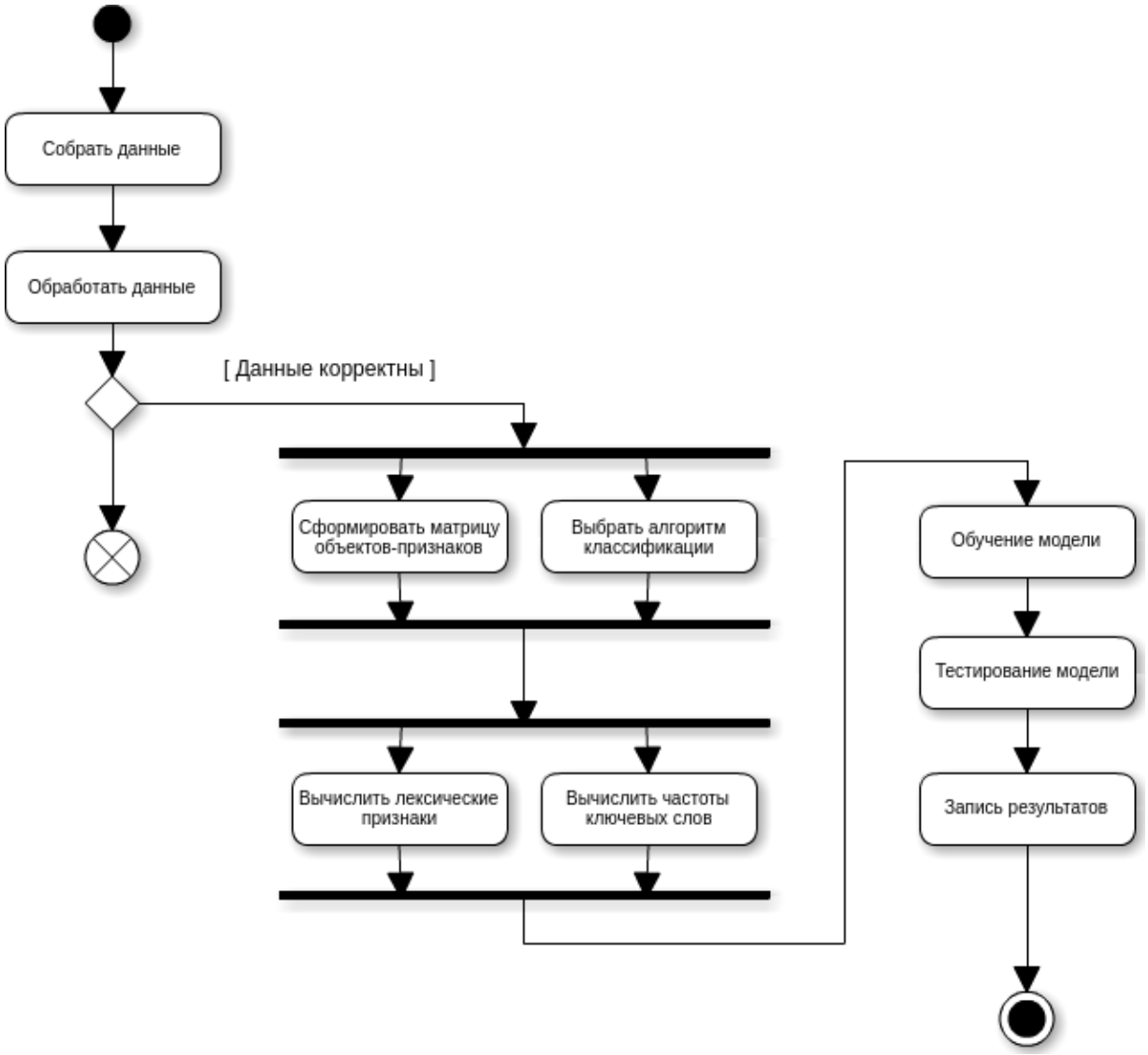

Рисунок 9.1 – Диаграмма действий алгоритма работы программы «WhoseCppCode»

