ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.512

АНАЛИЗ SEQ2SEQ МОДЕЛЕЙ И МЕТОДА ВЫДЕЛЕНИЯ КЛЮЧЕВЫХ СЛОВ ДЛЯ ГЕНЕРАЦИИ ВОПРОСОВ ПО ВХОДНОМУ ТЕКСТУ

Иванов Никита Владимирович,
Студент кафедры ИУК5 «Системы обработки информации»
Московский государственный технический университет имени Н.Э. Баумана
destira@mail.ru

Косова Ксения Алексеевна,
Студентка кафедры ИУК5 «Системы обработки информации»
Московский государственный технический университет имени Н.Э. Баумана
ksenya.kosova.04@mail.ru

Вершинин Евгений Владимирович,
К.ф.-м.н., доцент, заведующий кафедрой ИУК5 «Системы обработки информации»
Московский государственный технический университет имени Н.Э. Баумана
vershinin@bmstu.ru

Ильичев Владимир Юрьевич,
к.т.н. доцент кафедры ИУК5 «Системы обработки информации»
Московский государственный технический университет имени Н.Э. Баумана
ilychev.vyu@bmstu.ru

Аннотация

Статья посвящена описанию процесса и результатов сравнительного анализа двух подходов к автоматической генерации вопросов по исходному тексу, а именно: Seq2Seq моделей и методов выделения ключевых слов. Сформулирована цель исследования, приведены краткие сведения об используемых подходах. Подобраны программные средства языка Python и библиотеки, позволяющие реализовать каждый из подходов. Разработаны программные реализации указанных подходов. Проведено экспериментальное исследование свойств генерации по лингвистическим и числовым метрикам. Выполнен анализ полученных результатов. Сделан вывод по использованию каждого из подходов.

Ключевые слова: генерация, Seq2Seq модели, выделение ключевых слов, сравнительный анализ, вопросы

ANALYSIS OF SEQ2SEQ MODELS AND A METHOD FOR HIGHLIGHTING KEYWORDS TO GENERATE QUESTIONS ON THE INPUT TEXT

Ivanov Nikita Vladimirovich,
Student of the Department of IUK5 «Information Processing Systems»
Bauman Moscow State Technical University
destira@mail.ru

Kosova Ksenia Alekseevna,
Student of the Department of IUK5 «Information Processing Systems»
Bauman Moscow State Technical University
ksenya.kosova.04@mail.ru

Vershinin Evgeny Vladimirovich,
Ph.D., Associate Professor, Head of the Department of IUK5 «Information Processing Systems»
Bauman Moscow State Technical University
vershinin@bmstu.ru

Ilyichev Vladimir Yuryevich,
Ph.D. Associate Professor, Department of IUK5 «Information Processing Systems»
Bauman Moscow State Technical University
ilychev.vyu@bmstu.ru

ABSTRACT

The article describes the process and results of a comparative analysis of two approaches to the automatic generation of questions on the source text, namely: Seq2Seq models and methods for highlighting keywords. The purpose of the study is formulated, brief information on the approaches used is given. Python software and libraries have been selected to implement each of the approaches. Software implementations of these approaches have been developed. Experimental study of generation properties by linguistic and numerical metrics was carried out. The results were analyzed. A conclusion was made on the use of each of the approaches.

Keywords: generation, model Seq2Seq, selection of keywords, comparative analysis, questions.

Введение

В современном мире разработки приложений неотъемлемой частью практически каждого серьезного онлайн проекта стали внедрения ИИ в том или ином виде. Одной из тривиальных задач этой сферы является генерация вопросов по заданному тексту. Для реализации используют различные подходы, среди которых можно выделить метод выделения ключевых слов и Seq2Seq модели. [1, 2]

Различные отрасли предъявляют к такой генерации большое количество всевозможных требований, поэтому разработчики постоянно сталкиваются с проблемой выбора того или иного подхода. В данной работе будет сделан упор на их сравнение.

Метод выделения ключевых слов — это лингвистический подход, который обеспечивает прямое извлечение именованных сущностей из текста с последующим преобразованием их в вопросы по заранее определенным шаблонам, что позволяет генерировать релевантные вопросы без обучения на больших данных и с минимальными вычислительными затратами.

Метод Seq2Seq (Sequence-to-Sequence) — это нейросетевой подход, который обеспечивает сквозное преобразование исходного текста в вопросы через архитектуру энкодер-декодер, что позволяет генерировать разнообразные и грамматически правильные вопросы на основе изученных языковых паттернов, но требует значительных вычислительных ресурсов и размеченных данных для обучения.

