УДК 004.832.22
DOI: 10.26102/2310-6018/2025.48.1.030

Оценка качества полученного результата в задаче генерации исходного кода по изображению

И.В. Никитин✉

Российский экономический университет имени Г.В. Плеханова, Москва, Российская Федерация

Резюме. Исследование представляет собой оценку возможности построения системы выполнения функциональных тестов для задачи генерации исходного кода из изображения. Существует много различных метрик для оценки качества предсказанного нейронной сетью текста: от математических, таких как BLEU, Rogue, до таких, которые используют другую модель для оценки, как, например, BERTScore, BLEURT. Однако проблема генерации исходного кода программы состоит в том, что код представляет собой набор инструкций для выполнения определенной задачи. Актуальность состоит в том, что в публикациях, связанных с системой pix2code, отсутствовало упоминание об автоматизированной тестовой среде, которая сможет проверить соответствие полученного кода заданным условиям. В ходе проделанной работы была реализована подсистема, которая в автоматическом режиме может получить информацию о различиях между изображением, основанном на предсказанном коде, и изображении, основанном на эталонном коде. Также результаты работы этой системы сопоставлены с метрикой BLEU. Проведенный эксперимент позволяет сделать вывод о том, что значение BLEU и результаты выполнения тестов не имеют явной зависимости между собой, а значит, функциональные тесты необходимы для дополнительной проверки эффективности работы модели.

Ключевые слова: кодогенерация, изображение, машинное обучение, BLEU, функциональные тесты.

Для цитирования: Никитин И.В. Оценка качества полученного результата в задаче генерации исходного кода по изображению. *Моделирование, оптимизация и информационные технологии*. 2025;13(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1830 DOI: 10.26102/2310-6018/2025.48.1.030

Assessing the quality of the result in the problem of source code generation from an image

I.V. Nikitin✉

Plekhanov Russian University of Economics, Moscow, the Russian Federation

Abstract. This study is an assessment of the feasibility of building a system for executing functional tests for the task of generating source code from an image. There are many different metrics for assessing the quality of text predicted by a neural network: from mathematical ones, such as BLEU, Rogue, and those that use another model for evaluation, such as BERTScore, BLEURT. However, the problem with generating source code for a program is that the code is a set of instructions to perform a specific task. The relevance is that in publications related to the pix2code system, there was no mention of an automated test environment that can check whether the resulting code meets the specified conditions. In the course of the work done, a subsystem was implemented that can automatically obtain information about the differences between an image based on a predicted code and an image based on a reference code. Also, the results of this system are compared with the BLEU metric. As a result of the experiment, we can conclude that the BLEU value and the execution of tests do not have an obvious relationship between them, which means that tests are necessary for additional checks of the efficiency of the model.

Keywords: code generation, image, machine learning, BLEU, functional tests.

For citation: Nikitin I.V. Assessing the quality of the result in the problem of source code generation from an image. *Modeling, Optimization and Information Technology*. 2025;13(1). (In Russ.). URL: https://moitvivt.ru/ru/journal/pdf?id=1830 DOI: 10.26102/2310-6018/2025.48.1.030

Введение

Системы машинного обучения постепенно входят в повседневную жизнь, помогая решать те или иные задачи: создать изображение или видео-открытку по текстовому запросу за несколько минут вместо их ручного создания в течение нескольких часов; редактирование больших объемов текстов или выделение из таких текстов основных мыслей и т. д. Все эти задачи могут быть решены, например, с использованием ChatGPT или другого чат-бота. Еще одна задача, где может использоваться система машинного обучения – это создание исходного кода приложения. Одна из частей любого приложения – его интерфейс, с которым взаимодействует пользователь. Чтобы им было удобно пользоваться, его необходимо правильно структурировать, для чего создаются макеты будущего приложения. В данном случае системы машинного обучения могут использоваться для получения исходного кода программы на основе изображения того самого макета, который необходимо реализовать. Однако в процессе выполнения данной задачи необходимо решить вопрос о качестве обучения, чтобы понять, что полученный результат – это хороший результат.

