Компьютерные инструменты в образовании, 2026
№ 1: 74–90
УДК: 004.8:37.091.33
http://cte.eltech.ru
doi:10.32603/2071-2340-2026-1-74-90

МОДЕЛИ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ УЧЕБНЫХ ЗАДАНИЙ: СРАВНИТЕЛЬНЫЙ ОБЗОР

Бутенко Д. С.[1], ✉ dmitriy.butenko@moevm.info

[1]Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина), Российская Федерация,197022, Санкт-Петербург, ул. Профессора Попова, 5, к. 3

Аннотация

Статья посвящена сравнительному анализу моделей автоматической генерации заданий для учебных дисциплин в условиях растущей диспропорции между численностью студентов и преподавателей и растущих случаев академической нечестности. Цель исследования — сравнить существующие модели генерации по трём критериям: вариативность заданий, трудозатраты на применение и объяснимость результатов — для снижения нагрузки на преподавателей при сохранении качества образовательного процесса. Методология включает анализ публикаций 2020–2025 гг. с классификацией моделей на шаблонные, грамматические, статистические, графовые, рекуррентные нейронные сети, эволюционные алгоритмы и большие языковые модели (LLM). Основные результаты: установлено, что LLM превосходят альтернативные подходы по разнообразию генерируемого контента и вычислительной эффективности при использовании предобученных моделей. Шаблонные и грамматические системы ограничены низкой вариативностью, эволюционные алгоритмы требуют на порядки больше времени, а рекуррентные сети уступают в поддержании семантической целостности. Критическими недостатками LLM являются низкая объяснимость результатов и склонность к галлюцинациям, что требует обязательного экспертного контроля результата. Работа представляет практическую ценность для разработчиков образовательных систем и преподавателей, стремящихся к масштабированию учебного процесса при сохранении педагогического контроля.

Ключевые слова: *автоматическая генерация заданий, большие языковые модели, генерация кода, академическая нечестность, модели автоматической генерации.*

Цитирование: Бутенко Д. С. Модели автоматической генерации учебных заданий: сравнительный обзор // Компьютерные инструменты в образовании. 2026. № 1. С. 74–90. doi:10.32603/2071-2340-2026-1-74-90

1. ВВЕДЕНИЕ

Общее количество студентов высших учебных заведений по всему миру в 2025 году достигло рекордного уровня в 264 миллиона человек [1]. Это увеличение на 25 млн с 2020 года и более чем вдвое по сравнению с показателем 2000 года. При этом число преподавателей высшего образования увеличилось в меньшем объеме: с 4 млн в 1980 году до 13,1 млн в 2018 году. Средневзвешенное соотношение студент–преподаватель в мире изменилось с 12,6:1 в 1980 году до 17,1:1 в 2018 году, при этом тенденция является нараста-

ющей [2]. По данным федерального статистического наблюдения, на 1 октября 2023 года в системе высшего образования России обучалось 4,33 млн студентов, в то время как число преподавателей составляло 216,1 тыс. человек [3]. Таким образом, соотношение числа студентов к числу преподавателей находится выше общемирового показателя 2018 года и составляет 20.

Параллельно с недостаточным количеством преподавателей относительно количества студентов существует проблема академической нечестности. В исследовании 2024 г., проведенном среди студентов высших учебных заведений Израиля, отмечается, что почти половина студентов из выборки не заявляют, что «никогда не списывали» [4]. Особенно эта проблема актуальна для дистанционного образования: согласно систематическому обзору 2024 г. [5], охватывающему исследования как допандемийного, так и пандемийного периодов, самооценочное списывание при онлайн-экзаменах зафиксировано в среднем у 44,7 % студентов высших учебных заведений; при этом в период перехода на дистанционное обучение данный показатель возрастал до 54,7 % по сравнению с 29,9 % до пандемии. Это значительно подрывает эффективность учебного процесса и является косвенным фактором экономического ущерба, связанного с недостаточной квалификацией выпускников.

Ввиду сложившихся обстоятельств преподаватели вынуждены создавать множество вариантов учебных заданий для снижения вероятности списывания, организации персонализированного обучения и объективной оценки знаний студентов. В данной работе под учебным заданием понимается задание для специальностей ИТ-направления: условия, описывающие программу, которую необходимо разработать, либо фрагмент кода, с которым необходимо сделать определенные действия (модифицировать, отладить, проанализировать) в соответствии с заданными условием и ограничениями. Однако тенденция является общей и затрагивает почти все направления образования. Создание качественных учебных заданий осуществляется полностью вручную, и их проверка требует значительных временных затрат, что существенно ограничивает масштабируемость образовательного процесса: 6,5 ч на подготовку и 4,2 ч на проверку согласно международному исследованию OECD TALIS [6]. Подход к созданию, при котором преподаватель лично разрабатывает каждое учебное задание, обеспечивает качество и релевантность контента с полной объяснимостью результата. Тем не менее, такой метод крайне трудоёмок и медлителен: создание даже одной задачи требует много времени преподавателя, что делает генерацию большого количества вариантов практически неосуществимой с учетом упомянутого количества студентов и преподавателей [3], а также участившихся попыток списывания, по которым преподаватели склонны считать, что общее число случаев списывания растет [7]. В дополнение, время на составление и проверку может быть во внеурочные часы: в отчете “Teacher Workload Research 2024” отмечено в среднем 11,39 ч в неделю на работу, связанную с подготовкой, проверкой и обратной связью [8].

Для снижения нагрузки на преподавателей предпринимаются усилия по автоматизации составления учебных заданий. Однако применение данных инструментов не решает проблему полностью, так как в той или иной мере каждый из них не до конца соответствует требованиям учебного процесса [9–13].

Цель данного исследования — провести сравнительный анализ моделей автоматической генерации условий и кода для учебных заданий на основе существующих исследований. В работе рассматривается сравнение по параметрам, способствующим решению упомянутых ранее проблем:

1) разнообразие структуры генерируемых учебных заданий (вариативность) — способность предоставлять множество вариантов, различающихся по форме, содержа-

нию, структуре, стилю, семантике, но при этом удовлетворять требованиям исходного запроса [14–16];
2) трудозатраты на подготовку и применение — затрачиваемое время на обучение моделей и ручные действия;
3) объяснимость результатов [17–19] — способность предоставлять объяснение результатов модели так, чтобы преподаватель смог понять, почему модель получила такой результат.

Основная гипотеза исследования состоит в том, что большие языковые модели (LLM), несмотря на присущие им ограничения [20–23], обеспечивают наилучшее сочетание характеристик для генерации учебных заданий: разнообразие структуры заданий, приемлемые трудозатраты при использовании предобученных моделей и возможность управления результатом через ввод в свободной форме на естественном языке (запрос). Проверка данной гипотезы осуществляется путем сопоставления характеристик LLM с показателями других моделей, представленных в проанализированных исследованиях.

