УДК 004.89
DOI: 10.26102/2310-6018/2022.38.3.016

Идентификация автора исходного кода программы на основе неоднородных данных для решения задач кибербезопасности

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

Томский государственный университет систем управления и радиоэлектроники, Томск, Российская Федерация
av.kurtukova@gmail.com✉

Резюме. Статья посвящена идентификации автора неоднородного исходного кода программы на основе гибридной нейронной сети. Решения данной проблемы особенно актуальны для областей информационной безопасности, образовательного процесса и защиты авторского права. В статье представлен анализ современных методов решения поставленной задачи. Авторами предлагается собственная методика на основе гибридной нейронной сети, зарекомендовавшей себя в ранних исследованиях, направленных на оценку эффективности данного подхода в простых и сложных случаях.

Данная работа включает в себя эксперименты по ранее не рассмотренным случаям идентификации автора исходного кода на основе неоднородных данных. Рассматриваются случаи, актуальные для корпоративной разработки. Среди них анализ исходных кодов, представленных в виде коммитов, и обучение модели на наборах данных, включающих в себя два и более языка программирования. Также исследовано набирающее популярность направление определения авторства искусственно сгенерированного исходного кода. Для каждого случая сформирован набор данных и проведен эксперимент.

Оценка эффективности авторской методики для всех трех сложных случаев осуществлена при помощи перекрестной проверки по 10 блокам. Средняя точность для смешанных наборов данных составила 87 % для двух языков программирования и 76 % для трех и более языков, соответственно. Точность методики для решения задачи определения авторства искусственно сгенерированных исходных кодов в среднем составила 81,5 %. Идентификация автора исходного кода программы на основе коммитов осуществлялась с точностью 84 %. Эксперименты показали, что во всех трех случаях эффективность методики может быть повышена путем использования больших объемов обучающих данных.

Ключевые слова: авторство, исходный код, коммиты, генерация, нейронная сеть.

Благодарности: 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).

Для цитирования: Куртукова А.В., Романов А.С., Шелупанов А.А., Федотова А.М. Идентификация автора исходного кода программы на основе неоднородных данных для решения задач кибербезопасности. *Моделирование, оптимизация и информационные технологии*. 2022;10(3). Доступно по: https://moitvivt.ru/ru/journal/pdf?id=1227 DOI: 10.26102/2310-6018/2022.38.3.016

Authorship identification of a heterogeneous source code for the purposes of cybersecurity management

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

Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russian Federation
av.kurtukova@gmail.com✉

© Куртукова А.В., Романов А.С., Шелупанов А.А., Федотова А.М., 2022

Abstract. The article is devoted to the issue of identifying the author of a heterogeneous source code program by means of a hybrid neural network. The solutions to this problem are especially relevant to the fields of information security, educational process, and copyright protection. The article analyzes modern methods of addressing this problem. The authors propose their own methodology based on a proven in early studies hybrid neural network aimed at evaluating the effectiveness of this approach in simple and difficult cases.

This research incorporates experiments on previously unconsidered cases of source code author identification based on heterogeneous data. Cases relevant to corporate development are examined including the analysis of source codes presented as commits and model training on datasets with more than two programming languages. Additionally, the trend of determining the authorship of an artificially generated source code, which is gaining traction, is regarded. A dataset was generated, and an appropriate experiment was performed for each case.

The effectiveness of the author's methodology for all three difficult cases was evaluated using a 10 blocks cross-validation. The average accuracy for mixed datasets was 87 % for two programming languages and 76 % for three or more languages, respectively. The average accuracy of the methodology for authorship identification of artificially generated source codes was 81.5 %. Identification of the author of a program source code based on commits was carried out with an accuracy of 84 %. Experiments have shown that the effectiveness of the methodology can be improved in all three cases by using large amounts of training data.

***Keywords*:** authorship, source code, commits, generation, neural network.

Acknowledgements: the research was supported by the grant of the Ministry of Science and Higher Education of the Russian Federation under the core part of TUSUR state task for 2020–2022 (project No. FEWM-2020-0037).

For citation: Kurtukova A.V., Romanov A.S., Shelupanov A.A., Fedotova A.M. Authorship identification of a heterogeneous source code for the purposes of cybersecurity management. *Modeling, Optimization and Information Technology*. 2022;10(3). Available from: https://moitvivt.ru/ru/journal/pdf?id=1227 DOI: 10.26102/2310-6018/2022.38.3.016 (In Russ.).

Введение

Проблема идентификации автора исходного кода программы как искусственно-языкового текста является актуальной и важной. Ее решения состоят в определении индивидуального стиля автора-программиста, в частности, в выявлении персональных приемов, методов разработки. Такие решения могут быть особенно полезны в сферах информационной безопасности, образовательного процесса и защиты авторского права.

Исследователи заинтересованы как в совершенствовании подходов к идентификации автора исходного кода программы, так и в создании новых инструментов на их основе. Тем не менее множество разработанных и улучшенных методов позволяет добиться высокой точности лишь для простых случаев определения авторства. Под простыми, в данном случае, подразумеваются написанные разработчиком исходные тексты программ, не подвергнутые никаким преобразованиям, выполненных как вручную, так и при помощи среды программирования и встроенных в нее инструментов (линтеров, автоматического форматирования в соответствии с парадигмами языка) или сторонних инструментов (обфускаторов). Также простыми следует считать случаи, когда данные, предназначенные для анализа на предмет авторства, гарантированно являются подлинными и однородными.

Решение прикладных задач идентификации автора исходного кода предполагает анализ текстов программ, осложненный теми или иными факторами. Такие факторы можно разделить на две группы:

1. Факторы, возникающие в процессе или по завершению написания исходного кода. К данной группе относятся различные преобразования исходных кодов программ.

Например, озвученные ранее: обфускация исходного кода программы – приведение исходного кода программы к виду, затрудняющему его понимание и анализ, но сохраняющему функциональность; написание исходного кода в соответствии со стандартами кодирования – разработка исходного кода с учетом соглашений и правил, принятых группой программистов.

2. Факторы, возникающие ввиду особенностей процесса разработки. Данная группа включает в себя другие случаи, в значительной степени осложняющие процесс определения автора исходного кода. Например, идентификация автора на основе образцов, написанных на разных языках программирования одним и тем же программистом (на основе смешанных данных). Также разграничение авторства исходного кода программы между человеком и машиной или генеративной моделью. Еще одним случаем, который следует рассматривать в рамках второй группы, является определение авторства на основе фрагментов исходных кодов, написанных в рамках групповой разработки.

Научная новизна работы состоит в предлагаемой авторами методике, впервые учитывающей как простые, так и все сложные случаи идентификации автора исходного кода программы. Сложные случаи включают в себя идентификацию автора:

– исходного кода программы, сформированного из отдельных фрагментов кода (коммитов);

– искусственно-сгенерированного исходного кода программы;

– исходного кода программы, автор которой программирует на двух и более языках программирования;

– обфусцированного исходного кода программы;

– исходного кода программы, написанного в соответствии со стандартами кодирования.

Предшествующие работы

Простые случаи идентификации были рассмотрены в рамках ранних исследований, посвященных идентификации автора исходного кода программы [1]. С целью решения поставленной задачи в них были рассмотрены подходы на основе глубокой нейронной сети (НС) и метода опорных векторов (SVM) совместно с фильтром быстрой корреляции. Оба подхода продемонстрировали высокую точность – НС позволила добиться 97 % в среднем вне зависимости от языка программирования, а SVM уступила в эффективности лишь 1 %, составив 96 %, соответственно. Важным аспектом является то, что результат НС был получен на основе анализа необработанных исходных кодов программ, а следовательно, информативные признаки для принятия решения об авторстве выявлялись ей самостоятельно на глубоких слоях модели. В свою очередь, SVM была обучена на множестве признаков, сформированном экспертным путем и отфильтрованном алгоритмом на основе корреляции. Таким образом, в рамках научной работы была выдвинута гипотеза о способности глубокой НС к выявлению новых, неявных закономерностей в исходных данных, которые также могут быть неподконтрольны программисту на сознательном уровне. Классический метод SVM такой способности не продемонстрировал, то есть оказался неустойчивым к намеренным преобразованиям исходного кода с целью запутывания методов определения авторства. Проведенные эксперименты позволили сделать вывод о независимости обоих методов идентификации автора исходного кода программы от языка программирования, на котором осуществляется разработка анализируемого программного обеспечения (ПО). Квалификация программистов также не оказала негативного влияния на эффективность методики при решении реальных задач.