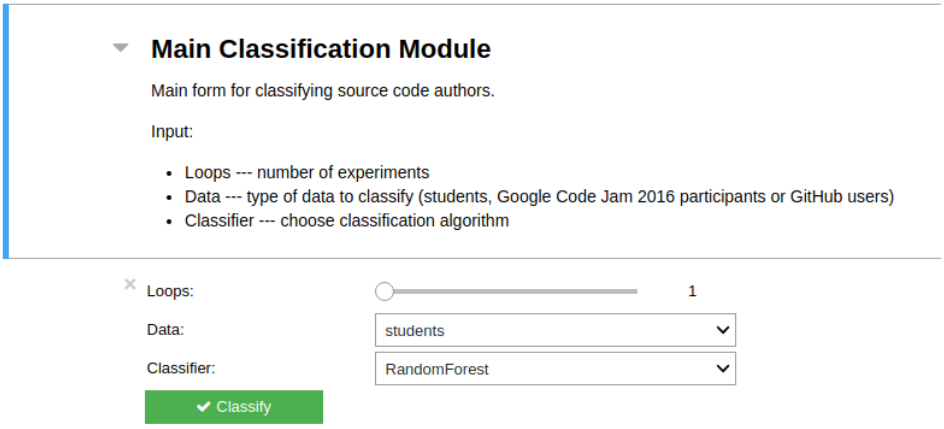

Рисунок 9.2 – Вид основной формы программного интерфейса

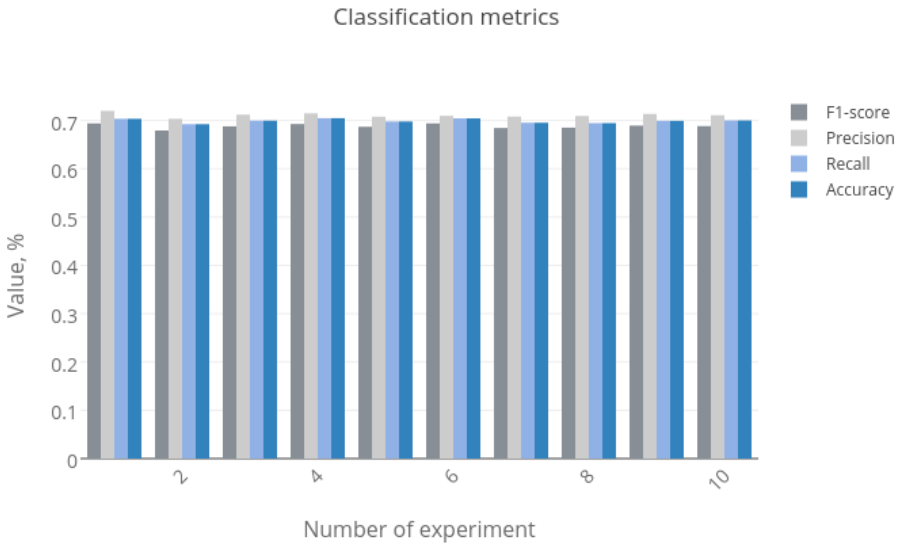

Рисунок 9.3 – Вывод диаграммы результатов классификации

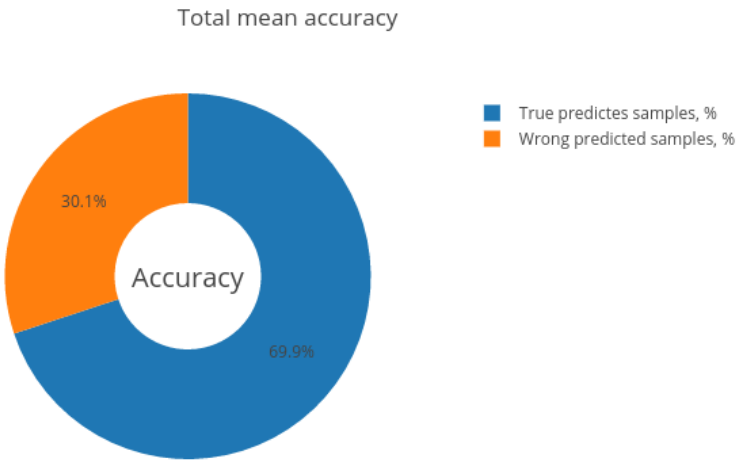

Рисунок 9.4 – Пример вывода диаграммы для средней точности классификации

10 Критерии оценки эффективности классификации

Критерии оценки работы классификатора [31] представлены в таблице 10.1, где:

– tp — истинно-положительное решение;

– tn — истинно-отрицательное решение;

– fp — ложно-положительное решение;

– fn — ложно-отрицательное решение;

– Accuracy (точность) — отношение количества документов, по которым классификатор принял правильное решение, к общему числу документов (примеров файлов исходного кода);

– Precision (правильность) — доля документов, действительно принадлежащих данному классу, относительно всех документов, которые система отнесла к этому классу;

– Recall (полнота) — доля найденных классфикатором документов, принадлежащих классу, относительно всех документов этого класса в тестовой выборке;

– F1-score (F1-мера) — гармоническое среднее между правильностью и полнотой.

Таблица 10.1 – Критерии оценки работы классификатора

Критерий	Формула	Луч. знач.	Худ. знач.
Accuracy (точность)	(tp + tn) / число примеров * 100 %	100 %	0 %
Precision (правильность)	tp / (tp + fp)	1	0
Recall (полнота)	tp / (tp + fn)	1	0
F1-score (F1-мера)	2 * (precision * recall) / (precision + recall)	1	0

11 Результаты классификации

При тестировании классификатора использовались критерии оценки, описанные в разделе 10, а также время работы программы. Результаты работы классификатора представлены в таблице 11.1, а также на рисунках 11.1, 11.2 и 11.3.

Таблица 11.1 – Результаты работы

Набор данных «Students»					
Классификатор	Accuracy, %	Precision	Recall	F1-score	Время работы, сек.
RandomForest	89,55	0,90	0,93	0,90	93,61
AdaBoost	70,45	0,70	0,74	0,70	53,22
ExtraTrees	91,85	0,92	0,95	0,92	53,70
Набор данных «Google Code Jam»					
Классификатор	Accuracy, %	Precision	Recall	F1-score	Время работы, сек.
RandomForest	86,66	0,86	0,88	0,87	110,65
AdaBoost	19,43	0,16	0,16	0,19	103,53
ExtraTrees	88,09	0,88	0,90	0,88	60,34
Набор данных «GitHub»					
Классификатор	Accuracy, %	Precision	Recall	F1-score	Время работы, сек.
RandomForest	69,92	0,69	0,71	0,70	223,77
AdaBoost	16,44	0,09	0,11	0,16	451,63
ExtraTrees	70,99	0,70	0,72	0,71	201,13

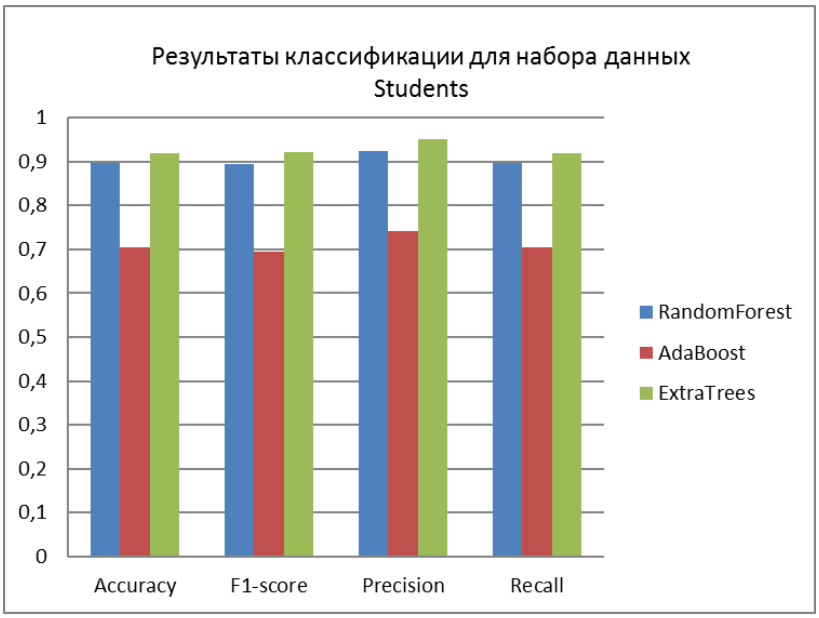

Рисунок 11.1 – «Students»

Наихудшие результаты показал алгоритм AdaBoost, в то время как наиболее точным и быстрым из трех представленных алгоритмов оказался ExtraTrees (см. раздел 6.3).