2. ОБЩАЯ ЧАСТЬ

Для подготовки обзора проведён поиск исследований в базах данных CyberLeninka [24] и Google Scholar [25]. Поиск осуществлялся по ключевым запросам, связанным с автоматической генерацией заданий, генерацией кода и применением языковых моделей в образовательных целях. Рассматривались публикации с 2020 года. Дополнительные критерии отбора включали принадлежность работ к области компьютерных и информационных наук (классификация OECD) и релевантность тематике автоматической генерации учебных заданий. Список поисковых запросов:

CyberLeninka:

— автоматическая генерация заданий;
— методы генерации заданий;
— генерация заданий;
— автоматическая проверка заданий;
— генерация кода;
— языковые модели.

Google Scholar:

— automatic programming task generator;
— ai question generation;
— ai programming exams generation;
— programming exams generation;
— exams generation.

Основным критерием отбора аналогов, по которому происходит выбор моделей автоматической генерации текста и кода, является как можно большая вариативность генерации учебных заданий. При этом, с одной стороны, вариативность не должна быть чем-то высчитываемым, что можно описать в виде формулы, с другой — получаемый результат должен быть корректным учебным заданием (соответствовать заданной теме и иметь решение). Трудозатраты оцениваются для того, чтобы снизить нагрузку на преподавателей при генерации учебных заданий. При этом в этот критерий попадают как трудозатраты на предварительную подготовку (обучение, формирование базы знаний и т. д.), так и на использование готового решения, которое и занимается генерацией.

Объяснимость результатов позволяет преподавателю понимать логику генерации, влиять на получаемый результат через корректировку исходных данных и тем самым сократить количество некорректных заданий, сохраняя контроль над качеством каждого сгенерированного учебного задания.

3. СРАВНИТЕЛЬНЫЙ ОБЗОР МОДЕЛЕЙ ГЕНЕРАЦИИ

Перечисленные модели широко описаны в статьях как основные модели автоматической генерации учебных заданий [9, 13]. Модели генерации могут объединяться по ряду признаков:

1. **Тип входных данных:** работа со структурированным вводом (шаблоны, грамматики) и со свободными источниками (тексты, онтологии). Так, шаблонные модели и онтологии требуют организованных моделей предметной области, тогда как модели NLP / ML генерируют вопросы из неструктурированных текстов.
2. **Уровень автоматизации:** ручная генерация не автоматизирована; шаблонные и грамматические модели автоматизированы частично (требуют ручной подготовки шаблонов / грамматик); нейросетевые и эволюционные — автоматизированы, нужны обучающие данные или правила для целевой функции.
3. **Требования к данным и знаниям:** модели без ML (шаблоны, грамматики, онтологии) требуют экспертных знаний и ручного задания правил, но мало нуждаются в подготовленном датасете; ML / LLM-модели, напротив, требуют обширных наборов примеров или предобученных моделей. Например, при генерации учебных заданий на естественном языке отмечено, что практически не существует специализированных датасетов для учебных предметов, и исследования выигрывают от наличия таких ресурсов [26].
4. **Масштабируемость и скорость генерации:** модели на основе ML. Модели на основе ML потенциально обеспечивают быструю генерацию множества вариантов после обучения, тогда как шаблонные модели масштабируются менее гибко (необходимо создавать новые шаблоны) [27]. Шаблонные и грамматические модели Для шаблонных моделей преподаватель задаёт структуру учебного задания с параметрами, автоматически подставляемыми при генерации.

Такие генераторы обеспечивают формальную корректность и позволяют создавать широкий спектр вариантов при строгом контроле формата, решая проблему плагиата и снижая трудозатраты. Так, в работе [28] на основе предметно-ориентированного языка (DSL) и принципов проектирования продуктовой линейки, ориентированной на функциональность, из одного шаблона без параметров порождалось 8 вариантов; с параметрами — до 432; с учётом ограничений — 90 вариантов. Всего при апробации было использовано 538 различных вопросов для экзамена по 40 вопросов каждому участнику. Однако разнообразие генерируемых учебных заданий у подобных систем остаётся весьма ограниченным: структура и содержание каждого учебного задания заранее задаются преподавателем, а качество заданий зависит в первую очередь от автора шаблона, а не от функциональности генератора [28]. Помимо этого, в распространённых системах электронного обучения, таких как Moodle, база вопросов теста является статической, что не позволяет достичь вариативности [29]. Кроме того, появление больших языковых моделей (LLM) существенно снижает эффективность шаблонного подхода: модели семейства GPT и более поздние за секунды решают задачи по параметризированным шаблонам [28], что требует поиска принципиально новых моделей построения оценочных заданий.

Примером такого учебного задания может служить следующий шаблон:

Напишите функцию solve(s: str, p: int) -> int, в которой:

s — строка, в которой записано число в бинарном, восьмеричном или шестнадцатеричном виде;
p — целое натуральное число.

Этапы работы функции:

1) функция высчитывает число res = s {{name_oper}} (2 ^ p);
2) функция устанавливает все p - 1 младших битов числа res в 1 (единицу);
3) функция возвращает полученный результат.

Примечание:

1) из математических операций вам доступна только операция «вычитание»;
2) у вас нет ограничений на использование битовых операций;
3) функция возвращает число в десятичной системе счисления.

В примере выше параметр шаблона {{name_oper}} может заменяться на действия умножения или деления в зависимости от значения генератора псевдослучайных чисел.

Альтернативным направлением стали модели на основе формальных грамматик, гарантирующие синтаксическую корректность выходных заданий через чёткие правила генерации [29]. Грамматические подходы позволяют задавать структуру упражнений и даже математические операции, обеспечивая логическую верность форматов. В работе [29] метод генерации тестовых заданий, основанный на формализме канонических исчислений Э. Поста с использованием формальных грамматик в нотации Бэкуса-Наура, был апробирован в расширенном формате: разработано более 300 порождающих грамматик по дисциплинам всех основных циклов университета. Одна грамматика по теме «Измерительные шкалы» порождала 52 различных вопроса, а грамматика по теме «Измерения с помощью осциллографа» — до 400 вариантов задач. Надёжность сгенерированных тестов, оценённая по формуле Спирмена-Брауна, составила от 0,66 до 0,83, что свидетельствует о достаточно высоком качестве. Тем не менее, создание грамматических правил для каждого нового типа учебных заданий остаётся трудоёмким процессом, требующим участия эксперта в соответствующей предметной области [30] и постоянного обновления спецификаций [30, 31].