Последующие исследования [2] охватывают всю первую группу осложняющих факторов идентификации автора исходного кода программы. Исходя из полученного ранее опыта, было решено использовать методику на основе глубокой НС, представленной в виде авторской гибридной НС (HNN), с целью подтверждения или, напротив, опровержения гипотезы о ее способности к поиску неявных информативных признаков. Во-первых, в рамках экспериментов были рассмотрены исходные коды на пяти языках программирования, преобразованные при помощи различных по своему виду действия обфускаторов. Результаты позволили сделать вывод об устойчивости выбранного подхода к лексической обфускации – модель продемонстрировала потерю в точности, в среднем не превышающую 10 %. Во-вторых, был проведен ряд экспериментов, направленных на оценку точности определения авторства в условиях следования программистами единому стандарту кодирования. В качестве данных для обучения были использованы исходные коды ядра Linux, написанные в соответствии с рядом правил, предъявляемым пользователям-контрибьюторам. В отличие от обфускации, стандарты кодирования оказали негативное влияние на точность. Однако в ходе дополнительных экспериментов было выявлено, что увеличение количества обучающих данных позволяет получить прирост точности в 40 % и делает эффективность модели достаточной для применения с целью решения реальных задач. Итак, выдвинутая в первом исследовании гипотеза была подтверждена – разработанная методика на основе HNN позволила классифицировать в соответствии с авторством как преобразованные при помощи обфускаторов, так и написанные согласно стандартам кодирования исходные коды программ. Вторая группа факторов требует такого же тщательного изучения и проработки.

Анализ предметной области

Применение простых статистических методов является недостаточным для эффективного принятия решения об авторстве в сложных случаях, поэтому необходимо использование моделей, способных находить новые закономерности и зависимости в данных, являющиеся неявными для исследователя, а также косвенно учитывать интеллектуальное содержание и особенности реализации программного кода. К таким моделям относятся глубокие архитектуры НС, в частности, модификации, разработанные исследователями за последние 5 лет.

Работа [3] посвящена системе «Deep Learning-based Code Authorship Identification System» (DL-CAIS). Эта система основана на классификаторе случайного леса. Он масштабируется и принимает решение об истинном авторе на основе глубоких представлений, полученных с использованием частот слов и обратных частот документов (TF-IDF) и многоуровневой рекуррентной НС (RNN). По мнению авторов, система не зависит от языка программирования. Это подтверждается на примере 4 языков: C ++, Java, Python, C. Средняя точность идентификации составила 95 %.

Авторы работы [4] предлагают собственный подход на основе глубокого обучения, называемый «Генератор шаблонов устойчивого стиля кодирования» (RoPGen). Данный подход изучает уникальные шаблоны стиля кодирования авторов, которые трудно имитировать злоумышленникам. Основная идея состоит в одновременной аугментации данных и градиента. Так увеличивается разнообразие обучающих примеров, создаются значимые возмущения градиентов глубоких НС и изучаются разнообразные представления стилей кодирования. Эффективность RoPGen оценивается на четырех наборах исходных кодов программ, написанных на языках C, C++ и Java. Экспериментальные результаты показывают, что RoPGen может значительно

повысить надежность атрибуции авторства кода на основе глубокого обучения, снизив вероятность успеха целевых и нецелевых атак до 22,8 % и 41 %.

В работе [5] для определения автора исходного кода предлагается использование графовых НС. Данный подход возник в связи с тем, что популярные для решения смежных задач сверточные НС (CNN) ограничивают выражение зависимостей и семантических отношений в исходных кодах программ. Программы в данном подходе представляются в виде графов, отражающих сложные взаимосвязи авторских признаков друг с другом. Оценка эффективности производится на наборе данных Google Code Jam. Точность подхода достигает 60 %.

Авторы исследований [6-7] предлагают решение проблем невозможности добавления новых авторов без переобучения и отсутствия интерпретируемости. Ими используются методы объяснимого искусственного интеллекта (XAI). Результаты экспериментов для разных языков программирования подтверждают, что модель обращает внимание на отличительные черты авторов исходных кодов программ. Средняя точность предложенных подходов составляет 75 %.

Работа [8] посвящена независимому от языка программирования подходу к атрибуции авторства исходного кода. Авторы обсуждают ограничения существующих синтетических наборов данных для установления авторства и предлагают свой подход к сбору данных, лучшим образом отражающим аспекты, важные для потенциального практического использования. Также авторы доказывают, что точность моделей современных решений для идентификации автора резко падает, когда их эффективность оценивается на более реалистичных данных. В качестве решений проблемы они предлагают две модели: НС на основе пути (PbNN) и модифицированный случайный лес (PbRF). Обе модели не зависят от языка и работают с представлениями кода на основе путей, которые могут быть построены для любого синтаксически правильного фрагмента кода. Точность модели для языка программирования Java составляет 97,9 % для PbNN и 98,5 % для PbRF соответственно.

Авторы перечисленных подходов хоть и заявляют о высокой эффективности при решении задачи идентификации автора исходного кода программы, но не оценивают их на устойчивость к сложным случаям. Исходя из опыта смежных исследований [9-10], можно сделать вывод о том, что преобразования исходных кодов, на основе которых происходит обучение НС, или непосредственно обучение НС на неоднородных данных приводят к запутыванию классификатора и падению его точности.

Формальная постановка задачи

Учитывая осложняющие факторы, предложенная авторами ранее [1] формальная постановка задачи обобщена до следующего вида. Идентификация автора исходного кода программы на основе неоднородных данных состоит в нахождении целевой функции $y^*: S \to P$, где S – множество исходных кодов программ, а P – множество авторов-программистов. В качестве $s \in S$ могут выступать обфусцированные или агрегированные из множества фрагментов исходные коды программ. В качестве $p \in P$ могут выступать генеративные языковые модели. Истинные авторы исходных кодов программ известны только на конечной обучающей выборке $S^m = \{(s_1, p_1), ..., (s_m, p_m)\}$, где m – количество исходных кодов программ с известным автором. В таком случае задача состоит в построении классификатора $\alpha: S \to P$, способного определить принадлежность произвольного исходного кода $s \in S$ истинному автору $p \in P$.

С целью нахождения целевой функции y^* классификатор α должен быть обучен на признаковом пространстве. Каждый отдельный признак данного пространства можно

описать как $f: S \to D_f$, где f – множество допустимых значений, а D – множество признаков текста. Тогда множеством признаков произвольного исходного кода $s \in S$ является вектор $s = \{(f_1,(s_1),...,(f_n,s_n)\}$, где n – размерность признакового пространства.

Методика определения автора анонимного текста

Положительный опыт предшествующих научных работ [1-2] подтверждает целесообразность применения авторской методики на основе HNN в рамках исследований сложных случаев идентификации автора неоднородного исходного кода программы. Модель, лежащая в основе методики, продемонстрировала высокую эффективность в простых случаях определения авторства программы и устойчивость перед такими осложняющими факторами, как обфускация и следование стандартам кодирования. Представленная на Рисунке 1 архитектура авторской HNN является глубокой и включает в себя несколько слоев, а именно:

1. Входной слой, размерность которого соответствует длине векторов, полученных на вход НС. В данном случае она составляет 256, что соответствует вектору из 255 нулей и 1 единицы на позиции кода символа согласно кодировке ASCII. Данный вектор формируется для каждого символа текстовой последовательности исходного кода программы. Непосредственно процесс векторизации происходит методом one-hot кодирования.

