В. М. Мальцев, И. А. Назаров, Е. А. Павлова
Тюменский государственный университет, г. Тюмень
УДК 004.94

ИССЛЕДОВАНИЕ ПРИМЕНИМОСТИ ГЕНЕРАТИВНОЙ НЕЙРОННОЙ СЕТИ В КАЧЕСТВЕ КЛАССИФИКАТОРА РЕШЕНИЙ ЗАДАНИЙ ПО ПРОГРАММИРОВАНИЮ

Аннотация. Данная статья посвящена исследованию проверки решений заданий по программированию при помощи генеративной нейронной сети. Был проведен анализ возможностей для проверки кода к задачам разных тем по программированию на языке Python при помощи нейронной сети. В результате были созданы запросы для различных тем по программированию и сравнение точности классификации решений для данных тем. По итогу были выявлены 2 темы, классификация которых происходит с высокими показателями метрик, и 2 темы, которые пока что показывают неудовлетворительные результаты.

Ключевые слова: генеративная нейронная сеть, проверка заданий, верификация программного кода, автоматизация, индивидуальное обучение.

Введение. В современной образовательной среде высших учебных заведений при изучении дисциплин связанных с программированием, задача проверки решений студентов преподавателями является одной из значимых, так как важно помочь студентам разобраться с трудностями при решении задач. Индивидуальная работа со студентом позволит ему разобраться со своими проблемами за менее продолжительное время, чем если бы он самостоятельно пробовал ее решить. Такой подход особенно важен при подготовке к промежуточной аттестации или контрольной работе. Однако при проверке решений и поиске ошибок с их последующим разбором индивидуально с каждым студентом преподаватели сталкиваются с нехваткой времени.

На данный момент существуют различные подходы по автоматической проверке решений, но их использование оставляет такие проблемы как: подготовка проверок вручную, отсутствие указаний на проблемные части решения и как их исправить. О нетривиальности автоматической проверки решений заданий по программированию описывается в статье [1] написанной в 2011 году. Особенно активно системы использующие автоматические проверки решений стали разрабатываться в последние несколько лет [2].

Уникальный метод был представлен в статье Массачусетского технологического института [3]. В данной работе разбирается вариант автоматического предоставления обратной связи при решении задач начального уровня по программированию на языке Python. Для использования метода необходимы эталонная реализация задания и модель ошибок, состоящая из потенциальных исправлений ошибок, которые могут совершить студенты. С помощью этой информации, система автоматически выводит минимальные количество правок к неверным решениям студентов, а также предоставляет им оценку того, насколько неверным было данное решение, и обратную связь о том, что они сделали неправильно.

Вариант использования модели GPT-4 в качестве ассистента преподавателя для проверок решений по программированию был рассмотрен в статье [5]. В данной статье авторы имеют четкие критерии к каждой задаче и запрашивают у модели оценивать задачи по критериям и давать обратную связь. Общая точность по 6 видам задач, включающих в себя 3 темы

(Одномерные массивы, Математические функции и выражения, Условные операторы if), которые будут рассматриваться в данной работе, составила примерно 75%, данный результат был основан на 157 решениях. Данное исследование подтверждает, что использование генеративной нейронной сети для классификации актуально и данный подход применяется.

Выдвинем гипотезу о том, что для проверки решений заданий по определенным темам программирования на языке Python, может быть достаточно проверки при помощи генеративной нейронной сети.

Целью данной работы является исследование для заданий каких тем автоматической проверки от генеративной нейронной сети будет достаточно в ходе тренировки навыков написания кода на языке Python.

Для этого были поставлены следующие задачи:

1. Собрать датасет заданий и решений по программированию на Python и разбить его по темам.
2. Разработать варианты запросов к нейросети для наилучшего результата проверки.
3. Реализовать модуль для проверки при помощи генеративной нейронной сети.
4. Сравнить результаты проверок по различным темам.

Правильным решением задачи считается, если код выполняет условия задачи и содержит использование ключевых слов темы задания, например выполнение задания при помощи цикла for в теме «циклы for».

Материалы и методы. Для исследования возможностей генеративной нейронной сети выступать в роли преподавателя в задаче проверки решений заданий по программированию на Python был собран датасет из 582 решений, написанных по задачам 4 тем. Общее распределение правильных к неправильным решениям равно 336:246.