Каждое из двух решений имеет как свои плюсы, так и минусы, которые всегда зависят от конкретной реализации. В данной работе будут рассмотрены и сравнены наиболее общие реализации генераторов вопросов с использованием метода выделения ключевых слов и Seq2Seq моделей.

Цель исследования

Целью работы является сравнительный анализ двух подходов к генерации вопросов по исходному тексту.

Задачами работы являются реализация двух вариантов генераторов вопросов и сравнение результатов генерации, с целью выявления основных закономерностей.

Материалы и методы исследования

В качестве программных средств разработки были выбраны язык программирования Python и соответствующие ему библиотеки, предназначенные для генерации вопросов по входному тексту.

Реализация генератора, использующего метод выделения ключевых слов.

На первом этапе создания приложения была загружена предобученная модель анализа русского языка SpaCy [3]. Были заданы типы именованных сущностей, которые должны извлекаться из исходного текста. Использовались 3 вида сущностей PER, LOC, ORG: персона, локация и организация соответственно. Также с помощью библиотеки dataclass созданы структуры хранения результатов [4].

На втором этапе были созданы шаблоны будущих вопросов для каждой сущности. Например, вопросы для сущности PER:

'PER': ["Кто такой {entity}?",

"Что сделал {entity}?",

"Чем известен {entity}?",]

Далее был реализован алгоритм генерации вопросов по заданному тексту. Основной метод generate_qa_pairs.

Сигнатура метода: generate_qa_pairs(self, text: str, text_id: int, title: str) -> GenerationResult

Данный метод специализируется на автоматической генерации вопросов по входному тексту. Процесс начинается с точного замера времени выполнения для оценки производительности системы. На первом этапе метод использует модель spaCy для извлечения именованных сущностей, приведенных ранее, из входного текста.

Для каждой обнаруженной сущности система применяет интеллектуальный подбор вопросов на основе предопределенных шаблонов, сгруппированных по типам сущностей. Интеллектуальный отбор заключается в автоматической адаптации к содержанию текста, отбору наиболее релевантных сущностей и преобразованию их в грамматически корректные вопросы. Каждый сгенерированный вопрос сопровождается метаданными, включающими тип исходной сущности и показателем уверенности системы, в том на сколько точно была отобрана сущность.

Итоговый результат представляет собой структурированную коллекцию вопросов, в которых были использованы сущности с наибольшей уверенностью системы.

В качестве инструментов для сбора и обработки числовых метрик были выбраны библиотеки: NumPy и Time [5]. Time – встроенная библиотека Python, используется в программе для измерения временных метрик. NumPy была выбрана, так как она

предоставляет удобный функционал для эффективного подсчета медианных и средних значений вычисляемых метрик.

В качестве исследуемых метрик были выбраны длина вопроса, измеряемая в словах, его среднее и медианное значение, а также число вопросов, генерируемое в секунду, среднее время, затрачиваемое на текст: время от получения текста до генерации вопросов, и среднее число вопросов, сгенерированное с помощью одного текста.

Рассмотрим реализацию генератора, использующего Seq2Seq модель.

Для реализации данного метода была выбрана библиотека для глубоко обучения PyTorch и предобученная модель для обработки естественного языка Transformers. [6] Для дообучения такого вида модели необходим большой объем данных. В качестве датасета использовался SberQuad - русскоязычный датасет для обучения и оценки моделей вопросно-ответных систем [7].

Работа с данным видом генерации происходила в несколько этапов, первый из которых формирование данных для обучения. Были созданы пары вида “контекст - вопрос”. На втором этапе тексты были токенизированы. Далее основа системы – модель ruT5-base была дообучена на сформированных данных с использованием алгоритма оптимизации AdamW [8, 9]. Обучение происходило за один цикл.

Метод generate_questions служит основным для генерации вопросов из текстовых данных.

Сигнатура метода: generate_questions(self, texts, num_questions=2).

Он принимает список текстов произвольной длины и опциональный параметр, определяющий целевое количество вопросов для каждого текста.

Процесс обработки организован как последовательный обход всех входных текстов. Для каждого текста запускается таймер производительности, после чего осуществляется попытка генерации заданного количества вопросов. Генерация выполняется через внутренний вспомогательный метод, который может создавать варианты вопросов с различными параметрами разнообразия.

