УДК 004.89

А.В. Куртукова, А.С. Романов, А.М. Федотова, А.А. Шелупанов

Архитектура интеллектуальной системы для идентификации автора исходного кода

Разработка решений, предназначенных для идентификации автора исходного кода, является актуальной для области защиты интеллектуальной собственности и авторского права. Особую важность такие решения представляют для анализа исходных кодов промышленных программных продуктов, как правило, разрабатываемых на базе узкоспециализированных научных исследований. В данной статье представлена архитектура интеллектуальной системы, предназначенной для идентификации автора исходного кода на основе глубокой гибридной нейронной сети. Описаны основные структурные элементы разрабатываемой системы и их интерфейсы, а также механизмы, необходимые для ее корректного и быстрого функционирования.

Ключевые слова: программная система, исходный код, машинное обучение, автор, глубокие нейронные сети.
DOI: 10.21293/1818-0442-2022-25-3-39-44

Актуальность проблемы идентификации автора исходного кода программы обусловлена активной цифровизацией всевозможных сфер жизни людей: социальной, финансовой, производственной, образовательной и многих других. Одним из главных аспектов цифровазиации является применение современных инструментов, в том числе программного обеспечения (ПО). Подавляющее большинство таких инструментов ориентировано на промышленность, товарооборот, предпринимательство, военное дело и другие сферы деятельности, использующие узкоспециализированное ПО. Отсюда возникает необходимость в создании решений, предназначенных для защиты авторских прав и интеллектуальной собственности, объектами которых могут являться исходные коды программных продуктов. Особую ценность такие решения представляют для деятельности, осуществляемой в промышленных масштабах – оборонной, авиационной, продовольственной, фармацевтической и т.д. Это подтверждается тем, что промышленные программные решения являются исключительной собственностью, так как разрабатываются на основе ряда узконаправленных исследований и с соблюдением строгих качественных требований.

Несмотря на большое количество трудов, посвященных проблеме определения автора естественно-языкового текста и их программным реализациям [1], подобные решения для искусственно-языковых текстов (в том числе исходных кодов), почти полностью отсутствуют. Поэтому возникает необходимость в разработке простой, эффективной и полнофункциональной интеллектуальной системы для идентификации автора исходного кода программы.

Главной сложностью при разработке интеллектуальной системы является выбор подхода к решению задачи определения авторства исходного кода программы. Существует множество потенциально эффективных подходов [2], основывающихся как на классических (дискриминантный, кластерный анализы, энтропийное сжатие и др.), так и на сложных моделях машинного обучения, в том числе на нейронных сетях (НС) [3–7]. Для создания ПО, гарантирующего высокую точность решения поставленной задачи, следует учесть опыт создания подобных решений и проблемы, с которыми сталкивались разработчики.

Единственным прямым аналогом разрабатываемого решения для идентификации автора исходного кода является ASAP [8]. Данное ПО функционирует на основе SCAP (авторские профили исходных кодов) и метода Борроуса [9]. Оба подхода реализуют идентификацию автора исходного кода путем сопоставления анонимного исходного кода с образцами, хранящимися в базе ASAP. Следует отметить, что в первом случае сравнение производится со сгруппированными по профилям образцами, во втором – без какой-либо группировки. Непосредственно ASAP представляет собой простое десктопное приложение. Данный инструмент имеет ряд недостатков. Эксперименты с применением методов, используемых в ASAP, проводились только для Java и C++ – инструмент не является универсальным для всех языков программирования. Кроме того, используемые методы являются эффективными исключительно при наличии в хранилище ASAP большого количества экземпляров исходных кода (более 3 тыс. для каждого автора). Также к недостаткам ПО относятся сложность его установки и тонкя ручная настройка.

Веб-сервис JPlag [10] является косвенным аналогом разрабатываемой системы и позволяет осуществлять проверку на плагиат, а также выявлять неоднородности исходных кодов, написанных на Java, C++, C и Scheme. Детальный отчет по найденным сходствам формируется в виде HTML-страниц с таблицами. Существенным недостатком сервиса является ограниченный функционал – он не может применяться для идентификации автора исходного кода.