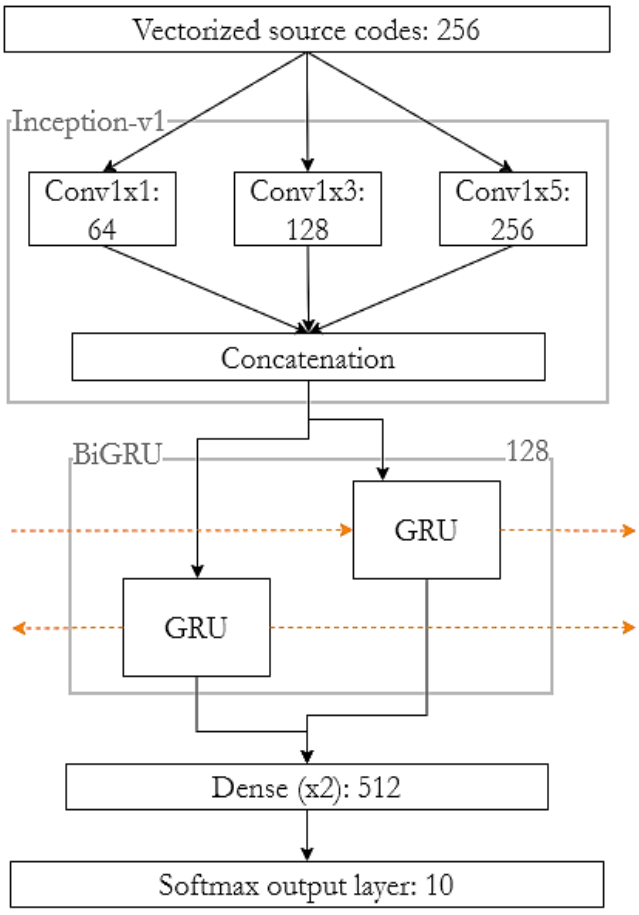

Рисунок 1 – Архитектура HNN
Figure 1 – HNN architecture

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

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

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 модели.

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

Представленная модель является полностью самостоятельным инструментом. Она не требует предварительной обработки исходных кодов программ и выделения из них информативных признаков, а работает с образцами на символьном уровне и выявляет их самостоятельно.

Экспериментальные данные

Процесс идентификации автора исходного кода должен быть построен на качественном и объемном наборе данных. Для получения такого набора необходимо было выбрать платформу, на которой в свободном доступе размещены подходящие данные. Поиск такой платформы был осуществлен среди самых популярных хостингов IT-проектов. Поскольку при сборе данных особенно важна актуальность языков и написанных на них исходных кодов программ, а также возможность использования программного интерфейса приложения (API), было решено рассмотреть хостинги GitHub [11] и GitLab [12]. Они являются наиболее популярными платформами, за счет чего обновления на них происходят регулярно даже для самых редких или совсем новых языков программирования. В связи с тем, что GitLab более распространен у разработчиков коммерческого ПО, на нем преобладают закрытые репозитории (виртуальные хранилища проектов), в то время как для сбора необходимо большое количество открытых, более свойственных платформе GitHub. Таким образом, в качестве источника для сбора и формирования набора данных исходных кодов программ был выбран хостинг GitHub.

После выбора хостинга необходимо было определить множество языков программирования для последующего сбора. Обязательными критериями являлась популярность языка и достаточное количество открытых репозиториев на GitHub. Согласно рейтингу, основанному на мнении разработчиков коммерческого ПО, были отобраны те языки, которые имели достаточное количество репозиториев на хостинге GitHub, а именно: JavaScript, Java, C#, Python, PHP, Kotlin, Swift, C++, Go, Ruby, C, Groovy, Perl. Итого для сбора данных было отобрано 13 языков программирования.

Процесс клонирования репозиториев с ресурса GitHub происходил путем создания запросов, включающих в себя адрес хранилища исходных кодов, язык программирования, страницу, на которой расположен репозиторий, само название репозитория и токен доступа пользователя. Структура выгруженного с хостинга репозитория при этом изменениям не подвергалась. Клонированные на локальный носитель с GitHub репозитории почти всегда включали в себя, помимо исходных кодов

программ на конкретном языке программирования, вспомогательные файлы. Такие файлы считались избыточными, так как зашумляли исходные коды, несущие в себе наибольшее число информативных признаков, поэтому были удалены.

Информация о полученном наборе исходных кодов программ представлена в Таблице 1.

Таблица 1 – Информация о наборе исходных кодов программ
Table 1 – Information about the program source code dataset

Количество строк кода	20967040
Количество проектов	569
Количество проектов с одним автором	71
Средняя длина коммита, строк	13
Минимальная длина коммита, строк	1
Максимальная длина коммита, строк	119892
Среднее число исходных кодов в проекте	231
Средняя длина исходного кода, строк	169
Суммарное число символов кода	212336673
Всего исходных кодов	102908
Авторов во всех проектах	383

С целью корректной постановки экспериментов необходимо детально рассмотреть все сложные случаи и выявить, во-первых, особенности данных как объекта исследования, отдельно для каждого из них, во-вторых, потенциальные сложности и узкие места их анализа.

Смешанные данные

Важной проблемой современных исследований, посвященных задаче идентификации автора исходного кода программы, является отсутствие экспериментов, направленных на оценку точности разработанных решений применительно к смешанным наборам данных.

Необходимость получения такой оценки обусловлена особенностями разработки современного ПО. Зачастую реализация продукта происходит на нескольких языках программирования одновременно. Набор языков программирования, используемых в рамках разработки того или иного продукта, может различаться и называется его стеком. Точно такой же стек существует и у каждого разработчика, то есть группа языков, на которых он осуществляет разработку исходных кодов программ или их модулей.

Реализация основной части программы нередко осуществляется на одном языке, а необходимость в другом возникает лишь для создания вспомогательных инструментов. Однако возникают ситуации, когда разработка выполняется в равной степени на двух или даже трех языках программирования. Исходя из этого, количество данных, собираемых с целью обучения модели идентификации автора исходного кода программы, может оказаться недостаточным для получения разработанным инструментом достоверного результата. Отсюда возникает необходимость в оценке способности предложенной методики демонстрировать сопоставимый однородным данным результат применительно к неоднородным, в данном случае, реализованным на различных языках программирования исходным кодам программ.

Проверке подлежит гипотеза о слабой зависимости авторского стиля от языка программирования, на котором пишет программист, и свободном переносе информативных признаков с одного языка на другой. Предполагается, что гипотеза

будет правдива, в том числе для разных по способу реализации языков программирования, то есть для компилируемых и интерпретируемых языков. Таким образом, модель, лежащая в основе методики идентификации автора исходного кода программы, должна обладать способностью к поиску и анализу как явных, так и неявных авторских признаков даже на основе данных, включающих в себя образцы на различных языках программирования.

Иллюстрация сложного случая представлена далее. Пример исходного кода автора N на языке программирования JavaScript представлен на Рисунке 2 сверху. Код аналогичной программы автора *N* на языке программирования C# – на Рисунке 2 снизу. Приведенный пример подтверждает, что авторские признаки автора *N* могут как частично, так и полностью сохраниться при переходе от одного языка программирования к другому. Форматирование блоков кода, названия переменных, расстановка пробельных символов – это те явные признаки, которые были перенесены автором с JavaScript на C# в данном случае.

```
#pragma strict
private var myFloat : float = 0.05;
public var myString : String = "test";
function Start(){
var bla : Array;
yield WaitForSeconds(2);
}
function GetScore(playerName : String) : int{
return 5;
}
@RPC
function Test(){}
```

```
using UnityEngine;
using System.Collections;

public class MYCLASSNAME : MonoBehaviour {

private float myFloat = 0.05f;
public string myString = "test";
void  Start (){
    Array bla;
    yield return new WaitForSeconds(2);
}
int GetScore ( string playerName  ){
    return 5;
}
[RPC]
void  Test (){}
}
```

Рисунок 2 – Исходные коды автора на языке JavaScript и C#
Figure 2 – Author's source codes in JavaScript and C#

Для получения обучающих наборов, включающих в себя смешанные данные, был осуществлен поиск авторов-программистов, являющихся контрибьюторами проектов на разных языках программирования. Извлекались исходные коды программ, написанные конкретным автором и размещенные в соответствующем языку репозитории. При формировании набора данных разделение образцов исходных кодов программ не осуществлялось.

Искусственно-сгенерированные исходные коды

Одной из перспективных тенденций глубокого обучения является генерация текста на различных языках. Современные модели, предназначенные для создания текстов с нуля или на основе контекста, демонстрируют высокую эффективность [13-18]. Зачастую написанные машиной тексты невозможно отличить от человеческих. Это

обусловлено способностью языковых моделей к сохранению авторских признаков, то есть отличительных черт стиля письма того или иного автора.