В статье [1] рассматривался вариант по актуализации версий библиотек TensorFlow и Keras. В качестве оценки полученного результата для сравнения использовались два показателя, которые изначально заложены в TensorFlow – это Accuracy и Loss. Первый из этих параметров отвечает за долю правильных ответов в процессе обучения, причем значение для конкретного предсказания бинарное, а второй – это сумма ошибок, совершаемых моделью во время предсказания. Использование этих метрик может дать общее представление о качестве обучения системы. Кроме того, их расчет уже встроен в библиотеку TensorFlow. Однако, так как это общие метрики, которые могут быть использованы для любого обучения с использованием библиотеки, от них сложно получить более точную информацию о происходящем в процессе обучения, а также действительно ли система обучается, а не просто научилась угадывать параметры. На эти же метрики, например, опирается исследование [2].

В работах [3, 4] для оценки качества полученных результатов используется процент полученных ошибок после обучения. В статье [3] уточняется, что именно относится к ошибке: либо ошибка в классификации DSL-токена, либо разница в длине полученного ответа. С одной стороны, такой подход лучше, чем просто ориентирование на общие параметры, так как в данном случае рассматриваются уже специфичные для системы генерации исходного кода ошибки в определении необходимого токена. Однако не совсем понятно, как именно считается и усредняется это значение, а одинаковая длина полученного результата также не может однозначно утверждать, что полученный результат верный.

В результате, на момент проведения данного исследования сложно однозначно оценить эффективность получившегося исходного кода на основе изображения макета. При этом приведенные примеры исследований не пытаются системно подойти к оценке качества полученного исходного кода в той среде исполнения, для которой делается предсказание. В работах [3, 4] есть примеры точечного сравнения полученных результатов, но этого может быть мало для однозначного утверждения, что система выдает качественный результат, плюс необходимо учитывать, что генерация исходного

кода ограничена в 150 токенов. При увеличении числа токенов в ответе генерации ручная проверка полученного результата может быть уже затруднительна. Актуальность данной работы связана как с улучшением потребности в оценке полученных результатов для понимания качества полученного результата и всего процесса обучения, так и с улучшением возможности масштабирования оценки качества для ее применения на более сложных и комплексных примерах.

В рамках данного исследования, проведен эксперимент по оценке качества предсказания исходного кода программы на основе изображения макета с использованием следующих метрик: Accuracy, Loss, BLEU, а также реализована подсистема для оценки качества получившегося результата на основе функционального теста. В конце сделан вывод относительно эффективности применения этих подходов.

Обзор существующих метрик для оценки качества предсказанного текста

Так как полученный исходный код, по сути, представляет из себя текстовую информацию (в более специфичном и структурированном виде, чем обычный текст), для оценки качества полученного результата можно использовать те же метрики, что и для обычного текста. Глобально, эти метрики можно разделить на две группы: те метрики, которые основаны на совпадении текстов и могут быть рассчитаны математически, и те, которые основаны на использовании нейронных сетей. Однако, этого набора метрик недостаточно, так как исходный код программы – это не просто текст, который несет смысловую нагрузку, а текст, набранный определенным образом с определенными правилами и определенной последовательностью в символах, который должен выполнять определенную задачу. То есть полученный результат должен выполняться в среде исполнения в зависимости от языка программирования. Для рассматриваемой системы по генерации исходного кода на основе макета такой средой может являться веб-браузер для веб-страниц или мобильная операционная система для мобильного приложения. Кроме того, полученный исходный код должен выполнять ту задачу, которую перед ним ставили: для рассматриваемой в рамках данной работы задачи, он должен быть таким же, как на исходном изображении.

