

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения
высшего образования «Национальный исследовательский ядерный университет «МИФИ»
(ИАТЭ НИЯУ МИФИ)

Отделение Интеллектуальные кибернетические системы

**Выпускная квалификационная работа —
магистерская диссертация**

по направлению подготовки **09.04.02 Информационные системы и технологии**

Направленность (профиль) **Информационные системы**

**«Разработка алгоритмов и программных средств компонента
выявления знаний обучаемых в интеллектуальной обучающей
системе»**

Выполнил:

студент гр. ИС-М23

Заволженский В.В.

Старший преподаватель ИАТЭ

НИЯУ МИФИ, к.т.н.

Фонталины Е.С.

Нормоконтроль

доцент отделения ИКС, к.ф.-м.н.

Качанов Б.В.

Выпускная квалификационная

работа допущена к защите

Руководитель образовательной

программы 09.04.02

Информационные системы и

технологии

канд. тех. наук

Мирзеабасов О.А.

Обнинск, 2025 г

РЕФЕРАТ

Отчет с. 81, рисунков 16, источников 27, приложений 7.

**ИНТЕЛЛЕКТУАЛЬНЫЕ ОБУЧАЮЩИЕ СИСТЕМЫ, АЛГОРИТМЫ
ИНТЕЛЛЕКТУАЛИЗАЦИИ ОБУЧЕНИЯ, АДАПТИВНОЕ ТЕСТИРОВАНИЕ,
ТЕОРИЯ ОТВЕТОВ НА ЗАДАНИЯ (IRT), RASCH-МОДЕЛЬ, ГЕНЕРАЦИЯ
ТЕСТОВЫХ ЗАДАНИЙ, ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ОБУЧЕНИЯ,
СОВРЕМЕННЫЕ ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ, РАЗРАБОТКА
ПРОГРАММНЫХ СИСТЕМ**

Объектом исследования выпускной квалификационной работы является интеллектуальная обучающая система, направленная на персонализированное выявление знаний обучаемых. Основное внимание уделяется алгоритмам интеллектуализации образовательного процесса, а именно механизму адаптивного тестирования, построенному на основе теории ответов на задание (IRT) с использованием Rasch-модели.

Цель работы заключается в разработке алгоритмов и программных средств компонента выявления знаний обучаемых в Интеллектуальной Системе Обучения с применением адаптивного тестирования на базе Rasch-модели IRT. Для этого реализован модуль генерации тестовых заданий с применением нейросетевых моделей и методов кластеризации, позволяющий получать уникальные и разнообразные вопросы, привязанные к содержанию учебных материалов.

Практическая значимость результатов работы заключается в их применении при разработке интеллектуальных обучающих систем, что способствует повышению эффективности образовательного процесса и соответствует современным тенденциям в образовательных технологиях.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	5
ГЛАВА 1: ТЕОРЕТИЧЕСКИЕ ОСНОВЫ.....	7
1.1. Что такое “Интеллектуальная Обучающая Система”.....	7
1.2 Принципы построения Интеллектуальных Обучающих Систем.....	8
1.3. Влияние использования интеллектуальных систем на эффективность обучения.....	10
1.4 Автоматическая генерация вопросов в ИОС.....	12
1.5 Готовые решения генерации вопросов.....	13
1.6 Основы адаптивного тестирования.....	15
1.7 Расширенные модели IRT, психометрические свойства и методы оценки параметров.....	17
1.8 Технологии обработки мультимедийных учебных материалов.....	19
1.9 Цели и задачи.....	20
ГЛАВА 2: РАЗРАБОТКА МОДУЛЯ ТЕСТИРОВАНИЯ В ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЕ ОБУЧЕНИЯ.....	22
2.1 Обзор инструментов.....	22
2.1.1 Обзор инструментов.....	22
2.1.2 Django.....	23
2.1.3. Плюсы и минусы Django.....	25
2.1.4. Устройство Django.....	26
2.2 Пользователи системы.....	28
2.2.1 Студенты.....	28
2.2.2 Преподаватели.....	29
2.2.3 Администраторы.....	30
2.3 Архитектура системы.....	30
2.4 База данных.....	32
2.5 Модель генерации вопросов как инструмент.....	35
2.6 Реализация модели генерации вопросов.....	36

2.7 Модели теории ответов на задание (Item Response Theory, IRT).....	38
2.8 Интеграция экспертных систем и IRT.....	40
2.10. Модели внутренней и внешней адаптации в тестировании.....	43
2.10.1. Модель внутренней адаптации.....	43
2.10.2. Модель внешней адаптации.....	44
2.10.3. Сравнение моделей внутренней и внешней адаптации.....	45
2.10.4. Трудности и перспективы реализации моделей.....	46
2.11 Система генерации и оценивания тестовых заданий.....	47
2.12 Алгоритм формирования тестовых вопросов.....	49
2.13 Улучшение алгоритма формирования тестовых вопросов.....	51
2.14 Внедрение адаптивного тестирования в текущую систему.....	52
ГЛАВА 3: ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ АДАПТИВНОГО ТЕСТИРОВАНИЯ НА БАЗЕ RASCH-МОДЕЛИ IRT.....	53
3.1. Реализация ключевого функционала.....	53
3.2 Повышение качества генерации вопросов.....	54
3.3. Параметры IRT.....	55
3.4. Визуализация в системе.....	57
3.5 Симуляция и автоматическая калибровка параметров тестирования.....	59
3.6 Повышение качества вопросов с помощью кластеризации.....	60
3.7 Работа модуля создания тестов в рамках ILS.....	62
ЗАКЛЮЧЕНИЕ.....	64
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ.....	66
ПРИЛОЖЕНИЕ А.....	69
ПРИЛОЖЕНИЕ Б.....	71
ПРИЛОЖЕНИЕ В.....	74
ПРИЛОЖЕНИЕ Г.....	75
ПРИЛОЖЕНИЕ Д.....	79
ПРИЛОЖЕНИЕ Е.....	80
ПРИЛОЖЕНИЕ Ё.....	81

ВВЕДЕНИЕ

Современное образование находится на этапе масштабной цифровой трансформации, которая предполагает внедрение электронных обучающих систем и ресурсов. Изменения направлены на повышение качества образовательного процесса за счёт использования инновационных технологий и индивидуального подхода к каждому учащемуся [1]. Важнейшей частью такого подхода становится качественная и своевременная оценка полученных знаний. В условиях, когда курсы состоят из десятков тем, содержащих текстовый, аудиовизуальный и мультимедийный контент, ручная подготовка проверочных материалов сновится практически невозможной задачей. Она требует огромных затрат времени преподавателя и неизбежно приводит к субъективным ошибкам. Очевидным выходом кажется автоматизация процесса создания проверочных материалов, например, тестов. Однако на практике это решение оказывается гораздо сложнее, чем выглядит на первый взгляд. Существует необходимость разработки интеллектуальных обучающих систем с использованием технологий помогающих решить эту проблему.

Использование универсальных нейросетевых моделей или шаблонных подходов не позволяет добиться требуемого уровня качества создания тестов. Так же не существует универсальных моделей созданных для такой специфической задачи находящихся в открытом доступе. Шаблонные методы, наоборот, слишком примитивны и ограничены, они не позволяют формулировать вопросы, требующие глубокого понимания темы и анализа материала. Необходимо создать целостную архитектуру, включая модели и алгоритмы, которая могла бы критически подходить к результатам генерации и отбирать наиболее удачные вопросы.

Интеграция системы создания тестов в интеллектуальную систему обучения с использованием технологий адаптивного тестирования позволит стандартизировать измерение учебных достижений и учитывать индивидуальные особенности учащихся. Адаптивное тестирование способно

динамически подбирать задания в зависимости от уровня подготовки студента, что повышает точность оценки и мотивацию учащихся [3].

Одной из ключевых моделей в адаптивном тестировании является Теория Ответов на Задания (Item Response Theory, IRT). IRT представляет собой современный статистический подход, который анализирует взаимодействие характеристик тестируемого и параметров тестовых заданий. В отличие от традиционной теории тестирования, где каждому заданию приписывается одинаковая значимость, IRT учитывает индивидуальные различия и уровень сложности вопросов [4].

IRT позволяет создавать тесты подстраивающиеся под способности учащегося, оценивая их на основе моделей, которые учитывают вероятность угадывания ответа, уровень сложности задания и дискриминацию вопроса [5]. Это способствует не только объективности тестирования, но и персонализации образовательного процесса.

ГЛАВА 1: ТЕОРЕТИЧЕСКИЕ ОСНОВЫ

1.1. Что такое “Интеллектуальная Обучающая Система”

Впервые концепция интеллектуальных обучающих систем (ИОС) была сформулирована Дж. Карбонеллом в 1970 году, однако реальные исследовательские и коммерческие ИОС начали появляться в 80-е годы XX века. По сравнению с обычными автоматизированными обучающими системами, где программа просто указывает правильность или неправильность ответа студента, ИОС ориентирована на диагностику, отладку и коррекцию поведения обучаемого. Эта система не только выявляет ошибки студента, но и проанализировав их причины, строит гипотезы, правила и планы исправления ошибок, предоставляет советы, опираясь на заранее определенные стратегии обучения и модель обучаемого [16].

ИОС определяется как комплекс организационно-методического, информационного, технологического и программного обеспечения, реализующего педагогическую цель на основе базы знаний в конкретной предметной области. Она управляет процессом обучения, проводит диагностику готовности к обучению и оценивает результаты обучения [17].

Современные тенденции в разработке автоматизированных обучающих систем включают их интеллектуализацию, что означает способность системы адаптировать уровень представления учебного материала к индивидуальным способностям и возможностям обучаемого. В этой связи, актуальной становится задача выявления основных составляющих интеллектуальных обучающих систем и их программная реализация.

Интеллектуальные обучающие системы стремятся рационально использовать информацию о знаниях, умениях и возможностях обучаемого. Существующие ИОС работают в сети и охватывают определенные аспекты образовательного процесса. Интеграция технологий также становится принципом работы ИОС, используя искусственный интеллект, аналитику данных, виртуальную реальность и другие технологии для создания

эффективной образовательной среды.

Принцип мгновенной обратной связи также является важным в работе ИОС. Обучающиеся могут получать обратную связь немедленно после выполнения заданий, что помогает им лучше понимать свои ошибки и улучшать свои знания.

Гибкость обучения является еще одним принципом, предоставляя обучающимся возможность выбирать удобное время и место для обучения, а также устанавливать собственный темп изучения материала.

Еще одним важным принципом работы ИОС является мультиплатформенность. Они доступны на различных устройствах, таких как компьютеры, планшеты и смартфоны, что делает обучение более доступным и удобным для обучающихся [18].

Обзор существующих ИОС выделяет технологии, такие как построение последовательности курса обучения, интеллектуальный анализ ответов обучаемого, интерактивная поддержка в решении задач, и помощь в решении задач на основе примеров [19].

1.2 Принципы построения Интеллектуальных Обучающих Систем

В настоящее время процесс образования в основном осуществляется в удаленном формате. Использование специализированных программ значительно повышает эффективность усвоения материала и упрощает задачи преподавателя. Например, в случае технических дисциплин необходимо обеспечить компьютерную визуализацию учебного материала, математическое моделирование объектов, процессов, явлений, а также имитацию работы различных устройств.

При рассмотрении сложных областей обучения при помощи интеллектуальных систем обучения возникает инвариантная и неинвариантная части ИОС.

Инвариантная часть подразумевает, что многие образовательные системы обладают данной характеристикой. Эта часть включает информацию о

обучаемом (ФИО, курс, группа, дисциплина и т.п.) и оценки за различные этапы обучения. Например, в нее входят данные о том, какие знания уже усвоены студентом.

Неинвариантная часть, в свою очередь, означает, что этой особенностью обладает только конкретная интеллектуальная обучающая система. Сюда включаются информация о целях системы, предназначенных для проведения определенного тестирования, и методиках проведения тестов. Важно подчеркнуть, что образовательные программы считаются интеллектуальными, если они обладают такими свойствами, как генерация учебных задач, решение задач с использованием методов представления знаний по изучаемой дисциплине, определение стратегии и тактики ведения диалога, моделирование состояния знаний обучаемого и способность к самообучению на основе анализа взаимодействия с обучаемыми [20].

Если в обычной автоматизированной обучающей системе программа лишь сообщает студенту, правильный ли у него ответ, то ИОС ориентирована на диагностику, отладку и коррекцию поведения обучаемого. Эта система не только выявляет и обозначает ошибки студента, но и анализирует их причины, формулирует гипотезы, устанавливает правила и разрабатывает планы исправления ошибок. Она предоставляет советы, основанные на заранее определенных стратегиях обучения и имеющейся модели обучаемого.

В блоке «модель обучаемого» содержится информация об индивидуальных особенностях студента, предпочитаемых им стратегиях обучения, типичных ошибках. Важно организовать представление текущего уровня обученности, провести диагностику текущих знаний студента, а также указать информацию о том, что он не понимает, вместе с данными о предпочтительной стратегии обучения, такой как обучение на примерах или обучение по аналогии.

Блок «модель процесса обучения» формирует информационную модель, предъявляет информацию и оценивает качество деятельности студента. В этот блок внедряются знания о планировании и организации процесса обучения, об

общих и частных методиках обучения. Этот блок также обеспечивает реализацию различных интерактивных режимов обучения, таких как тренировка в процессе развивающей игры, постановка тестовых задач и вопросно-ответные процедуры [21].

В контексте создания интеллектуальных систем образовательного назначения по принципам ИОС можно выделить пять основных типов знаний. Предметные знания связаны с конкретным курсом. Стратегические и методические знания относятся к организации процесса обучения. Педагогические знания управляют деятельностью студентов. Эргономические знания организуют интерфейс преподавателей и студентов. Метазнания отвечают за способы компьютерной интеграции знаний [21].

Общие принципы построения интеллектуальных средств обучения включают принцип прагматической диагностики, который поддерживает выполнение учебного плана, принцип сопоставления текущей модели обучаемого с моделью идеального обучаемого, принцип «порождающих интерфейсов» для индивидуальной адаптации формы предъявления материала, принцип неэквивинальности обучения, который предполагает, что конечный уровень подготовки обучаемого никогда не соответствует точке в абстрактном пространстве состояний, и принцип необходимого разнообразия обучающих воздействий для выбора наиболее эффективных вариантов воздействий из всего набора.

Применение ИОС пока ограничено из-за трудностей формализации учебного материала, необходимости участия профессиональных программистов и сложности интерфейса взаимодействия пользователей с системой. Однако создание более совершенных ИОС, лишенных перечисленных недостатков, представляется вполне возможным в ближайшем будущем [21].

1.3. Влияние использования интеллектуальных систем на эффективность обучения

В современном образовании интеллектуальные системы оказывают значительное влияние на эффективность учебного процесса, трансформируя традиционные методы преподавания и открывая новые горизонты как для студентов, так и для преподавателей. Их воздействие проявляется в различных аспектах — от персонализации обучения до повышения доступности образовательных ресурсов.

Одним из ключевых преимуществ внедрения интеллектуальных технологий является возможность создания персонализированных образовательных программ. Искусственный интеллект способен анализировать данные об успеваемости, предпочтениях и темпе усвоения информации конкретного учащегося, что позволяет формировать индивидуальные учебные планы. Такая адаптация способствует более глубокому усвоению материала и росту учебной мотивации, ведь каждый студент получает знания в наиболее подходящей для него форме и темпе.

Значительно улучшаются также процессы оценки знаний и предоставления обратной связи. Интеллектуальные алгоритмы способны быстро и точно проверять задания, минимизируя влияние человеческого фактора. Благодаря автоматической обратной связи студенты могут немедленно видеть допущенные ошибки и своевременно вносить коррективы, что способствует более эффективному обучению и формированию навыков самоконтроля.

Важным вкладом интеллектуальных систем становится расширение доступности образования. Платформы на основе ИИ позволяют осваивать учебные курсы из любой точки мира в удобное для студента время, что особенно актуально для людей, проживающих в отдалённых регионах или имеющих физические ограничения. Благодаря этому создаются условия для более инклюзивного образования, доступного каждому.

ИИ становится надёжным инструментом индивидуальной поддержки обучающихся. Он может своевременно выявлять пробелы в знаниях и предлагать дополнительные ресурсы, упражнения или разъяснения. Такой подход позволяет студентам укреплять слабые стороны и обеспечивает более прочное освоение сложного материала.

Интеллектуальные технологии способствуют также развитию критического мышления. Задания, генерируемые ИИ, могут быть направлены на анализ, сопоставление фактов и выработку решений, что стимулирует студентов к самостоятельному мышлению, формированию аргументации и аналитических навыков.