Генерация естественно-языковых текстов, зарекомендовавшая себя в различных сферах деятельности человека, спровоцировала появление нового направления – генерацию искусственно-языковых текстов, в том числе исходных кодов программ. Модели, позволяющие создавать исходный код самостоятельно или дописывать текст программы за пользователя позволяют сократить временные затраты программистов путем избавления их от написания рутинного кода, не требующего исследовательского или творческого вклада. Использование генеративных моделей с целью реализации исходных кодов программ для сферы разработки коммерческого ПО очень перспективно и полезно. Тем не менее появление описанных инструментов лишь повышает важность вопроса идентификации автора исходного кода программы. Таким образом, исследователи авторства исходного кода оказываются вынуждены ответить на три новых вопроса:

– Написан образец S человеком или языковой моделью?
– Сгенерированы ли образцы S_1 и S_2 одной языковой моделью?
– Каким из n методов сгенерирован заданный исходный код S?

Методика идентификации автора искусственно сгенерированного исходного кода программы должна не только эффективно выявлять информативные признаки авторов-программистов, но также быть способной, как минимум, разделять авторство между различными генеративными моделями, а следовательно, выявлять отличительные черты создания кода искусственным путем.

Принцип работы генеративной модели (Рисунок 3) заключается в получении на вход собственного предшествующего состояния. После подачи на вход неполного отрывка текста, например, «import ker», модель определяет распределение вероятностей для следующего за отрывком символа. Символ с наибольшей вероятностью добавляется в конец неполного отрывка. В данном примере таким символом является «a», а результатом итерации предсказания «import kera». Процесс повторяется до тех пор, пока не будет получен полносвязный и осмысленный текст.

Разработчики решений, предназначенных для генерации исходных кодов программ, также учли положительный опыт применения GPT-моделей. На их основе появились наиболее известные инструменты Open AI Codex, GitHub Copilot, AlphaCode, JARVIS Sber AI и PolyCoder [21-25]. Все пять решений дообучены на соответствующих данных – больших объемах исходных кодов программ. Для обучения некоторых из них применялись образцы, находящиеся в свободном доступе на IT-хостинге GitHub или на крупных соревновательных платформах, для других – собственная база исходных кодов. Избыточное количество эталонных данных, подходящих для обучения генеративных моделей, позволяет добиться точности, достаточной для создания ими компилируемого и работоспособного кода.

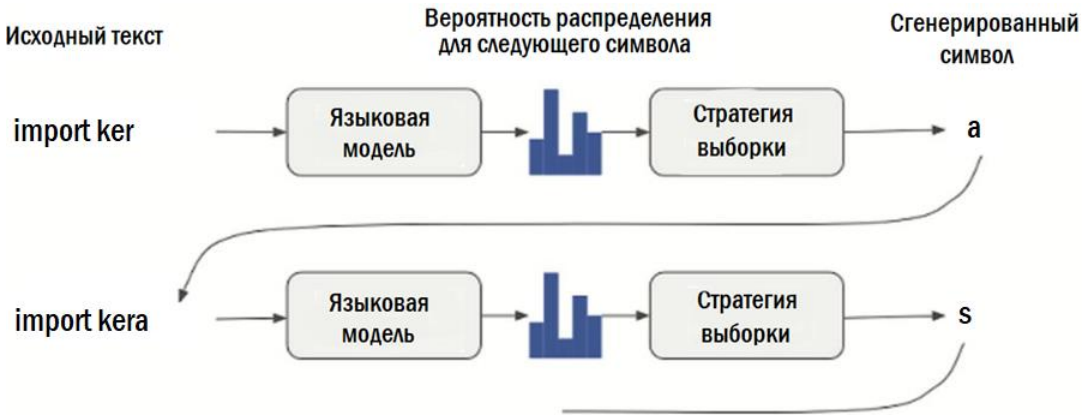

Рисунок 3 – Процесс генерации исходного кода программы
Figure 3 – Program source code generation

Несмотря на то, что большинство существующих решений в области генерации исходного кода, демонстрируют впечатляющие результаты, их применение в рамках данного исследования невозможно, так как они являются закрытыми для сообщества разработчиков. Единственное открытое решение, AlphaCode, не является подходящим в связи с тем, что предназначено исключительно для создания кода для соревнований, а, следовательно, использование готовых решений не представляется возможным. Однако, согласно проведенному анализу, все рассмотренные инструменты базируются на моделях семейства GPT. Таким образом, в рамках данного исследования решено использовать собственную дообученную на кодовой базе, собранную с хостинга GitHub модель GPT-3.

Коммиты исходных кодов программ

Наиболее распространенным среди сложных случаев анализа неоднородных данных является идентификация автора исходного кода программы на основании его коммитов – последних изменений исходного кода в репозиторий. Так как инструменты определения авторства программ считаются наиболее востребованными для коммерческой среды, следует учитывать специфику работы этой области. Современные IT-компании пользуются различными технологиями для гибкой совместной разработки программных продуктов и используют в своей ежедневной деятельности различные системы контроля версий.

Согласно принципам групповой разработки, большинство программ и программных комплексов создаются несколькими разработчиками совместно. То есть работа над одним и тем же исходным кодом или модулем ведется не единолично одним программистом, а целой командой. Вклад членов команды может быть различен по своей ценности: один разработчик может написать 90 % от всего кода, в то время как второй – лишь исправить найденные в ходе тестирования ошибки, написав, в общей сложности, пару строк. Для решения задачи идентификации автора исходного кода важно уметь правильно извлекать образцы конкретных авторов-программистов, а значит, разграничивать коммиты в соответствии с именами загрузивших их пользователей.

Для того, чтобы извлекать в репозиторий информацию о сделанных коммитах отдельных пользователей, аналогично формированию смешанных наборов данных, использовался API, предоставляемый GitHub. В данном случае, для создания обучающего набора данных используется информация о коммитах отдельных пользователей в репозитории. Строки исходного кода, загруженные конкретным пользователем, записываются в отдельный файл, создавая таким образом не всегда

компилируемый или работоспособный, но, тем не менее, отражающий авторский стиль текст программы. Пример сложного случая представлен на Рисунке 4.

```
    if priority not in ['HIGH', 'LOW']:
        priority = 'MEDIUM'
else:                                   if priority not in ['HIGH', 'LOW']:
    date_sent = ''                                  priority = 'MEDIUM'
    priority = None
```

Рисунок 4 – Пример файла, полученного на основании коммитов
Figure 4 – Example of a file obtained using commits

Постановка эксперимента и результаты

Для обучения и оценки авторской HNN в соответствии с каждым сложным случаем был сформирован соответствующий набор данных. Каждый из наборов включил в себя образцы, соответствующие следующим требованиям:

1. Длина исходного кода должна быть не менее 30 символов и не более 3000 символов.
2. На каждого автора отбирается не более 30 образцов исходных кодов программ.
3. Выбор образцов осуществляется случайным образом.

Таким образом, минимальное число строк, приходящихся на автора, участвующего в обучении, составляет 90, а максимальное – 90000. Границы были выбраны, исходя из статистики собранной базы данных исходных кодов программ. Число эпох обучения HNN составляет 5 итераций по 10 эпох, итого 50 эпох.

Для оценки точности модели применялась перекрестная проверка по 10-блокам (10-фолдовая кросс-валидация). Принцип работы данного алгоритма состоит в последовательном обучении модели на 9 блоках данных и оценке на 1 оставшемся, повторяющемся на протяжении 10 итераций.

В рамках данного исследования было проведено 6 экспериментов, включая оценки точности для:

– смешанных данных, включающих в себя два языка программирования (языковые пары);
– смешанных данных, включающих в себя три и более языка;
– разграничения авторства между человеком и машиной;
– определения единства генеративной модели для пар образцов;
– разграничения авторства между генеративными моделями;
– данных, полученных из коммитов контрибьюторов.

Согласно результатам экспериментов, средняя точность идентификации автора исходного кода программы, осуществляющего разработку на двух языках программирования, составляет 87 % (Рисунок 5). Можно отметить, что точность определения автора-программиста на основе пар языков, в той или иной степени схожих по синтаксису, превышает точность, полученную на основе пар языков, не имеющих сходств. Такой вывод особенно справедлив для пар языков JavaScript-C+, Swift-Java, JavaScript-Python.

По результатам экспериментов, средняя точность идентификации автора исходного кода программы, осуществляющего разработку на трех и более языках программирования, составляет 76 % (Рисунок 6). Так как наборы являются полностью смешанными и могут включать в себя более десяти языков программирования на одного автора, оценить степень влияния того или иного языка на результат не представляется

возможным. Следует отметить, что данный случай представляет скорее исследовательский интерес, так как на практике встречается крайне редко.