К первой группе метрик, основанной на совпадении текстов или n-грамм, относятся метрики, которые сравнивают полученный результат и ожидаемый результат, а далее на основе совпадения n-грамм и статистики получают итоговую метрику качества или совпадения текстов. К таким метрикам можно отнести BLEU, Nist, Rogue, chrF++. BLEU [5] – один из самых распространенных методов для получения оценки качества текстовой информации, основанный на сравнении n-грамм текстовой информации. Метод основан на отношении количества n-грамм в исходном тексте и в эталонном с добавлением таких коэффициентов, как «штраф» за слишком короткий ответ. Схожим образом работает Nist [6], но с присвоением весов в зависимости от частоты встречаемости. Rogue [7] продолжает и дополняет идеи, заложенные в BLEU. Так, кроме точности (precision), в Rogue метрики рассчитывается и учитывается полнота полученных результатов (recall) и F1 метрика (среднее гармоническое между точностью и полнотой). Одной из самых актуальных метрик, основанных на совпадении текстов, является chrF++ [8], которая в своей основе также содержит подсчет совпадений символов, но уже с учетом того, что эти символы должны составлять правильную последовательность. Общая идея, которая стоит за всеми этими метриками, состоит в математическом сравнении полученных текстов. С одной стороны, высокие показатели метрик между эталонным текстом и предсказанным может означать, что тексты схожи, но, с другой стороны, при этом по смыслу полученный результат может быть совершенно другой по сравнению с результатом эталонным или не иметь смысла, а

являться просто правильным по количеству и наличию набором слов. Так, например, в работе [9] было показано, что метрика BLEU плохо помогает оценить реальное качество полученного кода, не может оценить функциональное качество, то есть то, насколько исходный код рабочий и способен выполнять поставленную перед ним функцию.

Вторая группа метрик – эти метрики, которые основаны на использовании еще одной нейронной сети для оценки качества работы исследуемой модели. Примерами таких метрик являются, например, BERTScore [10], BLEURT [11] и Comet [12]. За ними стоит общая идея: использовать предобученную отдельную модель для оценки работы текущей модели (в зависимости от метрики различаются те самые предобученные модели, а также способ их использования и подсчета). Основное свойство, благодаря которому этим метрикам можно верить, заключается в эмбеддинге текстовых представлений, где слова могут быть представлены в виде векторов. Благодаря этому свойству есть возможность оценивать не точное совпадение слов, а делать оценку в зависимости от того, насколько ожидаемый и полученный результат схожи между собой по смыслу, что убирает необходимость в проверке точного совпадения слов между текстами. Однако данная группа метрик плохо подходит для оценки качества полученного исходного кода по причине того, что необходимо проверять не просто смысл и быть близким к ожидаемому результату, а выполнять именно то, что требуется от этого исходного кода.

Для оценки функционального качества кода были придуманы отдельные метрики, специализирующиеся именно на исходном коде программы: RUBY [13] и CodeBlue [14]. Идея, заложенная в метрику RUBY, заключается в том, чтобы оценить исходный код на разных уровнях абстракции при наличии соответствующей возможности: на уровне графа программы, на уровне абстрактного синтаксического дерева программы, на уровне лексем. При этом в исследовании было отмечено, что чем выше уровень абстракции, тем хуже результаты, поэтому сама метрика RUBY является составной: сначала необходимо попробовать оценить граф программы, если это не получается, то переходить к абстрактному синтаксическому дереву, а после уже к самим лексемам. Важным замечанием является то, что исследование было проведено для работы с миграцией исходного кода, то есть перенос алгоритма, реализованного на одном языке программирования на другой с сохранением оригинальной логики. Из-за этого нельзя однозначно утверждать, что метрика подходит для решения любой задачи, однако она может быть использована для других задач генерации исходного кода.

Другая метрика для оценки качества созданного исходного кода – CodeBLEU. Схожим с RUBY образом, данная метрика также состоит из нескольких метрик: обычного BLEU, взвешенного BLEU, совпадение синтаксического абстрактного дерева и совпадение потока выполнения программы. Однако в отличие от RUBY, в CodeBLEU все 4 подметрики принимают участие в том, чтобы получить итоговую метрику, а не только одна.

