СЕКЦИЯ 13 Проектирование программно-информационных систем

УДК [004.89](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwimpc_B9s_vAhXyiIsKHb_pDR4QFjAAegQIBhAD&url=https%3A%2F%2Fteacode.com%2Fonline%2Fudc%2F62%2F621.224.html&usg=AOvVaw0P_0eq0wyUbKiPPSPH_0qx)

**ОСНОВНЫЕ ПАРАДИГМЫ МАШИННОГО ОБУЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОГРАММИРОВАНИЯ**

***Сафронов Никита Сергеевич***safronovns@student.bmstu.ru

КФ МГТУ им. Н.Э. Баумана

***Белов Юрий Сергеевич*** ysbelov@bmstu.ru

КФ МГТУ им. Баумана

*В данной статье рассматриваются три основные парадигмы автоматизации кода с использованием машинного обучения: генерация кода по описанию (Description-to-Code), генерация описания по коду (Code-to-Description) и модификация кода на основе кода (Code-to-Code). Парадигма Description-to-Code позволяет автоматизировать создание кода на основе текстовых описаний. Парадигма Code-to-Description сосредоточена на автоматическом создании документации и пояснений к уже существующему коду. Парадигма Code-to-Code направлена на оптимизацию, рефакторинг и исправление ошибок в коде.*

***Ключевые слова:*** *автоматизация кода, машинное обучение, генерация кода по описанию, генерация описания по коду, модификация кода, создание кода, генерация документации, оптимизация кода, рефакторинг кода.*

Помимо традиционных инструментов, разработчики могут использовать модели машинного обучения для автоматизации рутинных задач. В рамках этой статьи рассмотрим три основные парадигмы автоматизации кода с использованием методов ML: генерация кода по описанию («Description-to-Code»), генерация описания по коду («Code-to-Description») и модификация кода на основе кода («Code-to-Code») [1]. Эти парадигмы ориентированы на различные аспекты работы с программным обеспечением.

Парадигма «Description-to-Code» предполагает, что модель создает исполняемый код на основе входных данных (текстовое описание или другие структурированные данные). Этот подход позволяет переводить задачи, описанные на естественном языке, в машинные инструкции, что упрощает процесс разработки, особенно в случаях, когда нет четкого требования к реализации, но известен желаемый результат. В рамках данной парадигмы выделяются три основных направления:

*Создание кода на основе текстовых описаний*. Модели могут генерировать код по запросам на естественном языке, создавая функции и методы на таких языках, как Python, JavaScript и SQL [2].

*Программирование по примерам (Programming by Example, PBE)*. В рамках этого направления модели создают код, основываясь на предоставленных примерах входных и выходных данных. Указывается несколько примеров, и модель создает функцию или алгоритм, который обеспечивает аналогичный результат для других входных данных.

*Генерация интерфейсов на основе изображений*. В некоторых случаях создание пользовательских интерфейсов можно автоматизировать с помощью моделей, которые преобразуют визуальные дизайны в код HTML/CSS или JavaScript.

Парадигма «Description-to-Code» требует моделей, которые могут эффективно обрабатывать текстовые данные и создавать логически корректный код. Наиболее подходящие для этой парадигмы архитектуры — трансформеры, которые обучены на больших объемах текстовых и кодовых данных. Эти модели учитывают контекст задачи, имеют гибкость в генерации и обладают высокой точностью [3].

Парадигма «Code-to-Description» направлена на преобразование программного кода в текстовое описание, которое делает код более понятным для разработчиков и других специалистов. Эта парадигма востребована для задач создания документации, комментирования, и упрощения понимания логики сложных алгоритмов. В рамках парадигмы выделяются следующие направления:

*Автоматическая генерация документации*. Модель может анализировать код и создавать описание функций и методов, объясняя их назначение и параметры.

*Суммаризация кода*. Часто бывает необходимо упростить сложный код, особенно если он использует многократные вычисления или нестандартные подходы. Модели, работающие по принципу «Code-to-Description», могут генерировать краткие и точные описания, делая код более доступным для широкого круга разработчиков.