Ключевой особенностью метода является система валидации генерируемых вопросов. Каждый созданный вопрос проверяется на соответствие нескольким критериям: непустота содержимого, уникальность в рамках текущего набора и минимальная длина. Это позволяет отфильтровывать некорректные или дублирующиеся результаты.

Метод реализует устойчивую к ошибкам архитектуру. При возникновении исключений в процессе обработки отдельного текста, система не прерывает общий поток выполнения, а переходит к следующему тексту, сохраняя детали ошибки для анализа.

В контексте управления ресурсами метод включает механизмы работы с GPU-памятью. При обнаружении ошибок, связанных с нехваткой памяти, выполняется очистка кэша, что предотвращает накопление проблем и обеспечивает стабильность длительных сеансов обработки.

Результирующая структура данных содержит комплексную информацию по каждому тексту: исходное содержание, набор валидных вопросов, метрики времени выполнения и служебную информацию об ошибках.

Инструменты для сбора и обработки числовых метрик, библиотеки для отображения метрик и сами метрики были выбраны те же, что и в первом случае.

Тестирование происходило на 50 текстах длинной примерно в 50 слов.

Пример входного текста: «Python является интерпретируемым языком программирования высокого уровня с динамической типизацией. Он поддерживает несколько парадигм программирования, включая объектно-ориентированное, императивное и функциональное программирование. Язык обладает чистым и понятным синтаксисом, что делает его отличным выбором для начинающих программистов. Python

широко используется в веб-разработке, научных исследованиях, анализе данных и искусственном интеллекте. Стандартная библиотека Python включает модули для работы с операционной системой, сетевых взаимодействий и обработки различных форматов данных.»

Пример сгенерированных вопросов.

Метод ключевых слов: «Что такое Python?».

Seq2Seq: «Как называется интерпретируемый язык программирования высокого уровня с динамической типизацией?».

Результаты и их обсуждение

В ходе экспериментального исследования были получены количественные и качественные данные по производительности и качеству генерации вопросов двумя методами: на основе выделения ключевых слов (Keyword-based) и с использованием архитектуры Seq2Seq. Ниже представлены результаты, проиллюстрированные соответствующими рисунками, и их интерпретация.

Рисунок 1 демонстрирует временные метрики, полученные при тестировании метода выделения ключевых слов. В среднем на обработку одного текста методу требовалось 9,4 миллисекунды. Такое низкое время обработки обусловлено простотой алгоритма: он выполняет токенизацию, фильтрацию по частеречным меткам и сопоставление с предварительно определёнными шаблонами вопросов. Отсутствие сложных вычислений или обучения моделей делает метод крайне эффективным с точки зрения вычислительных затрат.