Стоит обратить внимание на исследование [15]. В нем авторы пытаются оценить различные метрики качества полученного кода, сравнивая метрики как с эталонной разметкой, которую сделал человек, так и между собой. В частности, сравниваются такие метрики как BLEU, ROGUE-L, CodeBLEU, RUBY. В результате, авторы пришли к выводу о том, что ни одна из метрик не достигает хороших результатов по сравнению с разметкой, сделанной человеком. Этот результат говорит о том, что хоть различные метрики и пытаются быть максимально объективными с точки зрения оценки качества полученного результата, но они не всегда достигают его.

Эксперимент

Проведенный анализ не позволяет однозначно сказать, какую метрику лучше использовать для задачи генерации исходного кода по изображению макета. Однако можно увидеть, что часто упоминается необходимость проверки полученного предсказания с помощью тестов. Так как программа – это исходный код, который выполняет определенную поставленную задачу, можно написать такие тесты, которые проверят, выполняет ли полученное предсказание необходимую задачу или нет. Важно отметить, что эксперимент выполнен только для веб-страниц, которые в качестве итогового исходного кода представляют собой HTML-разметку, хотя pix2code предполагает предсказание также для iOS и Android приложений. Связано это с техническими ограничениями по созданию инфраструктуры по запуску тестов.

Исходная задача заключается в том, чтобы получить такой исходный код, который при выполнении в своей среде исполнения будет давать результат, аналогичный тому, который изначально закладывался эталоном. Поэтому для выполнения функциональных тестов была построена отдельная подсистема, работающая по следующему алгоритму. В обученную систему на вход передается эталонное изображение высокого разрешения (2400 на 1380 пикселей). После получения предсказания подсистема получает изображение того, что будет видеть пользователь при исполнении предсказанного исходного кода. Для этого используется библиотека Playwright[1], которая позволяет автоматически запустить веб-браузер, а после сделать снимок экрана (в данном случае делается снимок 1280 на 986). Выбор такого размера обусловлен особенностями самой библиотеки Playwright, которая на более высоких разрешениях делает изображения, на которых уже сложно что-то сравнивать. Аналогичным образом получается изображение на основе эталонного кода, связанного с изображением, которое было изначально загружено, и также делается снимок экрана размером 1280 на 986. После чего из эталонного изображения попиксельно вычитается предсказанное изображение, берется попиксельное отношение к эталонному изображению и переводится в проценты. В результате можно получить процент, на который отличается предсказанное изображение от эталонного. Чем меньше полученный процент, тем ближе предсказание к эталонному результату. При идеальном предсказании этот процент должен быть равен 0.

Для сравнения предлагается использовать значение метрики BLEU. Хотя ранее и упоминалось, что она не является самой эффективной для оценки качества исходного кода, однако согласно исследованию [15] утверждать, что она совсем не эффективна в сравнении с другими метриками нельзя. Кроме того, это самый распространенный метод оценки текста, поэтому эффективность его применения может быть понятна любому, кто занимается оценкой качества предсказанного текста.

Таким образом, в рамках эксперимента выполнено обучение системы из статьи [1] – pix2code, переделанной под версию библиотеки TensorFlow 2.17. После каждой эпохи обучения вычислялось 4 значения: Accuracy, Loss, BLEU и разница изображений, которая представляет собой результат выполнения функциональных тестов. Обучение проводилось на протяжении 50 эпох на процессоре M1. В качестве набора данных использовался набор из оригинальной системы pix2code. Исходный код программы можно найти по источнику[2]. Результаты изменения метрики Accuracy в процессе обучения можно увидеть на Рисунке 1, метрики Loss – на Рисунке 2, метрики BLEU – на Рисунке 3, результат выполнения функциональных тестов – на Рисунке 4.