Современные исследования показывают, что LLM способны автоматизировать процесс извлечения грамматик из спецификаций на естественном языке, преодолевая ключевое ограничение грамматического подхода — зависимость от ручного составления правил. Так, во фреймворке SAGE [30] на базе дообученной LLM (DeepSeek-R1-Distill-Qwen-14B) с применением обучения с подкреплением (GRPO) контекстно-свободные грамматики со счётчиками (CCFG) автоматически извлекаются из текстовых описаний задач. На датасете CodeContests [32] (1200 задач) система достигла 96,66 % валидности и 95,92 % общности порождаемых тестов, превзойдя 17 открытых и закрытых LLM и улучшив валидность на 15,92 п. п., а эффективности тестирования — на 12,34 п. п. Это демонстрирует, что LLM не только могут заменить работу эксперта, выполняемую вручную при составлении грамматик, но и обеспечивают более высокую обобщающую способность, автоматически адаптируясь к разнообразным форматам входных данных. Рассмотренный ранее пример шаблона можно изменить под грамматику:

Algorithm 1:

```
<задание> ::= <заголовок> <этапы_работы> <примечание> <возврат>

<заголовок> ::= "Напишите функцию solve(), которая принимает на вход " <входные_параметры>

<входные_параметры> ::=
 | "строку s (число в " <системы_счисления> ") и целое натуральное число p"
 | "строки s1, s2 (числа в " <системы_счисления> ") и целое натуральное число p"
 | "строку s (число в " <системы_счисления> "), целые натуральные числа p и k"

<системы_счисления> ::=
 | "бинарном, восьмеричном или шестнадцатеричном виде"
 | "двоичном или шестнадцатеричном виде"
 | "восьмеричном или шестнадцатеричном виде"

<этапы_работы> ::= "Этапы работы функции:\n"
<этап_вычисления>
<этап\модификации_битов>
[<дополнительный_этап>]

<этап_вычисления> ::=
 | "Функция высчитывает число res = s " <операция> " (2 ^ p)"
 | "Функция высчитывает число res = (s1 " <операция1> " s2) " <операция2> " (2 ^ p)"
 | "Функция высчитывает число res = s " <операция> " ((2 ^ p) " <операция2> " k)"
 | "Функция высчитывает число res = (s " <операция> " (2 ^ p)) "
                                      <логическая_операция> " ((2 ^ k) - 1)"

 <операция> ::= "*" | "/" | "+" | "-" | "<<" | ">>" | "&" | "|" | "^"

<логическая_операция> ::= "&" | "|" | "\^"

<этап_модификации_битов> ::=
 | "Функция устанавливает " <выбор_битов> " в " <значение_бита>
 | "Функция инвертирует " <выбор_битов>
 | "Функция обнуляет " <выбор_битов> " и устанавливает " <другие_биты> " в 1"

<выбор_битов> ::=
 | "все p - 1 младших битов числа res"
 | "все p младших битов числа res"
 | "p-й бит числа res (нумерация с 0)"
 | "биты с позиций 0 до p-1 включительно числа res"
 | "биты с позиций p до 2*p-1 включительно числа res"
 | "все четные биты в диапазоне [0, p) числа res"
 | "все нечетные биты в диапазоне [0, p) числа res"
 | "старшие k битов числа res"
 | "каждый второй бит, начиная с бита p"

<значение_бита> ::= "1 (единицу)" | "0 (ноль)"

<дополнительный_этап> ::=
 | "Функция циклически сдвигает биты результата вправо на k позиций"
 | "Функция меняет местами младшие p битов со следующими p битами"
 | "Если результат четный, функция инвертирует младший бит"
 | "Функция подсчитывает количество единичных битов и добавляет это к результату"

<примечание> ::= "Примечание:\n" <ограничения>

<ограничения> ::=
 | <набор_запретов> <набор_разрешений>
```

```
<набор_запретов> ::=
  | "Для решения из математических операций Вам доступна только операция " <список_операций>
  | "Для решения из математических операций Вам доступны только операции " <список_операций>
  | "Вы не можете использовать операции " <список_операций>

<список_операций> ::=
  | "\"вычитание\""
  | "\"сложение\" и \"вычитание\""
  | "\"умножение на степени двойки\""
  | "\"деление\" и \"остаток от деления\""

<набор_разрешений> ::=
  | "У Вас нет ограничений на использование битовых операций"
  | "Из битовых операций доступны только сдвиги и операция И (&)"
  | "Битовые операции запрещены, кроме сдвигов"
  | "Вы можете использовать любые операции"

<возврат> ::= "Функция возвращает результат в десятичной системе счисления"
```

Демонстрация грамматики для примера показывает, что усложняется структура учебного задания. При этом следует учитывать, что сочетания параметров для учебных заданий могут конфликтовать друг с другом, из-за чего полученные в результате генерации учебные задания могут быть нерешаемыми. Это можно контролировать через дополнительные параметры для грамматик, задавая варианты связей, но это сильнее усложнит структуру грамматики.

4. СТАТИСТИЧЕСКИЕ И НЕЙРОСЕТЕВЫЕ МОДЕЛИ

Статистические модели, включая модели на основе марковских цепей и n-грамм, предлагают более простой путь без необходимости длительного обучения, если сравнивать с нейросетевыми или эволюционными моделями. В работе [33] марковские цепи применены для анализа 1,08 млн студенческих сессий (37,5 млн действий) в образовательной системе Edulab (крупнейшая платформа цифрового обучения математике в Дании): комбинация из 6 марковских цепей, оценённая модифицированным алгоритмом k-средних, позволила выявить около 125 тыс. сессий, отнесенных к непродуктивному типу учебного поведения. Однако модель ограничена первым порядком зависимостей — каждый переход определяется только текущим состоянием, что делает невозможным моделирование многошаговых логических связей. Количественное подтверждение ограниченности марковских моделей для генерации текста получено в работе [34], где проведено сравнение текстов, порождённых марковскими моделями порядков 2–12 и LSTM-сетями, с оригинальным корпусом объёмом 24 млн символов. Анализ детрендированных флуктуаций DFA (Detrended Fluctuation Analysis) показал, что экспонента долгосрочных корреляций для марковских моделей любого порядка составила $\alpha \approx 0,5$ – значение, эквивалентное случайно перемешанному тексту, тогда как LSTM при оптимальной температуре генерации ($T \approx 0,8 - 1,0$) достигала значения $\alpha \approx 0,6 - 0,7$, сопоставимого с естественным языком. Марковские модели высоких порядков (10–12) фактически воспроизводили фрагменты обучающего корпуса: длина наибольшей общей подпоследовательности (LCS) составляла 73–125 символов против 40–60 у LSTM — значения, сопоставимого с уровнем повторов между различными произведениями одного автора (45 символов). Симметризированная KL-дивергенция между LSTM-текстом и оригиналом достигала минимума при $T \approx 1$, а энтропия наиболее точно соответствовала

оригиналу при $T \approx 0,9$ [34]. Тем не менее, авторы отмечают, что LSTM-тексты «всё ещё далеки от текстов, написанных человеком, по осмысленности» [34], что критически важно для формулировок учебных заданий, требующих точной и однозначной постановки условий.