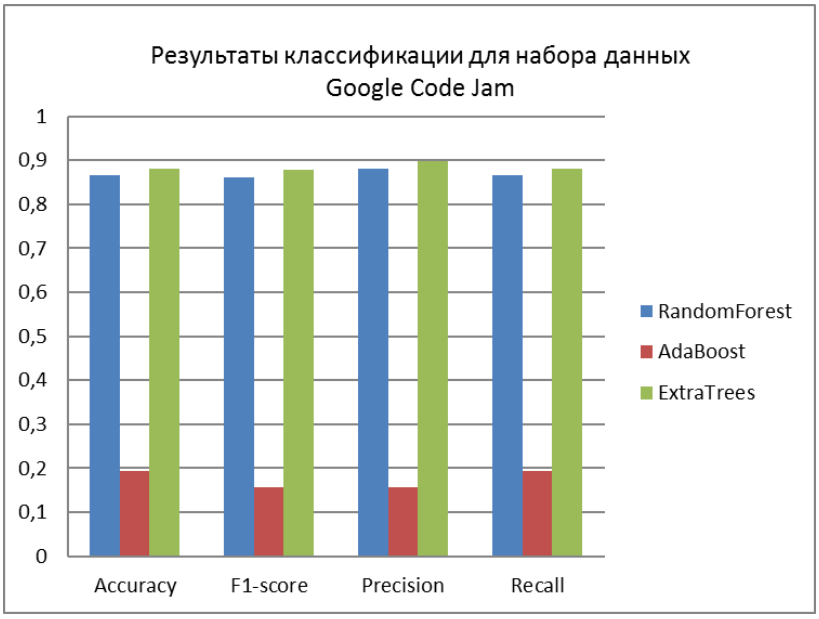

Рисунок 11.2 – «Google Code Jam»

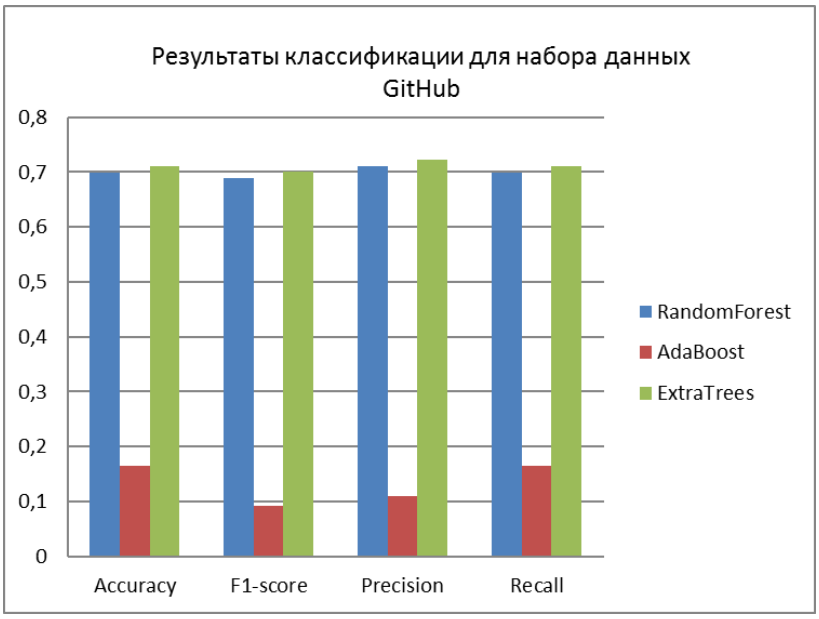

Рисунок 11.3 – «GitHub»

С использованием метода, основанного на извлечении лексических признаков и классификации с помощью алгоритма ExtraTrees (Extremly Randomized Trees) точность классификации составила 70-71 % на выборке данных из 30 авторов и 2334 неполных, немопилируемых файлов с веб-хостинга GitHub.

По результатам классификации можно сделать вывод, что данные методы могут применяться не только в «лабораторных» условиях, когда тестовая выборка генерируется на основе студенческих работ или результатов олимпиад по программированию, где решаются схожие задачи, ограниченные по времени и объему кода, но и в условиях реального мира.