[1] Playwright for Python. Playwright. URL: https://playwright.dev/python (дата обращения: 19.02.2025).
[2] IlyaNikk / html-code-generation. GitHub. URL: https://github.com/IlyaNikk/html-code-generation (дата обращения: 19.02.2025).

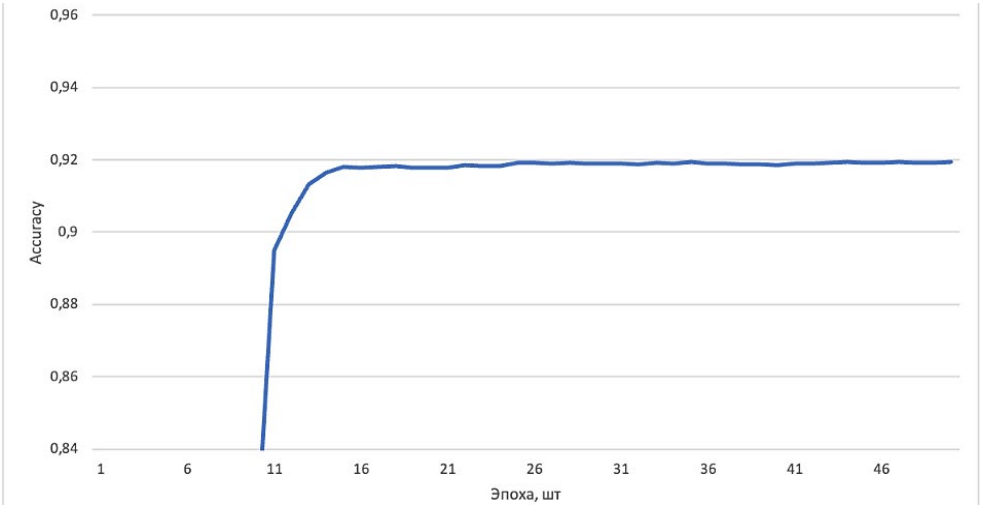

Рисунок 1 – Изменения значения метрики Accuracy в процессе обучения
Figure 1 – Changes of Accuracy's value during training

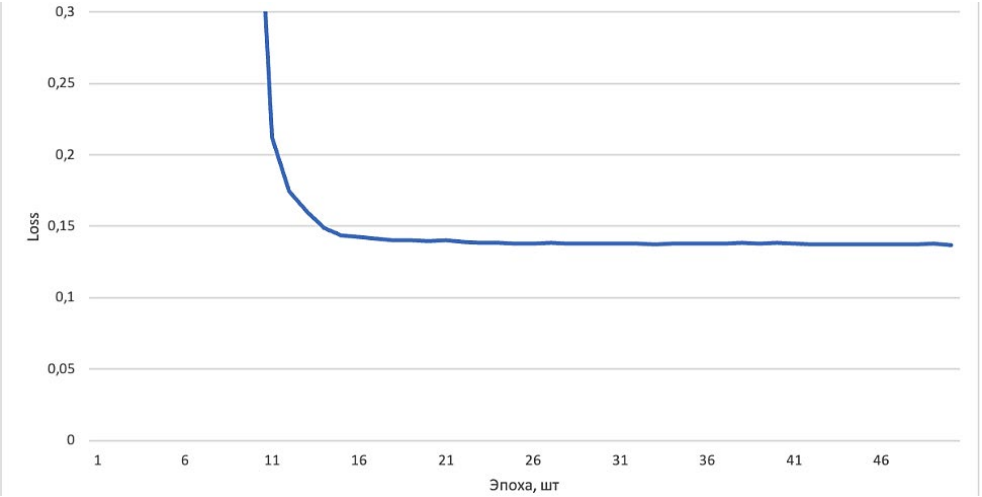

Рисунок 2 – Изменения значения метрики Loss в процессе обучения
Figure 2 – Changes of Loss's value during training