Не менее важным направлением является образовательная аналитика. ИИ способен обрабатывать большие объёмы данных, собранных в ходе обучения, выявлять закономерности, слабые места и общие тенденции, что позволяет образовательным учреждениям улучшать содержание программ, адаптировать методики преподавания и делать обучение более эффективным и целенаправленным.

1.4 Автоматическая генерация вопросов в ИОС

Система контроля результатов обучения в высшей школе традиционно включает экзамены, зачёты, устный опрос, контрольные работы, коллоквиумы, рефераты, семинары, лабораторные работы и отчёты по практике. Но эти методы имеют ряд недостатков. Существует субъективность оценки, различия в требованиях и уровне строгости преподавателей могут приводить к несоответствиям в оценивании одного и того же ответа [1]. Психологические факторы оказывают существенное влияние на результаты тестирования. Возможная предвзятость преподавателя, использование студентами шпаргалок и списывание искажают достоверность оценки знаний [3]. Отсутствие чётких стандартов оценки приводит к недостаточно конкретно определённым критериям и объёму умений, необходимых для получения той или иной оценки [7]. Неполное охватывание материала является ещё одним существенным

ограничением, когда экзамены с ограниченным числом вопросов не позволяют полноценно оценить, насколько хорошо студент усвоил весь объём учебного материала, что может стимулировать к списыванию и поверхностному изучению тем [5].

Тестирование представляет собой стандартизированные, краткие и ограниченные во времени испытания, предназначенные для установления количественных и качественных индивидуальных различий. Тесты могут использоваться как метод педагогического измерения, включающий разработку системы тестовых заданий с заданными характеристиками, стандартизированную процедуру проведения и методы статистической обработки результатов [6].

Большинство интеллектуальных обучающих систем уделяют внимание предоставлению учебного материала, например, лекциям, видеороликам, графикам и интерактивным элементам. Однако подача информации — это только одна сторона учебного процесса. Вторая, не менее важная — контроль и оценка того, как усвоен материал студентами.

Традиционно проверка знаний требует большого количества ресурсов преподавателя, таких как составление тестовых заданий, их корректировка, проверка ответов. Чем объемнее и сложнее курс, тем больше усилий и времени нужно для обеспечения качественного тестирования. Особенно остро эта проблема проявляется в системах персонализированного обучения, где каждому студенту желательно подбирать индивидуальные задания и вопросы. В таких условиях ручное создание тестов быстро теряет эффективность.

Автоматическая генерация вопросов кажется логичным решением проблемы. Простая схема «есть текст — создай вопрос» порождает вопросы поверхностного уровня, часто не соответствующие образовательным целям. В учебных текстах присутствует насыщенная терминология, сложные логические связи между понятиями, а также большое количество примеров, которые нельзя использовать напрямую в качестве проверочных фактов. Это означает, что вопросы, созданные на основе простого шаблона или общей нейросети

оказываются недостаточно глубокими и плохо отражают реальный уровень понимания материала учащимся.

1.5 Готовые решения генерации вопросов

Применение готовых открытых решений, таких как модель `valhalla/t5-small-qg-hl`, изначально предназначенных именно для генерации вопросов не отвечают приемлемому качеству. Обучающие данные таких моделей чаще всего составлены из коротких фрагментов Википедии и датасетов общего назначения, далеких от реалий узкопрофильного учебного материала. Модель одинаково воспринимает определение термина и его применение в конкретной ситуации, она не чувствует разницы между проверкой поверхностного знания и понимания причинно-следственных связей. В результате модель стабильно генерирует однотипные вопросы вида: «Что такое искусственный интеллект?» или «Какова область применения машинного обучения?». Преподаватель же ожидает от автоматизации совершенно иных вопросов, например, «Почему машинное обучение предпочтительнее классических статистических методов при диагностике заболеваний?» или «Какие проблемы возникают при принятии решений с помощью современных систем ИИ?». Генерация подобных вопросов требует глубокого семантического анализа текста, чего существующие открытые модели обычно обеспечить не могут.

Использование более продвинутых моделей, таких как GPT, способно существенно повысить качество вопросов. Но в реальных условиях образовательного учреждения использование закрытых коммерческих API часто неприемлемо по причине требований к оффлайн-режиму и конфиденциальности данных. К тому же, даже GPT требует значительных усилий для постобработки полученных результатов, что вновь ставит вопрос о рациональности такого подхода.

Значительной проблемой автоматической генерации вопросов так же является постоянное дублирование по смыслу. Учебные материалы, особенно по узкоспециализированным дисциплинам, неизбежно содержат многократные

повторения ключевых терминов. Например, текст о нейронных сетях обязательно упоминает этот термин во многих разных контекстах: определения, примеры использования, теоретические и практические аспекты. Модели генерации, видя в своей задаче формулировку вопроса из каждого предложения отдельно, неизбежно начинают повторять одни и те же темы и понятия, формулируя вопросы, отличающиеся лишь стилистически, но не несущие новой информации. Повторяющиеся вопросы «Что такое нейронная сеть?», «Какие существуют нейронные сети?» не имеют педагогической ценности и вызывают усталость у обучающихся. Задача автоматической генерации требует не только правильного подбора модели, но и наличия механизма, который может контролировать и устранять подобные смысловые дубли на более высоком уровне обработки.

Использование шаблонов «Что такое {термин}?», «Опишите признаки {понятия}» кажется удобным способом избежать дублирования вопросов, но такой подход страдает недостатком гибкости и не отражает глубокого понимания материала. Шаблоны могут быть полезны для оценки простых уровней освоения материала (по таксономии Блума это уровни знания и понимания), однако совершенно бесполезны для вопросов более высокого порядка, связанных с применением, анализом и синтезом знаний. Кроме того, шаблонные вопросы часто оказываются формально корректными, но абсолютно неуместными по отношению к конкретному учебному материалу.

Шаблонные подходы могут использоваться лишь в ограниченных ситуациях, и только в качестве вспомогательного инструмента в общей архитектуре интеллектуальной генерации вопросов, а не как её основной компонент.

1.6 Основы адаптивного тестирования

Тестирование обладает рядом преимуществ по сравнению с традиционными методами контроля. Индивидуальный характер контроля позволяет отслеживать работу каждого студента и его личную учебную деятельность [1]. Регулярность и систематичность проведения тестов обеспечивает возможность контроля знаний на всех этапах процесса обучения [6]. Объективность тестирования исключает субъективные оценочные суждения преподавателя и обеспечивает единообразие требований ко всем испытуемым [3]. Тестирование может охватывать все разделы учебной программы и проверять теоретические знания, интеллектуальные и практические умения студентов [7]. Использование современных технологий, таких как компьютерное и адаптивное тестирование, также является значительным преимуществом [5]. Надёжность и валидность тестов обеспечивают полноценное педагогическое измерение уровня обученности и высокую содержательную и прогностическую валидность результатов [4].

Адаптивное тестирование является современным методом контроля знаний, где порядок предъявления тестовых заданий или их сложность зависят от ответов испытуемого на предыдущие задания, что позволяет динамически подстраивать тест под уровень подготовки каждого обучаемого, обеспечивая более точную и эффективную оценку знаний [3].

Основные особенности адаптивного тестирования:

1. Принцип обратной связи, где после каждого ответа система выбирает следующее задание, сложность которого зависит от предыдущего результата. Это обеспечивает индивидуализацию траектории тестирования и повышает точность оценки [5].

2. Индивидуализация траектории тестирования строится на том, что каждый обучающийся движется по своей индивидуальной траектории, получая задания, соответствующие его текущему уровню знаний. Это позволяет более точно определить способности испытуемого и создать комфортные условия

тестирования.

3. Адаптивное тестирование позволяет существенно экономить время, сокращая число предъявляемых заданий без потери качества оценки, так как задания подбираются оптимально с точки зрения информативности [3].

4. Дифференцирующая способность теста или отдельного задания показывает, насколько хорошо он может отличить испытуемых с разным уровнем знаний. Если все тестируемые дают один и тот же ответ на вопрос, это задание не способно различать их и считается малоинформативным.

5. Информативность задания зависит от разнообразия ответов, если почти все отвечают одинаково, задание не помогает определить уровень знаний испытуемых [7].

В системах адаптивного тестирования очень важна трудность тестовых заданий. Этот параметр обычно определяется так: чем меньше процент испытуемых, давших правильный ответ на задание, тем выше его трудность. Задания, которые слишком лёгкие или слишком сложные, оказываются малоинформативными. Оптимальными считаются задания средней сложности, на которые правильно отвечают около 50% тестируемых [8].

В традиционных системах тестирования основными критериями качества являются надёжность и валидность. Однако для адаптивного тестирования необходим новый критерий — эффективность. Эффективность можно определить как точность измерения знаний испытуемых с разным уровнем подготовки. Эффективный тест индивидуально оценивает знания каждого испытуемого, используя оптимальное по сложности и минимальное по количеству количество заданий.

Стратегии адаптивного тестирования подразделяются на двухшаговые и многошаговые. Двухшаговые стратегии включают два этапа: предварительную дифференциацию испытуемых по уровню знаний и адаптивный режим на втором этапе. На первом этапе используется общий тест для определения начального уровня, после чего на втором этапе тест адаптируется к каждому испытуемому [1].

Многошаговые стратегии предполагают многократную адаптацию теста на основе ответов испытуемого. В таких стратегиях задания выбираются из банка вопросов на основе результатов предыдущих ответов, что позволяет постоянно корректировать сложность и содержание теста в соответствии с уровнем знаний испытуемого [6].

1.7 Расширенные модели IRT, психометрические свойства и методы оценки параметров

Помимо уже рассмотренных моделей IRT (Rasch, 2PL, 3PL), существует ряд других моделей, которые используются для более сложного оценивания компетенций. Например, многопараметрические модели позволяют учитывать различные аспекты заданий, такие как дискриминация и вероятность угадывания, что повышает точность оценивания [3].

Многомерные модели IRT (MIRT) позволяют учитывать несколько скрытых факторов (способностей) одновременно, что особенно полезно при оценивании комплексных компетенций. В отличие от унитарных моделей, MIRT могут моделировать зависимости между различными компетенциями и предоставлять более детализированную информацию о профиле способностей обучаемого [9].

Полиномиальные модели IRT расширяют традиционные подходы, позволяя учитывать неоднородность популяции обучаемых. Такие модели полезны, когда в выборке присутствуют различные группы студентов с разными характеристиками, что может влиять на параметры заданий и оценки способностей [10].

Для обеспечения качества адаптивного тестирования важно учитывать психометрические свойства тестов, такие как валидность, надежность и справедливость.

Валидность теста определяется его способностью измерять то, что он предназначен измерять. В контексте IRT это означает, что параметры модели точно отражают реальные способности и характеристики заданий [11].

Надежность связана с консистентностью результатов тестирования. В IRT надежность может быть оценена через информационную функцию теста, которая показывает, насколько точно тест оценивает способности при различных уровнях θ [7].

Справедливость теста подразумевает отсутствие систематических искажений при оценивании различных групп обучаемых. Важно проверять, чтобы параметры заданий не зависели от принадлежности к определённым группам, если это не предусмотрено моделью.

А что конкретно у нас? Примеры ниже

Процесс оценки параметров моделей IRT включает несколько этапов, каждый из которых требует использования специфических алгоритмов.

Метод максимального правдоподобия (MLE) является наиболее распространённым методом оценки параметров. Он основан на максимизации функции правдоподобия, которая определяет вероятность наблюдаемых ответов обучаемых при заданных параметрах модели [6].

Метод Байесовской оценки использует априорные распределения параметров и обновляет их на основе наблюдаемых данных. Этот подход особенно полезен при работе с небольшими выборками или при наличии априорной информации о параметрах.

Эмпирический метод Байеса (EAP) сочетает преимущества MLE и Байесовских методов, предоставляя более устойчивые оценки параметров, особенно при наличии неопределённости [5].

1.8 Технологии обработки мультимедийных учебных материалов

Современные интеллектуальные обучающие системы всё чаще работают не только с текстовыми, но и с мультимедийными учебными материалами, включая видео- и аудиофайлы. Это особенно важно при создании инклюзивной и доступной образовательной среды, в которой студенты могут потреблять контент в удобной для них форме. Однако эффективное использование таких материалов требует их унификации — преобразования в текстовую форму,

пригодную для дальнейшего анализа, генерации тестовых заданий и интеграции в адаптивные механизмы. В рамках представляемого проекта реализована автоматизированная обработка мультимедиа-контента, включающая транскрипцию речи и машинный перевод.

Для распознавания речи из видео и аудиофайлов используется нейросетевая модель Whisper, разработанная компанией OpenAI. Модель запускается на центральном процессоре, что делает систему кросс-платформенной и независимой от наличия GPU. При загрузке преподавателем видеофайла система автоматически извлекает из него аудиотрек с помощью утилиты ffmpeg, а затем разбивает полученный звук на сегменты по 60 секунд. Каждый сегмент обрабатывается моделью Whisper для получения текстовой транскрипции. Результат объединяется и сохраняется в базе данных в виде текстового представления исходного контента (поле `generated_text` модели TopicContent). Следующим этапом является перевод полученного текста на английский язык. Это необходимо, поскольку используемые в системе модели генерации вопросов обучены преимущественно на англоязычных корпусах.

Для перевода используется модель MarianMT (Helsinki-NLP/opus-mt-ru-en), интегрированная через библиотеку Hugging Face Transformers. Перевод осуществляется построчно (по предложениям), что позволяет сохранить логическую структуру текста и повысить точность последующей генерации вопросов. Если исходный контент изначально предоставлен в текстовом формате (например, вводится вручную или загружается как текстовый файл), он также проходит этап определения языка и, при необходимости, перевода. Таким образом, весь мультимедийный контент, связанный с конкретной учебной темой, автоматически приводится к унифицированной текстовой форме на английском языке, что делает его пригодным для использования в модулях генерации тестов и адаптивной диагностики. Подобный подход не только повышает масштабируемость системы, но и значительно снижает нагрузку на преподавателей, освобождая их от необходимости вручную расшифровывать и структурировать учебные материалы.

1.9 Цели и задачи

Целью данной работы является разработка программного модуля интеллектуальной системы обучения, обеспечивающего автоматическую генерацию тестовых заданий на основе образовательного контента, с учётом индивидуальных особенностей обучающихся и архитектурных ограничений системы ILS. Решение должно быть ориентировано на последующую интеграцию с адаптивной моделью оценки знаний на основе Теории Ответов на Задания (IRT).

Для достижения этой цели также требуется провести глубокий анализ существующих алгоритмов генерации тестов. Это позволит определить, какие из подходов наиболее адаптированы к работе с текстовыми и мультимедийными образовательными материалами. На основе результатов анализа необходимо спроектировать модуль, который сможет не только формировать осмысленные и разнообразные вопросы, но и обеспечивать контроль качества генерируемых заданий. Особое внимание должно быть уделено интеграции модуля в инфраструктуру системы ILS с возможностью последующего расширения в сторону адаптивного тестирования на базе модели IRT.

ГЛАВА 2: РАЗРАБОТКА МОДУЛЯ ТЕСТИРОВАНИЯ В ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЕ ОБУЧЕНИЯ

2.1 Обзор инструментов

Современные образовательные технологии активно внедряют интеллектуальные обучающие системы, которые играют значимую роль в улучшении процесса обучения и формировании навыков студентов. При проектировании таких систем существенно обращать внимание на выбор подходящих инструментов, обеспечивающих не только надежность и масштабируемость, но и высокую производительность. В этом контексте представляется важным ознакомиться с ключевыми технологическими инструментами, которые применялись в разработке нашей интеллектуальной обучающей системы.

2.1.1 Обзор инструментов

Разработка ИОС - это сложная задача, которая требует грамотного выбора инструментов для обеспечения надежности, эффективности и масштабируемости. В мире веб-разработки существует множество фреймворков, каждый из которых обладает своими особенностями, преимуществами и недостатками.

Один из наиболее распространенных фреймворков, применяемых при создании веб-приложений, - Django. Этот высокоуровневый Python веб-фреймворк предоставляет всесторонний инструментарий для создания безопасных и масштабируемых веб-приложений.

Flask фреймворк ориентируется на простоту и гибкость. Он предоставляет все основные инструменты для создания веб-приложений, оставляя свободу выбора компонентов. Но в отличие от Django, Flask не содержит встроенных модулей для администрирования, аутентификации и других часто используемых функций, из-за чего могут понадобиться дополнительные усилия реализации и поиска сторонних библиотек для

полноценной реализации подобных функциональностей.

Ruby, Ruby on Rails фреймворк предоставляет инструменты для создания полноценных веб-приложений с минимальным объемом кода, но в контексте Python-ориентированной разработки, интеграция между Ruby on Rails и существующими библиотеками Python может потребовать дополнительных усилий.

JavaScript, Express.js на платформе Node.js ориентирован на создание масштабируемых веб-приложений, но, как и Flask, требует использования дополнительных модулей для базовых функциональностей.