Заключение

В ходе преддипломной практики были выполнены все поставленные задачи:

– проведен обзор актуальных на сегодняшний день информационных источников в области деанонимизации авторов программного обеспечения по его исходному коду;

– построена модель процесса определения авторства исходного кода;

– сформирован и обработан набор данных, состоящий из трех подвыборок, имеющих характерные особенности, которые оказывают влияние на точность классификации;

– произведена программная реализация разработанной модели и ее тестирование;

– разработан интерфейс на основе технологии Jupyter Notebook;

– проведены вычислительные эксперименты и анализ полученных результатов.

Итогом практики является разработанное программное обеспечение «WhoseCppCode» на языке программирования Python, предназначенное для сбора и обработки данных, классификации авторов исходного кода на языке C/C++, визуализации полученных результатов при помощи интерфейса. Основной программный модуль может быть использован отдельно при разработке различных систем, интерфейсов и программ, а также дальнейших научных исследований задачи деанонимизации авторов исходного кода.

Список использованных источников

1 Образовательный стандарт ВУЗа ОС ТУСУР 01-2013 [Электронный ресурс]. — Режим доступа: https://storage.tusur.ru/files/40668/rules_tech_01-2013.pdf, свободный (дата обращения: 15.12.2016).

2 Google C++ Style Guide [Электронный ресурс]. — Режим доступа: https://google.github.io/styleguide/cppguide.html, свободный (дата обращения: 28.04.2017).

3 C++ Programming Style Guidelines [Электронный ресурс]. — Режим доступа: http://geosoft.no/development/cppstyle.html, свободный (дата обращения: 28.04.2017).

4 Кафедра комплексной информационной безопасности электронно-вычислительных систем [Электронный ресурс]. — Режим доступа: доступа: http://kibevs.tusur.ru/pages/kafedra/index, свободный (дата обращения: 01.05.2017).

5 Source Code Authorship Analysis For Supporting The Cybercrime Investigation Process. Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis [Электронный ресурс]. — Режим доступа: http://www.icsd.aegean.gr/lecturers/stamatatos/papers/ICETE2005.pdf, свободный (дата обращения: 17.02.2017).

6 Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method. Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis [Электронный ресурс]. — Режим доступа: https://www.semanticscholar.org/paper/Identifying-Authorship-by-Byte-Level-N-Grams-The-Frantzeskou-Stamatatos/3b2531ea2685b9fb9abf071d119974ac3405874d, свободный (дата обращения: 15.02.2017).

7 Using Classification Techniques to Determine Source Code Authorship. Brian N. Pellin Computer Sciences Department University of Wisconsin [Электронный ресурс]. — Режим доступа: https://pdfs.semanticscholar.org/f9aa/790191a50bed02a877e1696c7bb71ea9f33a.pdf, свободный (дата обращения: 25.02.2017).

8 Source Code Authorship Attribution using n-grams. Steven Burrows, S.M.M. Tahaghoghi [Электронный ресурс]. — Режим доступа: https://pdfs.semanticscholar.org/79a2/1998c2f0afe2c616c01d590d6d0f6e16e9eb.pdf, свободный (дата обращения: 23.02.2017).

9 Source Code Authorship Attribution. Steven Burrows [Электронный ресурс]. — Режим доступа: http://researchbank.rmit.edu.au/eserv/rmit:10828/Burrows.pdf, свободный (дата обращения: 23.02.2017).

10 Application of information retrieval techniques for source code authorship attribution. S. Burrows, A. Uitdenbogerd, T. Urpin [Электронный ресурс]. — Режим доступа:

https://www.researchgate.net/publication/220787332_Application_of_Information_Retrieval_Techniques_for_Source_Code_Authorship_Attribution, свободный (дата обращения: 23.02.2017).

11 De-anonymizing Programmers via Code Stylometry. Aylin Caliskan-Islam, Drexel University; Richard Harang, U.S. Army Research Laboratory; Andrew Liu, University of Maryland; Arvind Narayanan, Princeton University; Clare Voss, U.S. Army Research Laboratory; Fabian Yamaguchi, University of Goettingen; Rachel Greenstadt, Drexel University [Электронный ресурс]. — Режим доступа: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-caliskan-islam.pdf, свободный (дата обращения: 18.02.2017).

12 Random Forest. Applied Multivariate Statistics — Spring 2012 [Электронный ресурс]. — Режим доступа: http://stat.ethz.ch/education/semesters/ss2012/ams/slides/v10.2.pdf, свободный (дата обращения: 17.02.2017).