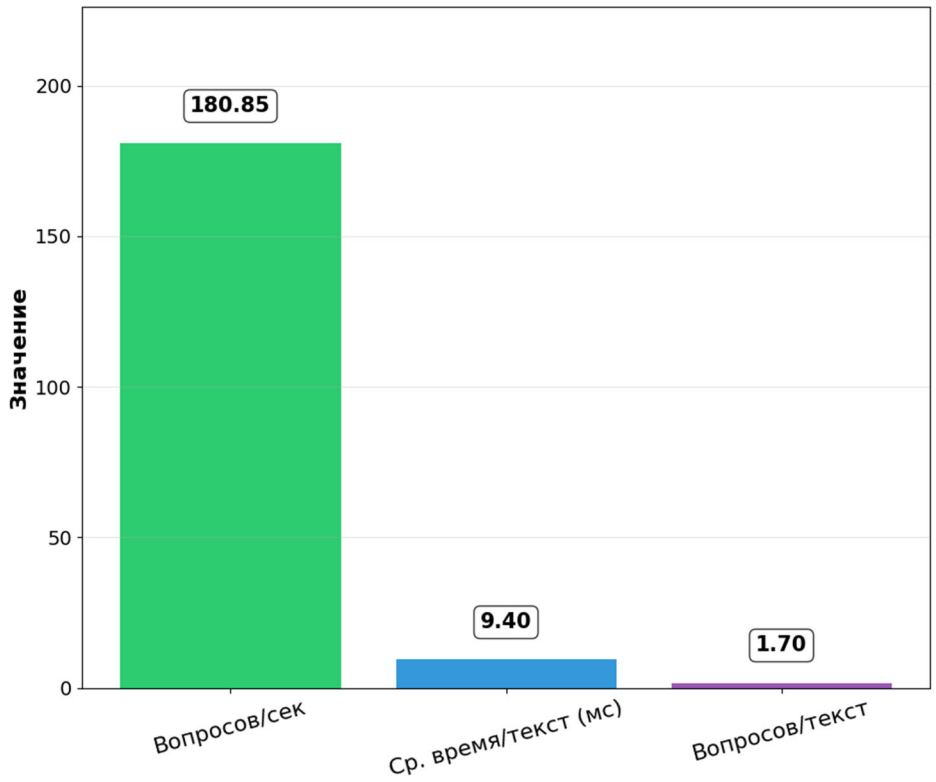

Рисунок 1. Временные метрики, полученные при тестировании метода ключевых слов

Рисунок 2 представляет временные метрики для Seq2Seq-модели, обученной на датасете вопросов-ответов. В среднем на генерацию одного вопроса по тексту модель затрачивала 950,6 миллисекунд, что во много раз больше, чем у метода ключевых слов. Это значительное увеличение времени связано с необходимостью выполнения полного прохода через нейронную сеть, включающего энкодинг текста, декодинг с вниманием (attention

mechanism) и генерацию последовательности токенов. Дополнительные накладные расходы обусловлены загрузкой модели в память, предварительной обработкой и постобработкой выходных данных. Такой уровень вычислительной нагрузки делает Seq2Seq менее пригодным для систем с жёсткими ограничениями по времени отклика, например, мобильных приложений или реальных систем поддержки.

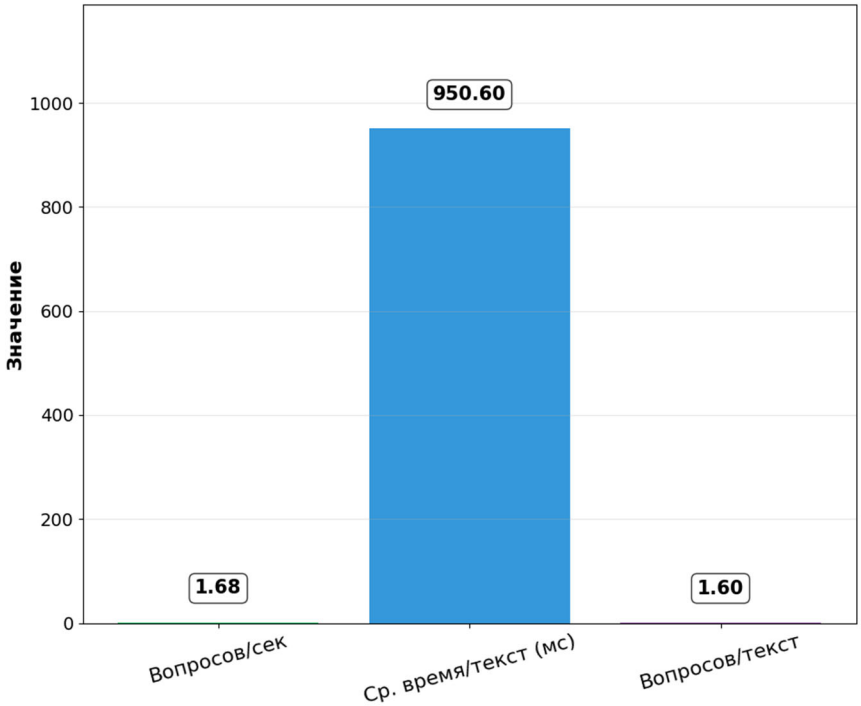

Рисунок 2. Временные метрики, полученные при тестировании Seq2Seq

Рисунок 3 показывает распределение средней длины вопросов, сгенерированных методом ключевых слов. Средняя длина составила 3,4 слова (медиана — 3 слова), что соответствует типичным шаблонным конструкциям вроде «Что такое…?», «Каковы причины…?». Вариативность ограничена фиксированными шаблонами, что приводит к высокой повторяемости формулировок даже на разных текстах. Несмотря на простоту, вопросы остаются грамматически корректными, но их семантическое разнообразие крайне ограничено.