Django сочетает в себе мощь и гибкость, предоставляя разработчикам инструменты, необходимые для создания высокопроизводительных веб-приложений. Встроенная административная панель, система аутентификации, ORM и широкий выбор встроенных компонентов делают Django полноценным решением для разработки ИОС.

Выбор фреймворка зависит от конкретных требований проекта и опыта команды разработчиков. Однако в контексте разработки Интеллектуальной обучающей системы Django предоставляет нам комплексное решение, сокращая время разработки и обеспечивая высокую производительность [22].

2.1.2 Django

Django представляет собой высокоуровневый веб-фреймворк, разработанный на языке программирования Python, и спроектированный для оперативного создания безопасных и удобно поддерживаемых веб-сайтов. С его помощью мы можем сосредоточиться на разработке нашего веб-приложения, избегая необходимости изобретения новых методов.

Одной из выдающихся особенностей Django является его модульность, встроенный ORM (Object-Relational Mapping) и удобная система маршрутизации, обеспечивающие быстрое создание и поддержку веб-приложений. Этот фреймворк предоставляет всесторонний инструментарий для решения различных задач веб-разработки, предоставляя при этом несколько

предпочтительных способов их реализации.

Django представляет собой мощный веб-фреймворк на языке Python, ориентированный на быстрое и структурированное создание надёжных веб-приложений. Его философия заключается в предоставлении разработчику всего необходимого инструментария "из коробки", что позволяет существенно сократить время на выбор и интеграцию сторонних решений. Такой целостный подход способствует совместимости компонентов, поддержанию единого архитектурного стиля и обеспечивает обширную документацию для каждого аспекта разработки.

Благодаря своей универсальности, Django успешно применяется для реализации веб-сервисов от систем управления контентом и корпоративных порталов до образовательных платформ и социальных сетей. Он легко интегрируется с разнообразными клиентскими приложениями и может отдавать данные в форматах HTML, JSON, XML и других, что делает его удобным как для классических сайтов, так и для API-ориентированных решений.

В Django особое внимание уделяется вопросам безопасности. Встроенные механизмы фреймворка автоматически защищают от наиболее распространённых уязвимостей, включая SQL-инъекции, XSS-атаки и подделку межсайтовых запросов. Также предусмотрены безопасные методы хранения паролей и управления сессиями пользователей, исключающие необходимость ручного контроля над этими аспектами.

Архитектура Django спроектирована с учётом масштабируемости: каждый компонент можно при необходимости заменить или масштабировать независимо от других. Такой подход успешно реализован в крупных проектах, например, Disqus, где надёжность и производительность имеют критическое значение.

Поддерживаемость кода обеспечивается принципами, такими как «Не повторяйся» (DRY), что поощряет повторное использование кода и улучшает читаемость проекта. А благодаря написанию на Python, Django обладает высокой переносимостью: приложения можно разворачивать на разных

операционных системах и серверах без необходимости серьёзных модификаций [23].

2.1.3. Плюсы и минусы Django

Одним из существенных преимуществ разработки на Django является наличие обширной экосистемы встроенных модулей и расширений, позволяющих существенно упростить реализацию функциональности. В наличии находится административная панель, механизмы аутентификации и авторизации, система форм с автоматической валидацией данных, встроенный сервер разработки, а также инструменты для работы со статикой и кэшированием. Такая модульность позволяет быстро адаптировать фреймворк под задачи различной сложности и направленности.

Django благодаря большому количеству дополнительных библиотек и пакетов успешно применяется как в простых проектах, так и в сложных корпоративных решениях. Одной из ключевых особенностей является объектно-реляционное отображение (ORM), которое обеспечивает взаимодействие с базой данных на уровне моделей Python, устраняя необходимость написания прямых SQL-запросов. Это не только упрощает разработку, но и способствует соблюдению принципов безопасности и читаемости кода.

Производительность Django поддерживается за счёт встроенных средств кэширования, гибкой системы управления статическими файлами и оптимизации запросов, что особенно важно при масштабировании приложений. В то же время фреймворк легко интегрируется с внешними сервисами и технологиями, включая WordPress, REST и GraphQL, а его фронтенд часть может быть реализована с использованием современных JavaScript-фреймворков, таких как React или Vue.js. Такая гибкость позволяет строить как монолитные, так и гибридные архитектуры, разделяя логику обработки между серверной и клиентской частями.

Django автоматически защищает от целого ряда типичных уязвимостей,

включая SQL-инъекции, XSS и CSRF-атаки. В сочетании с отличной документацией и активным сообществом это делает Django фреймворком, способствующим ускоренной разработке, особенно на этапе прототипирования и ввода в эксплуатацию.

Тем не менее, у Django есть и определённые ограничения. В частности, фиксированная структура проекта может оказаться громоздкой для небольших решений и ограничивает гибкость по сравнению с более лёгкими фреймворками вроде Flask. Кроме того, несмотря на все возможности оптимизации, Django не всегда подходит для систем с экстремально высокой нагрузкой, таких как крупные торговые площадки. Одним из технологических ограничений является преимущественно синхронный характер обработки запросов, что может быть критичным при большом количестве одновременных соединений. В таких случаях разработчикам рекомендуется использовать дополнения вроде Django Channels или асинхронные представления с использованием `asyncio`, что позволяет повысить отзывчивость системы при параллельной обработке входящих запросов [24].

2.1.4. Устройство Django

В типичной архитектуре информационного веб-приложения центральную роль играет обработка HTTP-запросов, поступающих от клиента. После получения запроса приложение анализирует URL-адрес, а также, при необходимости, извлекает данные из параметров запроса (GET) или из отправленных форм (POST). Далее производится логическая обработка — это может включать доступ к базе данных, выполнение вычислений или иные операции, соответствующие поставленной задаче. В результате приложение формирует ответ, чаще всего — в виде динамически сгенерированной HTML-страницы, структура которой определяется шаблоном, а содержимое — данными, полученными в процессе обработки.

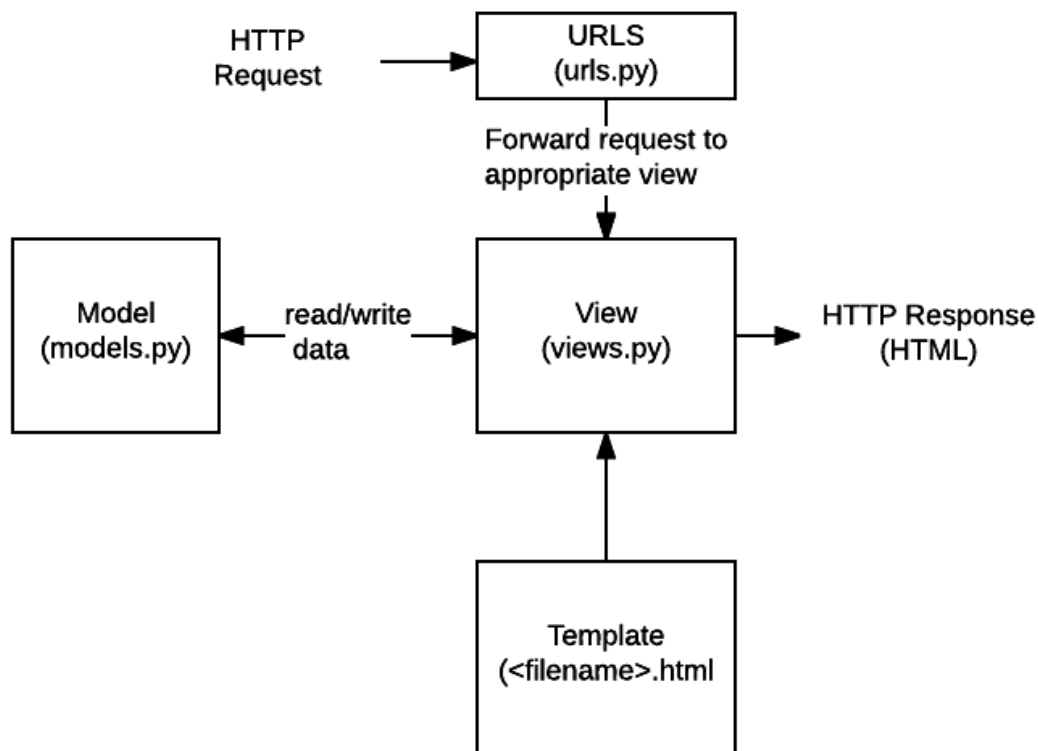


Рисунок 1 — Схема устройства Django

В Django подобная логика работы реализована с чётким разделением по компонентам, что способствует модульности и читаемости кода. Каждый этап обработки запроса делегирован отдельному элементу архитектуры.

Система маршрутизации (URLs) анализирует входящий URL и сопоставляет его с одним из заданных шаблонов. В отличие от монолитной обработки, Django позволяет направлять каждый запрос на строго определённую функцию или класс представления, автоматически передавая переменные из URL как аргументы.

Представление (Views) в Django — это функция или класс, принимающий запрос и формирующий ответ. Оно осуществляет основную бизнес-логику, взаимодействует с моделями и определяет, какие данные и в каком виде должны быть возвращены пользователю. Представления могут формировать HTML-страницы, возвращать JSON для API или перенаправлять пользователя на другие ресурсы.

Модель (Models) описывает структуру данных с помощью Python-

классов, которые напрямую связаны с таблицами базы данных. Django ORM позволяет выполнять операции чтения, записи и фильтрации данных без необходимости обращаться к SQL, делая работу с хранилищем данных более интуитивной и надёжной.

Система шаблонов (Templates) Django предназначена для отделения логики представления от визуального отображения. Шаблоны, как правило, используются для генерации HTML-страниц, но могут применяться и к другим форматам, включая XML или CSV. В шаблоны можно встраивать переменные, управляющие конструкции и фильтры, позволяющие динамически формировать содержание страниц на основе полученных из модели данных.

Такая организация проекта позволяет легко масштабировать приложение, изменять отдельные компоненты без затрагивания остальных и чётко разграничивать ответственность между обработкой логики и визуализацией данных [23].

2.2 Пользователи системы

2.2.1 Студенты

Основными пользователями интеллектуальной обучающей системы (ИОС) являются студенты, и именно на удовлетворение их образовательных потребностей ориентирована архитектура разрабатываемой платформы. Целью ИОС в данном контексте является обеспечение персонализированного подхода к обучению, адаптирующегося к индивидуальным особенностям, текущему уровню знаний и стилю восприятия каждого учащегося.

Одним из ключевых требований со стороны студентов является круглосуточный доступ к учебным материалам с различных устройств. Такая мобильность обеспечивает гибкость обучения, что особенно актуально в условиях насыщенного учебного или рабочего графика, когда возможность обучаться в индивидуально удобное время становится критически важной.

Для поддержания мотивации и повышения вовлеченности обучающихся

система должна содержать интерактивные элементы: автоматизированные тесты, тренировочные задания, викторины, а также мультимедийные лекции. Эти компоненты не только способствуют лучшему усвоению материала, но и делают образовательный процесс более живым, динамичным и интерактивным.

Дополнительно, студенты должны иметь возможность отслеживать собственный прогресс. Система должна предоставлять подробную и своевременную обратную связь, отображающую текущие достижения, пробелы в знаниях и направления, требующие дополнительного внимания. Такой подход способствует развитию навыков саморефлексии и формированию ответственности за собственное обучение.

2.2.2 Преподаватели

Преподаватели так же являются участниками образовательного процесса и одновременно основными модераторами и создателями учебного контента в ИОС. Для них система должна предоставлять расширенный инструментарий, обеспечивающий эффективное управление учебным процессом, контроль за прогрессом обучающихся и оперативную коррекцию методических стратегий.

Преподаватели должны иметь возможность создавать и структурировать учебные курсы, добавлять и обновлять материалы, планировать темы и задания. Интуитивный интерфейс и гибкая система управления контентом позволят адаптировать учебную программу под конкретные цели дисциплины и уровень подготовки студентов.

Одним из значимых компонентов является автоматизированная проверка заданий и тестов, а также возможность предоставления индивидуальной обратной связи. Это не только снижает нагрузку на преподавателя, но и обеспечивает оперативную оценку знаний обучающихся, позволяя в короткие сроки реагировать на затруднения студентов.

ИОС также должна обеспечивать преподавателям доступ к развернутой аналитике по каждому студенту и группе в целом. Сводные отчёты об успеваемости, динамике прохождения тем и результатах тестов позволяют

выявлять проблемные области и принимать обоснованные решения по корректировке учебного процесса, направленного на повышение эффективности обучения [25].

2.2.3 Администраторы

Администраторы системы ответственны за её техническую поддержку и обеспечение бесперебойной работы. Для них ИОС должна предоставлять инструменты для управления пользователями, мониторинга работы системы и обеспечения безопасности данных.

Администраторам необходима возможность управления пользователями. Возможность создания, редактирования и удаления учетных записей пользователей, управления их ролями и правами доступа. Это обеспечивает эффективное администрирование системы и контроль за доступом к различным функциям.

Не менее важны инструменты для мониторинга работы системы, включая журналирование действий пользователей, контроль за производительностью и генерацию отчетов о состоянии системы. Это позволяет своевременно выявлять и устранять технические проблемы.

Средства для защиты данных пользователей предотвращают несанкционированный доступ и обеспечивают соответствия требованиям законодательства о защите данных. Безопасность является критическим аспектом, особенно в образовательной среде, где обрабатываются персональные данные студентов и преподавателей [26].

2.3 Архитектура системы

Система включает четыре основных модуля: модуль предоставления знаний обучаемым, модуль выявления знаний обучающегося, модуль построения личностного портрета обучаемого и базу данных (см. рисунок 1).

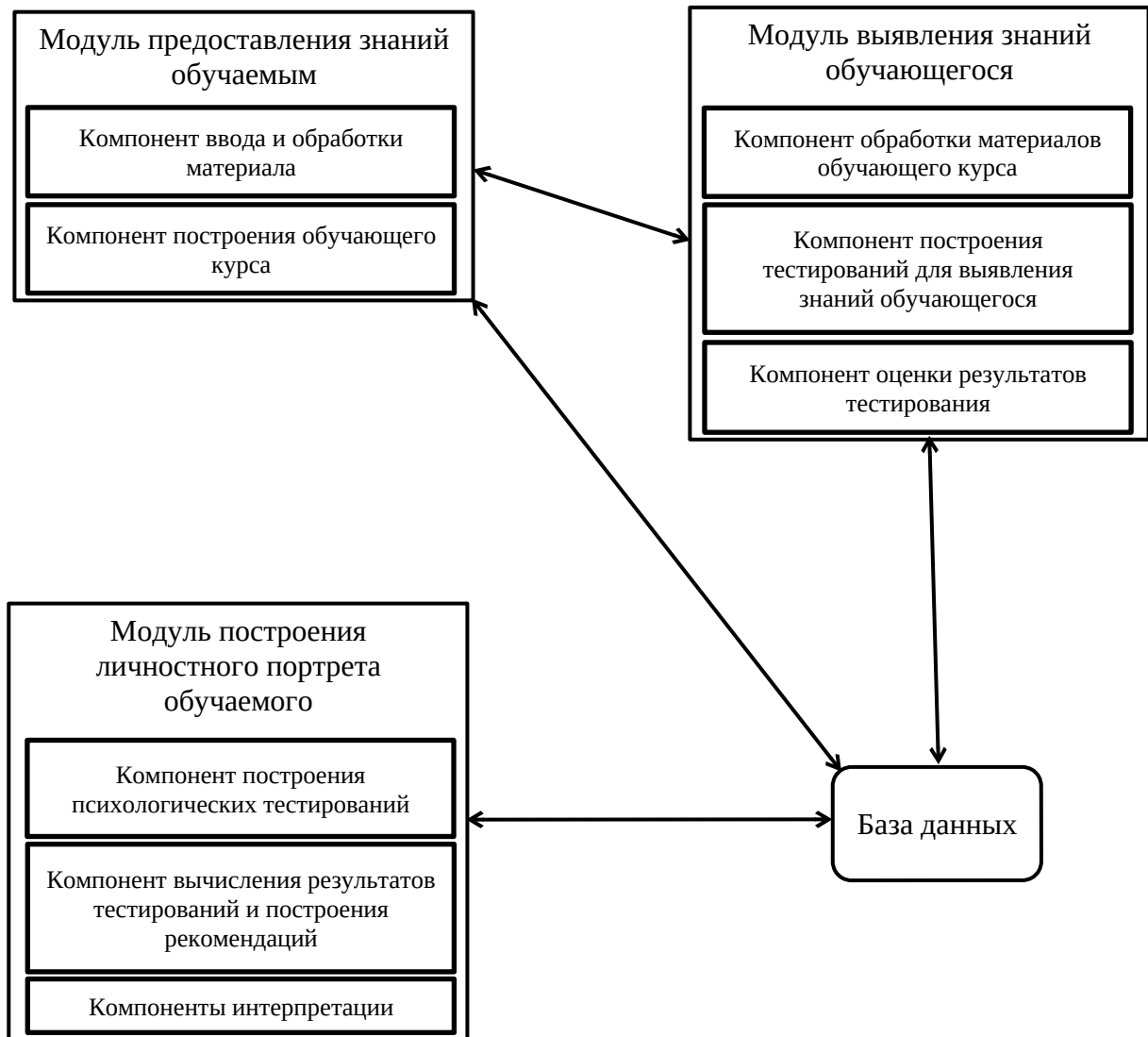


Рисунок 2 – Архитектурная схема интеллектуальной обучающей системы

Модуль предоставления знаний отвечает за формирование образовательного контента и является стартовой точкой взаимодействия студента с системой. Компонент ввода и обработки материала выполняет загрузку, анализ и структурирование предоставленных преподавателем текстов, видео, аудиозаписей и других источников. Включает этапы распознавания речи, перевода, извлечения ключевой информации и подготовки к генерации

тестов. Компонент построения обучающего курса организует структурированный учебный материал в соответствии с темами, подтемами и уровнем сложности. Результат передаётся в другие модули системы, в частности — в модуль выявления знаний обучающегося, что позволяет обеспечить целостность образовательной траектории.