13 Git Blame Who?: Stylistic Authorship Attribution of Small, Incomplete Source Code Fragments. Edwin Dauber, Aylin Caliskan, Richard Harang, Rachel Greenstadt [Электронный ресурс]. — Режим доступа: https://arxiv.org/abs/1701.05681, свободный (дата обращения: 10.03.2017).

14 Перекрёстная проверка [Электронный ресурс]. — Режим доступа: https://ru.wikipedia.org/wiki/, свободный (дата обращения: 01.05.2017).

15 Хэзфилд Р. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста / Р. Хэзфилд, Л. Кирби. — М: . ДиаСофт, 2001. – 736 с.

16 Макросы (C/C++) [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/503x3e3s.aspx, свободный (дата обращения: 23.02.2017).

17 Ключевые слова C++ [Электронный ресурс]. — Режим доступа: http://ru.cppreference.com/w/cpp/keyword, свободный (дата обращения: 12.03.2017).

18 Методы классификации и прогнозирования. Деревья решений. ИНТУИТ [Электронный ресурс]. — Режим доступа: http://www.intuit.ru/studies/courses/6/6/lecture/1740, свободный (дата обращения: 01.05.2017).

19 Алгоритм AdaBoost [Электронный ресурс]. — Режим доступа: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost, свободный (дата обращения: 27.04.2017).

20 Extremely randomized trees. Pierre Geurts, Damien Ernst, Louis Wehenkel [Электронный ресурс]. — Режим доступа: https://pdfs.semanticscholar.org/336a/165c17c9c56160d332b9f4a2b403fccbdbfb.pdf, свободный (дата обращения: 28.04.2017).

21 Code Jam Language Stats [Электронный ресурс]. — Режим доступа: https://www.go-hero.net/jam/16, свободный (дата обращения: 10.02.2017).

22 GitHub [Электронный ресурс]. — Режим доступа: https://github.com/, свободный (дата обращения: 10.05.2017).

23 GitHub Dominates the Forges [Электронный ресурс]. — Режим доступа: https://github.com/blog/865-github-dominates-the-forges, свободный (дата обращения: 10.05.2017).

24 The Jupyter Notebook [Электронный ресурс]. — Режим доступа: http://jupyter.org/, свободный (дата обращения: 02.04.2017).

25 Scikit-learn — Machine Learning in Python [Электронный ресурс]. — Режим доступа: http://scikit-learn.org/stable/, свободный (дата обращения: 02.03.2017).

26 Plotly Python Library [Электронный ресурс]. — Режим доступа: https://plot.ly/python/, свободный (дата обращения: 25.02.2017).

27 NumPy [Электронный ресурс]. — Режим доступа: http://www.numpy.org/, свободный (дата обращения: 25.02.2017).

28 SciPy [Электронный ресурс]. — Режим доступа: https://www.scipy.org/, свободный (дата обращения: 25.02.2017).

29 Python Data Analysis Library [Электронный ресурс]. — Режим доступа: http://pandas.pydata.org/, свободный (дата обращения: 25.02.2017).

30 ipywidgets: Interactive HTML Widgets [Электронный ресурс]. — Режим доступа: https://github.com/jupyter-widgets/ipywidgets, свободный (дата обращения: 09.03.2017).

31 Оценка классификатора (точность, полнота, F-мера) [Электронный ресурс]. — Режим доступа: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html, свободный (дата обращения: 17.02.2017).

Приложение А

(Обязательное)

Компакт-диск

Компакт-диск содержит:

– электронную версию пояснительной записки в форматах *.tex и *.pdf;

– итоговую презентацию результатов работы в форматах *.pptx и *.pdf;

– актуальную версию программы, реализованную на языке программирования Python, для определения авторсва исходного кода программ на языке C/C++;

– тестовые данные для работы с программой.