В качестве обязательных данных необходимы задания, и решения от студентов. Для этого были получены результаты контрольной по дисциплине «Программирование и основы алгоритмизации на языке Python», в сумме из которых было выделено 582 решений.

Анализ исходных данных показал, что текст 233 решений не содержат комментарии, и соответственно подходят для классификации при помощи генеративной нейронной сети. В остальных 349 решениях, присутствуют комментарии, которые необходимо убрать из решений, иначе комментарии могут интерпретироваться нейросетью как инструкции, например, комментарии с описанием верной работы программы может привести к положительному ответу на неверное решение, и проверка может быть некорректной.

Автоматическая проверка решений задач по программированию — это задача информационных технологий, связанная с проверкой решений с учетом различных факторов без участия человека. Основным критерием правильности программы является корректность ее исполнения и/или корректная обработка входных данных.

Математическая постановка задачи может быть представлена следующим образом.

$E = e_1, e_2, ..., e_n$ — множество заданий,

$C_i = c_{i1}, c_{i2}, \ldots, c_{ip}$ — множество требований к заданию e_i,

$T_i = t_{i1}, t_{i2}, \ldots, t_{im}$ — множество тестов для проверки задания e_i,

Решение s для задания e_i.

Требуется для e_i задания получить $A_i = a_{i1}, a_{i2}, \ldots, a_{im}$ — множество ответов о выполнении требований C_i в решении s, где a_{ij} — либо 1, если требование выполнено, либо 0, если оно

не выполнено, при $j \in [1;m]$. Если $\forall\, a_{ij} = 1$, тогда следует получить $O_i = o_{i1}, o_{i2}, \ldots, o_{im}$ — множество результатов тестов из T_i, необходимо из O_i получить p_i — оценку за решение s задания e_i.

Для определения корректности собственного метода проверки рассмотрим подходы к автоматической проверке решений задач по программированию на корректность и сравним их.

Простое сравнение. Суть данного метода заключается в сравнении выходных данных программы с эталонным ответом теста. Если строки не совпадают, программа считается не прошедшей тест, и переходит к следующему. После завершения тестирования анализируется количество пройденных и проваленных тестов.

Использование метрик. При данном способе для сравнения выходных данных программы с эталонным ответом используются метрики [4]. В данном случае оценивается расстояние между выводом программы и эталонным ответом. Чем меньше расстояние, тем выше оценка.

Использование регулярных выражений. При использовании данного метода выходные данные программы проверяются с помощью регулярных выражений. Регулярные выражения позволяют осуществлять различные операции со строками, включая поиск, проверку на соответствие шаблону и другие. Если вывод программы соответствует описанному шаблону, тест считается пройденным, и переходит к следующему. Как и в первом случае, возможны только две оценки при прохождении каждого теста: зачтено или незачтено.

В табл. 1 представлен сравнительный анализ описанных выше методов, с указаниями преимуществ и недостатков.

Таблица 1

Методы автоматической проверки решений задач по программированию

Метод	*Преимущества*	*Недостатки*
Простое сравнение	Простота реализации	Чтобы полностью проверить необходимо огромное количество тестов, низкая точность
Использование метрик	Точность оценивания выше	Необходимость знания формата ответа, который может отличаться в разных заданиях
Использование регулярных выражений	Меньшее количество тестов	Низкая точность, требования знаний составления регулярных выражений

Проверка при помощи нейросети. Метод состоит в том, чтобы передать нейросети на вход текст задачи и код решения, после чего нейросеть должна проверить код на соответствие условиям задачи.

Именно этот метод будет рассматриваться в данной статье. Преимуществами этого метода являются отсутствие необходимости написания тестов для каждого задания или групп заданий, а также возможность получения обратной связи об ошибках в решении.

Математическая постановка исследования данного метода:

Дано: $E = e_1, e_2, \ldots, e_n$ — множество заданий,

$C_i = c_{i1}, c_{i2}, \ldots, c_{ip}$ — множество требований к заданию e_i,

$S_i = s_{i1}, s_{i2}, \ldots, s_{im}$ — множество решений для задания e_i, где доля s_{ij} — неверные решения, а остальные — верные, при $j \in [1;m]$.

Шаблон запросов *prompt* использующийся для всех заданий из E.