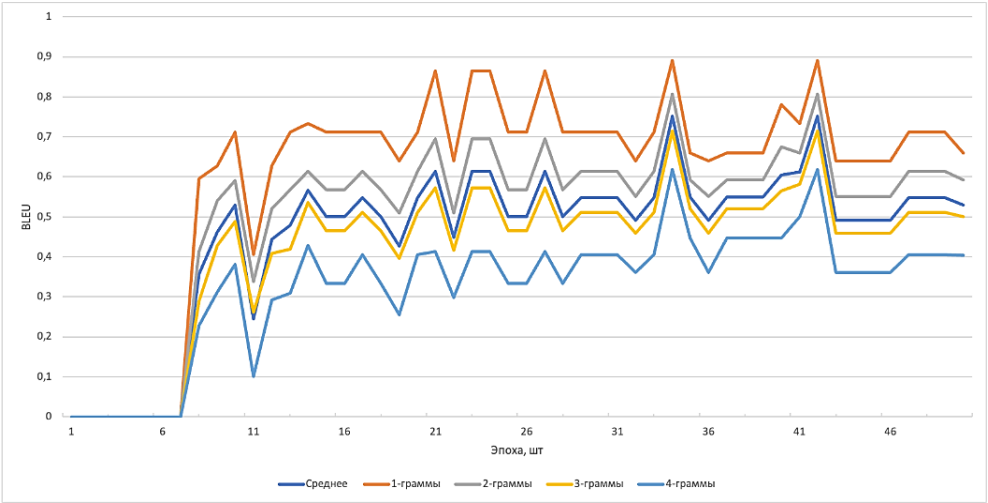

Рисунок 3 – Изменения значения метрики BLEU в процессе обучения
Figure 3 – Changes of BLEU's value during training

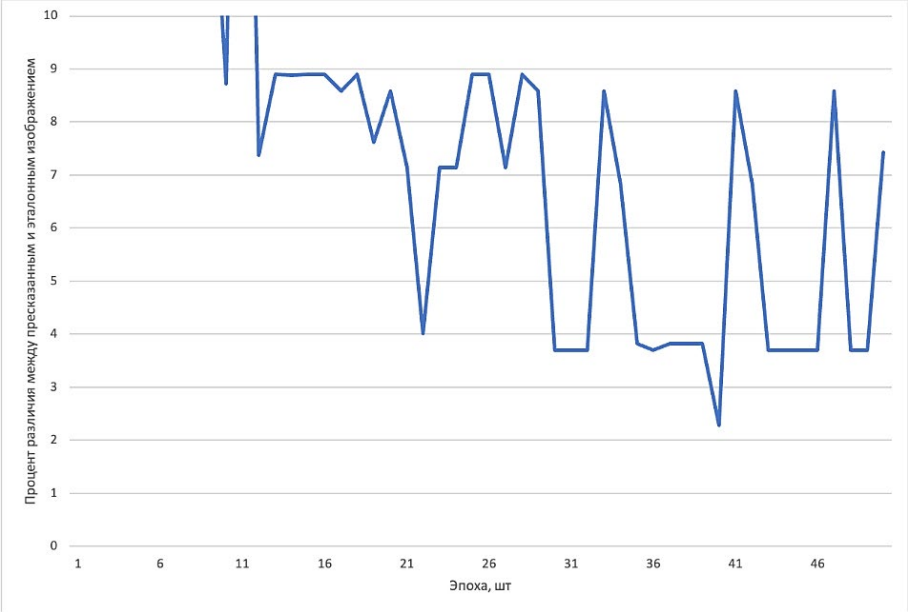

Рисунок 4 – Изменения результата выполнения функциональных тестов в процессе обучения
Figure 4 – Changes of functional test's result during training