Графовые модели, использующие онтологические сети, деревья разбора и семантические графы, предлагают иной подход, моделируя связи между понятиями и обеспечивая тематическую согласованность генерируемых учебных заданий. В исследовании [35] отмечено, что графовые модели получают положительную оценку экспертов и эффективнее работают на данных с заданной графовой структурой. Однако в задачах, связанных с обработкой естественного языка и кодогенерацией, эти подходы демонстрируют ограниченную эффективность [36]. В исследовании [37] проведено сравнение графовых нейронных сетей (GNN) и моделей-трансформеров (BERT, Longformer) на пяти наборах данных с документами различной длины (от 14 до 912 слов в среднем). На коротких текстах (средняя длина 51 слово) BERT достигал точности 98,3 % против 97,5 % у лучшей графовой конфигурации. Однако на длинных документах (средняя длина 912 слов) графовые модели превзошли трансформеры: точность 79,1 % и F1-метрика 78,5 % против 72,6 % / 70,6 % у BERT и 77,2 % / 75,5 % у Longformer, при этом используя на три порядка меньше обучаемых параметров (16–66 тыс. против 108 млн) и работая в 3,5–5,7 раза быстрее. Тем не менее, авторы подчёркивают, что эффективность графовых моделей существенно зависит от лексических особенностей и предметной области [37]. Применительно к генерации учебных заданий это означает, что графовые модели могут быть эффективны для представления структурных связей в условиях задач, но уступают трансформерам в задачах, требующих глубокого понимания семантики текста. Сравнительный анализ в контексте классификации текста подтвердил, что модели- трансформеры стабильно обеспечивают более высокую точность и F1-метрику [36], что объясняется их способностью лучше улавливать языковые нюансы и контекст, тогда как графовые сети ориентированы преимущественно на реляционные связи данных.

5. ЭВОЛЮЦИОННЫЕ МОДЕЛИ

Эволюционные алгоритмы представляют ещё одну альтернативу, осуществляя поиск решений посредством мутаций и скрещиваний [38]. Они способны генерировать корректный код, однако требуют обширного набора примеров и значительных вычислительных ресурсов. В работе Custode et al. [39] на бенчмарке HumanEval (73 задачи) грамматическая эволюция (GE) с популяцией 500 особей и 100 000 поколений решила лишь 6 % задач, тогда как LLM (DeepSeek-Coder 6.7B) — 26 %. При этом средняя доля пройденных тестов оказалась сопоставимой (GE — 37,5 %, LLM – 38,3 %), что указывает на различие результатов: LLM либо решает задачу полностью, либо не решает вовсе, тогда как GE способна частично удовлетворять тест-кейсы [39]. Критическим ограничением является время генерации: в отдельных случаях GE затрачивала несколько дней на синтез одного решения, тогда как LLM выдавала результат за секунды. Аналогичные результаты получены в работе Sobania et al. [40], где на бенчмарках PSB1 (29 задач) и PSB2 (25 задач) GitHub Copilot синтезировал 24 и 15 задач соответственно, а генетическое программирование (при агрегации результатов из 54 публикаций и типичном количестве 100 запусков на задачу) — до 25 и 17. При этом эволюционные модели «могут затрачивать дни на синтез одного решения», а генерируемый код «часто раздут и труден для понимания» [41]. Случайность мутаций делает результаты непредсказуемыми, что ухудшает объяснимость.

6. БОЛЬШИЕ ЯЗЫКОВЫЕ МОДЕЛИ

Предобученные модели, такие как GPT-2 и более поздние версии, способны генерировать «беглый и связный текст, который часто сложно отличить от написанного человеком», превосходя более ранние подходы в качестве результатов. Анализ сравнений различных моделей подтверждает, что трансформеры стабильно дают более качественные тексты по флюентности и целостности [41], чем рекуррентные сети и другие архитектуры. Это объясняется их способностью эффективно улавливать длинные зависимости благодаря механизму самовнимания (self-attention) [36].

LLM позволяют быстро создавать большое разнообразие вариантов учебных заданий без необходимости ручного кодирования каждого шаблона или правила. Подходы на основе LLM способны автоматически индуцировать грамматики и извлекать спецификации учебных заданий из ограниченного количества примеров, что обеспечивает высокую обобщаемость и эффективность генерации. В программировании LLM превосходят традиционные грамматические системы и эволюционные алгоритмы, решая задачи генерации кода быстрее и эффективнее.

Вместе с тем, применение LLM сопряжено с определёнными ограничениями. LLM остаются «чёрными ящиками»: трудно объяснить, почему модель сгенерировала тот или иной ответ, и обеспечить контроль её знаний. Как отмечают исследователи, генерируемый LLM код может содержать неизученные или даже потенциально небезопасные элементы, а преподаватель не знает точных данных, на которых модель обучена. У LLM отсутствует явный контроль над содержанием в сравнении с более прозрачными моделями: они могут допускать скрытые ошибки или выдавать неожиданно сложные или простые варианты. Например, опыт работы с GitHub Copilot показывает, что непрозрачность моделей может вызывать недоверие и опасения по поводу корректности результатов [40].

Для генерации учебных заданий свойства больших языковых моделей имеют практическое значение: флюентность обеспечивает естественность формулировок, что важно для понимания студентами условий учебного задания без дополнительных пояснений. В работе Meissner et al. [42] система ItemForge на базе GPT-3.5 и GPT-4 сгенерировала 240 математических заданий для бакалавров по таксономии Андерсон и Кратволь, оценённых тремя экспертами: лингвистическое качество охарактеризовано как «стабильно высокое», медиана соответствия концепции — 4,5 из 5, корректности условия — 4,5 из 5, однако корректность решений оказалась ниже — медиана 4,0 из 5, что подчёркивает необходимость экспертной проверки. Целостность текста позволяет генерировать задания со сложной структурой (например, многоступенчатые учебные задания с взаимосвязанными подпунктами), где все части логически согласованы. Scaria et al.čitelib-43 провели исследование 5 LLM (включая GPT-3.5, GPT-4, Llama-2) на генерации 2550 вопросов по 6 уровням таксономии Блума: 78 % вопросов получили высокую оценку качества от экспертов, а совпадение с целевым уровнем Блума составило 65,56 %. GPT-4 показала лучшие результаты (89 % качественных вопросов, 70 % совпадение уровня), а наиболее эффективной стратегией ввода стала комбинация цепочки рассуждений (CoT), указания навыка и примера. Способность улавливать длинные зависимости критична для создания учебных заданий, требующих понимания контекста предметной области: модель может учитывать не только локальные правила формулировки, но и общую логику курса или тематического раздела [43].

7. РЕЗУЛЬТАТ СРАВНЕНИЯ

В таблице 1 представлено качественное сравнение моделей генерации учебных заданий по ключевым характеристикам: разнообразие структуры генерируемых учебных заданий, объяснимость результата, время подготовки и скорость работы.