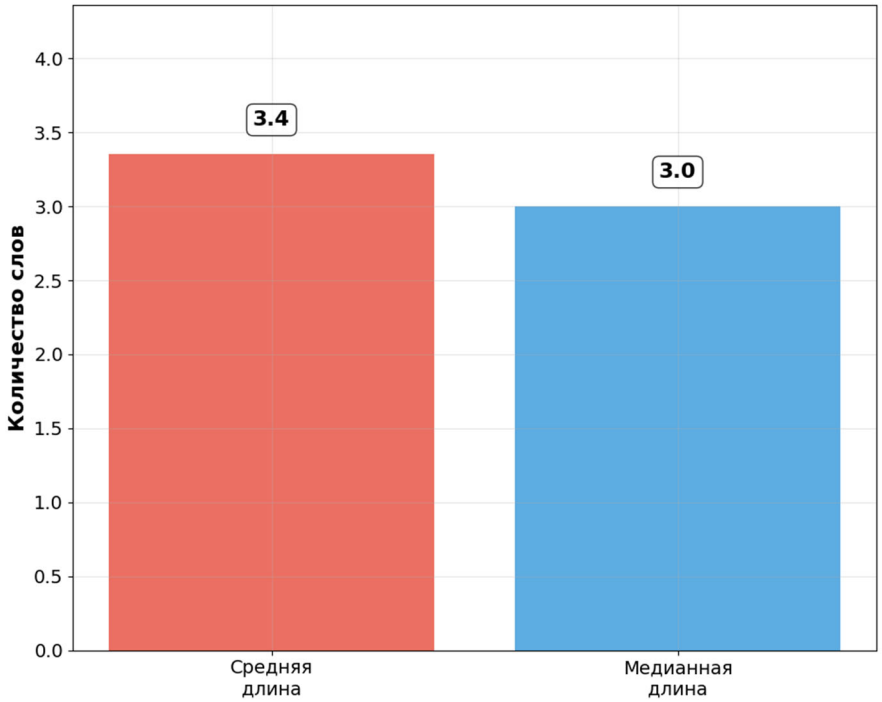

Рисунок 3. Длина вопроса, полученная при тестировании метода ключевых слов

Рисунок 4 отражает распределение длины вопросов, сгенерированных Seq2Seq-моделью. Средняя длина составила 6,5 слов (медиана — 6 слов), что примерно в 2 раза больше, чем у метода ключевых слов. Более длинные вопросы отражают способность модели генерировать сложные, контекстно-зависимые формулировки, такие как: «Как влияет изменение температуры на скорость химической реакции, если учитывать закон Аррениуса?». Такие вопросы не могут быть сгенерированы шаблонным подходом, так как требуют понимания семантики текста и логических связей между понятиями.

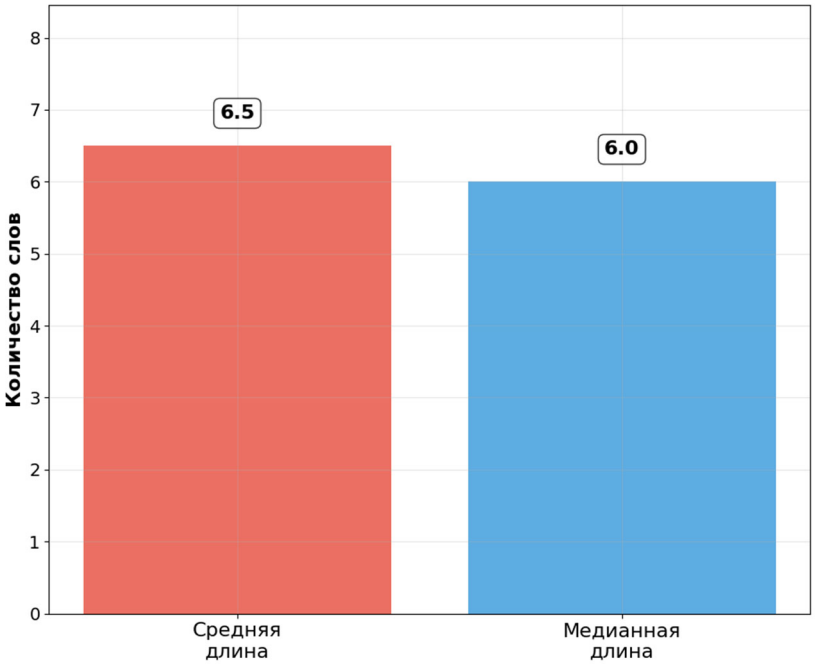

Рисунок 4. Длина вопроса, полученная при тестировании Seq2Seq

Сравнение результатов. Временные метрики показывают, что Seq2Seq требует гораздо больше вычислительных ресурсов, чем метод выделения ключевых слов. При этом в среднем генерируется одинаковое количество вопросов на один текст. Результаты

измерения средней длины вопросов показывают, что из-за своей специфики Seq2Seq предлагает большее разнообразие вопросов, чем метод ключевых слов. Субъективный анализ лингвистических особенностей сгенерированных вопросов показал, что Seq2Seq чаще генерирует осмысленные вопросы.