Как можно увидеть, значения Accuracy и Loss довольно быстро достигают своего максимума к порядку 20-й эпохи и больше не меняются, оставаясь в пределах одного значения. Данные для построения графиков BLEU и выполнения функционального теста взяты для одного изображения с целью проследить, как меняются значения. Из графика BLEU можно увидеть, что чем длиннее n-граммы, тем менее точное предсказание. В среднем значение получается около 0,5, а значение для 1-граммы – в среднем 0,8. В свою очередь значение для функциональных тестов ближе к концу обучения принимает значение порядка 4 %. Для изображения 1280×986 это составляет порядка 50000 пикселей, что довольно много. Однако можно заметить, что значения BLEU и функциональных тестов меняются на протяжении всех 50-ти эпох, хотя значения Loss и Accuracy уже нет. Из этого можно сделать вывод о том, что использование только метрик Loss и Accuracy для принятия решения о том, что обучение завершено, недостаточно. При этом какой-то четкой зависимости между BLEU и функциональными тестами нет. В результате, однозначно сказать, что какой-то из этих подходов лучше другого, нельзя. Однако в предыдущих работах, связанных с системой pix2code, не было реализовано каких-либо функциональных тестов, которые позволят проверить работоспособность полученного исходного кода и его соответствие тому, что изначально закладывалось в него. Использование данной подсистемы в будущем позволит получать данные о состоянии обучения и способности модели к точному предсказанию исходного кода во время будущих изменений и доработок данной системы.

Заключение

В ходе исследования была выдвинута гипотеза о том, что значение стандартных метрик Accuracy и Loss, которые предоставляются библиотекой TensorFlow, недостаточно для определения качества обучения модели. В процессе изучения материалов, связанных с используемыми метриками для оценки качества полученных результатов для текстовой информации и, в частности, исходного кода, во-первых, было решено использовать BLEU-метрику как самый популярный способ оценки качества текста, и, во-вторых, результаты выполнения функциональных тестов, так как эти тесты

позволяют ответить на вопрос о том, выполняет ли полученный исходный код ту задачу, которая перед ним ставится.

В рамках эксперимента, исходный код программы, реализованный в рамках статьи [1] был доработан для того, чтобы появилась возможность получить все необходимые метрики и оценить их результат. По итогам проведенного эксперимента можно увидеть, что использование одних только метрик Accuracy и Loss недостаточно и необходимо использование других метрик в том числе. Однако нельзя однозначно утверждать, что необходим выбор только метрики BLEU или функциональных тестов так как какой-либо явной зависимости между результатами данных нет. В результате сделан вывод о том, что необходимо учитывать, как минимум, обе эти метрики.

В будущих исследованиях планируется активно использовать реализованный модуль для выполнения функциональных тестов. Ранее такая подсистема не была реализована в работах, связанных с системой pix2code. При этом текущий результат показывает довольно большой процент ошибки. Теперь же такая подсистема может помочь в оценке будущих доработок системы и дать более точный ответ о том, насколько предсказанный код ближе к эталону с точки зрения его отображения.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Никитин И.В. Влияние версии библиотеки TensorFlow на качество генерации кода по изображению. *Моделирование, оптимизация и информационные технологии.* 2024;12(4). https://doi.org/10.26102/2310-6018/2024.47.4.040
   Nikitin I.V. Influence of the TensorFlow library's version on the quality of code generation from an image. *Modeling, Optimization and Information Technology.* 2024;12(4). (In Russ.). https://doi.org/10.26102/2310-6018/2024.47.4.040
2. Zou D., Wu G. Automatic Code Generation for Android Applications Based on Improved Pix2code. *Journal of Artificial Intelligence and Technology.* 2024;4(4):325–331. https://doi.org/10.37965/jait.2024.0515
3. Beltramelli T. pix2code: Generating Code from a Graphical User Interface Screenshot. In: *EICS '18: Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 19–22 June 2018, Paris, France.* New York: Association for Computing Machinery; 2018. https://doi.org/10.1145/3220134.3220135
4. Zhu Zh., Xue Zh., Yuan Z. Automatic Graphics Program Generation Using Attention–Based Hierarchical Decoder. In: *Computer Vision – ACCV 2018: 14th Asian Conference on Computer Vision: Revised Selected Papers: Part VI, 02–06 December 2018, Perth, Australia.* Cham: Springer; 2019. pp. 181–196. https://doi.org/10.1007/978-3-030-20876-9_12
5. Papineni K., Roukos S., Ward T., Zhu W.-J. BLEU: a Method for Automatic Evaluation of Machine Translation. In: *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 07–12 July 2002, Philadelphia, USA.* Stroudsburg: Association for Computational Linguistics; 2002. pp. 311–318. https://doi.org/10.3115/1073083.1073135
6. Doddington G. Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In: *HLT '02: Proceeding of the Second International Conference on Human Language Technology Research, 24–27 March 2002, San Diego, USA.* San Francisco: Morgan Kaufmann Publishers Inc.; 2002. pp. 138–145. https://doi.org/10.3115/1289189.1289273
7. Lin Ch.-Ye. ROGUE: A Package for Automatic Evaluation of Summaries. In: *Proceedings of the Workshop on Text Summarization Branches Out, 25–26 July 2004, Barcelona, Spain.* Association for Computational Linguistics; 2004. pp. 74–81.