Таблица 1. Сравнение моделей генерации учебных заданий

Метод	Разнообразие генерируемых заданий	Объяснимость результата	Время подготовки	Скорость работы
Ручная генерация	целиком зависит от составителя	полная объяснимость из-за ручного составления	не требуется	к моменту составления задание готово к использованию
Шаблонные системы	низкое, (ограничение по структуре учебного задания [28], ограничение в количестве [29])	легко контролировать формат, так как все результаты из одного шаблона	обучение не требуется, требуется время эксперта на создание шаблонов	напрямую зависит от скорости интерпретации шаблона
Грамматические методы	ограничения, привязанные к составлению грамматики человеком [30, 31]), неэффективность [30] в сравнении с другими методами (генетические алгоритмы, LLM)	выходы соответствуют заданным правилам, включая структурированность кода [29]	обучение не требуется, большие временные затраты эксперта на создание грамматик [30]	напрямую зависит от скорости интерпретации грамматик
Марковские цепи	могут генерировать бессмысленные или очень шаблонные описания [33, 34]	усложняется по мере увеличения данных, есть проблемы с объяснимостью	реализуются детерминированными/частотными методами оценки (оценка переходных вероятностей), что не требует сложного	увеличивается по мере роста количества связей обучения
Графовые методы	могут приближаться к LLM при особом подходе к формированию датасета [35, 37], в общем случае ограничены заданным графом	можно проследить, какие узлы и связи графа привели к появлению конкретного задания	быстрее в общем случае, на больших данных сопоставимые с LLM затраты	проигрывают в точности и производительности языковым моделям [35, 36]
RNN/LSTM	могут быть эффективны на более коротком контексте [41] в сравнении с LLM, в общем случае менее точны [41]	множество скрытых слоев препятствуют объяснимости	обучение может занять больше времени или сопоставимо с LLM	могут быть менее производительны в сравнении с LLM [41]
Эволюционные алгоритмы	результат может быть неприменим для текста, код более сложен для понимания [40]	труднообъяснимы, так как есть элемент случайности	GA требует специального датасета для того, чтобы быть эффективнее [39]	на порядки более медленная генерация [39, 40]
LLM	наибольшее разнообразие в сравнениии с другими методами	множество скрытых слоев препятствуют объяснимости	наиболее длительное обучение (общая проблема всех нейросетевых методов)	сопоставимая с нейросетевыми методами

Оценки основаны на анализе результатов, представленных в проанализированных исследованиях. Следует отметить, что количественные метрики варьируются в зависимости от конкретной реализации модели и используемых данных, поэтому приведены качественные характеристики с указанием общих тенденций.

Проведенный анализ позволяет сформулировать практические рекомендации по формированию архитектуры системы автоматической генерации учебных заданий. На основе выявленных преимуществ LLM и необходимости контроля качества результатов, предлагается модульная архитектура, изображенная в виде диаграммы на рисунке 1 и состоящая из пяти компонентов:

1. **Интерфейс взаимодействия с пользователем** необходим для сбора вводимых пользователем данных и передачи ввода в следующий компонент.
2. **Компонент подготовки ввода** обрабатывает пользовательский ввод таким образом, чтобы обработчик мог с этим вводом работать: подготовить запрос.
3. **Компонент генерации учебных заданий** обрабатывает запрос и отвечает за формирование результата на основе запроса. Может быть выполнен в виде языковой модели или конвейера из языковой модели, моделей или комбинирования языковой модели с другими моделями генерации из обзора. Конечная архитектура компонента и детальное рассмотрение компонента является одной из целей дальнейших исследований.
4. **База данных** используется для хранения сгенерированных заданий, оценок, решений, историй запросов и шаблонов для запросов.
5. **Служебный компонент** необходим для системного администрирования компонента генерации учебного задания и подготовки ввода. Системное администрирование может включать: настройку параметров языковой модели и констант, дообучение, переобучение, корректировка ответов, правила обработки ввода, настройки мониторинга (проверка доступности запущенных компонентов, нагрузка на оборудование, сеть, логирование служебных сообщений).

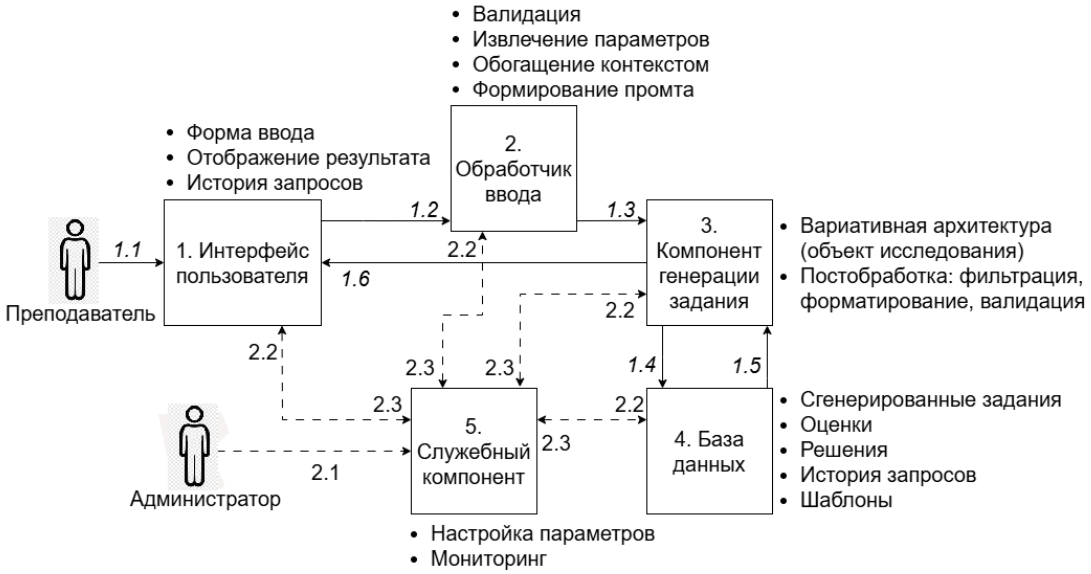

Рис. 1. Архитектура системы автоматической генерации учебных заданий

8. ЗАКЛЮЧЕНИЕ

Проведённый сравнительный анализ демонстрирует, что LLM представляют собой перспективное направление для автоматизации создания учебных заданий. LLM показывают преимущества перед другими подходами по ключевым критериям: качеству и связности генерируемого текста, разнообразию результата и скорости генерации. В отличие от шаблонных систем, которые оказались уязвимы перед современными интеллектуальными системами и ограничены низким разнообразием, LLM способны генерировать принципиально новые варианты учебных заданий без необходимости ручного кодирования каждого шаблона. По сравнению с грамматическими моделями, требующими трудоемкой разработки формальных спецификаций, LLM демонстрируют гибкость и обобщаемость, автоматически извлекая паттерны из ограниченного количества примеров. Статистические модели и рекуррентные сети уступают трансформерам в способности поддерживать семантическую целостность и учитывать длинные зависимости, а эволюционные алгоритмы значительно проигрывают по вычислительной эффективности, требуя на порядки больше времени для решения тех же задач.

На перспективность применения LLM также указывают авторы других работ [27]. Вместе с тем, применение LLM требует учёта их ограничений, прежде всего проблемы объяснимости. Для преподавателя это означает невозможность предсказать и контролировать логику генерации до получения результата и необходимость экспертного контроля генерируемого контента. Непрозрачность работы моделей и возможность генерации некорректных или небезопасных решений делают обязательным внедрение механизмов валидации и верификации результатов. Перспективным решением проблемы контроля качества выступает использование многоуровневых систем на базе LLM, где одна модель выполняет генерацию контента, а другая осуществляет его предварительную фильтрацию и модерацию [44]. Такой подход позволяет автоматизировать первичную проверку сгенерированных заданий на соответствие заданным критериям. А также автоматизировать выявление фактических ошибок, несоответствий учебным целям или потенциально некорректного содержания до передачи результатов на экспертную оценку [45]. Применение LLM-модераторов снижает нагрузку на преподавателей, отфильтровывая заведомо неподходящие варианты, при этом сохраняя необходимость финальной экспертной валидации для обеспечения педагогической корректности [44] . Дальнейшим направлением исследований представляется разработка гибридных подходов, сочетающих преимущества LLM в генерации разнообразного контента с элементами формального контроля, характерными для традиционных моделей. Таким образом, несмотря на существующие ограничения, LLM на сегодняшний день являются эффективным инструментом для автоматической генерации учебных заданий, обеспечивающим оптимальный баланс между многовариантным результатом и вычислительной эффективностью при условии экспертного контроля.