Модуль выявления знаний отвечает за адаптивное тестирование и формирование обратной связи на основе анализа уровня усвоения учебного материала. Компонент обработки материалов обучающего курса принимает и анализирует подготовленные лекционные данные из модуля предоставления знаний. Компонент построения тестирований использует алгоритмы автоматической генерации заданий (в том числе на базе моделей машинного обучения) и включает преподавателя в контур подтверждения качества. Применяется кластеризация, фильтрация дублей, типизация вопросов и отбор дистракторов. Компонент оценки результатов тестирования фиксирует и интерпретирует ответы студентов, рассчитывает баллы, определяет уровень усвоения и предоставляет данные для обновления учебного плана. Полученные результаты записываются в базу данных и становятся входом для дальнейшей адаптации.

База данных центральное хранилище всей информации. Включает персональные данные пользователей, курсы, темы, структуры тестов и учебных материалов, результаты тестирований и психологических опросов, метаданные об активности пользователей и их прогрессе. Архитектура базы данных построена с учётом нормализации, масштабируемости и обеспечения безопасности. Обеспечивает централизованный обмен данными между модулями и позволяет отслеживать индивидуальную образовательную траекторию каждого пользователя.

2.4 База данных

В рамках работы разработана реляционная база данных. Архитектура базы данных построена с учётом требований к масштабируемости, безопасности и адаптивности.

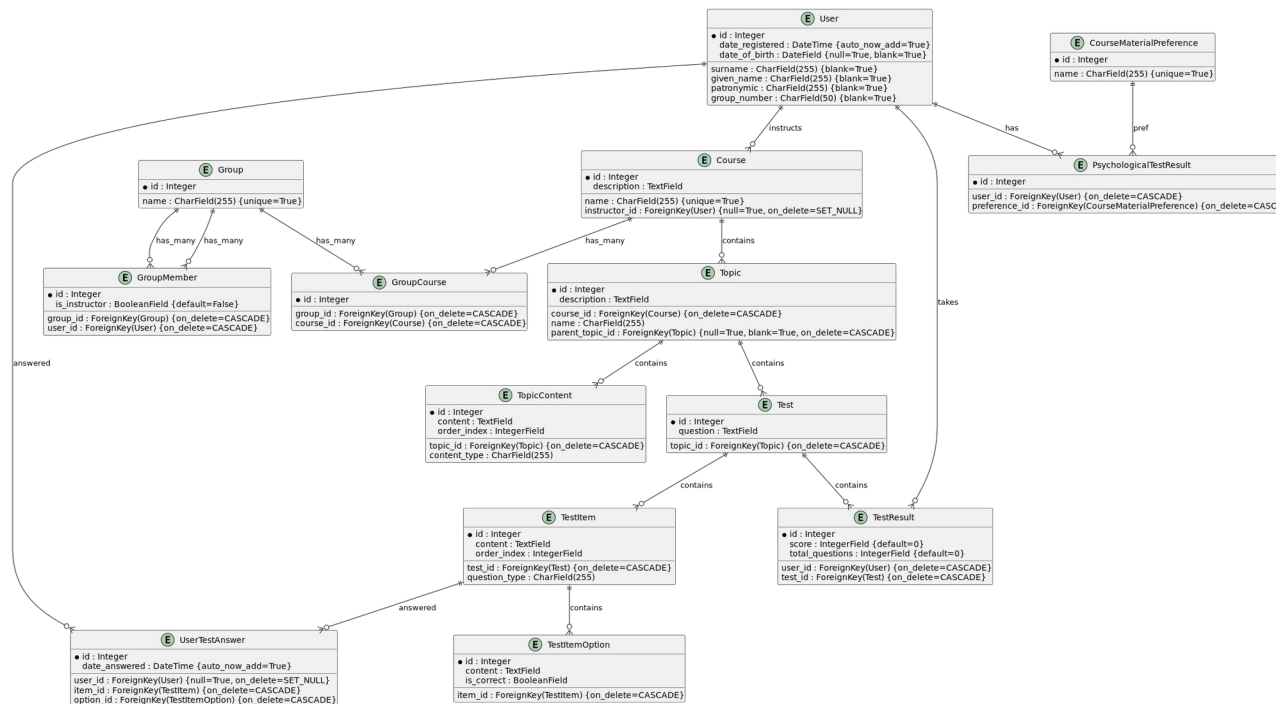


Рисунок 3 – Физическая модель базы данных системы

Модели, содержащие персональные данные используют встроенные механизмы безопасности Django, включая защиту от SQL-инъекций, XSS-атак и CSRF-уязвимостей. Для разграничения доступа применяются механизмы аутентификации и авторизации. На уровне модели User реализовано разделение ролей, где преподаватели имеют доступ к только своим функциям, а студенты — только к собственному обучению. Доступ к административному интерфейсу ограничен и предоставляется только суперпользователям или назначенным администраторам.

User — модель пользователя, расширяющая стандартную модель Django AbstractUser. Хранит фамилию, имя, отчество, номер учебной группы и дату рождения. Пользователи могут быть как студентами, так и преподавателями,

обеспечивая гибкое распределение ролей и разграничение доступа в системе.

Course — модель учебного курса. Содержит название и описание курса, а также ссылку на преподавателя (модель User). Каждый курс включает в себя набор тем (модель Topic) и может быть привязан к одной или нескольким учебным группам.

StudyGroup — модель учебной группы студентов. Хранит название группы и список курсов, назначенных этой группе (связь многие ко многим с моделью Course). При удалении группы автоматически удаляются все участники, связанные через модель GroupMember.

GroupMember — промежуточная модель, связывающая студентов и учебные группы. Содержит ссылки на модели StudyGroup и User. Ограничение unique_together гарантирует, что один студент может состоять только в одной группе.

Topic — модель темы курса. Содержит ссылку на курс (Course), название, описание темы, а также может включать иерархическую структуру через поле parent_topic. С темой связаны учебные материалы и тесты.

TopicContent — модель учебных материалов, привязанных к конкретной теме. Хранит тип контента (текст, файл, изображение, видео, аудио), текстовое или файловое содержимое, порядок отображения.

Test — модель теста, связанного с определённой темой (Topic). Содержит название, описание и флаг, указывающий, разрешено ли повторное прохождение теста.

TestItem — модель вопроса в тесте. Хранит ссылку на тест (Test), тип вопроса (один вариант, несколько, текстовый ответ), формулировку вопроса, правильный ответ (если применимо), порядок отображения и источник контента (если вопрос сгенерирован на основе материала). Также включает параметры модели IRT (используется Rasch-модель: $a = 1$, $c = 0$, b — параметр сложности).

TestItemOption — модель вариантов ответов на вопрос. Содержит ссылку на TestItem, текст варианта и флаг, обозначающий, является ли он правильным.

UserTestAnswer — модель, отражающая ответы пользователя. Хранит информацию о том, кто и на какой вопрос ответил, выбранные варианты (многие ко многим), текстовый ответ (если есть) и дату ответа.

TestResult — модель итогов прохождения теста. Содержит пользователя, тест, количество набранных баллов, общее число вопросов, оценку способности (θ) по IRT и дату прохождения теста.

TopicProgress — модель прогресса пользователя по теме. Включает связь с пользователем и темой, статус (не начато, в процессе, завершено), дату начала и окончания работы с темой, а также статистику по прохождению тестов.

TestRetakePermission — модель, фиксирующая разрешения на повторное прохождение теста. Хранит пользователя, тест, флаг разрешения и дату выдачи разрешения.

AdaptiveTestSession — модель, представляющая сессию адаптивного тестирования. Содержит пользователя, тест, текущую оценку способности (θ), список уже заданных вопросов, статус завершения сессии, текущий уровень сложности (легкий, средний, сложный) и дату начала теста. Благодаря этой модели система может динамически подбирать задания соответствующего уровня, что повышает точность оценки и мотивацию обучающегося.

PsychTest, PsychQuestion, PsychAnswer, PsychTestResult — блок моделей, отвечающих за реализацию психологических тестов. PsychTest — описание теста. PsychQuestion — вопрос с указанием психологического фактора (например, А, В). PsychAnswer — возможный ответ с соответствующим баллом. PsychTestResult — результат по конкретному фактору для пользователя. FactorInterpretation — модель, содержащая интерпретации значений психологических факторов.

2.5 Модель генерации вопросов как инструмент

Нейросетевая модель как инструмент не способен сразу выдавать качественный итог. В условиях образовательной системы, где приоритетом являются не количество, а релевантность и смысловая точность вопросов, попытки добиться результата только нейронной моделью приводят лишь к наращиванию вычислительных ресурсов, но не к росту качества.

На практике нейросеть выполняет задачу генерации чернового материала — большого массива возможных вопросов, без понимания их значимости или актуальности. Она не анализирует контекст курса, не знает, какие вопросы уже были заданы, не различает формальные перефразировки от содержательных различий. Наибольшую проблему представило дублирование вопросов. Модель, ориентированная на предложение-ответную генерацию формировала десятки вариантов одного и того же вопроса, изменяя лишь формулировку. Для преподавателя разницы между «Что такое машинное обучение?» и «Каковы особенности машинного обучения?» зачастую нет. Для модели — это совершенно разные запросы.

Поэтому в рамках архитектуры ILS модель была осознана как источник вариативного черновика, а не как финальный результат. Весь дальнейший процесс обработки — фильтрация, типизация, добавление дистракторов — стал обязательным этапом.

Дистрактор — это неправильный вариант ответа, включённый в задание с выбором, цель которого отвлечь внимание от правильного ответа и проверить глубину понимания материала. В тестовых заданиях с выбором дистракторы формируют контекст, в котором студенту нужно не просто узнать правильный ответ, а продемонстрировать осознанность выбора. В рамках системы дистракторы генерируются автоматически с помощью моделей смыслового анализа текста. Они подбираются так, чтобы быть похожими по теме, но семантически отличными от правильного ответа. Это позволяет избежать очевидно неверных или слишком простых вариантов и повысить

диагностическую ценность вопроса.

Фокус разработки сместился с "поиска лучшей модели" на построение многоуровневого конвейера доработки результата. Все же в конечном итоге нельзя полностью полагаться на результат модуля генерации вопросов, за учителем остаётся задача фильтрации и переработки вопросов при необходимости.

2.6 Реализация модели генерации вопросов

Изначально система воспринимала все вопросы как текстовые, что не отражало разнообразия педагогических целей. Разные форматы вопросов позволяют проверять разные уровни знаний, от банальных фактов до способности к анализу и синтезу информации.

Вместо ручной разметки типов вопросов был реализован автоматический алгоритм классификации [11]:

Single choice проверяет знание факта.

Multiple choice позволяет оценить полноту понимания.

Текстовый ответ ориентирован на развернутое объяснение.

Попытка вручную размечать типы вопросов сразу показала свою неэффективность при автоматической генерации. Было решено автоматизировать этот процесс на основе признаков самого ответа. Алгоритм учитывал длину ответа, его структуру (наличие перечислений, сложных конструкций) и семантическую полноту. Простые короткие ответы классифицировались как single choice, списочные или комплексные — как multiple choice, а всё, что требовало разъяснения, — как текст. Таким образом, типизация вопросов стала частью архитектуры, а не субъективным выбором преподавателя.

Стандартные методы вроде TF-IDF или косинусного сходства на bag-of-words-векторах показали свою слабость: они фиксируют лишь поверхностные различия в наборе слов, не улавливая смысловую близость. Было принято решение применить Sentence-BERT — модель, способную кодировать именно

смысл предложения.

Для группировки схожих вопросов использовался алгоритм DBSCAN. Заранее неизвестно сколько уникальных вопросов получится из конкретного учебного текста, поэтому алгоритмы вроде k-means были неприменимы. DBSCAN позволил автоматически выявлять смысловые кластеры и отсеивать выбросы, обеспечив тем самым фильтрацию повторов не по форме, а по содержанию.

Качественные дистракторы (неправильные, но правдоподобные варианты ответа) определяют реальную сложность теста. Примитивные или очевидно ложные дистракторы обесценивают вопрос, превращая тест в формальность [4].

Попытки автоматически генерировать дистракторы через ту же модель, что и вопросы, показали низкую эффективность: ответы были либо откровенно глупыми, либо очевидно неверными. В результате вектор поиска сместился к анализу самого контекста лекционного материала.

С помощью Sentence-BERT измерялась семантическая близость между правильным ответом и предложениями из исходного текста. В качестве дистракторов выбирались фрагменты, которые сохраняют тематическую связь, но не совпадают по смыслу с правильным ответом.

Начальный этап проекта использовал Google Translate, но зависимость от внешнего сервиса, нестабильность API, невозможность локальной работы и низкий уровень контроля над качеством перевода не позволяли отавить его в системе.

Решением стало внедрение модели MarianMT, позволяющей выполнять перевод локально, без подключения к внешним сервисам [12]. Подход к переводу был изменен, текст обрабатывался на уровне предложений с предварительной детекцией языка, что позволило корректно работать со смешанным контентом и минимизировать смысловые искажения.

2.7 Модели теории ответов на задание (Item Response Theory, IRT)

Теория ответов на задание (IRT) — это статистическая модель, используемая для анализа взаимодействия между характеристиками тестируемого и параметрами тестовых заданий. В отличие от классической тестовой теории (КТТ), которая предполагает, что все задания имеют одинаковую значимость и сложность, IRT учитывает индивидуальные различия в способностях тестируемых и параметры заданий, такие как сложность и дискриминация [7].

Наиболее распространена трёхпараметрическая логистическая модель IRT, которая описывается следующей формулой:

$$P_i(\theta) = c_i + (1 - c_i) \cdot \frac{1}{1 + e^{-a_i(\theta - b_i)}}$$

$P_i(\theta)$ — вероятность правильного ответа на задание i при уровне способности θ ,

a_i — параметр дискриминации задания i ,

b_i — параметр сложности задания i ,

c_i — параметр угадывания задания i .

Компоненты модели IRT:

Дискриминация (a_i): Определяет, насколько хорошо задание различает между тестируемыми с разными уровнями способностей. Высокий a_i указывает на высокую способность задания к дискриминации.

Сложность (b_i): Определяет уровень способности, при котором вероятность правильного ответа составляет 50%. Более высокое значение b_i означает более сложное задание.

Угадывание (c_i): Отражает вероятность правильного ответа даже при низком уровне способности, что особенно актуально для заданий с несколькими вариантами ответа.

Способность (θ): Отражает уровень знаний или умений тестируемого. Более высокие значения θ соответствуют более высоким способностям.

Алгоритм адаптивного тестирования на основе IRT включает следующие шаги:

1. Выбор следующего задания. Рассчитывается информативность каждого доступного задания по формуле:

$$I_i(\theta) = \frac{[P'_i(\theta)]^2}{P_i(\theta)(1 - P_i(\theta))}$$

$P'_i(\theta)$ — производная функции вероятности.

Выбирается задание с наибольшей информативностью для текущего уровня способности θ .

2. Предъявление задания. Испытуемому предлагается выбранное задание.

3. Обновление оценки способности (θ). После ответа пересчитывается уровень способности с использованием метода максимального правдоподобия или байесовских методов.

4. Повторение процесса. Шаги 1–3 повторяются до достижения заданной точности оценки или других условий остановки (например, максимальное количество заданий).

Основное преимущество подхода IRT заключается в возможности определения уровня способности с высокой точностью, используя меньшее количество заданий по сравнению с традиционным тестированием [5].

