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)

Определение авторства исходного кода

ОТЧЕТ

по преддипломной практике

Студентка гр. 722

_____ Мейта М.В.

«_____»_____2017г.

СОГЛАСОВАНО

Руководитель практики
от предприятия
к.т.н., доцент каф. КИБЭВС
_____ Давыдова Е.М.
«_____»_____2017г.

Руководитель практики
к.т.н., доцент каф. БИС
_____ Романов А.С.
«_____»_____2017г.

Томск 2017

РЕФЕРАТ

Отчет содержит 22 страниц, 1 рисунков, 7 таблиц, 16 источников, 1 приложение.

СТИЛОМЕТРИЯ, ИСХОДНЫЙ КОД, ДЕАНОНИМИЗАЦИЯ АВТОРА, C++, КЛАССИФИКАЦИЯ, PYTHON, SKLEARN, RANDOM FOREST CLASSIFIER, LATEX.

Цель работы — разработка алгоритма для определения авторства программного обеспечения, основанного на стилометрическом анализе исходного кода программ на языках высокого уровня.

В рамках научно-исследовательской работы на текущий семестр были поставленны следующие задачи:

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

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

– создание программной реализации разработанного алгоритма;

– исследование эффективности разработанного алгоритма анализа исходных кодов.

Объект исследования: деанонимизация автора программного обеспечения.

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

В результате работы было выполнено слудующее:

– произведен аналитический обзор существующих методов анализа исходного кода программ с целью деанонимизации автора;

– выбран набор признаков для классификации авторов программного кода на языке C++;

– в качестве алгоритма классификации выбран Random Forest Classifier;

– разработан алгоритм определения авторства исходного кода программ на языке C++;

– произведена программная реализация разработанного алгоритма на языке программирования высокого уровня Python;

– подготовлен тестовый набор данных;

– выбраны критерии оценки эффективности разработанного алгоритма;

– произведены вычислительные эксперименты на данном наборе данных;

– сделаны некоторые выводы на основе полученных результатов.

Отчет о НИР выполнен согласно ОС ТУСУР 01-2013 [1] при помощи системы компьютерной вёрстки LaTeX.

Содержание

Ведение

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.

Целью данной работы является исследование методов стилометрии — статистического анализа текста для выявления его стилистических особенностей, а также методов машинного обучения для решения задачи деанонимизации разработчика по исходному коду программного обеспечения.

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.

1 Кафедра КИБЭВС

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

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 на сегодняшний день является ректор ТУСУРа, Александр Александрович Шелупанов, лауреат премии Правительства Российской Федерации, действительный член Международной Академииnнаук 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.

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 [1].

Официальный сайт КИБЭВС [Электронный ресурс]. – Режим доступа: http://kibevs.tusur.ru/pages/kafedra/index (дата обращения:

2 Обзор информационных источников

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

В работе [2] 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.

Среди проблем (задач) анализа авторства исходного кода выделены:

– 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 [3] предложен способ определения авторства программного обеспечения. в основе которого лежит система, состоящая из 100 метрик, отражающих «почерк создателя» программного обеспечения. На основе метрик составлен «профиль почерка» пяти разных программистов по текстам трех разработанных ими программных систем и проверено соответствие этому профилю других программ, написанных в том числе и другими программистами. Однако авторы привели сомнительные результаты вычислений, не указали способ составления «профиля почерка» и полученную точность, с которой программная система определяла авторство.

В [4] исходный код транслировался в абстрактные синтаксические деревья, после чего разбивался 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 статье [5] 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. Производилось ранжирование документов по схожести, после чего производилась оценка результатов. При этом составителям удалось успешно определить плагиат в 67% случаев.

В [6] применялся алгоритм классификации Random Forest [7] 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++.

По результатам анализа вышеперечисленных источников было принято решение использовать подход, основанный на построении абстрактных синтаксических деревьев и классификации при помощи алгоритма Random Forest.

2.1 Новый (подробный) обзор источников

3 Выбор набора признаков, характеризующих автора программы

Идентификация авторов исходного кода производилась на основании различных признаков, которые каким-либо образом могут идентифицировать так называемый стиль написания программ, присущий конкретному автору. Помимо деанонимизации автора, стиль написания кода может дать представление об уровне его квалификации, образовании, личных и профессиональных предпочтениях, а также некоторых других особенностях, которые могут представлять интерес в различного рода исследованиях.

Перечисленные в данном разделе признаки 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.

3.1 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 (табл. 3.1) — данная подгруппа определяет преобладающий в тексте вид комментариев (однострочные или многострочные), а также общее их количество.

– Стиль расстановки фигурных скобок (табл. 3.4) — к наиболее известным относят «K&R», «Whitesmith», «One True Bracing Style», стиль Алмена и другие. [8]

– Стиль разметки (табл. 3.5) — расстановка пробелов, табуляций, число переносов строки к общей длине файла.

Дополнительно вычисляются (табл. 3.3):

– Число макросов — использует ли программист директивы препроцессора.[9]

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

В таблице 3.1 приведено описание использованных при классификации авторов признаков, а также формулы для их вычисления.