Полученные на основе смешанных наборов данных результаты позволяют сделать вывод о высокой эффективности авторской методики. Полученные точности в 76 % для наборов данных, включающих в себя три и более языка программирования, и 87 % для наборов данных, состоящих из языковых пар, превышают точность некоторых аналогов методики [26-27], полученных для простых случаев идентификации.

В рамках оценки точности идентификации автора искусственно-сгенерированного исходного кода программы было рассмотрено сразу три сложных случая. Первый из них и самый простой – разграничение авторства между человеком и генеративной моделью (Рисунок 7). В данном случае оценка производилась следующим образом. В каждом эксперименте для 5, 10 и 20 авторов одним из авторов являлась генеративная модель. В ряде исследований участвовали образцы, написанные авторами-программистами, и образцы, сгенерированные тремя моделями семейства GPT: GPT-2, GPT-3, RuGPT-3 (от Sber AI), соответственно. Все образцы были реализованы на наиболее популярном языке Java. Средняя точность для GPT-2 составила 97 %, для GPT-3 и RuGPT-3 – 94 %. Разница в точности обусловлена тем, что модели 3-го поколения демонстрируют лучшую генеративную способность, нежели 2-го. Следовательно, разделительная способность авторской модели так же снижается.

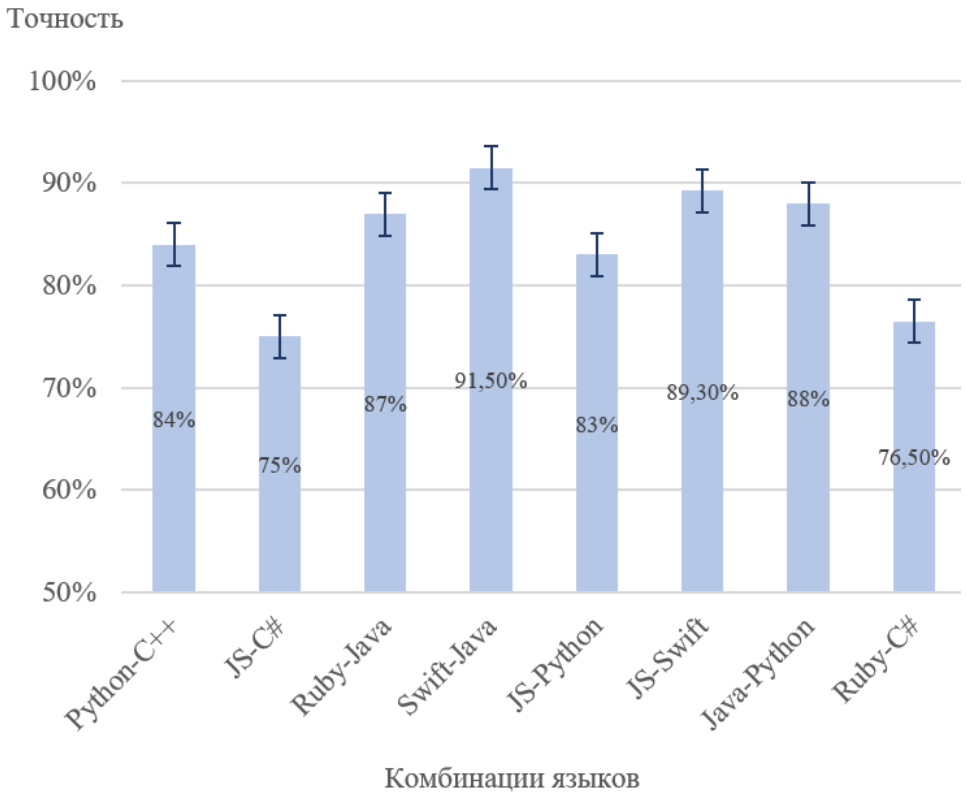

Рисунок 5 – Точность для пар языков
Figure 5 – Classification accuracy for language pairs

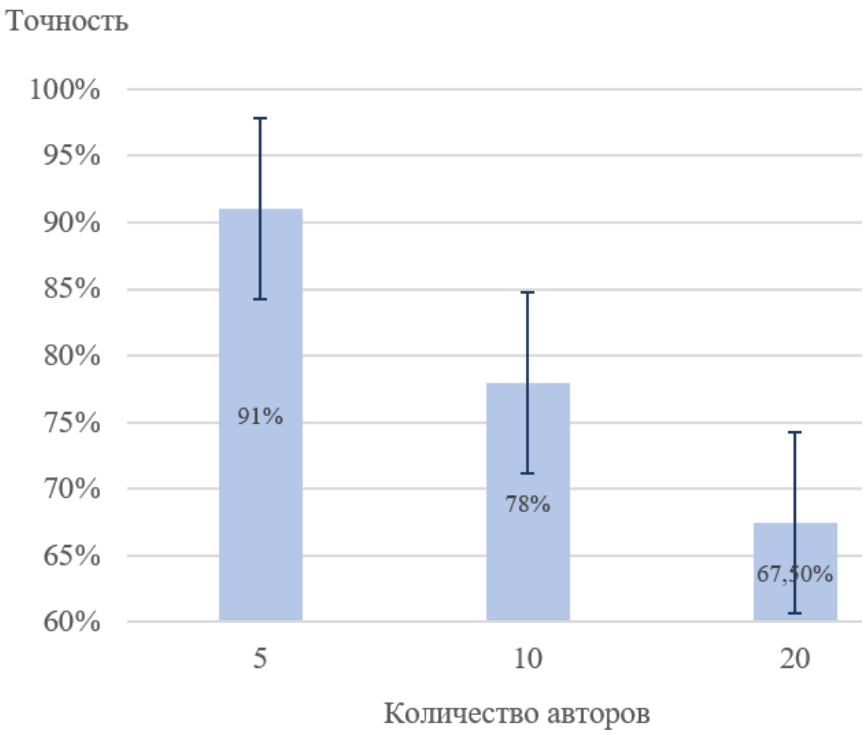

Рисунок 6 – Точность для разных комбинаций языков
Figure 6 – Classification accuracy for different language combinations

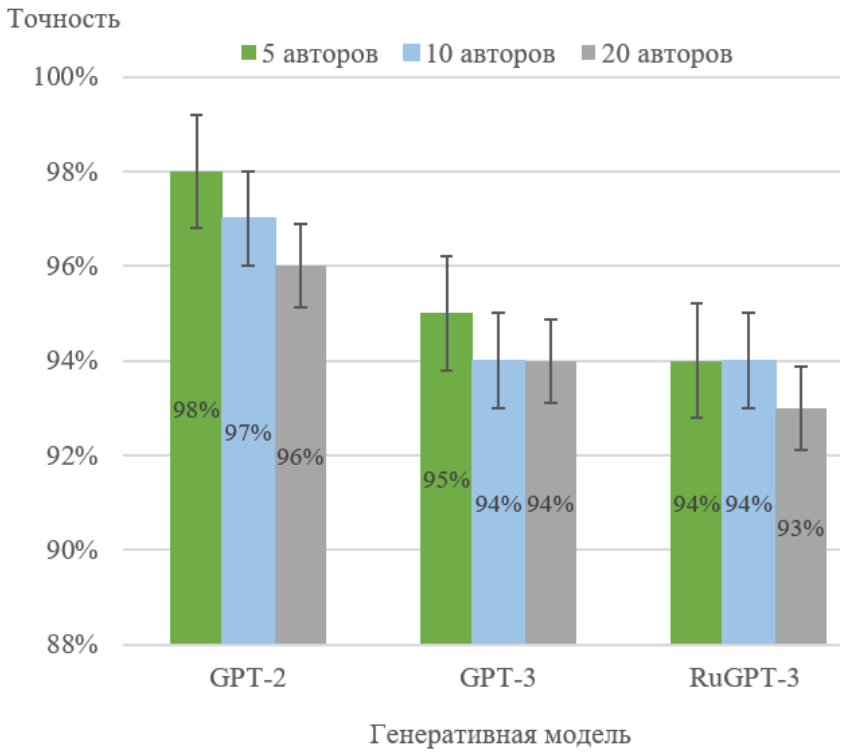

Рисунок 7 – Разграничение авторства между человеком и машиной
Figure 7 – Difference between a human and a machine

Второй эксперимент был посвящен оценке точности определения единства генеративной модели для двух образцов исходных кодов программ (Рисунок 8).