8. Popović M. chrF++: words helping character n-grams. In: *Proceedings of the Second Conference on Machine Translation, 07–08 September 2017, Copenhagen, Denmark.* Association for Computational Linguistics; 2017. pp. 612–618. https://doi.org/10.18653/v1/W17-4770
9. Hendrycks D., Basart S., Kadavath S., et al. Measuring Coding Challenge Competence With APPS. In: *35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks, 06–14 December 2021, Online.* https://doi.org/10.48550/arXiv.2105.09938
10. Zhang T., Kishore V., Wu F., Weinberger K.Q., Artzi Yo. BERTScore: evaluating Text Generation with BERT. In: *8th International Conference on Learning Representations, ICLR 2020, 26–30 April 2020, Addis Ababa, Ethiopia.* 2020. https://doi.org/10.48550/arXiv.1904.09675
11. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 02–07 June 2019, Mineapolis, USA.* Association for Computational Linguistics; 2019. pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423
12. Rei R., Stewart C., Farinha A.C., Lavie A. COMET: A Neural Framework for MT Evaluation. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 16–20 November 2020, Online.* Association for Computational Linguistics; 2020. pp. 2685–2702. https://doi.org/10.18653/v1/2020.emnlp-main.213
13. Tran N., Tran H., Nguyen S., Nguyen H., Nguyen T. Does BLEU Score Work for Code Migration? In: *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC), 25–26 May 2019, Montreal, USA.* IEEE; 2019. pp. 165–176. https://doi.org/10.1109/ICPC.2019.00034
14. Ren Sh., Guo D., Lu Sh., et al. CodeBLEU: a Method for Automatic Evaluation of Code Synthesis. arXiv. URL: https://doi.org/10.48550/arXiv.2009.10297 [Accessed 19th February 2025].
15. Evtikhiev M., Bogomolov E., Sokolov Ya., Bryksin T. Out of the BLEU: How Should We Assess Quality of the Code Generation Models? *Journal of Systems and Software.* 2023;203. https://doi.org/10.1016/j.jss.2023.111741

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Никитин Илья Владимирович, аспирант, Российский экономический университет имени Г.В. Плеханова, Москва, Российская Федерация.
e-mail: vic096@yandex.ru

Ilya V. Nikitin, postgraduate, Plekhanov Russian University of Economics, Moscow, the Russian Federation.

Статья поступила в редакцию 20.02.2025; одобрена после рецензирования 04.03.2025; принята к публикации 11.03.2025.

The article was submitted 20.02.2025; approved after reviewing 04.03.2025; accepted for publication 11.03.2025.