Список литературы

1. Юнеско. Higher education: figures at a glance [Электронный ресурс]. Париж, Франция: Юнеско, 2025. URL: https://unesdoc.unesco.org/ark:/48223/pf0000394112
2. W. M. To and B. T. Yu, “Rise in higher education researchers and academic publications,” Emerald Open Research, vol. 1, no. 3, 2023.
3. Министерство науки и высшего образования Российской Федерации, “оклад о реализации го-

сударственной политики в сфере высшего образования и соответствующего дополнительного профессионального образования," 2024.
4. R. Yavich and N. Davidovitch, "Plagiarism among higher education students," Education Sciences, vol. 14, no. 8, p. 908, 2024.
5. Newton, P.M. and Essex, K., 2024. How common is cheating in online exams and did it increase during the COVID-19 pandemic? A systematic review. Journal of Academic Ethics, 22(2), pp.323-343.
6. OECD, "TALIS 2018 Results (Volume I): Teachers and School Leaders as Lifelong Learners," TALIS, OECD Publishing, 2019.
7. R. Denkin, "On perception of prevalence of cheating and usage of generative AI," arXiv preprint arXiv:2405.18889, 2024.
8. M. Hulme, G. Beauchamp, J. Wood, and C. Bignell, "Teacher workload research report 2024," University of the West of Scotland, 2024.
9. G. Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari, "A systematic review of automatic question generation for educational purposes," International Journal of Artificial Intelligence in Education, vol. 30, no. 1, pp. 121-204, 2020.
10. R. Weegar and P. Idestam-Almquist, "Reducing workload in short answer grading using machine learning," International Journal of Artificial Intelligence in Education, vol. 34, no. 2, pp. 247-273, 2024.
11. A. Gobrecht, F. Tuma, M. Möller, T. Zöller, M. Zakhvatkin, A. Wuttig, H. Sommerfeldt, and S. Schütt, "Beyond human subjectivity and error: a novel AI grading system," arXiv preprint arXiv:2405.04323, 2024.
12. A. Formica, I. Mele, and F. Taglino, "A template-based approach for question answering over knowledge bases," Knowledge and Information Systems, vol. 66, no. 1, pp. 453-479, 2024.
13. М. А. Маслова, "Обзор существующих методов автоматической генерации тестовых заданий на естественном языке," Computational nanotechnology, vol. 10, no. 4, pp. 46-55, 2023.
14. L. Yun, C. An, Z. Wang, L. Peng, and J. Shang, "The Price of Format: Diversity Collapse in LLMs," arXiv preprint arXiv:2505.18949, 2025.
15. J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," Frontiers in Robotics and AI, vol. 3, p. 40, 2016.
16. D. Wright, S. Masud, J. Moore, S. Yadav, M. Antoniak, P. E. Christensen, C. Y. Park, and I. Augenstein, "Epistemic Diversity and Knowledge Collapse in Large Language Models," arXiv preprint arXiv:2510.04226, 2025.
17. T. Speith, B. Crook, S. Mann, A. Schomäcker, and M. Langer, "Conceptualizing understanding in explainable artificial intelligence (XAI): an abilities-based approach," Ethics and Information Technology, vol. 26, no. 2, p. 40, 2024.
18. F. Sovrano and F. Vitali, "An objective metric for Explainable AI: How and why to estimate the degree of explainability," Knowledge-Based Systems, vol. 278, p. 110866, 2023.
19. Д. Н. Бирюков and А. С. Дудкин, "Объяснимость и интерпретируемость—важные аспекты безопасности решений, принимаемых интеллектуальными системами (обзорная статья)," Научно-технический вестник информационных технологий, механики и оптики, vol. 25, no. 3, p. 373, 2025.
20. H. Ye, T. Liu, A. Zhang, W. Hua, and W. Jia, "Cognitive mirage: A review of hallucinations in large language models," arXiv preprint arXiv:2309.06794, 2023.
21. A. Saha, B. Gupta, A. Chatterjee, and K. Banerjee, "You believe your LLM is not delusional? Think again! a study of LLM hallucination on foundation models under perturbation," Discover Data, vol. 3, no. 1, p. 20, 2025.
22. Z. Zhang, C. Wang, Y. Wang, E. Shi, Y. Ma, W. Zhong, J. Chen, M. Mao, and Z. Zheng, "Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation," Proceedings of the ACM on Software Engineering, vol. 2, no. ISSTA, pp. 481-503, 2025.
23. E. Cambria, L. Malandri, F. Mercorio, N. Nobani, and A. Seveso, "Xai meets llms: A survey of the relation between explainable ai and large language models," arXiv preprint arXiv:2407.15248, 2024.
24. CyberLeninka, "Научная электронная библиотека "КиберЛенинка"," 2025. [Online]. Available: https://cyberleninka.ru/
25. Google Scholar, "Google Scholar," 2025. [Online]. Available: https://scholar.google.com/
26. R. Circi, J. Hicks, and E. Sikali, "Automatic item generation: foundations and machine learning-based