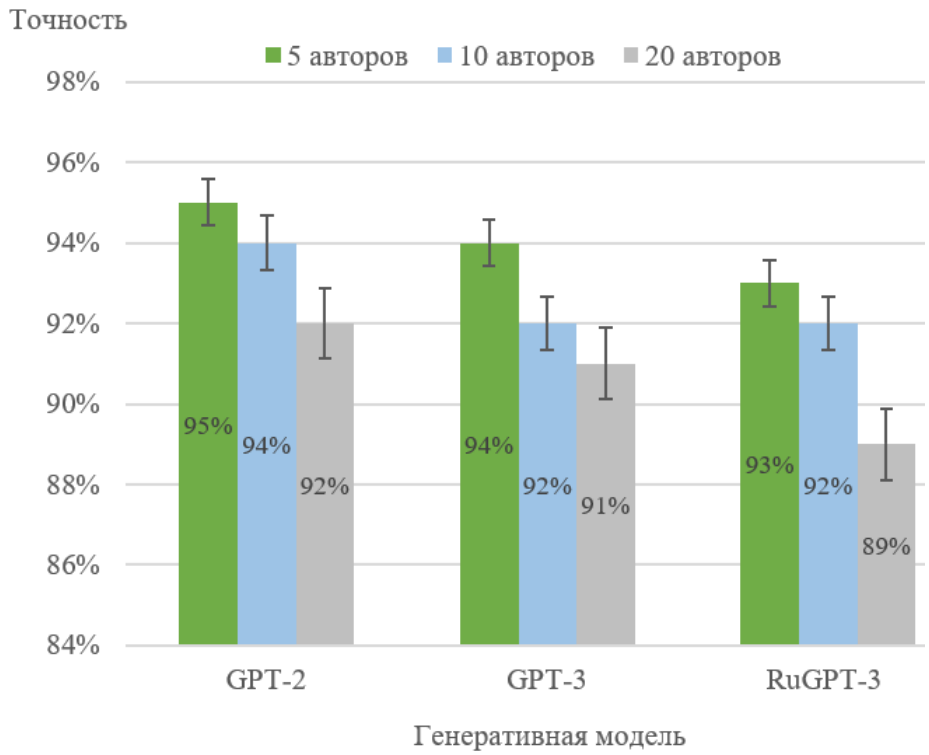

Рисунок 8 – Единство генеративной модели для пар образцов исходных кодов
Figure 8 – The unity of the generative model for pairs of source code samples

В рамках данного эксперимента использовались два образца исходного кода, сгенерированные одной и той же языковой моделью. В исследовании участвовали 50 образцов исходных кодов на языке Java, разбитых на пары, для каждой генеративной модели соответственно. Данные пары поочередно подавались на вход модели, обученной для 5, 10 и 20 авторов, одним из которых являлась целевая генеративная модель. Для GPT-2 средняя точность составила 94 %, для GPT-3 – 92 % и для RuGPT-3 – 91 %.

Третий и последний эксперимент, посвященный искусственно-сгенерированным исходным кодам, был нацелен на оценку точности модели при разграничении авторства между тремя различными генеративными моделями (Рисунок 9). В данном случае эксперимент был поставлен таким образом, что в обучении участвовали 3 автора-модели: GPT-2, GPT-3 и RuGPT-3. Целью HNN являлось определение принадлежности анонимного исходного кода, так же сгенерированного искусственно, к одной из языковых моделей. Эксперимент проводился для 5 языков программирования: Java, C++, Python, JavaScript и Go. Средняя точность определения авторства исходного кода, сгенерированного языковой моделью, составила 94 %.

Последнее исследование было посвящено коммитам контрибьюторов репозитория (Рисунок 10). Набор данных формировался на основе образцов, компилирующих в себе коммиты (фрагменты, измененные в общем файле исходного кода конкретным пользователем) отдельных пользователей. Единственным условием для отбора данных был выбор коммитов длиной 3 строки кода и более. С целью выявления методических рекомендаций по применению методики для идентификации авторов-программистов, разрабатывающих коммерческое ПО и использующих системы контроля версий, было решено провести эксперимент для разного количества файлов – 5, 10 и 20 соответственно. Согласно результатам, для получения высокоточного инструмента принятия решений об авторстве следует использовать не менее 10 файлов

исходного кода на каждого автора-программиста. При несоблюдении данного условия точность может быть недостаточной для эффективного решения задач.

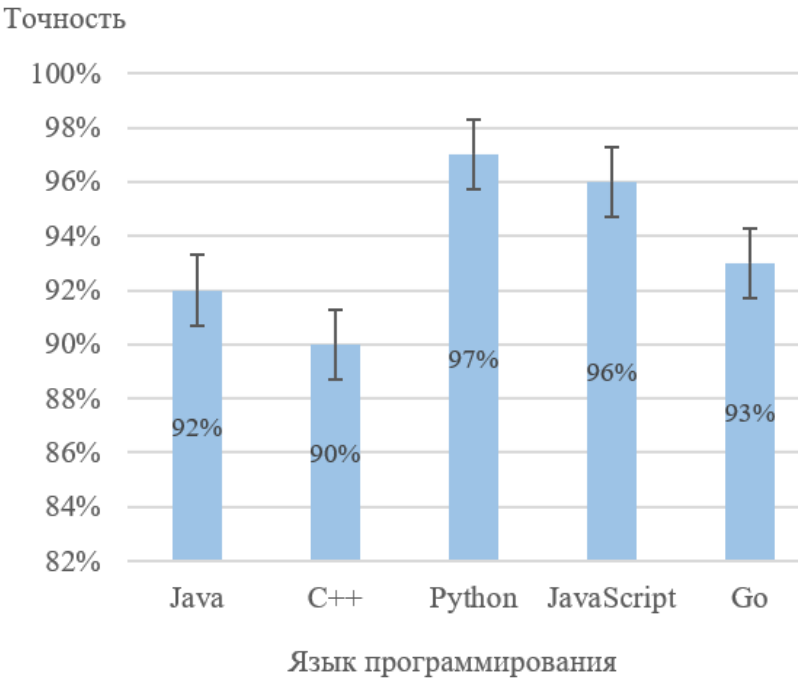

Рисунок 9 – Разграничение авторства между моделями
Figure 9 – Authorship differentiation between models

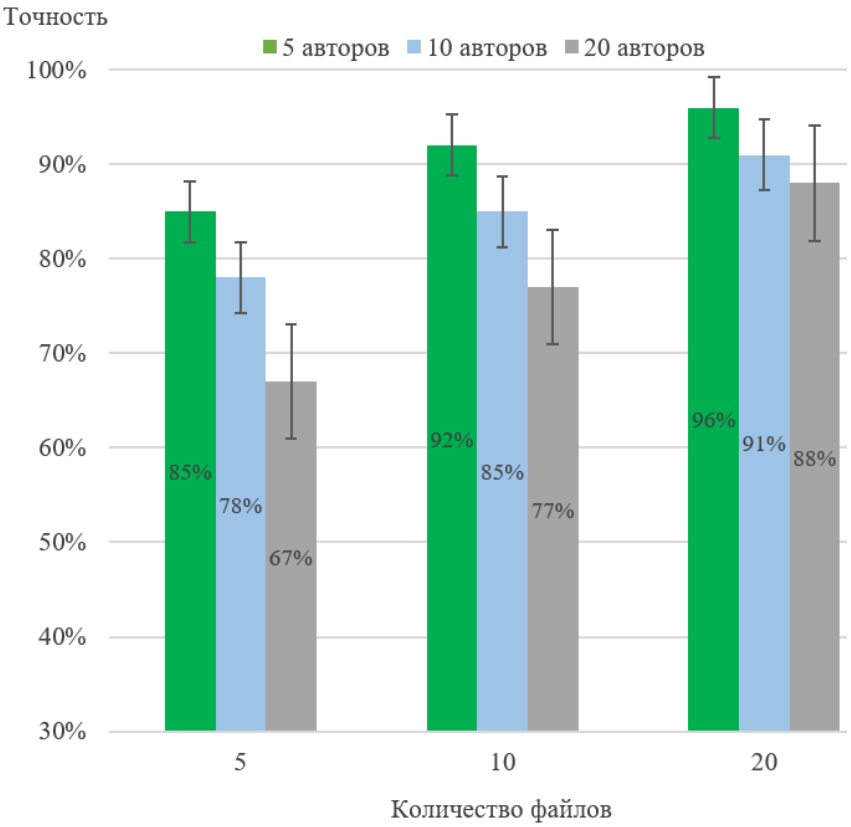

Рисунок 10 – Точность идентификации автора на основании коммитов
Figure 10 – Authorship identification accuracy based on commits

В Таблице 2 приведено сравнение результатов исследований, посвященных идентификации автора исходного кода, выполненных за последние 7 лет, в сравнении с результатами, полученными в рамках данной работы.