Таким образом, была поставлена цель – спроектировать архитектуру интеллектуальной системы для идентификации автора исходного кода на основе НС.

Требования к интеллектуальной системе для идентификации автора исходного кода

Интеллектуальная система должна анализировать исходные коды программ как сложные текстовые структуры с неотъемлемыми элементами, таки-

ми как комментарии и макросы, доли вклада каждого программиста и, в частности, разнородность данных, т.е. учитывать такие критические факторы, как

– количество языков. Существующие подходы и системы анализируют не более 4 языков программирования, что является их явным недостатком. Решение современных задач требует применения большего многообразия языков. Также привычки и стиль автора могут гибко переноситься с одного языка на другой в случаях, когда автор владеет двумя и более языками. Оптимальный подход, используемый в программной системе, должен это учитывать;

– опыт. С профессиональным ростом разработчик совершенствует свои навыки и процесс написания кода. Этот факт также важен, поэтому при анализе кода следует задействовать образцы из разных временных интервалов для одного и того же программиста;

– развитие команды. Обсуждение кода – стандартная практика коммерческой разработки, которая оказывает влияние на стиль написания кода внутри одной команды. Некоторые особенности кода меняются от одной команды к другой, а неявные и неконтролируемые программистом особенности – нет;

– творческая составляющая. У каждого разработчика есть свои предпочтения в использовании различных шаблонов и структур, поэтому написание программного кода является творческим процессом.

Важно учитывать, что исходный код не соответствует строгим правилам и изменяется в зависимости от личных предпочтений разработчика;

– продвинутые кейсы. В настоящее время коммиты, обфускация, стандарты кодирования, смешанные данные и сгенерированные исходные коды – важная часть процесса разработки ПО. Интеллектуальная система для анализа исходных кодов должна быть устойчива к сложным задачам. Создаваемая система должна соответствовать современным методам и инструментам разработки.

Система, учитывающая все обозначенные факторы, должна быть гибкой и полнофункциональной, т.е. исключающей любые дополнительные действия со стороны пользователя, помимо выбора данных для дальнейшего анализа. Кроме того, система не должна требовать установки и настройки вспомогательных дистрибутивов, обеспечивающих функционирование отдельно взятых модулей.

Архитектура интеллектуальной системы для идентификации автора исходного кода

При проектировании архитектуры программной системы необходимо учитывать ее наиболее важные структурные элементы, их интерфейсы, а также основные аспекты ее взаимодействия с пользователем и внешней средой. Модули архитектуры, представленной на рис. 1, можно сгруппировать по их функциональному назначению.