Таблица 3.1 – Стиль комментирования

Стиль комментирования		
Признак	Обозначение	Определение
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

Таблица 3.2 – Стиль разметки

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

Таблица 3.3 – Дополнительные признаки

Дополнительные признаки		
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

Таблица 3.4 – Стиль расстановки фигурных скобок

Стиль расстановки фигурных скобок		
Число раскрывающихся скобок, одиночных в строке	ln_open_brace_alone	Натуральный логарифм отношения числа раскрывающихся скобок, одиночных в строке, к длине файла в символах
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_first	Натуральный логарифм отношения числа раскрывающихся скобок, которым предшествует код, к длине файла в символах
Число закрывающихся скобок, одиночных в строке	ln_open_brace_alone	Натуральный логарифм отношения числа закрывающихся скобок, одиночных в строке, к длине файла в символах
Число закрывающихся скобок, первых в строке	ln_open_brace_first	Натуральный логарифм отношения числа закрывающихся скобок, после которых следует код, к длине файла в символах
Число закрывающихся скобок, последних в строке	ln_open_brace_first	Натуральный логарифм отношения числа закрывающихся скобок, которым предшествует код, к длине файла в символах

3.2 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++ используются 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 [10] и представлен в таблице 3.5.

Таблица 3.5 – 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

4 Классификация

4.1 Random Forest

В качестве алгоритма классификации в данной работе был использован Random Forest Classifier, который 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. 4.1);

– 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. [7]

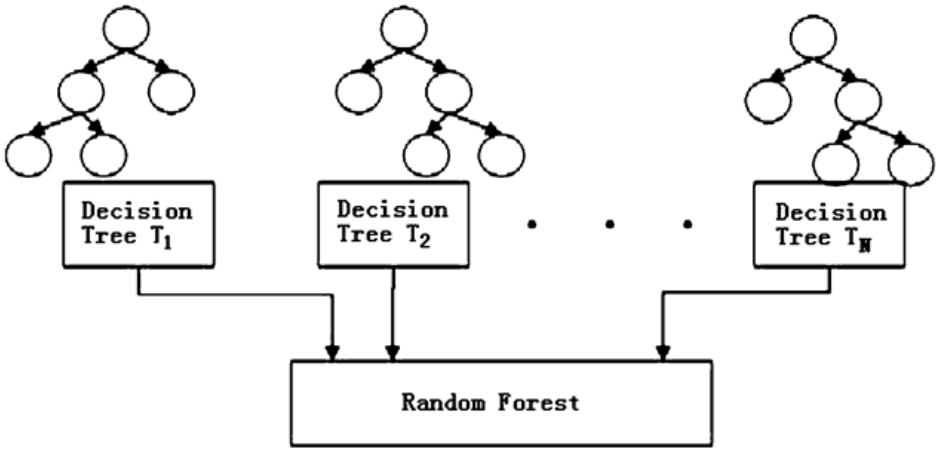

Рисунок 4.1 – Random Forest Classifier

4.2 AdaBoost

4.3 ExtraTrees

5 Инструменты разработки

В качестве инструментов разработки были выбраны:

1) интерпретируемый высокоуровневый язык программирования Python версии 3.5.2; [11]

2) CastXML — синтаксический анализатор программ на языке C/C++ для построения абстрактных синтаксических деревьев в формате XML; [12]

3) pygccxml — библиотека на языке программирования Python для чтения XML-вывода CastXML (или ранней версии — GCCXML); [13]

4) scikit-learn — свободно распространяемая Python-библиотека для машинного обучения; [14]

5) matplotlib — библиотека на языке программирования Python для визуализации данных; [15]

6) интегрированная среда разработки PyCharm;

7) операционная система Linux Ubuntu 16.10.

6 Тестовые данные

Согласно статье [6], обучение классификатора необходимо производить на программных файлах, решающих схожие задачи, для повышения точности классификации. Данное условие необходимо для того, чтобы раделять файлы исходного кода по стилю программирования, а не по функциональному назначению.

В качестве тестовых данных для обучения и классификации использовались лабораторные работы трех студентов первого курса обучения по предмету «Основы программирования». При этом каждым студентом было выполнено 7 лабораторных работ, в каждой из которых по 2 задачи на программирование на языке C/C++. Таким образом, выборка состояла из 14 примеров (файлов исходного кода) на каждого из трех объектов (авторов), что в общей сложности составило 42 файла формата *.cpp.

Тестовая и обучающая выборки генерировались случайным образом без повторений из начальной выборки, описанной выше. При этом четверть всех примеров использовалась для тестирования (классификации).

7 Критерии оценки эффективности классификации

Критерии оценки работы классификатора представлены в таблице 7.1, 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. [16]

Таблица 7.1 – 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

8 Результаты классификации

В ходе научно-исследовательской работы был построен классификатор, позволяющий определить автора исходного кода программы с точностью около 73%. При тестировании классификатора использовались критерии оценки, описанные в разделе 7, а также время работы программы. Результаты работы классификатора представлены в таблице 8.1.