2.8 Интеграция экспертных систем и IRT

Модель IRT (Item Response Theory) представляет собой эффективный инструмент для построения адаптивных тестов, позволяющий оценивать способности обучающихся на основе их ответов. Однако её традиционное применение ограничивается исключительно оценкой «правильно» или «неправильно», не учитывая при этом качественные характеристики ответа, такие как наличие частичного понимания или систематических ошибок. Для преодоления этих ограничений целесообразно использовать интеграцию моделей IRT с экспертными системами (ЭС), что открывает новые возможности для более тонкой настройки адаптивного тестирования.

Экспертные системы — это интеллектуальные программные средства,

основанные на базе знаний специалистов и способные принимать решения в определённой предметной области. Интеграция ЭС с моделью IRT позволяет учесть дополнительные параметры ответа: выявление типичных ошибок, анализ уровней частичного знания, а также использование правил перехода между заданиями. Применение продукционных правил типа «ЕСЛИ — ТО» даёт возможность гибко управлять логикой формирования последовательности тестовых заданий, опираясь не только на уровень сложности, но и на содержательные характеристики ответов.

Основное преимущество такого подхода заключается в расширении адаптивных возможностей тестирования. Система может адаптироваться не только количественно, изменяя сложность заданий, но и качественно — подбирая задания, соответствующие выявленным пробелам в знаниях. Это позволяет достигать более высокой точности в оценке способностей обучаемого, интегрируя поведенческие и содержательные данные в процесс диагностики.

Интеграция IRT и ЭС может быть реализована как на макроуровне, так и на микроуровне. На макроуровне экспертная система определяет приоритетные области знаний, на которых следует сосредоточить тестирование. На микроуровне она управляет выбором следующего задания, основываясь на анализе предыдущих ответов и заданных экспертных правил [13].

В этом контексте особенно значимыми становятся принципы дифференциации и индивидуализации обучения, широко обсуждаемые в современных образовательных парадигмах. Дифференциация предполагает учет индивидуальных особенностей обучающихся через формирование групп по уровню подготовки, интересам или иным критериям. В рамках адаптивного тестирования это может выражаться в создании тестов разного уровня сложности: базового, среднего и продвинутого. Например, начальные задания для всех обучаемых могут быть одинаковыми, а дальнейшие — настраиваться в зависимости от результатов, продемонстрированных ранее.

Индивидуализация же направлена на создание уникального

образовательного маршрута для каждого обучаемого, учитывая его когнитивные особенности, предпочтительный стиль восприятия информации и темп усвоения материала. Адаптивные алгоритмы, в том числе основанные на данных психологических тестов, могут использовать параметры, такие как уровень тревожности, мотивация или тип мышления, для подбора наиболее подходящих заданий.

Адаптивное тестирование, совмещающее в себе возможности IRT, экспертных систем, дифференциации и индивидуализации, формирует персонализированную траекторию прохождения теста. Это не только повышает объективность и точность диагностики, но и способствует росту учебной мотивации, снижению уровня стресса и созданию условий для максимально эффективного усвоения знаний.

2.9 Использование технологий адаптивного тестирования

Адаптивное тестирование представляет собой современный подход к контролю знаний, позволяющий не только точно оценивать уровень подготовки обучающихся, но и эффективно реализовывать принципы индивидуализации и дифференциации в образовательном процессе. Основное его назначение — создание гибкой системы оценки, способной подстраиваться под особенности каждого обучаемого и обеспечивать высокую точность диагностики.

Одним из ключевых преимуществ адаптивного тестирования является высокая объективность получаемых оценок, благодаря чему снижается влияние субъективных факторов, нередко присутствующих при традиционных способах оценивания [5]. Помимо этого, такая система способствует значительной экономии времени: за счёт интеллектуального подбора заданий по уровню обучаемого удаётся минимизировать общее количество вопросов без ущерба для точности результата. Ещё одним положительным эффектом является повышение учебной мотивации: тестируемый получает задания, соответствующие его текущему уровню, что предотвращает преждевременную фрустрацию или, наоборот, скуку, и делает процесс тестирования

психологически комфортным и интересным [14].

В современной практике применяются различные технологии адаптивного тестирования. Одной из распространённых является двухшаговая модель, предполагающая начальную дифференциацию — предварительное определение уровня знаний — с последующим адаптивным этапом, на котором система динамически подбирает задания в соответствии с ранее выявленным уровнем. Более сложные, многошаговые технологии предполагают последовательное проведение нескольких адаптивных раундов и применяются, например, в целях текущего контроля, промежуточной и итоговой оценки, а также в рамках долговременного мониторинга индивидуального прогресса обучающегося.

Алгоритм адаптивного тестирования, как правило, включает несколько этапов. На первом этапе производится определение стартового уровня знаний, который служит отправной точкой для подбора начальных заданий. В дальнейшем процесс тестирования осуществляется по принципу итеративной адаптации: после каждого ответа система пересчитывает оценку способности обучаемого и на основании этого результата выбирает следующее задание соответствующей сложности. Финальный этап заключается в предоставлении обобщённой оценки, индивидуально адаптированной под особенности прохождения теста конкретным пользователем. Такая модель позволяет не только объективно зафиксировать уровень знаний, но и отразить динамику усвоения материала, учитывая индивидуальные траектории обучающихся.

2.10. Модели внутренней и внешней адаптации в тестировании

Адаптивное тестирование может быть реализовано с использованием различных моделей адаптации. Среди них выделяют модель внутренней адаптации и модель внешней адаптации.

2.10.1. Модель внутренней адаптации

Модель внутренней адаптации представляет собой подход в адаптивном тестировании, при котором каждое тестовое задание сопровождается набором подсказок различной степени информативности и стоимости. Ключевая особенность данной модели заключается в том, что адаптация происходит не за счёт изменения состава заданий, а через предоставление возможности тестируемому самостоятельно управлять уровнем сложности посредством обращения к подсказкам. Таким образом, обучающийся может скорректировать сложность задания в зависимости от своих текущих знаний и уверенности, что повышает точность диагностики его реальных способностей [14].

Основными характеристиками модели являются наличие у каждого задания нескольких заранее подготовленных подсказок и механизм учёта их использования в итоговой оценке. Решение о применении подсказки принимается самим тестируемым, что способствует формированию навыков саморегуляции и контроля за собственным образовательным процессом. Система оценки при этом учитывает количество использованных подсказок: итоговый балл за выполнение задания пропорционально снижается в зависимости от объема полученной помощи.

Модель внутренней адаптации обладает рядом значительных преимуществ. Во-первых, она способствует индивидуализации тестирования, обеспечивая более точное соответствие заданий уровню подготовки каждого обучаемого. Во-вторых, предоставление возможности получить подсказку снижает тревожность и стресс, которые часто сопровождают процесс тестирования, особенно в условиях повышенной ответственности. В-третьих,

обучение с использованием данной модели способствует развитию метакогнитивных стратегий, таких как самоконтроль и осознанное принятие решений.

Однако, несмотря на обозначенные достоинства, модель имеет и определённые ограничения. В первую очередь, это высокие требования к разработке контента: создание заданий с несколькими уровнями подсказок требует значительных временных и трудовых ресурсов. Кроме того, возникают сложности с масштабированием системы — внедрение модели в условиях массового тестирования или в крупных образовательных учреждениях требует значительных затрат на поддержку, сопровождение и верификацию заданий. Эти факторы существенно ограничивают широкое применение модели в реальной практике [5].

2.10.2. Модель внешней адаптации

Модель внешней адаптации предполагает адаптацию тестирования за счёт изменения последовательности и сложности предъявляемых заданий на основе ответов тестируемого. В этой модели система автоматически выбирает следующее задание в зависимости от результатов предыдущих ответов, что обеспечивает индивидуализацию процесса тестирования.

Характеристики модели внешней адаптации включают диагностику по разделам, что позволяет оценивать знания по каждому разделу учебного курса отдельно. Используется метод дихотомии, при котором банк вопросов делится на классы сложности, и система выбирает задания соответствующей сложности на основе ответов тестируемого. Адаптивное тестирование по разделам обеспечивает итоговый подсчёт средней оценки по всему курсу, а особенности предоставления заданий предусматривают предложение двух вопросов из одного класса сложности для снижения вероятности угадывания ответов.

Преимуществами данной модели являются автоматическая адаптация сложности заданий, что делает процесс тестирования более точным и эффективным. Простота реализации без необходимости создания подсказок

облегчает внедрение системы, а более высокая объективность оценивания способствует справедливой оценке знаний тестируемых [1]. Однако модель внешней адаптации имеет и некоторые недостатки. Она обладает меньшей гибкостью в сравнении с моделью внутренней адаптации, что может ограничивать возможности индивидуализации процесса тестирования. Кроме того, отсутствует возможность развития навыков саморегуляции у тестируемых, что может снижать их способность самостоятельно оценивать и регулировать свой уровень подготовки [6].

Модель внешней адаптации предлагает эффективный способ автоматической настройки сложности заданий, обеспечивая объективность оценивания и простоту реализации. Тем не менее, её ограниченная гибкость и отсутствие механизма развития саморегуляции у тестируемых могут стать препятствием для её полного применения в некоторых образовательных контекстах.

2.10.3. Сравнение моделей внутренней и внешней адаптации

Общие черты обоих подходов заключаются в том, что они направлены на индивидуализацию процесса тестирования, стремятся повысить точность оценки знаний и адаптируют сложность заданий в соответствии с уровнем подготовки тестируемого.

Различия между внутренней и внешней адаптацией состоят в том, что внутренняя адаптация позволяет тестируемому самостоятельно регулировать сложность заданий через использование подсказок, что способствует развитию навыков саморегуляции. Однако этот подход требует значительных ресурсов для разработки заданий с подсказками. Внешняя адаптация, напротив, контролируется системой, которая автоматически выбирает задания на основе ответов тестируемого. Это упрощает процесс разработки тестов, но не способствует развитию самостоятельности у тестируемых [14].

2.10.4. Трудности и перспективы реализации моделей

Трудности реализации модели внутренней адаптации включают сложность разработки качественных подсказок, которые должны быть эффективными и соответствовать различным уровням подготовки тестируемых. Кроме того, создание базы данных заданий требует значительных временных и ресурсных затрат, так как необходимо разработать множество вариантов заданий с различными подсказками. Также возникает проблема оценки эффективности модели, поскольку для этого требуется наличие большого количества заданий с подсказками, что усложняет процесс анализа и проверки результатов [5].

В случае модели внешней адаптации существуют свои трудности. Во-первых, требуется высокое качество и актуальность базы данных заданий, чтобы система могла правильно подбирать задания в зависимости от ответов тестируемого. Во-вторых, необходимо тщательно калибровать задания по уровням сложности, что требует значительных усилий и экспертизы. База данных должна регулярно обновляться для поддержания её актуальности и соответствия современным образовательным стандартам, что также связано с дополнительными затратами и ресурсами [14].

Перспективы развития моделей адаптивного тестирования включают комбинирование внутренних и внешних подходов для создания гибридных систем, которые сочетают преимущества обоих методов. Такие системы позволяют повысить гибкость и точность тестирования, обеспечивая более индивидуализированный подход к каждому тестируемому. Кроме того, использование технологий искусственного интеллекта открывает новые возможности для автоматизации создания подсказок, анализа ответов и прогнозирования затруднений с помощью машинного обучения. Это не только снижает трудозатраты, но и повышает эффективность адаптации тестов. Также перспективным направлением является расширение областей применения адаптивных моделей для различных предметных областей, уровней

образования и типов обучаемых, включая взрослых и обучающихся с особыми потребностями.

2.11 Система генерации и оценивания тестовых заданий

Разработка эффективной системы генерации и оценивания тестовых заданий является ключевым элементом адаптивного тестирования. Такая система должна обеспечивать индивидуализацию процесса тестирования, точность оценки знаний и способствовать развитию необходимых компетенций у обучаемых.

Построение теста с учётом сложности и уровня компетенции предполагает, что тестируемому предлагается тест, состоящий из заданного количества тестовых заданий разной сложности и соответствующих уровней компетенции. Сложность тестового задания влияет на количество баллов, которые тестируемый может получить за верный ответ, а уровень компетенции задания зависит от текущего уровня компетенции пользователя по данной теме. Чем больше баллов имеет пользователь по теме, тем выше его уровень компетенции [5].

Система оценивания баллов устроена следующим образом: задания низкой сложности оцениваются в 1 балл, задания средней сложности — в 2 балла, а задания высокой сложности — в 3 балла. За неверный ответ пользователь теряет 1 балл, а за частично верный ответ получает от 0 до 2 баллов в зависимости от степени совпадения его ответа с эталонным.

Типы тестовых заданий включают задания закрытого типа, где тестируемый выбирает один или несколько правильных ответов из предложенных вариантов, задания открытого типа, требующие самостоятельного формулирования ответа без предложенных вариантов, а также задания на установление соответствия, предполагающие сопоставление элементов двух множеств.

Тест: Приложения искусственного интеллекта.

Текущая сложность: easy

В каких сферах применяется искусственный интеллект?

В каких сферах применяется искусственный интеллект? _____

- ☐ Здравоохранение
- ☐ Финансы
- ☐ Транспорт
- ☐ Образование
- ☐ Производство
- ☐ Торговля
- ☐ Религия
- ☐ Медиа

[Ответить](#)

Рисунок 4 – Вопрос сопоставления элементов двух множеств

По окончании теста тестируемому выводится сообщение о результате тестирования, содержащее количество набранных баллов из максимально возможного и итоговую оценку, которая является результатом тестирования.

Результаты теста: Приложения искусственного интеллекта.

Ваш результат: 4 из 10

Вопрос	Ваш ответ	Правильный ответ	Статус
В каких сферах применяется искусственный интеллект?	Здравоохранение, Финансы, Транспорт, Образование	Здравоохранение, Финансы, Транспорт, Образование, Производство	Неверно

Рисунок 5 – Результат тестирования ученика

Эффективная система генерации и оценивания тестовых заданий должна учитывать различные аспекты сложности и типов заданий, обеспечивая при этом прозрачную и справедливую систему оценивания. Это позволяет не только точно оценивать текущий уровень знаний обучаемого, но и способствовать его дальнейшему развитию и совершенствованию компетенций.

Алгоритм оценивания ответов на задания с открытым ответом представляет особую сложность и требует использования специальных

методов. Существует три основных подхода к оцениванию таких ответов.

Первый подход — абсолютная оценка, при котором происходит полное совпадение ответа пользователя с эталонным ответом. Это обеспечивает высокую точность оценивания, однако может быть не гибким в случаях, когда существуют различные формулировки правильного ответа.

Второй подход — проверка по маске, в рамках которого используется маска с отмеченными ключевыми словами. Этот метод позволяет учитывать наличие определённых элементов в ответе, что делает оценивание более гибким и адаптивным к различным вариантам правильных ответов.

Третий подход — проверка по формуле, который позволяет учитывать орфографические ошибки и отклонения от эталонного ответа, используя коэффициенты ошибок. Такой метод повышает устойчивость системы оценивания к неточным формулировкам и опечаткам, обеспечивая более справедливую оценку знаний тестируемых [14].

2.12 Алгоритм формирования тестовых вопросов

Первоначально система принимает различные типы учебных материалов, включая текстовые документы, аудио и видео файлы. Для видеофайлов извлекается аудиодорожка, которая затем транскрибируется с помощью модели распознавания речи. Полученный текст анализируется на предмет языка, и при необходимости переводится на английский для дальнейшей обработки.

На основе обработанного текста система использует модели машинного обучения для генерации вопросов. Модель, обученная на задаче создания вопросов, формирует вопросы, соответствующие содержанию материала. Параллельно применяется модель вопросно-ответной системы, которая генерирует корректные ответы на созданные вопросы. После этого вопросы и ответы при необходимости переводятся обратно на исходный язык.

Для предотвращения дублирования и повышения разнообразия вопросов система проводит кластеризацию сгенерированных вопросов с использованием эмбедингов и алгоритма DBSCAN. Из каждого кластера выбирается наиболее

представительный вопрос, обеспечивая тем самым уникальность и разнообразие тестовых заданий.

Сформированные вопросы и ответы сохраняются в базе данных как элементы тестов, связанные с соответствующими темами и подтемами. Каждый вопрос помечается параметрами сложности, что позволяет системе учитывать уровень подготовки обучающегося при подборе заданий.

Во время прохождения теста система динамически подбирает вопросы, максимально соответствующие текущему уровню знаний студента. Это достигается за счет сравнения параметров сложности вопросов с оценкой способности обучающегося (θ). Правильные ответы приводят к повышению сложности последующих вопросов, а неправильные — к её снижению, обеспечивая тем самым точную оценку компетенций.

После завершения тестирования система анализирует ответы обучающегося, выявляет типичные ошибки и пробелы в знаниях. На основе этого анализа обновляется индивидуальный учебный план, который включает рекомендации по дополнительным темам и ресурсам для устранения выявленных недостатков. При необходимости назначаются дополнительные занятия или консультации с преподавателем.