Требуется: Получить $Y_i = y_{i1}, y_{i2}, \dots, y_{im}$ — оценки каждого решения из S_i по e_i заданию, где $y_{ij} = 1$, если решение верно, и 0, если решение не верно. Требуется отправить запросы по шаблону *prompt* по каждому решению из S_i для всех заданий из E с условием выполнения всех требований из C_i и получить $Y`_i = y`_{i1}, y`_{i2}, \dots, y`_{im}$ — оценки каждого решения из S_i по e_i заданию, где $y_{ij} = 1$, если нейросеть оценивает решение как верное и 0, если наоборот. Необходимо сравнить оценки y_{ij} и $y`_{ij}$ для всех $i \in [1;n]$ и $j \in [1;m]$ и получить долю правильно оцененных нейронной сетью решений.

Для данной задачи обозначим следующие ограничения:

Задания из E представляют из себя текст с формулами размеченными при помощи LaTex. Решения из S_i написаны на языке Python.

Для проверки кода студентов была использована модель GPT-3.5-Turbo-0125, доступная через OpenAI API. Помимо этой модели были также рассмотрены GPT-4.0 и GPT-3.0. Однако, использование этих моделей сопряжено с определенными ограничениями. Например, GPT-4.0 обладает значительно более высокой стоимостью использования по сравнению с GPT-3.5-Turbo-0125 — до 40-60 раз выше, чтобы обработать то же количество токенов. GPT-3.0 не имеет инструментов для вывода ответов в формате JSON, а также, модель больше не получает обновлений.

Перед проверкой кода, модели задается поведение при помощи промпта, изображенного на рис. 1.

①	Ты автоматизированная система для проверки кода студентов на языке Python. Твоя работа — проверять код на решение поставленных целей в задаче. Тебе будут присылать сообщения содержащие описание задачи и код который необходимо будет проверить. Отвечай "Верно" или "Неверно"
②	Присланные задачи будут на тему "Циклы While". Проверяй, чтобы в коде решения обязательно было использовано ключевое слово while. Если используется другой цикл для решения задачи, задача считается решенной "Неверно". Если в описании задачи не предполагается использования бесконечных циклов с условием прерывания, то считай задачи с бесконечным циклом решенными "Неверно".

Рис. 1. Промпт поведения системы для темы «циклы While»

В примере из рис. 1 запрос разделен на две части:

1. Неизменяемая часть.
2. Часть, содержание которой зависит от темы проверяемой задачи.

Для проверки кода к заданию составляется промпт изображенный на рис. 2.

```
### Задача
Текст задачи
### Код решения
```ру
Код
```
```

Рис. 2. Промпт к нейросети для проверки решений

После получения результатов для каждого задания из E и каждого решения из S_i требуется провести оценку проверки при помощи метрик *Accuracy*, *precision* и *recall*.

Результаты

Проверка точности классификации 582 решений заданий по программированию на языке Python при помощи генеративной нейронной сети, показала, что 489 решений были определены правильно и 93 неправильно. Общая точность классификации составляет примерно 0,84.

В табл. 2 представлена матрица ошибок классификации.

Таблица 2

Матрица ошибок

Матрица ошибок		*Фактические*	
		Positive	*Negative*
Предсказанные	Positive	302	59
	Negative	34	187

По результатам классификации, в случае 48 неверных решений, которые выполняют основную задачу, но не используют указанные темы или используют альтернативные варианты решений (например, в случаях если указана тема «циклы while», а задача решена при помощи цикла for) точность нейросети составила примерно 0,813.

В случае 61 неверного решения, использующих правильную логику решения задачи, но имеют опечатку в каком-то цифровом значении (например, в проверке условия) точность определения составила примерно 0,377. Данный результат свидетельствует, что модель больше обращает внимания на логику алгоритма в решении, чем на значения переменных из условий задачи.

В случае 137 неверных решений, которые делали что-то абсолютно отличное от условий задачи, точность нейросети составила примерно 0,912.

В табл. 3 представлены результаты проверок решений для каждой темы.