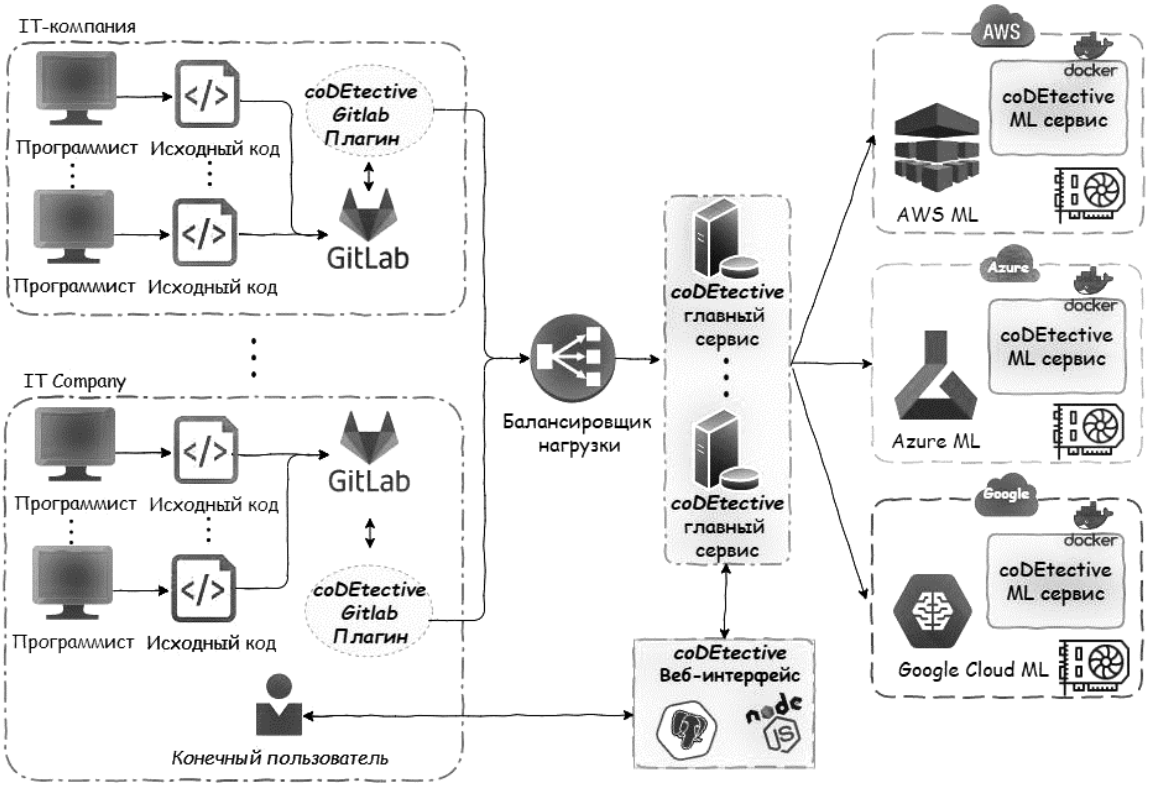

Рис. 1. Архитектура интеллектуальной системы

Модули сбора, агрегации и преобразования исходных кодов используют специальный разработанный плагин для системы контроля версий GitLab [11], предустановленной внутри компании. Так, осуществляется непрерывное получение последних версий исходных кодов и их последующего преобразования в векторный вид методом прямого кодирования (one-hot encoding).

Модуль взаимодействия с пользователем позволяет корректировать процесс работы интеллектуальной системы в зависимости от запросов и команд пользователя, передаваемых через веб-интерфейс. Кроме того, здесь же осуществляется процедура преобразования анонимного образца исходного кода, подлежащего идентификации, в векторный, аналогичный образцам с GitLab вид.

Главный сервис – модуль интеллектуальной системы, предназначенный для формирования из полученных векторизованных данных тренировочных и тестовых наборов, а также конфигурирования НС в соответствии с пользовательскими или базовыми настройками. Авторская архитектура НС, применяемая в системе, представлена на рис. 2.

Архитектура представляет собой гибрид из сверточных и рекуррентных сетей, что позволяет ей эффективно выявлять как временные зависимости, так и локальные информативные авторские признаки.

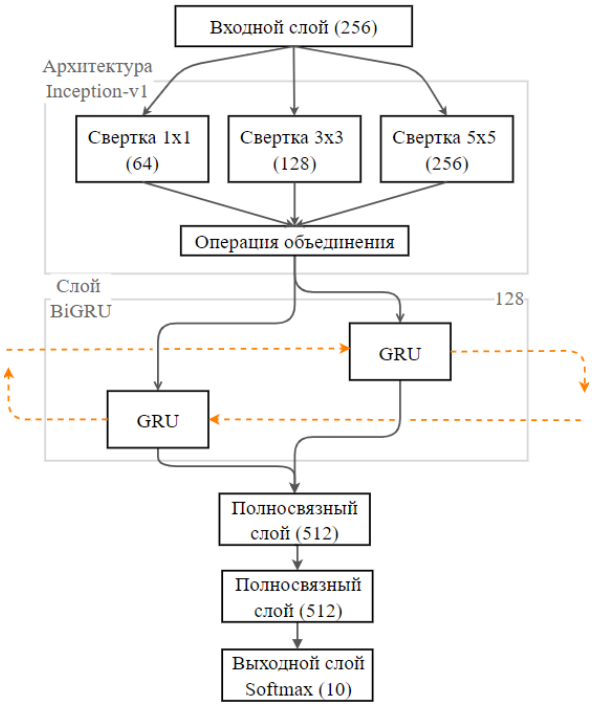

Рис. 2. Архитектура авторской НС

Архитектура состоит из следующих слоев:

1. Входной слой. Размерность входного слоя соответствует длине векторов, полученных от подсистемы анализа текстов.

2. Связка слоев Inception-v1. Данная связка включает в себя сверточные слои разных размерностей с фильтрами от 1 до 5. Вход каждой из сверток отличен по размеру от двух других. Таким образом, свертки образуют «бутылочное горлышко», через которое пропускают только самые информативные из исходных признаков. По завершению работы сверток их результаты объединяются в единый выход.

3. Двунаправленные управляемые рекуррентные блоки (BiGRU). Полученный от сверточной части выход поступает на вход рекуррентной сети со входной размерностью в 128. Текст анализируется в прямой и обратной последовательности.

4. Слои с прямой связью. Результат работы рекуррентной сети передается на вход двух последовательных слоев с прямой связью. Оба слоя имеют размерность в 512, за счет чего происходит масштабирование НС.

5. Выходной слой. В качестве выходного слоя используется Softmax. Данный слой позволяет получить распределение вероятности о принадлежности входного текста каждому из классов. Размерность слоя – 10, что применимо для 10 классов. Размерность напрямую зависит от количества классов предсказания для конкретной задачи.

Результатом работы этого модуля является передача конфигурации процесса обучения, архитектуры НС и наборов тренировочных данных на платформу некоторого крупного вендора облачных вычислений. Отличительной особенностью процесса обучения НС на платформе является его осуществление внутри docker-контейнера. Такое решение позволяет избежать необходимости в адаптации НС под фреймворк, используемый конкретным вендором. Центральная часть способна масштабироваться в зависимости от имеющихся ресурсов и требуемых объемов вычислений, обеспечивая оптимальную нагрузку.

Постановка эксперимента и результаты

С целью применения интеллектуальной системы для решения реальных практических задач необходимо было осуществить валидацию авторской НС. С ресурса GitHub был осуществлен сбор базы исходных кодов, содержащей свыше 250 тыс. образцов, написанных на 13 языках программирования. Данная база была сформирована с учетом критических факторов, встречающихся при решении реальных задач идентификации автора исходного кода программы.

В рамках эксперимента были сформированы корпуса образцов исходных кодов 5, 10 и 20 авторов-программистов. Для каждого автора производился отбор не более 20 образцов исходных кодов длиной не более 3000 символов.

Результаты экспериментов и общая информация о базе исходных кодов представлены в табл. 1. Оценка точности НС осуществлялась при помощи перекрестной проверки по 10 блокам. Для объективности заключения об эффективности авторской НС было решено осуществить дополнительные эксперименты. Данные эксперименты направлены на оценку точности авторской НС в сравнении с двумя наиболее популярными у исследователей методами идентификации автора исходного кода программы: BERT [12] и fastText [13]. Параметры обеих моделей были установлены в соответствии с рекомендациями [13]. Для оценки были взяты образцы авторов, программирующих на языке Java. Для каждого автора использовалось по 20 файлов для обучения. Результаты экспериментов представлены в табл. 2.