Таблица 8.1 – Результаты работы классификатора

	Число признаков	Accuracy	Precision	Recall	F1-score	Время работы
1	21	54,55 %	0,58	0,55	0,53	43,74 с
2	19	45,45 %	0,56	0,45	0,48	51,75 с
3	22	90,91 %	0,94	0,91	0,91	43,70 с
4	24	72,73 %	0,84	0,73	0,72	43,85 с
5	23	72,73 %	0,89	0,73	0,74	44,74 с
6	25	90,91 %	0,93	0,91	0,91	52,05 с
7	20	90,91 %	0,94	0,91	0,91	44,17 с
8	26	63,67 %	0,73	0,64	0,64	44,87 с
9	20	81,82 %	0,82	0,82	0,82	43,73 с
10	20	63,64 %	0,76	0,64	0,61	44,31 с
Ср. знач.	22	72,73 %	0,80	0,80	0,73	45,69 с

Заключение

По результатам, полученным в ходе научно-исследовательской работы, можно сделать следующие выводы:

1) Тестовые данные (см. раздел 6) имели определенные недостатки, которые в конечном итоге привели к снижению точности классификации:

– программы, на которых производилось обучение и тестирование классификатора, были написаны студентами в ходе изучения основ программирования, т.е. у авторов в выборке отсутствовал опыт программирования на C/C++;

– среднее количество строк кода на файл составило всего 45 строк;

– в основном примеры содержали множество конструкций ввода-вывода и простые расчеты (например, площадей различных геометрических фигур);

– задания выполнялись по примерам из методического обеспечения, что повлияло на предпочтение использования тех или иных конструкций языка.

2) Не все признаки, используемые в классификации, равнозначны. Так, например, частоты ключевых слов «int» или «float» (см. раздел 3.2) не так важны для определения стилистических особенностей написания программы, как, к примеру, определенные предпочтения автора при назначении идентификаторов.

3) Оптимальные параметры для задач классификации с помощью алгоритма Random Forest (см. раздел 4.1) были подобраны эмпирически (см. раздел **??**) и соответствовали рекомендациям из различных источников.

В ходе работы был построен классификатор, позволяющий отнести исходный код к конктретному автору с точностью около 73%.

Среди задач на будущее можно выделить следующие:

– расширение набора признаков, а также расчет (с использованием экспертной оценки) веса для каждого из них (см. раздел 3) для повышения точности классификации;

– подбор тестовых данных более сложных программ для большего количества авторов; при этом желательно, чтобы авторы программ имели некоторый опыт программирования на языке C/C++;

– исследование набора признаков для программ на других языках высокого уровня;

– исследование алгоритмов обнаружения плагиата в исходных кодах программ.

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 (дата обращения: 15.12.2016).

2 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 (дата обращения: 23.10.2016).

3 Маевский Д.А. Определение авторства программного обеспечения по исходному коду программ [Электронный ресурс]. — Режим доступа: http://www.khai.edu/csp/nauchportal/Arhiv/REKS/2014/REKS614/Maevsky.pdf (дата обращения: 17.12.2016).

4 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 (дата обращения: 23.10.2016).

5 Source Code Authorship Attribution using n-grams. Steven Burrows, S.M.M. Tahaghoghi [Электронный ресурс]. — Режим доступа: https://pdfs.semanticscholar.org/79a2/1998c2f0afe2c616c01d590d6d0f6e16e9eb.pdf (дата обращения: 23.10.2016).

6 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 (дата обращения: 07.09.2016).

7 Random Forest. Applied Multivariate Statistics — Spring 2012 [Электронный ресурс]. — Режим доступа: http://stat.ethz.ch/education/semesters/ss2012/ams/slides/v10.2.pdf (дата обращения: 23.10.2016).

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

9 Макросы (C/C++) [Электронный ресурс]. — Режим доступа: https://msdn.microsoft.com/ru-ru/library/503x3e3s.aspx (дата обращения: 23.02.2016).

10 Ключевые слова C++ [Электронный ресурс]. — Режим доступа: http://ru.cppreference.com/w/cpp/keyword (дата обращения: 12.10.2016).

11 Python 3.5.2 [Электронный ресурс]. — Режим доступа: https://www.python.org/ (дата обращения: 25.10.2016).

12 CastXML C-family Abstract Syntax Tree XML Output [Электронный ресурс]. — Режим доступа: https://github.com/CastXML/CastXML (дата обращения: 25.10.2016).

13 Pygccxml — pygccxml 1.8.3 documentation [Электронный ресурс]. — Режим доступа: http://pygccxml.readthedocs.io/en/develop/ (дата обращения: 25.10.2016).

14 Scikit-learn — Machine Learning in Python [Электронный ресурс]. — Режим доступа: http://scikit-learn.org/stable/ (дата обращения: 25.10.2016).

15 Matplotlib: Python Plotting [Электронный ресурс]. — Режим доступа: http://matplotlib.org/ (дата обращения: 25.10.2016).

16 Оценка классификатора (точность, полнота, F-мера) [Электронный ресурс]. — Режим доступа: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html (дата обращения: 17.12.2016).

Приложение А

(Обязательное)

Компакт-диск

Компакт-диск содержит:

– электронную версию пояснительной записки в форматах *.tex 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.