Система регулярно проводит калибровку параметров сложности вопросов на основе накопленных данных о прохождении тестов реальными студентами. Это позволяет поддерживать актуальность модели и повышать точность оценивания знаний. Кроме того, проводится симуляция прохождений тестов виртуальными студентами для проверки корректности работы алгоритмов и выявления потенциальных улучшений.

2.13 Улучшение алгоритма формирования тестовых вопросов

Интеллектуальная система тестирования предназначена для адаптивного формирования тестовых заданий, учитывая индивидуальные особенности и уровень знаний каждого обучающегося. В основе системы лежит иерархическая структура базы знаний, которая организует учебный материал по темам и подтемам, расположенным по уровням сложности и взаимосвязям между ними [5].

Алгоритм работы системы включает несколько этапов. На первом этапе система анализирует какие темы и элементы базы знаний присутствуют в системе в конкретной теме.

Следующим этапом является формирование первичного набора тестовых вопросов. На основе выявленных тем система формирует набор вопросов, направленных на оценку усвоения материала на соответствующем уровне иерархии базы знаний. Сложность вопросов соответствует уровню подготовки ученика и может быть адаптирована в зависимости от его успеваемости [7].

После этого обучающийся приступает к прохождению сформированного теста. В процессе тестирования ему последовательно предъявляются вопросы из первичного набора.

По завершении тестирования система проводит детальный анализ ответов учащегося. Оценивается правильность каждого ответа. Система сопоставляет результаты с установленными критериями оценки, включая процент правильных ответов и уровень сложности успешно решённых заданий. Этот анализ позволяет определить, насколько хорошо ученик усвоил изученный материал, и выявить темы, требующие дополнительного внимания [8].

Если по результатам анализа выясняется, что ученик недостаточно хорошо усвоил определённые темы и результат ниже заданного порога, система предоставляет возможность сгенерировать вторичный набор вопросов по теме.

2.14 Внедрение адаптивного тестирования в текущую систему

Архитектура системы построена на основе иерархической структуры, которая включает модели для пользователей, курсов, тем, контента, тестов и ответов пользователей. Модель теста (Test) связана с моделью темы (Topic), которая, в свою очередь, относится к определённому курсу. Тесты состоят из вопросов (TestItem) различных типов: выбор одного или нескольких вариантов ответа, текстовые ответы и другие. Процесс генерации вопросов автоматизирован с использованием моделей машинного обучения. Модуль генерации вопросов и ответов реализован в файле `questionanswergeneration.py` (см. приложение А). Контент, связанный с темами курсов, проходит несколько этапов предварительной обработки — извлечение аудио из видео, транскрипция речи, автоматический перевод и семантический анализ (см. приложение В). Это обеспечивает унификацию и стандартизацию материалов для формирования тестовых заданий.

Разработанный интерфейс системы удобен как для студентов, так и для преподавателей. Дашборды для разных типов пользователей (`StudentDashboardView`, `InstructorDashboardView`) показывают необходимую информацию о прогрессе обучаемого, доступных тестах и управлении учебными материалами. Подробный код, реализующий описанные функциональности, размещён в соответствующих приложениях: Приложение А — Генерация вопросов и ответов (`questionanswergeneration.py`), Приложение В — Обработка контента (`contentprocessing.py`), Приложение С — Представления для генерации тестов (`views.py`), Приложение D — Проведение тестирования и адаптивное тестирование (`views.py`), Приложение Е — Анализ результатов и адаптация (`views.py`).

ГЛАВА 3: ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ АДАПТИВНОГО ТЕСТИРОВАНИЯ НА БАЗЕ RASCH-МОДЕЛИ IRT

3.1. Реализация ключевого функционала

Изначально система позволяла создавать курсы, темы, материалы и статические тесты. Однако задача адаптации потребовала введения нескольких ключевых моделей и полей. Была разработана модель `AdaptiveTestSession`, которая создаёт адаптивные тесты. Хранится текущая оценка способности (θ), список уже заданных вопросов ученику и признак завершённости теста.

В модели `TestItem` были добавлены параметры IRT, в частности параметр b , отражающий сложность задания. В рамках Rasch-модели параметры a и c фиксируются и принимают значения $a=1$ и $c=0$, что упрощает калибровку и интерпретацию модели. На практике параметры a и c задаются статически, а параметр b калибруется на основе собранных данных о прохождении тестов реальными студентами.

При разработке модуля генерации важно определить, отвечает ли результат поставленным требованиям качества. В отличие от задачи классификации или поиска ошибок, здесь не существует однозначного критерия "правильности". Вопрос либо выполняет свою функцию в обучении, либо нет — и это всегда зависит от контекста.

Нельзя оценивать эффективность модуля создания тестов по количеству сгенерированных вопросов. Чрезмерное количество повторяющихся или бессмысленных вопросов создает впечатление низкого качества работы системы.

Поэтому главным критерием стало качество отобранных вопросов после всей цепочки обработки: фильтрации, кластеризации, типизации [12]. Необходимо ориентироваться не на абсолютные числа, а на собственное восприятие следующих факторов:

Насколько сгенерированные вопросы соответствуют содержанию учебной темы.

Удалось ли отфильтровать очевидные повторы и формальные перефразировки.

Выглядит ли итоговый список вопросов как полноценный, логичный тест, а не как случайный набор строк.

3.2 Повышение качества генерации вопросов

На ранних этапах генерации вопросов возникла проблема: модель воспроизводила однотипные вопросы с незначительными вариациями формулировок. При этом вопросы часто касались одних и тех же понятий, которые многократно встречались в исходном тексте.

После внедрения смысловой кластеризации на базе Sentence-BERT ситуация значительно улучшилась. Итоговый список вопросов стал восприниматься как более осмысленный и разнообразный. Вопросы перестали "конкурировать сами с собой" за внимание, а дублирующие формулировки — растворяться среди уникальных вариантов.

Количество вопросов после фильтрации заметно сократилось. Но при этом ощущение полноты теста, наоборот, усилилось: стало проще видеть, какие аспекты темы действительно охвачены, а какие требуют доработки вручную.

Формирование дистракторов стало важной частью работы над качеством тестов. Генерацией случайных "неправильных" ответов оказалась бесполезна. Подобные дистракторы либо слишком явно проигрывали по смыслу правильному ответу, либо выпадали из логики темы.

Решение пришло через семантический анализ: использовалась смысловую дистанцию между правильным ответом и другими предложениями текста. Этот подход позволил подбирать дистракторы, которые оставались в рамках темы, но не дублировали правильный ответ по смыслу.

Этот метод не идеален. В процессе работы неоднократно были ситуации, когда дистрактор оказывался либо слишком близким к истине, либо, наоборот, слишком очевидно неверным. Тем не менее, общее качество тестов благодаря дистракторам заметно повысилось: они превратили тесты из списка открытых

вопросов в полноценные задания с выбором ответа, что упростило как прохождение теста, так и его проверку.

3.3. Параметры IRT

Сложность вопросов "easy", "medium" и "hard" привязана к базовым значениям параметра b в Rasch-модели: задания "easy" имеют значение $b=-1.0$, "medium" — $b=0.0$, а "hard" — $b=1.0$. Простые задания характеризуются более низким значением b , что соответствует меньшему уровню θ для 50% успеха, тогда как сложные задания имеют более высокое значение b . Параметры $a=1$ и $c=0$ остаются фиксированными для всех заданий, что соответствует упрощённой реализации Rasch-модели [5].

Количество прохождений теста: 1

Вопрос	Сложность	a	b	c	Эмпир. P	Модельная P при $\theta=0$
Что такое философия?	Средний	1.000	0.000	0.000	0.000	0.500
Философия является частью нашей культуры?	Средний	1.000	0.000	0.000	0.000	0.500
Что ищет философ?	Средний	1.000	0.000	0.000	0.000	0.500
Что такое философия?	Средний	1.000	0.000	0.000	0.000	0.500
Как философия влияет на всех?	Средний	1.000	0.000	0.000	0.000	0.500

Рисунок 6 – Сложность тестов

Rasch-модель описывается следующей формулой:

$$P_{ij} = \frac{1}{1 + e^{-(\theta_j - b_i)}}$$

где: P_{ij} — вероятность правильного ответа на задание i студентом j при уровне способности θ_j , b_i — параметр сложности задания i , θ_j — уровень способности студента j .

Калибровка параметра b_i осуществляется методом максимального правдоподобия. После накопления данных о прохождении тестов реальными студентами система оценивает значения b_i , при которых наблюдаемые ответы наиболее вероятны. Для этого используется итеративный градиентный метод, который корректирует значения b_i на основе разницы между эмпирической частотой правильных ответов и предсказанной моделью вероятностью P_{ij} [5].

Каждый раз, когда студент отвечает на вопрос, система пересчитывает

его уровень способности θ с помощью метода максимального правдоподобия. В реализации используется градиентный подход для обновления значения θ . Следующее задание выбирается исходя из текущего уровня способности θ студента, оптимально приближаясь к нему по сложности, что обеспечивает высокую информативность каждого вопро

После обновления θ система выбирает следующее задание, максимально соответствующее текущему уровню способности студента. В Rasch-модели это задание с параметром b , близким к θ .

Адаптивный тест: Тест по теме 'Тема 1. Что такое философия?' #1

Текущая сложность: medium

Что такое философия?

наука

Что такое философия?

Ответить

Адаптивный тест: Тест по теме 'Тема 1. Что такое философия?' #1

Текущая сложность: easy

Философия является частью нашей культуры?

Философия является частью нашей культуры?

Ответить

Рисунок 7 – Адаптивный тест

Калибровка параметров b_i выполняется после накопления достаточного объёма данных о прохождении тестов. Для стабильной калибровки требуется большое количество прохождений тестов. В реальной системе параметры b корректируются периодически для учёта новых данных. Упрощённая реализация позволяет демонстрировать принцип, но для точной калибровки необходимо использовать более сложные методы и большие объёмы данных [5].

Добро пожаловать, Рученов Иван Иванович!

Основы Искусственного Интеллекта

В этом курсе вы познакомитесь с основами Искусственного Интеллекта.

Курс в процессе

Прогресс по курсу: **28.57%**

Завершено тем: 2/7

Средний результат по тестам: **61.11%**

Оценка (5-балльная): **3**

Темы в этом курсе:

Представление знаний в интеллектуальных системах. — Завершена (результат: 50.0%)

Основные понятия и определения в области искусственного интеллекта. Направления исследований. — Не начато

Этические аспекты использования искусственного интеллекта. — Не начато

Приложения искусственного интеллекта. — Завершена (результат: 70.0%)

Методы работы со знаниями в интеллектуальных системах. — Не начато

Искусственный интеллект сегодня. — Не начато

История искусственного интеллекта. — Не начато

Рисунок 8 – Сводка по успеваемости ученика

3.4. Визуализация в системе

Для преподавателей реализовано отображение текущих значений параметров b , эмпирической вероятности правильных ответов и итоговой оценки θ . Это позволяет отслеживать, как калибровка улучшает соответствие между моделью и реальными данными.

Количество прохождений теста: 1204

Вопрос	Сложность	a	b	c	Эмпир. P	Модельная P при $\theta=0$
Что такое философия?	Сложный	1.000	3.000	0.000	0.100	0.047
Философия является частью нашей культуры?	Легкий	1.000	-3.000	0.000	0.904	0.953
Что ищет философ?	Легкий	1.000	-3.000	0.000	0.914	0.953
Что такое философия?	Сложный	1.000	3.000	0.000	0.095	0.047
Как философия влияет на всех?	Легкий	1.000	-3.000	0.000	0.899	0.953

Рисунок 9 – Калибровка параметров IRT

Реализована возможность многократного прохождения тестов студентами. При повторном прохождении создаётся новая сессия тестирования, не зависящая от предыдущих попыток.

Управление повторными прохождением теста

Тест:

Выберите студентов для разрешения повторного прохождения:

- ☐ sim_user_b89097af
- ☐ sim_user_f9eacc05
- ☐ sim_user_4bde9848

Рисунок 10 – Повторное прохождение тестов.

Для быстрой калибровки параметров была реализована функция симуляции прохождений тестов виртуальными студентами с различными значениями θ . Это позволило протестировать корректность алгоритмов калибровки и адаптивного подбора заданий.

Количество симуляций:

Отладочная информация

Количество прохождений теста: 1204

Рисунок 11 – Симуляция прохождений

После того как достаточное количество студентов прошли тест, преподаватель запускает процедуру калибровки параметров b . Система анализирует собранные данные и обновляет значения параметров для каждого задания, улучшая точность модели.

Отладочная информация

Количество прохождений теста: 1204

Вопрос	Сложность	a	b	c	Эмпир. P	Модельная P при $\theta=0$
Что такое философия?	Сложный	1.000	3.000	0.000	0.100	0.047
Философия является частью нашей культуры?	Легкий	1.000	-3.000	0.000	0.904	0.953
Что ищет философ?	Легкий	1.000	-3.000	0.000	0.914	0.953
Что такое философия?	Сложный	1.000	3.000	0.000	0.095	0.047
Как философия влияет на всех?	Легкий	1.000	-3.000	0.000	0.899	0.953

Рисунок 12 – Калибровка после накопления данных

3.5 Симуляция и автоматическая калибровка параметров тестирования

Для надёжной работы адаптивного тестирования на основе модели IRT, необходимо, чтобы параметры тестовых заданий (в частности параметр сложности b в Rasch-модели) соответствовали реальной статистике прохождений тестов. Однако на ранних этапах внедрения адаптивного тестирования таких данных недостаточно. Чтобы решить эту проблему, в рамках проекта была реализована система симуляции прохождений тестов виртуальными студентами, что позволяет автоматически накапливать данные и проводить калибровку параметров заданий.

Виртуальные прохождения выполняются с использованием представления `SimulatePassesView`, вызываемого вручную через интерфейс преподавателя. Виртуальные пользователи создаются динамически, каждому назначается случайная, но контролируемая истинная способность θ (например, из нормального распределения), и далее они проходят тест в соответствии с алгоритмом адаптивного тестирования. Каждый шаг симуляции включает выбор следующего вопроса с учётом текущего значения θ , вероятностное принятие решения о правильности ответа, фиксацию ответа и пересчёт θ с использованием градиентного подхода. В результате создаются реалистичные пользовательские траектории прохождений, которые можно использовать как основу для калибровки.

После накопления достаточного объёма данных включается механизм рекалибровки параметров Rasch-модели, реализованный в функции `calibrate_item_parameters`. Он использует оценки θ , сохранённые в результатах прохождений (`TestResult`), и вычисляет новые значения параметров b для каждого задания на основе метода Ньютона-Рафсона. Алгоритм итеративно приближает значения, минимизируя отклонение между эмпирической и модельной вероятностью правильного ответа. Все вычисления оптимизированы для больших объёмов данных и выполняются фоном, при необходимости,

каждые 10 прохождений.

На практике внедрение симуляции и автоматической рекалибровки позволило значительно повысить надёжность адаптивного алгоритма на ранних этапах. Без реальных пользователей система начинала с плохо откалиброванных параметров, что приводило к неинформативному подбору заданий. Благодаря симуляции, параметры b покрывают весь спектр от лёгких до сложных заданий, обеспечивая высокую точность оценки способности студентов уже после первых реальных попыток. Кроме того, итеративная калибровка накапливает эмпирические данные, что делает выбор заданий более обоснованным и стабильным. Такой подход также позволяет безопасно тестировать адаптивные сценарии, не подвергая реальных студентов риску некорректной оценки.

3.6 Повышение качества вопросов с помощью кластеризации

Одной из проблем, возникающих при автоматической генерации тестовых заданий из учебного материала, является дублирование и однотипность вопросов. На раннем этапе проекта было выявлено, что при генерации из одного текста система может создавать несколько схожих по смыслу и формулировке вопросов. Это снижает ценность теста как инструмента диагностики, увеличивает субъективную нагрузку на студента и ухудшает репутацию интеллектуальной системы как "умной". Для решения данной проблемы был реализован механизм кластеризации сгенерированных вопросов с использованием Sentence-BERT и DBSCAN.

После того как модель генерирует пары "вопрос-ответ", каждый вопрос преобразуется в векторное представление с помощью предобученной модели SentenceTransformer('paraphrase-MiniLM-L6-v2'), которая обучена на задачах семантического сравнения предложений. Эти эмбединги используются в алгоритме кластеризации DBSCAN (Density-Based Spatial Clustering of Applications with Noise), который объединяет вопросы по признаку близости их смыслового содержания в векторном пространстве.

Каждый кластер отражает группу схожих вопросов, из которых выбирается только один представитель — как правило, первый входящим или наиболее типичный пример. Остальные вопросы игнорируются, что позволяет устранить дублирование. Алгоритм не требует предварительного задания количества кластеров и эффективно справляется даже в случае перемешанных и "шумных" наборов генераций.