Таблица 3

Результаты метрик по различным темам

Тема	*Условные операторы if*	*Мат. функции и выражения*	*Одномерные массивы*	*Циклы while*
Accuracy	0,876	0,732	0,9	0,846
Precision	0,949	0,796	0,848	0,633
Recall	0,902	0,836	0,987	0,861

Результаты метрик говорят о том, что решения заданий по теме «математические функции и выражения» значительно хуже остальных классифицируются верно. Это может быть связанно с тем, что в заданиях данной темы, гораздо больше переменных и числовых значений в условиях задачи чем в других темах. Так же из таблицы результатов можно сделать вывод, что в теме «Циклы While» из всех решений названных верными лишь 63% являются действительно верными. Однако это не говорит о том что нейросеть часто ошибалась с неверными решениями, потому что в данной теме, верных решений было меньше, из-за того что данная тема изначально дается студентам труднее других. Из результатов так же можно отметить что модель имеет высокие показатели метрик у оставшихся тем, а именно: «Условные операторы if» и «Одномерные массивы».

Заключение. В результате проведенного исследования можно сделать вывод о действенности предложенного подхода о использовании модели GPT 3.5 Turbo 0125 в качестве классификатора для проверки решений задач по программированию на языке Python. По итогам был разработан инструмент для классифицирования правильности решений по программированию. С его помощью в статье объективно оценена взаимосвязь между ответом нейронной сети с темами задач, решение которых она проверяла.

Для объективности оценки возможностей классификатора были введены ограничения о правильности решения, а конкретно: решение считается правильным если оно интерпретируется без ошибок, выполняет поставленные цели и написано на языке программирования Python. В то же время исследование проблемы поднятой в статье, при других ограничениях или отсутствии некоторых из указанных представляет большой интерес.

Конечно, нейросеть не идеально подходит для проверки заданий по программированию. В то же время, благодаря данной статье, можно отметить то, что с некоторыми темами модель справляется лучше, возможно это связанно с тем как формулируются задачи данных тем. Эту информацию можно использовать в дальнейшей работе по исследованию использования нейросети в качестве классификатора для задач по программированию. С предложенным подходом классификатор можно использовать для тем «Условные операторы if» и «Одномерные массивы», возможно, не для обучения программированию с нуля, но, например, для тренировки навыков студентов.

В дальнейшем предполагается попытаться улучшить предложенный подход о классификации решений при помощи генеративной нейронной сети. Для этого можно рассмотреть использование других нейросетей для данной задачи. Еще одним вариантом улучшения является исследование взаимосвязи формулировок заданий с точностью классификации.

СПИСОК ЛИТЕРАТУРЫ

1. Катаев А.В. Способы проверки решений заданий по программированию в обучающих системах / А.В. Катаев, О.А. Шабалина, В.А. Камаев. — Текст: непосредственный // Прикаспийский журнал: управление и высокие технологии. — 2011. — № 3. — С. 19-25.
2. Автоматическая обучающая экспертная система / В.Н. Буинцев, И.А. Рыбенко, Е.А. Мартусевич, Д.Ю. Белавенцева. — Текст: непосредственный // Вестник сибирского государственного индустриального университета. — 2022. — № 4. — С. 19-26.
3. Singh R. Automated Feedback Generation for Introductory Programming Assignments / R. Singh, S. Gulwani, A. Solar-Lezama. — URL: https://arxiv.org/abs/1204.1751 (date of the application: 16.11.2012). — Text: electronic.

4. Кузовенков А.О. Автоматическая проверка оригинальности исходного кода программы при онлайн-обучении / А.О. Кузовенков. — Текст: непосредственный // Прикладная математика и информатика: современные исследования в области естественных и технических наук: материалы VII Междунар. науч.-практ. конф. — Тольятти, 2021. — С. 41-47.
5. F. Nilsson GPT-4 as an Automatic Grader: The accuracy of grades set by GPT-4 on introductory programming assignments / F. Nilsson. — URL: https://www.divaportal.org/smash/record.jsf?dswid=8569&pid=diva2%3A1779778&c=1&searchType=SIMPLE&language=en&query=GPT-4+as+an+Automatic+Grader&af=%5B%5D&aq=%5B%5B%5D%5D&aq2=%5B%5B%5D%5D&aqe=%5B%5D&noOfRows=50&sortOrder=author_sort_asc&sortOrder2=title_sort_asc&onlyFullText=false&sf=all (date of the application 01.08.2023). — Text: electronic.