Таблица 1

Информация о данных и результаты экспериментов с различными корпусами

Язык	Общая информация о данных		Результаты экспериментов для корпусов из 5–20 авторов		
	Кол-во образцов	Кол-во авторов	5	10	20
C++	12366	72	0,92	0,92	0,90
Java	39708	73	0,97	0,93	0,89
JS	18375	69	0,92	0,82	0,76
Python	16783	57	0,95	0,92	0,92
C	17274	62	0,96	0,95	0,94
C#	19378	71	0,96	0,88	0,83
Ruby	19150	58	0,95	0,82	0,77
PHP	17158	80	0,92	0,89	0,86
Swift	12672	74	0,98	0,94	0,89
Go	14067	81	0,93	0,86	0,83
Groovy	14002	68	0,99	0,96	0,93
Kotlin	15274	72	0,91	0,85	0,81
Perl	11189	61	0,96	0,91	0,87

Таблица 2

Результаты экспериментов с различными методами

Метод	Кол-во авторов		
	5	10	20
HNN	0,97	0,93	0,89
BERT	0,94	0,90	0,87
fastText	0,93	0,86	0,83

Для того чтобы убедиться в наличии статистически значимой разницы между HNN, BERT и fastText, использовались апостериорные тесты Фридмана и Неменьи. Тесты был применены к результатам кроссвалидации трех моделей. Нулевая гипотеза – разница между результатами моделей случайна. Альтернативная – разница между результатами моделей статистически значима. По результатам расчетов *p*-значение не превысило порог в 0,05, а нулевая гипотеза была отвергнута. Статистический тест подтвердил значимость разницы между результатами. Считается, что эффективность сравниваемых моделей значительно различается, если средние ранги моделей отличаются на величину критической разницы (CD) и более. Для того чтобы графически изобразить результат теста, была построена диаграмма значимости Демшара (рис. 3). Эта диаграмма демонстрирует различия в точности пар методов. Если разница между средними рангами пары методов меньше расчетной CD, то разница в их эффективности также незначительна и представляется горизонтальной линией на диаграмме.

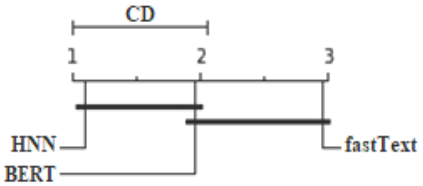

Рис. 3. Диаграмма значимости Демшара

Согласно диаграмме, HNN является наиболее точной. BERT демонстрирует незначительную разницу в эффективности по сравнению с HNN и с fastText. Однако fastText имеет низший ранг, поэтому BERT считается менее эффективным.

Полученные точности идентификации автора исходного кода позволяют сделать вывод о высокой эффективности применяемой НС и ее конкурентоспособности в сравнении с иными современными методами. HNN обеспечивает получение высокой точности для каждого из языков программирования в то время, как методы конкурентов адаптированы под 2–3 наиболее популярных языка. Также она учитывает изменения стиля программиста в результате улучшения его навыков и получения опыта командной разработки.

Заключение

В рамках данной работы была спроектирована архитектура интеллектуальной системы для идентификации автора исходного кода на основе НС, представлены ее основные структурные элементы, их интерфейсы и механизмы для быстрого и эффективного функционирования.

Результаты экспериментов свидетельствуют о возможности применения системы для решения реальных задач идентификации автора исходного кода программы. Авторская НС продемонстрировала точность идентификации более 90% для 13 различных языков программирования, что является наилучшим результатом среди современных решений. Тесты Фридмана и Неменьи подтвердили статистическую значимость полученных результатов.

В дальнейшем планируется тестирование разработанной интеллектуальной системы в целом, а также оценка ее эффективности на практике.

Работа выполнена при финансовой поддержке Министерства науки и высшего образования РФ в рамках базовой части государственного задания ТУСУРа на 2020–2022 гг. (проект № FEWM-2020-0037).

Литература

1. Романов А.С. Обобщенная методика идентификации автора неизвестного текста / А.С. Романов, А.А. Шелупанов, С.С. Бондарчук // Доклады ТУСУР. – 2010. – № 1(21). – С. 108–112.

2. Куртукова А.В. Идентификация автора исходного кода методами машинного обучения / А.В. Куртукова, А.С. Романов // Труды СПИИРАН. – 2019. – № 18(3). – С. 741–765.