До внедрения кластеризации вопросы в автоматически сгенерированном тесте были смысловыми дубликатами. После добавления кластеризации, повторяемость снизилась, а разнообразие тем и формулировок существенно возросло. Это позволило в дальнейшем повысить доверие к автоматической генерации тестов и сократить необходимость ручного редактирования. Кроме того, благодаря кластеризации стала возможна типизация заданий и отбор только уникальных вопросов для последующего распределения по уровням сложности и построения адаптивной модели теста.

Вопрос 5

Тип вопроса: Несколько вариантов ▾

Какие сферы деятельности используют искусственный интеллект в целях оптимизации процессов?

Вопрос:

- Образование ✓ [Ред.](#) [Удал.](#)
- Производство ✓ [Ред.](#) [Удал.](#)
- Здравоохранение [Ред.](#) [Удал.](#)
- Финансы [Ред.](#) [Удал.](#)
- Транспорт [Ред.](#) [Удал.](#)

[Добавить вариант ответа](#)

a	b	c	$P(\theta=0)$	Эмпирическая P
1.000	0.000	0.000	0.500	0.000

[Удалить вопрос](#)

Вопрос 6

Тип вопроса: Несколько вариантов ▾

В каких отраслях технологии искусственного интеллекта наиболее активно используются?

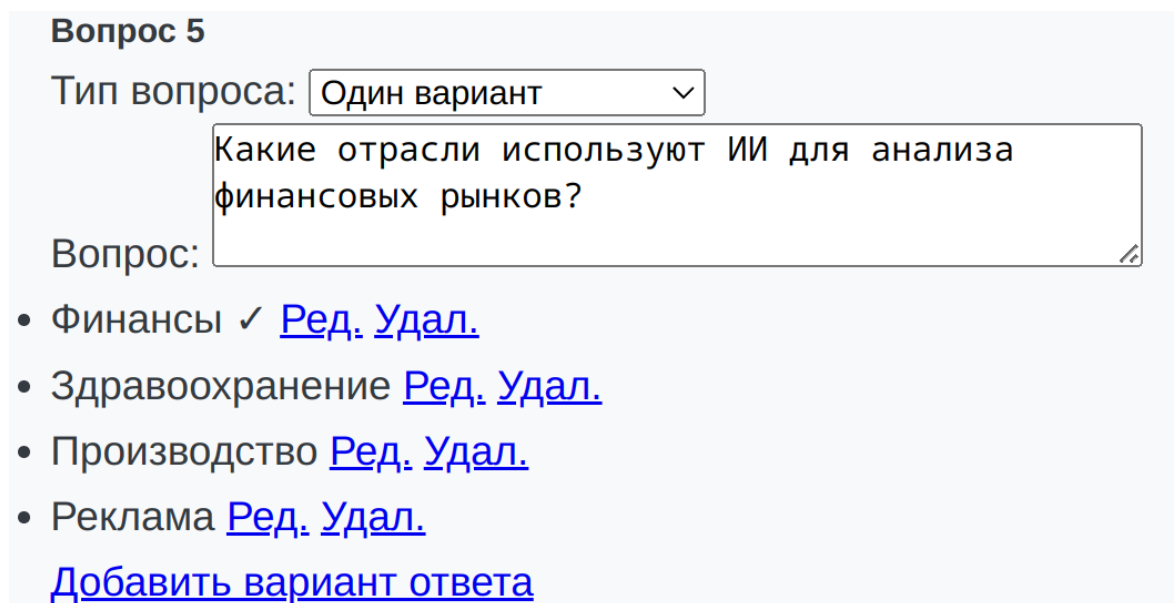
Вопрос:

- Здравоохранение ✓ [Ред.](#) [Удал.](#)
- Финансы ✓ [Ред.](#) [Удал.](#)
- Транспорт ✓ [Ред.](#) [Удал.](#)
- Производство ✓ [Ред.](#) [Удал.](#)

Рисунок 13 – Дублирующиеся вопросы

3.7 Работа модуля создания тестов в рамках ILS

До внедрения улучшенного модуля генерации вопросов результат работы модуля чаще всего служил лишь отправной точкой для ручной доработки. Система создавала "черновик", который требовал значительных усилий по очистке от дублирующих и нерелевантных вопросов.



Вопрос 5

Тип вопроса: Один вариант ▾

Вопрос:

- Финансы ✓ [Ред.](#) [Удал.](#)
- Здравоохранение [Ред.](#) [Удал.](#)
- Производство [Ред.](#) [Удал.](#)
- Реклама [Ред.](#) [Удал.](#)

[Добавить вариант ответа](#)

Рисунок 14 – Генерация вопросов без дополнительных улучшений

После интеграции нового пайплайна (с кластеризацией, типизацией, дистракторами и локальным переводом) работа с модулем изменилась. Модуль стал восприниматься как инструмент, с которого начинается составление теста. Результат, хоть и по-прежнему требует финального просмотра учителем, стал значительно более осмысленным уже "на выходе". Привязка каждого вопроса к конкретному источнику контента (TopicContent) позволила анализировать возникающие ошибки студентов не в общем виде, а относительно конкретных фрагментов учебного материала. Подход усилил связку между тестами и содержанием курса [15].

Вопрос 4

Тип вопроса: Несколько вариантов ▾

Вопрос: В каких сферах применяется искусственный интеллект?

- Здравоохранение ✓ [Ред.](#) [Удал.](#)
- Финансы ✓ [Ред.](#) [Удал.](#)
- Транспорт ✓ [Ред.](#) [Удал.](#)
- Образование ✓ [Ред.](#) [Удал.](#)
- Производство ✓ [Ред.](#) [Удал.](#)
- Торговля [Ред.](#) [Удал.](#)
- Религия [Ред.](#) [Удал.](#)
- Медиа [Ред.](#) [Удал.](#)

[Добавить вариант ответа](#)

Рисунок 15 – Генерация вопросов после дополнительных улучшений

Благодаря расширенному функционалу модуля преподаватель может запускать повторную генерацию вопросов по обновлённым материалам темы, а также использовать фильтры для отбора вопросов по типу, сложности и связанным компетенциям. Модуль обладает возможностью ручного редактирования отдельных элементов теста без необходимости пересоздания всего набора вопросов. Преподаватель может скорректировать формулировку, изменить сложность или внести правки в дистракторы, сохраняя гибкость и контроль над качеством итогового теста.

Тест: Ручной тест

None

[Сгенерировать вопросы для теста](#)

[Добавить вопрос](#) [Редактировать тест](#) [Удалить тест](#)

Всего прохождений: 0

[Сохранить изменения](#)

Рисунок 16 – Инструменты учителя по работе с тестами

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа была направлена на создание компонента интеллектуальной обучающей системы, обеспечивающего адаптивное выявление знаний обучаемых на основе модели Раша. В ходе исследования была разработана целостная архитектура, сочетающая в себе нейросетевые технологии генерации заданий и статистические модели адаптивного тестирования. Такая интеграция позволила не только автоматизировать процесс формирования проверочных материалов, но и учесть индивидуальные особенности каждого обучающегося.

Практическая реализация системы включала создание программного прототипа, обеспечивающего полный цикл работы с учебным контентом, от загрузки и обработки мультимедийных материалов до формирования тестов и анализа результатов. В процессе работы был реализован механизм генерации вопросов, основанный на современных трансформерных моделях, что позволило достигнуть приемлемого качества создаваемых заданий. Для повышения разнообразия и устранения смысловых дублей был использован алгоритм кластеризации, а также реализованы методы семантического анализа дистракторов, что значительно повысило диагностическую ценность итоговых тестов.

Особое внимание уделялось построению адаптивного тестирования. Использование модели Раша позволило гибко управлять сложностью заданий и динамически корректировать уровень оценки обучаемого в процессе тестирования. Благодаря реализации механизма симуляции и рекалибровки, система способна адаптироваться к изменениям в выборке обучающихся, сохраняя точность оценивания. Визуализация результатов и аналитика, доступные преподавателю, дополнили функциональность, обеспечивая целостную картину успеваемости и давая возможность адресной корректировки учебной траектории.

Сочетание методов машинного обучения, адаптивных моделей

тестирования и семантического анализа открывает новые перспективы в области интеллектуального образования. Разработанная система отвечает современным требованиям цифровой трансформации образования и может быть использована как в рамках традиционного обучения, так и в дистанционных форматах. Полученные результаты подтверждают достижение поставленных целей и создают основу для дальнейших исследований и расширения функциональности платформы.

Полученные в рамках дипломного проекта результаты получили научное признание и были частично представлены в публикации, подготовленной для сборника тезисов Международной математической конференции «Современные математические модели в энергетике», посвящённой памяти профессора В. А. Тупчиева. В совместной работе под названием «Персонализированный подход к формированию индивидуальных стратегий обучения рабочего персонала тепловых электростанций» в соавторстве: Фонталина Е., Заволженский В., Щепалов С., Сергиенко Н., были изложены ключевые идеи, связанные с адаптацией образовательного процесса под личностные характеристики обучающихся [27]. Концепции, представленные в этой работе, во многом легли в основу подходов, реализованных при проектировании и разработке интеллектуальной обучающей системы, описанной в настоящем исследовании.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Smith, J. (2021). Digital transformation in education: Challenges and opportunities. *Education & Technology Journal*, 34(4), 45-60.
2. Brusilovsky P., Millán E. User Models for Adaptive Hypermedia and Adaptive Educational Systems // *The Adaptive Web*. – 2007.
3. Brown, T. (2019). Adaptive testing: Advancing the quality of student assessments. *Learning Analytics & Assessment*, 11(3), 78-92.
4. Rasch, G. (1960). Probabilistic models for some intelligence and attainment tests. Copenhagen: Danmarks Paedagogiske Institut.
5. Embretson, S. E., & Reise, S. P. (2000). Item response theory for psychologists. Mahwah, NJ: Lawrence Erlbaum Associates.
6. Johnson, R., Brown, T., & Lee, K. (2020). Assessment practices in higher education: Trends and gaps. *Academic Review of Education*, 28(2), 123-136.
7. Lord, F. M. (1980). Applications of item response theory to practical testing problems. Hillsdale, NJ: Lawrence Erlbaum Associates.
8. Weiss, D. J. (1982). Improving measurement quality and efficiency with adaptive testing. *Applied Psychological Measurement*, 6(4), 473-492.
9. Reckase, M. D. (2009). Multidimensional item response theory. Springer.
10. Samejima, F. (1997). Graded response model. In W. J. van der Linden & R. K. Hambleton (Eds.), *Handbook of modern item response theory* (pp. 85–100). Springer.
11. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // *EMNLP 2019*.
12. Tiedemann J., Thottingal S. OPUS-MT — Building open translation services for the World // *EAMT 2020*.
13. Tomlinson, C. A. (2001). How to differentiate instruction in mixed-ability classrooms. Alexandria, VA: Association for Supervision and

Curriculum Development (ASCD).

14. Wainer, H., & Dorans, N. J. (2000). Computerized adaptive testing: A primer. Hillsdale, NJ: Lawrence Erlbaum Associates.

15. Anderson L.W., Krathwohl D.R. A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. – Longman, 2001.

16. Воробьева И. А. РАЗВИТИЕ ИНТЕЛЛЕКТУАЛЬНЫХ ОБУЧАЮЩИХ СИСТЕМ КАК ПЕДАГОГИЧЕСКОЕ ЯВЛЕНИЕ: ПЕРИОДЫ С 1998 Г. ПО НАСТОЯЩЕЕ ВРЕМЯ // Международный научно-исследовательский журнал № 11 (113) Часть 3 Ноябрь 2021 г. - С 43-48. DOI: <https://doi.org/10.23670/IRJ.2021.113.11.083>

17. Информационные и коммуникационные технологии в образовании: учебно-методическое пособие / И. В. Роберт, С. В. Панюкова, А. А. Кузнецов, А. Ю. Кравцова; под ред. И. В. Роберт. — М. : Дрофа, 2008. — 312, [8] с. : ил. — (Высшее педагогическое образование).

18. Е.А. Мясоедова, Г.А. Будникова Информационная образовательная среда учреждения: понятие, структура, проектирование // Астраханский институт повышения квалификации и переподготовки: Вестник РУДН, серия Информатизация образования, 2012, № 2 - С 82 - 90.

19. Гладкая В. С., Панасюк А. А. Интеллектуальные ситемы обучения // Белорусский государственный университет информатики и радиоэлектроники г. Минск, Республика Беларусь: БГУИР, 2018 г. - С 359

20. Белянская О.В., Привалов А.Н. Модели интеллектуальной системы обучения по техническим дисциплинам // Научный поиск: личность, образование, культура. 2022. No 4. С. 17–20. <https://doi.org/10.54348/2022.4.3>

21. А.С. Омарбекова, Б. Исмагамбетов, А.Сундетова Интеллектуальные обучающие системы // Евразийский национальный университет им.Л.Н.Гумилева, г.Астана, Казахский Агротехнический университет им.С.Сейфуллина, г.Астана - С 1- 5

22. <https://blog.back4app.com/> (2023). Курс [Электронный ресурс]. Режим доступа: Top 10 Backend Frameworks In 2023: Which One Is The Best? (дата обращения: 15.11.2023).
23. <https://developer.mozilla.org/> (2023). Курс [Электронный ресурс]. Режим доступа: Django введение - Изучение веб-разработки (дата обращения: 18.11.2023).
24. <https://proglib.io/> (2023). Курс [Электронный ресурс]. Режим доступа: Django - что это такое и для чего нужно: обзор и установка фреймворка для Python, разработка первого проекта (дата обращения: 22.11.2023).
25. "Educational Data Mining: A Review of Methods and Applications," IEEE Transactions on Learning Technologies, 2018.
26. "Managing and Supporting Educational Technologies: A Guide for Administrators," Educational Technology & Society, 2019.
27. Фонталина Е., Заволженский В., Пивень Н., Щепалов С., Сергиенко Н. Персонализированный подход к формированию индивидуальных стратегий обучения рабочего персонала тепловых электростанций // Современные математические модели в энергетике: сб. тез. Междунар. науч. конф., Обнинск, 25–26 окт. 2024 г. – Обнинск: ИАТЭ НИЯУ МИФИ, 2024. – С. 83.