Приложение Б

(Справочное)

Сравнительный обзор информационных источников

Приложение Б

(Справочное)

Сравнительный обзор информационных источников

Таблица 2.1 — Обзор источников

Название работы	Авторы, год публикации	Методы, использованные в работе	Описание данных	Достигнутая точность классификации	Язык программирования
Using classification techniques to determine source code authorship [7]	B. Pellin, 2008	ACT, SVM	4 схожие программы, 2 автора	67 — 88 %	Java
Source code authorship attribution using n-grams [8]	S. Burrows, S. Tahaghoghi, 2007	N-граммы	Выборка из 1640 файлов исходного кода и 100 авторов	67 %	C
Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method [6]	G. Frantzeskou, E. Stamatatos, S. Gritzalis, 2007	Составление профиля программиста на основе статистических метрик, подсчет отклонения от профиля	Не указано	88 % для C++, 100 % для Java	Java, C++
Application of information retrieval techniques for source code authorship attribution [10]	S. Burrows, A. Uitdenbogerd, T. Urpin, 2009	N-граммы, рейтинговые схемы	100 авторов, классифицировались по 10, 1579 программных файлов	77 %	C

Продолжение таблицы 2.1

Название работы	Авторы, год публикации	Методы, использованные в работе	Описание данных	Достигнутая точность классификации	Язык программи-рования
De-anonymizing Programmers via Code Stylometry [11]	A. Caliskan-Islam, R. Harang, A. Liu, F. Yamaguchi, 2015	Статистический подсчет признаков, нечеткие АСТ	250 авторов, 1600 файлов	94 — 98 %	C/C++, Python
Git Blame Who?: Stylistic Authorship Attribution of Small, Incomplete Source Code Fragments [13]	A. Caliskan-Islam, E. Dau-ber, R. Harang, R. Greenstadt, 2017	Калибровочные кривые, нечеткие АСТ, классифика-тор Random Forest	Некомпили-руемые неполные образцы кода с ресурса GitHub	70 — 100 %	C/C++

Приложение В

(Справочное)

Описание стилистических признаков

Приложение В

(Справочное)

Описание лексических признаков

Таблица 3.1.1 — Описание различных лексических признаков

Группа признаков	Признак	Обозначение	Определение
Стиль комментирования	Число однострочных комментариев	ln_inline_ comments	Натуральный логарифм отношения числа однострочных комментариев к длине файла в символах
	Число многострочных комментариев	ln_multiline_ comments	Натуральный логарифм отношения числа многострочных комментариев к длине файла в символах
	Число комментариев	ln_comments	Натуральный логарифм отношения числа комментариев к длине файла в символах
Стиль разметки	Число пробелов	ln_spaces	Натуральный логарифм отношения числа пробелов к длине файла в символах
	Число символов табуляции	ln_tabs	Натуральный логарифм отношения числа символов табуляции к длине файла в символах
	Число переводов строки	ln_newlines	Натуральный логарифм отношения числа переводов строки к длине файла в символах
	Коэффициент пробельных символов	whitespace_ ratio	Натуральный логарифм отношения суммы всех пробельных символов (пробелов, символов табуляции, переводов строки) к длине файла в символах
Стиль расстановки фигурных скобок	Число одиночных раскрывающихся скобок	ln_open_brace_ alone	Натуральный логарифм отношения числа раскрывающихся скобок, одиночных в строке, к длине файла в символах

Продолжение таблицы 3.1.1

Группа признаков	Признак	Обозначение	Определение
Стиль расстановки фигурных скобок	Число раскрывающихся скобок, первых в строке	ln_open_ brace_first	Натуральный логарифм отношения числа раскрывающихся скобок, после которых следует код, к длине файла в символах
	Число раскрывающихся скобок, последних в строке	ln_open_ brace_last	Натуральный логарифм отношения числа раскрывающихся скобок, которым предшествует код, к длине файла в символах
	Число закрывающихся скобок, одиночных в строке	ln_closing_ brace_alone	Натуральный логарифм отношения числа зарывающихся скобок, одиночных в строке, к длине файла в символах
	Число закрывающихся скобок, первых в строке	ln_closing_ brace_first	Натуральный логарифм отношения числа закрывающихся скобок, после которых следует код, к длине файла в символах
Дополнительные признаки	Число закрывающихся скобок, последних в строке	ln_closing_ brace_last	Натуральный логарифм отношения числа закрывающихся скобок, которым предшествует код, к длине файла в символах
	Число макросов	ln_macros	Натуральный логарифм отношения числа макросов к длине файла в символах
	Число строк кода	lines_of_code	Число строк кода, не включающее пустые строки