approaches for assessments," in Frontiers in Education, vol. 8, p. 858273, Frontiers Media SA, May 2023.
27. В. В. Кручинин and В. В. Кузовкин, "Обзор существующих методов автоматической генерации задач с условиями на естественном языке," Компьютерные инструменты в образовании, no. 1, pp. 85–96, 2022. doi: 10.32603/2071-2340-2022-1-85-96.
28. N. Willert and J. Thiemann, "Template-Based Generator for Single-Choice Questions," Tech Know Learn, vol. 29, pp. 355–370, 2024.
29. А. Н. Швецов and А. П. Сергушичева, "Опыт применения метода автоматической генерации тестовых заданий," ОТО, no. 4, 2017. [Online]. Available: https://cyberleninka.ru/article/n/opyt-primeneniya-metoda-avtomaticheskoy-generatsii-testovyh-zadaniy
30. H. Park et al., "SAGE: Specification-Aware Grammar Extraction for Automated Test Case Generation with LLMs," arXiv preprint arXiv:2506.11081, 2025.
31. P. M. Maurer, "Generating test data with enhanced context-free grammars," IEEE Software, vol. 7, no. 4, pp. 50-55, July 1990. doi: 10.1109/52.56422.
32. deepmind/code_contests [Электронный ресурс] // Hugging Face. – URL: https://huggingface.co/datasets/deepmind/code_contests (дата обращения: 15.03.2026).
33. C. Hansen et al., "Sequence modelling for analysing student interaction with educational systems," arXiv preprint arXiv:1708.04164, 2017.
34. M. Lippi et al., "Natural language statistical features of LSTM-generated texts," IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3326-3337, 2019.
35. L. Galke et al., "Are we really making much progress in text classification? A comparative review," arXiv preprint arXiv:2204.03954, 2022.
36. S. Kuntur, M. Krzywda, A. Wróblewska, M. Paprzycki, and M. Ganzha, "Comparative Analysis of Graph Neural Networks and Transformers for Robust Fake News Detection: A Verification and Reimplementation Study," Electronics, vol. 13, p. 4784, 2024.
37. M. Bugueño and G. De Melo, "Connecting the Dots: What Graph-Based Text Representations Work Best for Text Classification using Graph Neural Networks?," arXiv preprint arXiv:2305.14578, 2023.
38. J. Protopopova and S. Kulik, "Educational intelligent system using genetic algorithm," Procedia Computer Science, vol. 169, pp. 168-172, 2020.
39. L. L. Custode et al., "Comparing large language models and grammatical evolution for code generation," in Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2024.
40. D. Sobania, M. Briesch, and F. Rothlauf, "Choose your programming copilot: a comparison of the program synthesis performance of github copilot and genetic programming," in Proceedings of the Genetic and Evolutionary Computation Conference, 2022.
41. J. Xing, "Comparative and Performance Analysis of Different Deep Learning Models in Text Generation," Applied and Computational Engineering, vol. 154, pp. 212-218, 2025.
42. R. Meissner, A. Pögelt, K. Ihsberner, M. Grüttmüller, S. Tornack, A. Thor, N. Pengel, H. W. Wollersheim, and W. Hardt, "LLM-generated competence-based e-assessment items for higher education mathematics: Methodology and evaluation," Frontiers in Education, 2024.
43. N. Scaria, S. Dharani Chenna, and D. Subramani, "Automated educational question generation at different bloom's skill levels using large language models: Strategies and evaluation," in International Conference on Artificial Intelligence in Education, July 2024.
44. S. Pasch, "AI vs. Human Judgment of Content Moderation: LLM-as-a-Judge and Ethics- Based Response Refusals," arXiv preprint arXiv:2505.15365, 2025.
45. S. Saadaoui and E. Alonso, "Coordinated LLM Multi-Agent Systems for Collaborative Question-Answer Generation," Knowledge-Based Systems, p. 114627, 2025.

Поступила в редакцию 11.01.2026, окончательный вариант — 20.02.2026.

Бутенко Дмитрий Сергеевич, ассистент кафедры МОЭВМ, СПбГЭТУ «ЛЭТИ», ✉ **dmitriy.butenko@moevm.info**

Computer tools in education, 2026
№ 1: 74–90
http://cte.eltech.ru
doi:10.32603/2071-2340-2026-1-74-90

Models for the Automatic Generation of Assessment Tasks: a Comparative Review

Butenko D. S.[1], ✉ dmitriy.butenko@moevm.info

[1]Saint Petersburg State Electrotechnical University "LETI",
Professora Popova ul. 5, building 3, St. Petersburg, 197022, Russian Federation

Abstract

The article presents a comparative analysis of models for the automatic generation of assessment tasks for university courses in the context of a growing mismatch between student and instructor numbers and an increase in cases of academic dishonesty. The aim of the study is to compare existing generation models according to three criteria—task variability, effort required for implementation, and explainability of results—in order to reduce instructors' workload while preserving the quality of the educational process. The methodology includes an analysis of publications from 2020–2025 and a classification of models into template-based, grammar-based, statistical, graph-based, recurrent neural networks, evolutionary algorithms, and large language models (LLMs). Key findings: LLMs outperform alternative approaches in the diversity of generated content and computational efficiency when pre-trained models are used. Template-based and grammar-based systems are constrained by low variability, evolutionary algorithms require significantly more time, and recurrent networks are inferior in maintaining semantic coherence. Critical drawbacks of LLMs are limited explainability and a tendency to hallucinate, which necessitates mandatory expert oversight of outputs. The work has practical relevance for developers of educational systems and for instructors seeking to scale instruction while retaining pedagogical control.

Keywords: *automatic task generation, large language models, code generation, academic dishonesty, automated generation models.*

Citation: D. S. Butenko, "Models for the Automatic Generation of Assessment Tasks: a Comparative Review," *Computer tools in education*, no. 1, pp. 74–90, 2026 (in Russian); doi:10.32603/2071-2340-2026-1-74-90

References

1. UNESCO, *Higher education: figures at a glance*. Paris, France: UNESCO, 2025. [Online]. Available: https://unesdoc.unesco.org/ark:/48223/pf0000394112
2. W. M. To and B. T. Yu, "Rise in higher education researchers and academic publications," *Emerald Open Research*, vol. 1, no. 3, pp. 1–15, 2023; doi:10.1108/eor-03-2023-0008
3. Ministry of Science and Higher Education of the Russian Federation, "Report on the implementation of state policy in the field of higher education and corresponding additional professional education," 2024 (in Russian).
4. R. Yavich and N. Davidovitch, "Plagiarism among higher education students," *Education Sciences*, vol. 14, no. 8, p. 908, 2024; doi:10.3390/educsci14080908
5. P. M. Newton and K. Essex, "How Common is Cheating in Online Exams and did it Increase During the COVID-19 Pandemic? A Systematic Review,"*Journal of Academic Ethics*, vol. 22, no. 2, pp. 323–343, 2023; doi:10.1007/s10805-023-09485-5