Таблица 2 – Сравнение подходов к идентификации автора исходного кода
Table 2 – Comparison of approaches to source code authorship identification

Автор	Метод	Рассмотрение сложных случаев	Язык	Средняя точность
Данная работа	HNN	Обфускация Стандарты кодирования Смешанные данные Искусственно-сгенерированные данные Данные на основе коммитов	Java C++ C# JS Ruby Python Swift Perl Groovy Go PHP Kotlin C	95 %
Abuhamad M. AbuHmed T. Mohaisen A. Nyang D [3]	DL-CAIS	Обфускация	C C++ Java Python	98 %
Zhen L. Chen G. Chen C. Zou Y. Xu S. [4]	RoPGen	-	C C++ Java	88,5 %
Holland C. Khoshavi N. Jaimes L.G. [5]	GNN	-	C# C++ Java	60 %
Bogdanova A. Romanov V. [6-7]	XAI	-	C++ Java Python	75 %
Bogomolov E. Kovalenko V. Rebryk Y. Bacchelli A. Bryksin T. [8]	PbNN, PbRF	-	Java	98 %
Caliskan-Islam A. Harang R. [10]	FuzzyAST, Random forest (RF)	Обфускация	C C++ Python	96,7 %

Сравнительный анализ демонстрирует, что предложенная методика учитывает все возможные случаи идентификации автора исходного кода программы, распространенные при решении прикладных задач. Авторская методика является эффективной как в простых, так и сложных случаях. Ни осложняющие факторы первой группы (обфускация, стандарты кодирования), ни второй (неоднородные данные) не оказывают существенного негативного влияния на точность методики.

Так как большинство современных подходов не рассматривает сложные случаи, помимо обфускации, было решено осуществить сравнение трех методов на примере идентификации автора исходного кода программы, обфусцированного с помощью инструмента Tigress. Результаты представлены в Таблице 3.

Таблица 3 – Сравнение подходов для обфусцированных данных
Table 3 – Comparison of approaches to obfuscated data

Автор	Метод	Язык	Обфускатор	Точность
Данная работа	HNN	JS	JS Obfuscator Tool	86 %
			JS-obfuscator	86 %
		Python	Opy	87 %
			Payrmor	70 %
		PHP	Yakpro-po	89 %
			PHP Obfuscator	82 %
		C++	C++ Obfuscator	71 %
		C	Tigress	90 %
Автор	Метод	Язык	Обфускатор	Точность
Abuhamad M. AbuHmed T. Mohaisen A. Nyang D. [3]	DL-CAIS	C	Tigress	93 %
Caliskan-Islam A. Harang R. [10]	FuzzyAST, Random forest (RF)	C	Tigress	67,2 %

Результат, демонстрируемый авторской методикой, уступает 3 % точности подходу [3]. Однако важно учитывать, что предлагаемая методика охватывает не только случаи обфускации для одного языка программирования, но также адаптирована для идентификации автора исходного кода программы в других сложных случаях, в том числе и для других языков программирования.

Заключение

Данная статья посвящена идентификации автора неоднородного исходного кода программы на основе HNN. Данная методика зарекомендовала себя при решении задач определения авторства в простых и сложных, связанных с преобразованиями исходного кода случаях.

В рамках исследования проведен ряд экспериментов, направленных на оценку эффективности авторской методики для случаев смешанных наборов данных, искусственно-сгенерированных данных и неоднородных данных, представленных в виде коммитов в репозитории хостинга.

Результаты проведенных экспериментов подтверждают высокую эффективность разработанной методики на основе HNN как в сложных, так и простых случаях идентификации автора исходного кода программы.

Для случаев, когда автор-программист владеет двумя языками программирования, точность методики составляет 87 %, тремя и более – 76 %. Еще один случай, связанный с корпоративной разработкой, представляет собой обучение модели, лежащей в основе методики, на основе агрегированного из коммитов исходного кода. Для данного случая средняя точность варьируется от 87 % до 96 % при соблюдении условия о достаточном количестве обучающих данных.

Наконец, самый актуальный ввиду активного развития технологий искусственной генераций текста случай – идентификация автора искусственно-сгенерированного кода. В исследовании рассматривается три различных вопроса: разграничение авторства между машиной и человеком, принадлежность двух образцов кода одной и той же языковой модели и определение языковой модели, сгенерировавшей исходный код программы. Для данных, сгенерированных дообученными на исходных кодах с хостинга GitHub моделях семейства GPT, средняя точность составляет 94 %. Следовательно, предложенная методика позволяет не только анализировать как исходные коды, написанные человеком, так и машиной, но и разделять авторство между ними, а также выявлять отличительные особенности каждой языковой модели.

Разработанная методика имеет следующие преимущества:

1. Высокая точность идентификации автора исходного кода.
2. Независимость от языка программирования и квалификации автора-программиста анализируемого исходного кода.
3. Устойчивость к намеренным преобразованиям исходного кода программы посредством применения обфускаторов или следования стандартам кодирования.
4. Способность к обучению на исходных кодах программ, полученных в результате групповой разработки ПО.
5. Способность к выявлению информативных признаков, указывающих на принадлежность исходного кода программы той или иной генеративной модели.

Данная методика легла в основу интеллектуальной системы для идентификации автора исходного кода программы, которая, согласно полученным оценкам эффективности, может применяться для решения реальных прикладных задач.

СПИСОК ИСТОЧНИКОВ

1. Куртукова А.В., Романов А.С. Идентификация автора исходного кода методами машинного обучения. *Труды СПИИРАН*. 2019;18(3):741–765.
2. Kurtukova A., Romanov A., Shelupanov A. Source Code Authorship Identification Using Deep Neural Networks. *Symmetry*. 2020;12:2044.
3. Abuhamad M., AbuHmed T., Mohaisen A., Nyang D. Large-Scale and Language-Oblivious Code Authorship Identification. *In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018;101–114.
4. Zhen L., Chen G., Chen C., Zou Y., Xu S. RoPGen: Towards Robust Code Authorship Attribution via Automatic Coding Style Transformation. *2022 IEEE 44th International Conference on Software Engineering (ICSE)*. 2022;1906–1918.
5. Holland C., Khoshavi N., Jaimes L.G. Code authorship identification via deep graph CNNs. *In Proceedings of the 2022 ACM Southeast Conference (ACM SE '22)*. 2022;144–150.
6. Bogdanova A., Romanov V. Explainable source code authorship attribution algorithm. *Journal of Physics: Conference Series.* 2021;2134:012011. DOI: 10.1088/1742-6596/2134/1/012011.