Выводы

Таким образом, можно сделать вывод, что цель данного исследования, заключающаяся в сравнении подходов к генерации вопросов по исходному тексту, является полностью достигнутой. Реализованы два варианта генераторов вопросов. Проведено тестирование, результатом которого являются графики зависимости метрик от конкретного метода. По результатам сравнения результатов, можно сказать, что более предпочтительным подходом с точки зрения лингвистического разнообразия является Seq2Seq, но при отсутствии больших серверных мощностей и низком требовании к разнообразию вопросов метод выделения ключевых слов также является приемлемым подходом.

Список литературы:

1. Milvus.io. What is a sequence-to-sequence model? [Электронный ресурс]. URL: https://milvus.io/ai-quick-reference/what-is-a-sequencetosequence-model (дата обращения: 13.11.2025).
2. GeeksforGeeks. Keyword Extraction Methods in NLP [Электронный ресурс]. URL: https://www.geeksforgeeks.org/nlp/keyword-extraction-methods-in-nlp/ (дата обращения: 13.11.2025).
3. Ильичев В.Ю. Автоматизированный синтаксический разбор русского текста с помощью средств языка Python. // В сборнике: Мировые стратегии развития науки и образования в новой реальности: междисциплинарные исследования. Материалы I Международной научно-практической конференции. Москва, 2024. С. 32-34.
4. Python Standard Library. Dataclasses — Data Classes. [Электронный ресурс]. URL: https://docs.python.org/3/library/dataclasses.html (дата обращения: 13.11.2025).
5. Ильичев В.Ю. Использование алгоритма дифференциальной эволюции для решения оптимизационных задач. // Системный администратор. 2021. № 4 (221). С. 80-83.
6. Ultralytics. Pytorch: The Deep Learning Framework [Электронный ресурс]. URL: https://www.ultralytics.com/glossary/pytorch (дата обращения: 13.11.2025).
7. Kuznetsov Andrey et al. Sberquad – Russian Reading Comprehension Dataset. [Электронный ресурс] URL: https://umc.urfu.ru/fileadmin/user_upload/site-25894/sberquad_CLEF.pdf (дата обращения: 13.11.2025).
8. AI-Forever. Rut5-base model. [Электронный ресурс]. URL: https://huggingface.co/ai-forever/rut5-base (дата обращения: 13.11.2025).
9. Pytorch Documentation. Adamw — pytorch 2.7 documentation. [Электронный ресурс]. URL: https://pytorch.org/docs/stable/generated/torch.optim.adamw.html (дата обращения: 13.11.2025).

References:

1. Milvus.io. What is a sequence-to-sequence model? [Electronic resource]. URL: https://milvus.io/ai-quick-reference/what-is-a-sequencetosequence-model (accessed: 13.11.2025).

2. GeeksforGeeks. Keyword Extraction Methods in NLP [Electronic resource]. URL: https://www.geeksforgeeks.org/nlp/keyword-extraction-methods-in-nlp/ (accessed: 13.11.2025).
3. Ilyichev V. Yu. Automated syntactic parsing of Russian text using Python. // In the collection: World Strategies for the Development of Science and Education in the New Reality: Interdisciplinary Research. Proceedings of the 1st International Scientific and Practical Conference. Moscow, 2024. pp. 32-34.
4. Python Standard Library. Dataclasses — Data Classes. [Electronic resource]. URL: https://docs.python.org/3/library/dataclasses.html (accessed: 13.11.2025).
5. Ilyichev V. Yu. Using the Differential Evolution Algorithm to Solve Optimization Problems. // System Administrator. 2021. No. 4 (221). Pp. 80-83.
6. Ultralytics. Pytorch: The Deep Learning Framework [Electronic resource]. URL: https://www.ultralytics.com/glossary/pytorch (accessed: 13.11.2025).
7. Kuznetsov Andrey et al. Sberquad – Russian Reading Comprehension Dataset. [Electronic resource] URL: https://umc.urfu.ru/fileadmin/user_upload/site-25894/sberquad_CLEF.pdf (date of access: 11/13/2025).
8. AI-Forever. Rut5-base model. [Electronic resource]. URL: https://huggingface.co/ai-forever/rut5-base (date of access: 11/13/2025).
9. Pytorch Documentation. Adamw — pytorch 2.7 documentation. [Electronic resource]. URL: https://pytorch.org/docs/stable/generated/torch.optim.adamw.html (date of access: 11/13/2025).