6. OECD, “TALIS 2018 Results (Volume I): Teachers and School Leaders as Lifelong Learners,” TALIS, OECD Publishing, 2019; doi:10.1787/1d0bc92a-en
7. R. Denkin, “On perception of prevalence of cheating and usage of generative AI,” 2024. [Online]. Available: https://arxiv.org/abs/2405.18889
8. M. Hulme, G. Beauchamp, J. Wood, and C. Bignell, “Teacher workload research report 2024,” University of the West of Scotland, Paisley, Scotland, 2024.
9. G. Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari, “A systematic review of automatic question generation for educational purposes,” *International Journal of Artificial Intelligence in Education*, vol. 30, no. 1, pp. 121–204, 2020; doi:10.1007/s40593-019-00186-y
10. R. Weegar and P. Idestam-Almquist, “Reducing workload in short answer grading using machine learning,” *International Journal of Artificial Intelligence in Education*, vol. 34, no. 2, pp. 247–273, 2024; doi:10.1007/s40593-022-00322-1
11. A. Gobrecht et al., “Beyond human subjectivity and error: a novel AI grading system,” 2024. [Online]. Available: https://arxiv.org/abs/2405.04323
12. A. Formica, I. Mele, and F. Taglino, “A template-based approach for question answering over knowledge bases,” *Knowledge and Information Systems*, vol. 66, no. 1, pp. 453–479, 2024; doi:10.1007/s10115-023-01966-8
13. M. A. Maslova, “Review of existing methods for automatic generation of test tasks in natural language,” *Computational Nanotechnology*, vol. 10, no. 4, pp. 46–55, 2023 (in Russian); doi:10.33693/2313-223X-2023-10-4-46-55
14. L. Yun et al., “The Price of Format: Diversity Collapse in LLMs,” 2025. [Online]. Available: https://arxiv.org/abs/2505.18949
15. J. K. Pugh, L. B. Soros, and K. O. Stanley, “Quality diversity: A new frontier for evolutionary computation,” *Frontiers in Robotics and AI*, vol. 3, no. 40, pp. 1–17, 2016; doi:10.3389/frobt.2016.00040
16. D. Wright et al., “Epistemic Diversity and Knowledge Collapse in Large Language Models,” 2025. [Online]. Available: https://arxiv.org/abs/2510.04226
17. T. Speith, B. Crook, S. Mann, A. Schomäcker, and M. Langer, “Conceptualizing understanding in explainable artificial intelligence (XAI): an abilities-based approach,” *Ethics and Information Technology*, vol. 26, no. 40, pp. 1–15, 2024; doi:10.1007/s10676-024-09769-3
18. F. Sovrano and F. Vitali, “An objective metric for Explainable AI: How and why to estimate the degree of explainability,” *Knowledge-Based Systems*, vol. 278, p. 110866, 2023; doi:10.1016/j.knosys.2023.110866
19. D. N. Biryukov and A. S. Dudkin, “Explainability and interpretability—important aspects of the safety of decisions made by intelligent systems (review),” *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 25, no. 3, pp. 373–386, 2025 (in Russian); doi:10.17586/2226-1494-2025-25-3-373-386
20. H. Ye, T. Liu, A. Zhang, W. Hua, and W. Jia, “Cognitive mirage: A review of hallucinations in large language models,” *arXiv preprint*, arXiv:2309.06794, 2023. [Online]. Available: https://arxiv.org/abs/2309.06794
21. A. Saha, B. Gupta, A. Chatterjee, and K. Banerjee, “You believe your LLM is not delusional? Think again! a study of LLM hallucination on foundation models under perturbation,” *Discover Data*, vol. 3, no. 1, p. 20, 2025; doi:10.1007/s44248-025-00029-1
22. Z. Zhang et. al., “Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation,” *Proceedings of the ACM on Software Engineering*, vol. 2, no. ISSTA, pp. 481–503, 2025; doi:10.1145/3728894
23. E. Cambria et al., “Xai meets llms: A survey of the relation between explainable ai and large language models,” 2024. [Online]. Available: https://arxiv.org/abs/2407.15248
24. *Scientific electronic library ’CyberLeninka’*, 2025. [Online]. Available: https://cyberleninka.ru/
25. *Google Scholar*, 2025. [Online]. Available: https://scholar.google.com/
26. R. Circi, J. Hicks, and E. Sikali, “Automatic item generation: foundations and machine learning-based approaches for assessments,” *Frontiers in Education*, vol. 8, p. 858273, 2023; doi:10.3389/feduc.2023.858273
27. V. V. Kruchinin and V. V. Kuzovkin, “Review of existing methods for automatic generation of problems with conditions in natural language,” *Computer Tools in Education*, no. 1, pp. 85–96, 2022 (in Russian); doi:10.32603/2071-2340-2022-1-85-96
28. N. Willert and J. Thiemann, “Template-Based Generator for Single-Choice Questions,” *Tech Know Learn*, vol. 29, pp. 355–370, 2024; doi:10.1007/s10758-023-09659-5
29. A. N. Shvetsov and A. P. Sergusheva, “Experience in applying the method of automatic test task generation,” *Open Education*, no. 4, pp. 318–333, 2017 (in Russian).
30. H. Park et al., “SAGE: Specification-Aware Grammar Extraction for Automated Test Case Generation with LLMs,” 2025. [Online]. Available: https://arxiv.org/abs/2506.11081
31. P. M. Maurer, “Generating test data with enhanced context-free grammars,” *IEEE Software*, vol. 7, no. 4, pp. 50–55, 1990; doi:10.1109/52.56422
32. DeepMind, “code_contests,” *Hugging Face Datasets*. [Online]. Available: https://huggingface.co/datasets/deepmind/code_contests

33. C. Hansen et al., “Sequence modelling for analysing student interaction with educational systems,” *arXiv preprint*, 2017. [Online]. Available: https://arxiv.org/abs/1708.04164
34. M. Lippi et al., “Natural language statistical features of LSTM-generated texts,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3326–3337, 2019; doi:10.1109/tnnls.2019.2890970
35. L. Galke et al., “Are we really making much progress in text classification? A comparative review,” 2022. [Online]. Available: https://arxiv.org/abs/2204.03954
36. M. Bugueño and G. De Melo, “Connecting the Dots: What Graph-Based Text Representations Work Best for Text Classification using Graph Neural Networks?,” 2023. [Online]. Available: https://arxiv.org/abs/2305.14578
37. S. Kuntur et al., “Comparative Analysis of Graph Neural Networks and Transformers for Robust Fake News Detection: A Verification and Reimplementation Study,” *Electronics*, vol. 13, p. 4784, 2024; doi:10.3390/ electronics13234784
38. J. Protopopova and S. Kulik, “Educational intelligent system using genetic algorithm,” *Procedia Computer Science*, vol. 169, pp. 168-172, 2020; doi:10.1016/j.procs.2020.02.130
39. L. L. Custode et al., “Comparing large language models and grammatical evolution for code generation,” in *Proc. of the Genetic and Evolutionary Computation Conference Companion*, pp. 1830–1837, 2024; doi:10.1145/3638530.3664162
40. D. Sobania, M. Briesch, and F. Rothlauf, “Choose your programming copilot: a comparison of the program synthesis performance of github copilot and genetic programming,” in *Proc. of the Genetic and Evolutionary Computation Conference*, pp. 1019–1027, 2022; doi:10.1145/3512290.3528700
41. J. Xing, “Comparative and Performance Analysis of Different Deep Learning Models in Text Generation,” *Applied and Computational Engineering*, vol. 154, pp. 212–218, 2025; doi:10.54254/2755-2721/2025.tj23212
42. R. Meissner et al., “LLM-generated competence-based e-assessment items for higher education mathematics: Methodology and evaluation,” *Frontiers in Education*, vol. 9, 2024; doi:10.3389/feduc.2024.1427502
43. N. Scaria, S. Dharani Chenna, and D. Subramani, “Automated educational question generation at different bloom’s skill levels using large language models: Strategies and evaluation,” in *International Conference on Artificial Intelligence in Education*, pp. 165–179, 2024; doi:10.1007/978-3-031-64299-9_12
44. S. Pasch, “AI vs. Human Judgment of Content Moderation: LLM-as-a-Judge and Ethics-Based Response Refusals,” 2025. [Online]. Available: https://arxiv.org/abs/2505.15365
45. S. Saadaoui and E. Alonso, “Coordinated LLM Multi-Agent Systems for Collaborative Question-Answer Generation,” *Knowledge-Based Systems*, vol. 330, p. 114627, 2025; doi:10.1016/j.knosys.2025.114627

Received 11-01-2026, the final version — 20-02-2026.

Dmitry Butenko, Assistant Lecturer at Department of Software Engineering and Computer Applications, ETU “LETI”, ✉ dmitriy.butenko@moevm.info