3. Caliskan-Islam A. Deanonymizing programmers via code stylometry / A. Caliskan-Islam, R. Harang, A. Liu // Proceedings of the 24th USENIX Security Symposium, Washington. – 2015. – P. 255–270. – URL: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-caliskan-islam.pdf, свободный (дата обращения: 25.07.2022).

4. Caliskan-Islam A. Git Blame Who? Stylistic Authorship Attribution of Small, Incomplete Source Code Fragments / A. Caliskan-Islam, E. Dauber, R. Harang. – arXiv preprint. arXiv:1701.05681. 2019. – URL: https://arxiv.org/pdf/1701.05681.pdf, свободный (дата обращения: 25.07.2022).

5. Alsulami B. Source Code Authorship Attribution using Long Short-Term Memory Based Networks / B. Alsulami, E. Dauber, R. Harang, S. Mancoridis, R. Greenstadt // Proceedings of the 22nd European Symposium on Research in

Computer Security, Oslo, Norway, September 11–15. – 2017. – P. 65–82. – URL: https://www.cs.drexel.edu/~greenie/stylometry-esorics.pdf, свободный (дата обращения: 25.07.2022).

6. Kurtukova A. Source Code Authorship Identification Using Deep Neural Networks / A. Kurtukova, A. Romanov, A. Shelupanov. – Symmetry. – 2020. – No. 12. – URL: https://www.mdpi.com/2073-8994/12/12/2044/htm, свободный (дата обращения: 25.07.2022).

7. Wang N. Integration of Static and Dynamic Code Stylometry Analysis for Programmer De-anonymization / N. Wang, S. Ji // Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security. – 2018. – P. 74–84. – URL: https://dl.acm.org/doi/abs/10.1145/3270101.3270110, свободный (дата обращения: 25.07.2022).

8. Tennyson M.F. ASAP: A Source Code Authorship Program // International Journal on Software Tools for Technology Transfer. – 2019. – Vol. 21. – P. 471–484.

9. Burrows S. Source code authorship attribution using *n*-grams / S. Burrows, S. Tahaghoghi // Proceedings of the 12th Australasian Document Computing Symposium, Melbourne, Australia, RMIT University. – 2007. – P. 32–39. – URL: https://www.icsd.aegean.gr/publication_files/982250907.pdf, свободный (дата обращения: 25.07.2022).

10. JPlag [Электронный ресурс]. – URL: https://jplag.ipd.kit.edu/, свободный (дата обращения: 25.07.2022).

11. GitLab [Электронный ресурс]. – URL: https://gitlab.com/, свободный (дата обращения: 25.07.2022).

12. BERT [Электронный ресурс]. – URL: https://gitlab.com/, свободный (дата обращения: 25.07.2022).

13. FastText [Электронный ресурс]. – URL: https://huggingface.co/docs/transformers/model_doc/bert, свободный (дата обращения: 25.07.2022).

Куртукова Анна Владимировна
Аспирант каф. комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС) Томского государственного университета систем управления и радиоэлектроники (ТУСУР)
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7-905-991-67-13
Эл. почта: av.kurtukova@gmail.com

Романов Александр Сергеевич
Канд. техн. наук, доцент каф. КИБЭВС ТУСУРа
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7 (382-2) 41-34-26
Эл. почта: alexx.romanov@gmail.com

Федотова Анастасия Михайловна
Студент каф. безопасности информационных систем (БИС) ТУСУРа
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7-923-444-41-25
Эл. почта: fedotova.a.747@e.tusur.ru

Шелупанов Александр Александрович
Д-р техн. наук, проф., президент ТУСУРа
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7 (382-2) 90-71-55
Эл. почта: saa@tusur.ru

Kurtukova A.V., Romanov A.S.,
Fedotova A.M., Shelupanov A.A.

Intelligent System Architecture for Identification of the Source Code Author

The development of solutions for identifying the source code author is relevant for the field of intellectual property and copyright protection. Such solutions are important for the analysis of the industrial software product source codes, which are developed on the basis of a large number of professional studies. This paper presents the architecture of the intelligent system for identifying the source code author based on a deep hybrid neural network. The paper also describes the main structural elements of the system being developed and their interfaces, as well as the mechanisms necessary for its correct and high performance.