*Автоматическое комментирование кода*. Эта задача ориентирована на создание комментариев к коду для улучшения его читаемости. ML-модели могут анализировать логические блоки кода и предлагать поясняющие комментарии, указывающие на цель каждого блока и общую логику выполнения, что упрощает работу с чужим кодом.

Для задач «Code-to-Description» хорошо подходят модели, обученные на обработке текстов, такие как BERT и T5. Они способны интерпретировать код и преобразовывать его в содержательные описания, что делает их полезными для создания документации. Преимуществами этих моделей являются: понимание семантики кода, высокая точность генерации описаний и адаптируемость под конкретные языки программирования и стили кода.

Парадигма «Code-to-Code» ориентирована на преобразование и улучшение уже существующего кода. Она охватывает широкий спектр задач, связанных с улучшением структуры и качества кода, таких как:

*Автоматическое исправление ошибок*. Модели могут находить и устранять ошибки, что особенно актуально для автоматического «ремонта» программного обеспечения (Automated Program Repair, APR). Такие модели анализируют код и его выполнение, выявляют потенциальные проблемы и предлагают исправления, что способствует повышению стабильности.

*Рефакторинг кода*. Рефакторинг направлен на улучшение структуры и читаемости кода без изменения его функционала. Модели «Code-to-Code» могут автоматизировать этот процесс, преобразовывая сложные участки кода в более понятные и поддерживаемые версии.

*Оптимизация производительности*. «Code-to-Code» также помогает в оптимизации, заменяя неэффективные конструкции более производительными. Это особенно важно для программ, работающих с высоконагруженными системами или обработкой больших объемов данных.

Для задач «Code-to-Code» обычно используют специализированные трансформеры, обученные на больших объемах данных из репозиториев программного обеспечения. Эти модели могут находить оптимальные паттерны и предлагать улучшения в коде. Преимущества этих моделей: контекстное понимание ошибок, поддержка структуры и читаемости, оптимизация затрат на обслуживание написанного кода.

Применение методов машинного обучения для автоматизации программирования открывает перед разработчиками новые возможности. Каждая из рассмотренных парадигм позволяет автоматизировать различные этапы работы с программным кодом: «Description-to-Code» помогает автоматизировать написание кода, «Code-to-Description» упрощает процесс создания документации и комментирования, а «Code-to-Code» улучшает структуру и производительность. Автоматизация на базе машинного обучения способствует ускорению процесса разработки, снижению затрат и повышению качества создаваемого программного обеспечения.

**Литература**

[1] Dehaerne E., Dey B., Halder S., Gendt S., Meert W. Code Generation Using Machine Learning: A Systematic Review // IEEE Access. — 2022.

[2] Yin Pengcheng, Neubig Graham. A Syntactic Neural Model for General-Purpose Code Generation // CoRR. – 2017.

[3] Program synthesis using natural language / Aditya Desai, Sumit Gulwani, Vineet Hingorani et al. // Proceedings of the 38th International Conference on Software Engineering / ACM. – 2016. – P. 345–356.

**MACHINE LEARNING PARADIGMS FOR CODE AUTOMATION**

**Safronov Nikita Sergeevich** safronovns@student.bmstu.ru

BMSTU (KB)

**Belov Yuri Sergeevich** ysbelov@bmstu.ru

BMSTU (KB)

*This paper examines three main paradigms of code automation using machine learning: Description-to-Code, Code-to-Description, and Code-to-Code. The Description-to-Code paradigm enables the automated generation of code from textual descriptions. The Code-to-Description paradigm focuses on the automatic creation of documentation and explanations for existing code. The Code-to-Code paradigm is aimed at optimizing, refactoring, and correcting errors within code.*

***Keywords:*** *сode automation, machine learning, Description-to-Code, Code-to-Description, Code-to-Code, code generation, documentation generation, code optimization, code refactoring.*