ПРИЛОЖЕНИЕ А

Листинг А.1 – Генерация вопросов и ответов.

```
translator = Translator()
tokenizer_qg = AutoTokenizer.from_pretrained("valhalla/t5-small-qg-hl", use_fast=False)
model_qg = AutoModelForSeq2SeqLM.from_pretrained("valhalla/t5-small-qg-hl")
tokenizer_qa = AutoTokenizer.from_pretrained("deepset/roberta-base-squad2")
model_qa = AutoModelForQuestionAnswering.from_pretrained("deepset/roberta-base-squad2")
qa_pipeline = pipeline('question-answering', model=model_qa, tokenizer=tokenizer_qa)
embedding_model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
def translate_text(text, src_lang='ru', dest_lang='en'):
    try:
        translation = translator.translate(text, src=src_lang, dest=dest_lang)
        return translation.text
    except Exception as e:
        logger.error(f"Ошибка при переводе текста: {e}")
        return text
def generate_questions_and_answers(text):
    logger.debug(f"Начало генерации вопросов и ответов для текста: {text[:100]}...")
    try:
        questions_and_answers = []
        nltk.download('punkt', quiet=True)
        from langdetect import detect
        language = detect(text)
        text_en = translate_text(text, src_lang=language, dest_lang='en') if language != 'en' else
text
        sentences_en = nltk.sent_tokenize(text_en, language='english')
        for sentence_en in sentences_en:
            input_text = f"answer: {sentence_en} context: {text_en}"
            inputs = tokenizer_qg.encode(input_text, return_tensors='pt', max_length=512,
truncation=True)
            outputs = model_qg.generate(inputs, max_length=64, num_beams=5,
early_stopping=True)
            question_en = tokenizer_qg.decode(outputs[0],
skip_special_tokens=True).replace("question:", "").strip()
            qa_input = {'question': question_en, 'context': text_en}
            answer_en = qa_pipeline(qa_input)['answer']
            question = translate_text(question_en, src_lang='en', dest_lang=language)
            answer = translate_text(answer_en, src_lang='en', dest_lang=language)
            questions_and_answers.append({'question': question.strip(), 'answer':
answer.strip()})
        questions_and_answers = cluster_and_select_questions(questions_and_answers,
eps=0.5, min_samples=1)
        for qa in questions_and_answers:
            Qa['question'] = translate_text(qa['question'], src_lang='en', dest_lang='ru')
            qa['answer'] = translate_text(qa['answer'], src_lang='en', dest_lang='ru')
            logger.debug(f"Всего сгенерировано вопросов и ответов:
{len(questions_and_answers)}")
        return questions_and_answers
    except Exception as e:
```

```
error_message = traceback.format_exc()
```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ А

```
        logger.error(f"Ошибка при генерации вопросов и ответов: {error_message}")
        raise e
def cluster_and_select_questions(questions_and_answers, eps=0.5, min_samples=1):
    questions = [qa['question'] for qa in questions_and_answers]
    embeddings = embedding_model.encode(questions)
    clustering = DBSCAN(eps=eps, min_samples=min_samples,
metric='cosine').fit(embeddings)
    labels = clustering.labels_
    unique_qas = []
    seen = set()
    for idx, label in enumerate(labels):
        if label not in seen:
            seen.add(label)
            unique_qas.append(questions_and_answers[idx])
    return unique_qas
```


ПРИЛОЖЕНИЕ Б

Листинг Б.1 – Обработка мультимедийного и текстового контента.

```
def get_directml_device():
    return torch.device("cuda" if torch.cuda.is_available() else "cpu")

def get_whisper_model():
    global WHISPER_MODEL
    if WHISPER_MODEL is None:
        device = torch.device("cpu")
        WHISPER_MODEL = whisper.load_model("small").to(device)
        logger.info(f"Whisper модель загружена на устройство: {device}")
    return WHISPER_MODEL

def transcribe_single_chunk(chunk_path):
    model = get_whisper_model()
    result = model.transcribe(chunk_path, language=None)
    return result.get('text', "").strip()

def transcribe_audio(file_path):
    chunks = split_audio(file_path)
    full_transcription = ""
    for chunk in chunks:
        try:
            transcription = transcribe_single_chunk(chunk)
            full_transcription += transcription + " "
        except Exception as e:
            logger.error(f"Ошибка при транскрипции части {chunk}: {e}", exc_info=True)
        finally:
            if os.path.exists(chunk):
                os.remove(chunk)
    return full_transcription

def extract_audio(video_path):
    with tempfile.NamedTemporaryFile(suffix='.mp3', delete=False) as temp_audio:
        temp_audio_name = temp_audio.name
        command = ['ffmpeg', '-i', video_path, '-q:a', '0', '-map', 'a', temp_audio_name, '-y']
        subprocess.run(command, stdout=subprocess.DEVNULL, stderr=subprocess.DEVNULL)
    return temp_audio_name

def get_translator():
    global TRANSLATOR_MODEL, TRANSLATOR_TOKENIZER
    if TRANSLATOR_MODEL is None or TRANSLATOR_TOKENIZER is None:
        model_name = 'Helsinki-NLP/opus-mt-ru-en'
        TRANSLATOR_TOKENIZER = MarianTokenizer.from_pretrained(model_name)
        TRANSLATOR_MODEL = MarianMTModel.from_pretrained(model_name)
        device = get_directml_device()
        TRANSLATOR_MODEL.to(device)
        logger.info(f"Мариянская модель перевода загружена на устройство: {device}")
    return TRANSLATOR_MODEL, TRANSLATOR_TOKENIZER

def translate_ru_to_en(text):
    model, tokenizer = get_translator()
    download('punkt', quiet=True)
    nltk.data.path.append("/home/vladdarkstalker/nltk_data")
```

```

sentence_tokenizer = PunktSentenceTokenizer()
sentences = sentence_tokenizer.tokenize(text)
translated_sentences = []
for sentence in sentences:
    inputs = tokenizer(sentence, return_tensors="pt", padding=True, truncation=True,
max_length=512)
    device = next(model.parameters()).device
    inputs = {k: v.to(device) for k, v in inputs.items()}
    translated = model.generate(inputs)
    translated_sentence = tokenizer.decode(translated[0], skip_special_tokens=True)
    translated_sentences.append(translated_sentence)
return ''.join(translated_sentences)
def split_audio(file_path, chunk_length_ms=60000):
    audio = AudioSegment.from_file(file_path)
    chunks = []
    for i in range(0, len(audio), chunk_length_ms):
        chunk = audio[i:i + chunk_length_ms]
        chunk_filename = f"{file_path}_chunk_{i // chunk_length_ms}.mp3"
        chunk.export(chunk_filename, format="mp3")
        chunks.append(chunk_filename)
    return chunks
def process_content(content_instance):
    content_type = content_instance.content_type
    if content_type in ['audio', 'video']:
        file_path = content_instance.content.path
        try:
            transcribed_text = transcribe_audio(file_path)
            logger.debug(f"Распознанный текст: {transcribed_text[:100]}...")
            from langdetect import detect
            detected_language = detect(transcribed_text)
            logger.debug(f"Определён язык аудио: {detected_language}")
            translated_text = (
                translate_ru_to_en(transcribed_text) if detected_language == 'ru' else
transcribed_text
            )
            content_instance.generated_text = translated_text
            content_instance.save()
            logger.debug(f"Контент ID {content_instance.id} успешно обработан.")
            return True, "Контент успешно обработан."
        except Exception as e:
            logger.error(f"Ошибка при обработке контента ID {content_instance.id}: {e}",
exc_info=True)
            return False, f"Ошибка при обработке файла: {e}"
    elif content_type == 'text':
        try:
            from langdetect import detect
            language = detect(content_instance.text_content)
            logger.debug(f"Определён язык текста: {language}")
            translated_text = (
                translate_ru_to_en(content_instance.text_content) if language == 'ru' else
content_instance.text_content
            )

```


ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Б

```
content_instance.generated_text = translated_text
content_instance.save()
    logger.debug(f"Текстовый контент ID {content_instance.id} успешно
обработан.")
    return True, "Текстовый контент успешно обработан."
except Exception as e:
    logger.error(f"Ошибка при обработке контента ID {content_instance.id}: {e}",
exc_info=True)
    return False, f"Ошибка при обработке текста: {e}"
else:
    return False, "Тип контента не поддерживается для обработки."
```

ПРИЛОЖЕНИЕ В

Листинг В.1 – Представления для генерации тестов.

```
class GenerateQuestionsView(LoginRequiredMixin, UserPassesTestMixin, View):
    def post(self, request, *args, kwargs):
        test = get_object_or_404(Test, pk=self.kwargs['pk'])
        topic = test.topic
        contents = topic.contents.filter(generated_text__isnull=False)
        added = 0
        for content in contents:
            try:
                questions = generate_questions_from_text(content.generated_text,
content_id=content.id)
                for q in questions:
                    if not q.get('question') or not (q.get('answer') or q.get('answers')):
                        continue
                    q_type = q.get('type', 'text')
                    item = TestItem.objects.create(
                        test=test,
                        content=q['question'],
                        question_type=q_type,
                        correct_text_answer=q.get('answer', '') if q_type == 'text' else '',
                        source_content_id=content.id
                    )
                    if q_type in ['single_choice', 'multiple_choice']:
                        corrects = [q['answer']] if q_type == 'single_choice' else q.get('answers', [])
                        for c in corrects:
                            TestItemOption.objects.create(item=item, content=c, is_correct=True)
                        for d in q.get('distractors', []):
                            TestItemOption.objects.create(item=item, content=d, is_correct=False)
                        added += 1
            except Exception as e:
                logger.error(f"Ошибка генерации вопросов: {e}")
        if added:
            messages.success(request, f"Добавлено {added} новых вопрос(ов) к тесту.")
        else:
            messages.warning(request, "Вопросы не были добавлены.")
        return redirect('learnsys:test_detail', pk=test.id)
    def test_func(self):
        test = get_object_or_404(Test, pk=self.kwargs['pk'])
        return is_instructor(self.request.user) and test.topic.course.instructor ==
self.request.user
```

ПРИЛОЖЕНИЕ Г

Листинг Г.1 – Проведение тестирования и адаптивное тестирование.

```
class AdaptiveTestView(LoginRequiredMixin, View):
    template_name = 'tests/adaptive_test.html'
    MAX_QUESTIONS = 10
    COMPLEXITY_ORDER = ['easy', 'medium', 'hard']
    def get(self, request, *args, kwargs):
        test_id = kwargs.get('test_id')
        test = get_object_or_404(Test, id=test_id)
        user = request.user
        if is_instructor(user) and test.topic.course.instructor != user:
            raise PermissionDenied("У вас нет доступа к этому тесту.")
        if not (is_instructor(user) or is_student(user)):
            messages.error(request, "Нет доступа.")
            return redirect('learnsys:home')
        session, created = AdaptiveTestSession.objects.get_or_create(user=user, test=test)
        if session.finished:
            messages.info(request, "Тест уже завершён.")
            result = TestResult.objects.filter(user=user, test=test).order_by('-date_taken').first()
            if result:
                return redirect('learnsys:test_result_detail', test_result_id=result.id)
            return redirect('learnsys:home')
        if session.asked_items.count() >= self.MAX_QUESTIONS:
            self.finish_test(session)
            return self.redirect_to_results(user, test)
        next_item = self.select_next_item(test, session)
        if not next_item:
            self.finish_test(session)
            return self.redirect_to_results(user, test)
        form = TestAnswerForm(test_items=[next_item])
        return render(request, self.template_name, {
            'form': form,
            'question': next_item,
            'test': test,
            'current_complexity': session.current_complexity
        })
    def post(self, request, *args, kwargs):
        test_id = kwargs.get('test_id')
        test = get_object_or_404(Test, id=test_id)
        user = request.user
        session = get_object_or_404(AdaptiveTestSession, user=user, test=test, finished=False)
        item_id = request.POST.get('item_id')
        item = get_object_or_404(TestItem, id=item_id, test=test)
        form = TestAnswerForm(request.POST, test_items=[item])
        if form.is_valid():
            correct = self.process_answer(user, item, form)
            session.theta = self.estimate_theta(session)
            session.current_complexity = self.update_complexity(session.current_complexity,
correct)
            session.asked_items.add(item)
```


ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Г

```
session.save()
return redirect('learnsys:adaptive_test', test_id=test_id)
else:
    return render(request, self.template_name, {
        'form': form,
        'question': item,
        'test': test,
        'current_complexity': session.current_complexity
    })
def process_answer(self, user, item, form):
    user_answer = form.cleaned_data.get(f"item_{item.id}")
    uta = UserTestAnswer.objects.create(
        user=user,
        item=item
    )
    if item.question_type == 'single_choice':
        selected_options = [int(user_answer)]
        for oid in selected_options:
            uta.option.add(item.options.get(id=oid))
        correct_options = set(item.options.filter(is_correct=True))
        selected_set = set(uta.option.all())
        return selected_set == correct_options
    elif item.question_type == 'multiple_choice':
        selected_options = [int(i) for i in user_answer]
        for oid in selected_options:
            uta.option.add(item.options.get(id=oid))
        correct_options = set(item.options.filter(is_correct=True))
        selected_set = set(uta.option.all())
        return selected_set == correct_options
    elif item.question_type == 'text':
        user_text = user_answer
        uta.text_answer = user_text
        uta.save()
        return self.check_text_answer(user_text, item.correct_text_answer)
    return False
def check_text_answer(self, user_answer, correct_answer):
    normalized_user = ''.join(user_answer.lower().split())
    normalized_correct = ''.join(correct_answer.lower().split())
    if normalized_user == normalized_correct:
        return True
    from difflib import SequenceMatcher
    similarity = SequenceMatcher(None, normalized_user, normalized_correct).ratio()
    return similarity > 0.8
def select_next_item(self, test, session):
    theta = session.theta
    answered_item_ids = session.asked_items.values_list('id', flat=True)
    available_items = test.items.exclude(id__in=answered_item_ids)
    if not available_items.exists():
        return None
    min_diff = float('inf')
```



```

next_item = None
for item in available_items:
    diff = abs(theta - item.b)
    if diff < min_diff:
        min_diff = diff
        next_item = item
return next_item

def update_complexity(self, current_complexity, correct):
    idx = self.COMPLEXITY_ORDER.index(current_complexity)
    if correct and idx < len(self.COMPLEXITY_ORDER)-1:
        idx += 1
    elif not correct and idx > 0:
        idx -= 1
    return self.COMPLEXITY_ORDER[idx]

def estimate_theta(self, session, max_iter=10, step=0.1):
    user = session.user
    test = session.test
    answered_items = session.asked_items.all()
    answers = UserTestAnswer.objects.filter(user=user, item__in=answered_items)
    results = []
    for ans in answers:
        if ans.item.question_type in ['single_choice', 'multiple_choice']:
            correct_options = set(ans.item.options.filter(is_correct=True))
            selected = set(ans.option.all())
            correct = (correct_options == selected)
        elif ans.item.question_type == 'text':
            correct = self.check_text_answer(ans.text_answer, ans.item.correct_text_answer)
        else:
            correct = False
        results.append((ans.item, correct))
    theta = session.theta
    for _ in range(max_iter):
        grad = 0.0
        for (item, correct) in results:
            p = item.probability_of_correct_answer(theta)
            grad += (1.0 if correct else 0.0) - p
        theta = theta + step*grad
    return theta

def finish_test(self, session):
    session.finished = True
    session.save()
    user = session.user
    test = session.test
    asked_items = session.asked_items.all()
    answers = UserTestAnswer.objects.filter(user=user, item__in=asked_items)
    correct_count = 0
    total = asked_items.count()
    for ans in answers:
        if ans.item.question_type in ['single_choice', 'multiple_choice']:
            correct_options = set(ans.item.options.filter(is_correct=True))
            selected = set(ans.option.all())

```

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЯ Г

```
        if correct_options == selected:
            correct_count += 1
        elif ans.item.question_type == 'text':
            if self.check_text_answer(ans.text_answer, ans.item.correct_text_answer):
                correct_count += 1
    TResult.objects.create(
        user=user,
        test=test,
        score=correct_count,
        total_questions=total,
        date_taken=timezone.now(),
        theta=session.theta
    )
    topic_progress, _ = TopicProgress.objects.get_or_create(user=user, topic=test.topic)
    topic_progress.mark_test_completed(correct_count, total)
    count = TResult.objects.filter(test=test).count()
    if count % 10 == 0:
        from .views import calibrate_item_parameters
        calibrate_item_parameters(test, max_iter=20, tol=1e-3)
def redirect_to_results(self, user, test):
    result = TResult.objects.filter(user=user, test=test).order_by('-date_taken').first()
    if result:
        return redirect('learnsys:test_result_detail', test_result_id=result.id)
    return redirect('learnsys:home')
```

ПРИЛОЖЕНИЕ Д

Листинг Д.1 – Анализ результатов и адаптация.

```
def finish_test(self, session):
    session.finished = True
    session.save()
    user = session.user
    test = session.test
    asked_items = session.asked_items.all()
    answers = UserTestAnswer.objects.filter(user=user, item__in=asked_items)
    correct_count = 0
    for ans in answers:
        if ans.item.question_type in ['single_choice', 'multiple_choice']:
            correct_options = set(ans.item.options.filter(is_correct=True))
            selected = set(ans.option.all())
            if correct_options == selected:
                correct_count += 1
        elif ans.item.question_type == 'text':
            if self.check_text_answer(ans.text_answer, ans.item.correct_text_answer):
                correct_count += 1

    TestResult.objects.create(
        user=user,
        test=test,
        score=correct_count,
        total_questions=asked_items.count(),
        date_taken=timezone.now(),
        theta=session.theta
    )

    topic_progress, _ = TopicProgress.objects.get_or_create(user=user, topic=test.topic)
    topic_progress.mark_test_completed(correct_count, asked_items.count())
```

ПРИЛОЖЕНИЕ Е

Листинг Е.1 – Кластеризация вопросов с использованием Sentence-BERT и DBSCAN.

```
embedding_model = SentenceTransformer('paraphrase-MiniLM-L6-v2')
def cluster_and_select_questions(questions_and_answers, eps=0.5, min_samples=1):
    questions = [qa['question'] for qa in questions_and_answers]
    embeddings = embedding_model.encode(questions)
    clustering = DBSCAN(eps=eps, min_samples=min_samples,
metric='cosine').fit(embeddings)
    labels = clustering.labels_
    unique_qas = []
    seen = set()
    for idx, label in enumerate(labels):
        if label not in seen:
            seen.add(label)
            unique_qas.append(questions_and_answers[idx])
    return unique_qas
```

ПРИЛОЖЕНИЕ Ё

Листинг Ё.1 – Интерпретация результатов тестирования и формирование рекомендаций.

```
def interpret_educational_guidance(guidance_results):
    recommendations = {}
    for code, value in guidance_results.items():
        base = code[:-1]
        rtype = code[-1]
        if base not in recommendations:
            recommendations[base] = {"T": "", "R": "", "C": ""}
        if rtype == "T":
            recommendations[base]["T"] = get_teacher_support(value)
        elif rtype == "R":
            recommendations[base]["R"] = get_feedback(value)
        elif rtype == "C":
            recommendations[base]["C"] = get_community(value)
    return recommendations
```