Keywords: software system, source code, machine learning, author, deep neural networks.
DOI: 10.21293/1818-0442-2022-25-3-39-44

References

1. Romanov A.S., Shelupanov A.A., Bondarchuk S.S. Generalized authorship identification technique]. *Proceedings of TUSUR University*, 2010, vol. 1, no. 21. pp. 108–112 (in Russ.).

2. Kurtukova A.V., Romanov A.S. [Identification author of source code by machine learning methods]. *SPIIRAS Proceedings*, 2019, vol. 18, no. 3, pp. 741–765 (in Russ.).

3. Caliskan-Islam A., Harang R., Liu A. Deanonymizing programmers via code stylometry. *Proceedings of the 24th USENIX Security Symposium*, 2015, pp. 255–270. Available at: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-caliskan-islam.pdf, free (Accessed: July 25, 2022).

4. Git Blame Who? Stylistic Authorship Attribution of Small, Incomplete Source Code Fragments. arXiv preprint arXiv:1701.05681. 2017. Available at: https://arxiv.org/pdf/1701.05681.pdf, free (Accessed: July 25, 2022).

5. Alsulami B., Dauber E., Harang R., Mancoridis S., Greenstadt R. Source Code Authorship Attribution using Long Short-Term Memory Based Networks. *Proceedings of the 22nd European Symposium on Research in Computer Security*, Oslo, Norway, September 11-15, 2017. Part I. pp. 65–82. Available at: https://www.cs.drexel.edu/~greenie/stylometry-esorics.pdf, free (Accessed: July 25, 2022).

6. Kurtukova A., Romanov A., Shelupanov A. Source Code Authorship Identification Using Deep Neural Networks. *MDPI Symmetry*, 2020, no. 12. Available at: https://www.mdpi.com/2073-8994/12/12/2044, free (Accessed: July 25, 2022).

7. Wang N., Ji S. Integration of Static and Dynamic Code Stylometry Analysis for Programmer De-anonymization. *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, 2018. pp. 74–84. Available at: https://dl.acm.org/doi/abs/10.1145/3270101.3270110, free (Accessed: July 25, 2022).

8. Tennyson, M.F. ASAP: A Source Code Authorship Program. *International Journal on Software Tools for Technology Transfer*, 2019. Vol. 21. pp. 471–484.

9. Burrows, S., Tahaghoghi, S.: Source code authorship attribution using *n*-grams. *Proceedings of the 12-th Australasian Document Computing Symposium*, 2007. Available at: https://www.icsd.aegean.gr/publication_files/982250907.pdf, free (Accessed: July 25, 2022)

10. JPlag. Available at: https://jplag.ipd.kit.edu/, free (Accessed: July 25, 2022).

11. GitLab. Available at: https://gitlab.com/, free. (Accessed: July 25, 2022).

12. BERT. Available at: https://huggingface.co/docs/transformers/model_doc/bert, free (Accessed: July 25, 2022).

13. FastText. Available at: https://fasttext.cc/, free (Accessed: July 25, 2022).

Anna V. Kurtukova
Postgraduate student, Department of Complex Information Security of Electronic Computer Systems (KIBEVS), Tomsk State University of Control Systems and Radioelectronics (TUSUR)
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7-905-991-67-13
Email: av.kurtukova@gmail.com

Aleksandr S. Romanov
Candidate of Science in Engineering, KIBEVS, TUSUR
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7 (382-2) 41-34-26
Email: alexx.romanov@gmail.com

Anastasia M. Fedotova
Student, Department of Information System Security (BIS), TUSUR
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7-923-444-41-25
Email: fedotova.a.747@e.tusur.ru

Alexandr A. Shelupanov
Doctor of Science in Engineering, Professor, President of TUSUR University
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7 (382-2) 90-71-55
Email: saa@tusur.ru