7. Bogdanova A. Source code authorship attribution using file embeddings. *Companion Proceedings of the 2021 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*. 2021;31–33.
8. Bogomolov E., Kovalenko V., Rebryk Y., Bacchelli A., Bryksin T. Authorship attribution of source code: a language-agnostic approach and ap-plicability in software engineering. *In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2021;932–944.
9. Romanov A., Kurtukova A., Fedotova A., Meshcheryakov R. Natural Text Anonymization Using Universal Transformer with a Self-attention. *Proceedings of the III International Conference on Language Engineering and Applied Linguistics*. 2019;22–37
10. Caliskan-Islam A. Deanonymizing programmers via code stylometry. *Proceedings of the 24th USENIX Security Symposium*. 2015;255–270.
11. GitHub. Доступно по: https://GitHub.com/ (дата обращения: 14.08.2022).
12. GitLab. Доступно по: https://gitlab.com/ (дата обращения: 14.08.2022).
13. Rothe S., Narayan S., Severyn A. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*. 2020;8:264–280.
14. Du Z. *All NLP tasks are generation tasks: A general pretraining framework*. arXiv preprint arXiv:2103.10360. 2021.
15. Floridi L., Chiriatti M. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*. 2020;30(4):681–694.
16. Lee J. S., Hsiang J. Patent claim generation by fine-tuning OpenAI GPT-2. *World Patent Information*. 2020;62:101983.
17. Душейко А. *Генерация лидов новостных текстов с помощью нейронной сети ruGPT-3: магистерская диссертация по направлению подготовки: 45.04.03 Фундаментальная и прикладная лингвистика*. 2022.
18. Pisarevskaya D., Shavrina T. *WikiOmnia: generative QA corpus on the whole Russian Wikipedia*. arXiv preprint arXiv:2204.08009. 2022.
19. Li Z. RoPGen: towards robust code authorship attribution via automatic coding style transformation. *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE). IEEE*. 2022;1906–1918.
20. Cruz-Benito J. Automated source code generation and auto-completion using deep learning: Comparing and discussing current language model-related approaches. *AI*. 2021;2(1):1–16.
21. Open AI. Доступно по: https://openai.com/blog/openai-codex (дата обращения 14.08.2022).
22. GitHub Copilot. Доступно по: https://copilot.GitHub.com (дата обращения 14.08.2022).
23. AlphaCode. Доступно по: https://deepmind.com/blog/article/Competitive-programming-with-AlphaCode (дата обращения: 14.08.2022).
24. Sber AI ruGPT-3. Доступно по: https://developers.sber.ru/portal/tools/rugpt-3 (дата обращения: 14.08.2022).
25. PolyCoder. Доступно по: https://venturebeat.com/2022/03/04/researchers-open-source-code-generating-ai-they-claim-can-beat-openais-codex/ (дата обращения: 14.08.2022).
26. Frantzeskou G., Stamatatos E., Gritzalis S. Identifying authorship by bytelevel n-grams: The source code author profile (SCAP) method. *Int. J. Digit. Evid*. 2007;1:1–18.
27. Wisse W., Veenman C.J. Scripting DNA: Identifying the JavaScript Programmer. *Digit. Investig*. 2015;15:61–71.

REFERENCES

1. Kurtukova, A.V., Romanov, A.S. Identifikaciya avtora iskhodnogo koda metodami mashinnogo obucheniya. *Trudy SPIIRAN = Proceedings of SPIIRAS*. 2019;18:741–765. (In Russ.).
2. Kurtukova A., Romanov A., Shelupanov A. Source Code Authorship Identification Using Deep Neural Networks. *Symmetry*. 2020;12:2044.
3. Abuhamad M., AbuHmed T., Mohaisen A., Nyang D. Large-Scale and Language-Oblivious Code Authorship Identification. *In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018;101–114.
4. Zhen L., Chen G., Chen C., Zou Y., Xu S. RoPGen: Towards Robust Code Authorship Attribution via Automatic Coding Style Transformation. *2022 IEEE 44th International Conference on Software Engineering (ICSE)*. 2022;1906–1918.
5. Holland C., Khoshavi N., Jaimes L.G. Code authorship identification via deep graph CNNs. *In Proceedings of the 2022 ACM Southeast Conference (ACM SE '22)*. 2022;144–150.
6. Bogdanova A., Romanov V. Explainable source code authorship attribution algorithm. *Journal of Physics: Conference Series.* 2021;2134:012011. DOI: 10.1088/1742-6596/2134/1/012011.
7. Bogdanova A. Source code authorship attribution using file embeddings. *Companion Proceedings of the 2021 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*. 2021;31–33.
8. Bogomolov E., Kovalenko V., Rebryk Y., Bacchelli A., Bryksin T. Authorship attribution of source code: a language-agnostic approach and ap-plicability in software engineering. *In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2021;932–944.
9. Romanov A., Kurtukova A., Fedotova A., Meshcheryakov R. Natural Text Anonymization Using Universal Transformer with a Self-attention. *Proceedings of the III International Conference on Language Engineering and Applied Linguistics*. 2019;22–37
10. Caliskan-Islam A. Deanonymizing programmers via code stylometry. *Proceedings of the 24th USENIX Security Symposium*. 2015;255–270.
11. GitHub. Available at: https://GitHub.com/ (accessed 08.14.2022).
12. Gitlab. Available at: https://gitlab.com/ (accessed 08.14.2022).
13. Rothe S., Narayan S., Severyn A. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*. 2020;8:264–280.
14. Du Z. *All NLP tasks are generation tasks: A general pretraining framework*. arXiv preprint arXiv:2103.10360. 2021.
15. Floridi L., Chiriatti M. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines.* 2020;30(4):681–694.
16. Lee J. S., Hsiang J. Patent claim generation by fine-tuning OpenAI GPT-2. *World Patent Information*. 2020;62:101983.
17. Dusheiko A. *Lead generation of news texts using the ruGPT-3 neural network: Master's thesis in the field of preparation. 45.04.03 Fundamental and applied linguistics*. 2022. (In Russ.).
18. Pisarevskaya D., Shavrina T. *WikiOmnia: generative QA corpus on the whole Russian Wikipedia*. arXiv preprint arXiv:2204.08009. 2022.
19. Li Z. RoPGen: towards robust code authorship attribution via automatic coding style transformation. *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE). IEEE*. 2022;1906–1918.

20. Cruz-Benito J. Automated source code generation and auto-completion using deep learning: Comparing and discussing current language model-related approaches. *AI*. 2021;2(1):1–16.
21. Open AI. Available at: https://openai.com/blog/openai-codex (accessed 08.14.2022).
22. GitHub Copilot. Available at: https://copilot.GitHub.com (accessed 08.14.2022).
23. AlphaCode. Available at: https://deepmind.com/blog/article/Competitive-programming-with-AlphaCode (accessed 08.14.2022).
24. Sber AI ruGPT-3. Available at: https://developers.sber.ru/portal/tools/rugpt-3 (accessed 08.14.2022).
25. Polycoder. Available at: https://venturebeat.com/2022/03/04/researchers-open-source-code-generating-ai-they-claim-can-beat-openais-codex/ (accessed 08.14.2022).
26. Frantzeskou G., Stamatatos E., Gritzalis S. Identifying authorship by bytelevel n-grams: The source code author profile (SCAP) method. *Int. J. Digit. Evid*. 2007;1:1–18.
27. Wisse W., Veenman C.J. Scripting DNA: Identifying the JavaScript Programmer. *Digit. Investig*. 2015;15:61–71.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Куртукова Анна Владимировна, аспирант каф. комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС) Томского государственного университета систем управления и радиоэлектроники (ТУСУР), Томск, Российская Федерация.
e-mail: av.kurtukova@gmail.com

Anna V. Kurtukova, Postgraduate Student, Department of Complex Information Security of Electronic Computer Systems, Tomsk State University of Control Systems and Radioelectronics (TUSUR), Tomsk, Russian Federation.

Романов Александр Сергеевич, канд. техн. наук, доцент каф. комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС) Томского государственного университета систем управления и радиоэлектроники (ТУСУР), Томск, Российская Федерация.
e-mail: alexx.romanov@gmail.com

Aleksandr S. Romanov, Candidate of Technical Sciences, Associate Professor, Department of Complex Information Security of Electronic Computer Systems, Tomsk State University of Control Systems and Radioelectronics (TUSUR), Tomsk, Russian Federation.

Шелупанов Александр Александрович, д-р техн. наук, профессор, зав. каф. комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС), президент Томского государственного университета систем управления и радиоэлектроники (ТУСУР), Томск, Российская Федерация.
e-mail: saa@fb.tusur.ru

Aleksandr A. Shelupanov, Doctor of Technical Sciences, Professor, Head of the Department of Complex Information Security of Electronic Computer Systems, President of Tomsk State University of Control Systems and Radioelectronics (TUSUR), Tomsk, Russian Federation.

Федотова Анастасия Михайловна, студент каф. безопасности информационных систем (БИС) Томского государственного университета систем управления и радиоэлектроники (ТУСУР), Томск, Российская Федерация.
e-mail: fedotova.a.747@e.tusur.ru

Anastasia M. Fedotova, Student, Information Systems Security Department, Tomsk State University of Control Systems and Radioelectronics (TUSUR), Tomsk, Russian Federation.

Статья поступила в редакцию 04.09.2022; одобрена после рецензирования 18.09.2022; принята к публикации 24.09.2022.

The article was submitted 04.09.2022; approved after reviewing 18.09.2022; accepted for publication 24.09.2022.