УДК 004.75

РАЗРАБОТКА И РЕАЛИЗАЦИЯ МЕТОДОВ ОПТИМИЗАЦИИ МАЛЫХ ЯЗЫКОВЫХ МОДЕЛЕЙ ДЛЯ ЭФФЕКТИВНОЙ ГЕНЕРАЦИИ SQL-ЗАПРОСОВ В ОБРАЗОВАТЕЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Босенко Тимур Муртазович[1],
e-mail: bosenkotm@mgpu.ru,
[1]*Московский городской педагогический университет (МГПУ), г. Москва, Россия*

В статье представлены исследования по оптимизации малых языковых моделей для эффективной генерации SQL-запросов в условиях ограниченных вычислительных ресурсов образовательных учреждений. Цель исследования – разработка и реализация методов оптимизации, направленных на снижение требований к вычислительным ресурсам при сохранении высокой точности генерации SQL-запросов. Методология основана на комплексном подходе, включающем квантование параметров модели, оптимизацию весовых коэффициентов и адаптацию словаря под предметную область. Эмпирическая база включает бенчмарки Spider, CoSQL и специализированный набор данных DSBInf (Data Solutions Business Intelligence) из 3424 запросов. Разработанная модель SQL_DS_60M_ft (модель для генерации ответов на основе структурированного языка запросов), основанная на архитектуре CodeS, демонстрирует значительные улучшения: сокращение размера на 33 %, увеличение скорости обработки в 1.4 раза и снижение использования RAM на 18 % при сохранении точности 65.0 %. Практическое внедрение модели в образовательный процесс показало повышение эффективности обучения на 35–40 % при работе с базами данных и бизнес-аналитикой.

Ключевые слова: малые языковые модели, оптимизация, генерация SQL-запросов, Text-to-SQL, квантование, прунинг, образовательные информационные системы

THE DEVELOPMENT AND IMPLEMENTATION OF METHODS OF OPTIMIZATION OF SMALL LANGUAGE MODELS FOR EFFICIENT GENERATION OF SQL QUERY IN EDUCATIONAL INFORMATION SYSTEMS

Bosenko T.M.[1],
e-mail: bosenkotm@mgpu.ru,
[1]*Moscow City University (MCU), Moscow, Russia*

The article presents research on the optimization of small language models for efficient generation of SQL queries in conditions of limited computing resources of educational institutions. The objective of the study is to develop and implement optimization methods aimed at reducing the requirements for computing resources while maintaining high accuracy of SQL query generation. The methodology is based on an integrated approach, including quantization of model parameters, optimization of weight coefficients and adaptation of the dictionary to the subject area. The empirical base includes Spider, CoSQL benchmarks and a specialized DSBInf (Data Solutions Business Intelligence) dataset of 3424 queries. The developed SQL_DS_60M_ft model (a model for generating responses based on a structured query language), based on the CodeS architecture, demonstrates significant improvements: a 33 % reduction in size, a 1.4-fold increase in processing speed and an 18 % decrease in RAM usage while maintaining 65.0 % accuracy. Practical implementation of the model in the educational process showed an increase in learning efficiency by 35–40 % when working with databases and business analytics.

Keywords: small language models, optimization, SQL query generation, Text-to-SQL, quantization, pruning, educational information systems

DOI 10.21777/2500-2112-2025-1-54-67

Введение

Для эффективной генерации SQL-запросов из текста на естественном языке (Text-to-SQL) малые языковые модели (SLM) приобретают все большую популярность, так как они предлагают значительные преимущества в сравнении с большими языковыми моделями (LLM). Ключевыми преимуществами SLM являются их меньшие вычислительные требования и возможность адаптации под специфические задачи. Однако для достижения высокой точности при генерации сложных SQL-запросов необходимо применять методы оптимизации архитектуры таких моделей.

Актуальность исследования обусловлена растущей потребностью образовательных учреждений в эффективных инструментах для обучения работе с базами данных через естественно-языковой интерфейс [1]. Особую значимость приобретает разработка решений, учитывающих ограниченные вычислительные ресурсы учебных лабораторий и специфику образовательного процесса. Последние исследования показывают, что LLM, такие как GPT-4, достигают впечатляющих результатов в задачах Text-to-SQL [2], однако их использование связано со значительными вычислительными затратами, они демонстрируют мощные способности к рассуждениям, но часто ошибаются в сопоставлении таблиц и столбцов. В то же время разработка специализированных SLM, основанных на таких моделях, как BART, T5 и CodeS [3; 4], демонстрирует возможность достижения сопоставимой точности при существенно меньших ресурсных требованиях.

Современные архитектуры трансформеров [5] и методы предварительного обучения [6; 7] создали основу для развития эффективных языковых моделей. Особую роль в этом направлении играют специализированные архитектуры для работы с табличными данными [8] и унифицированные подходы к обучению с использованием трансформеров [9]. Важным аспектом является также применение методов рассуждений по цепочке (chain-of-thought) [10], которые позволяют улучшить качество генерации сложных SQL-запросов.

Хотя методы квантования и прунинга уже применялись для оптимизации языковых моделей, их адаптация специально для задач Text-to-SQL в контексте малых моделей представляет собой новое направление исследований. Специфика генерации SQL-запросов требует учета особенностей структуры запросов, схем баз данных и предметной области. Стандартные датасеты для оценки качества Text-to-SQL моделей, такие как Spider [11], Seq2SQL [12], CoSQL [13] и SParC [14], дают возможность комплексно проанализировать эффективность предлагаемых улучшений. Появление специализированных моделей, например, SQLCoder [15; 16] и WPAIGPT-sql [17], свидетельствует о растущем интересе к созданию оптимизированных решений для задач Text-to-SQL именно на основе малых моделей, адаптированных под конкретные области применения.

В данной работе представлена архитектура малых языковых моделей SQL_DS_100M и SQL_DS_60M_ft, разработанных в Институте цифрового образования Московского городского педагогического университета (МГПУ) на основе CodeS для решения задач Text-to-SQL (рисунок 1).

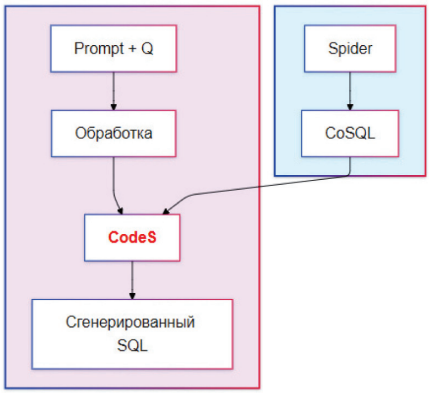

Рисунок 1 – Архитектура малой языковой модели SQL_DS_100M на базе CodeS для решения задач Text-to-SQL[1]

[1] Составлено автором.

SQL_DS_100M – базовая модель для генерации ответов на основе структурированного языка запросов в приложении к задачам бизнес-аналитики и обработке баз данных с размером 100 миллионов параметров.

SQL_DS_60M_ft – оптимизированная модель для генерации ответов на основе структурированного языка запросов в приложении к задачам бизнес-аналитики и обработке баз данных с размером 60 миллионов параметров.

Модели построены на основе стандартных бенчмарков Spider и CoSQL, что обеспечивает базовый уровень производительности для задач генерации SQL-запросов.

Методы и алгоритмы оптимизации малых языковых моделей

Цель данной работы – разработка и реализация методов оптимизации малых языковых моделей для эффективной генерации SQL-запросов, основанных на архитектуре CodeS, в образовательных информационных системах, направленных на снижение требований к вычислительным ресурсам при сохранении высокой точности генерации структурированных запросов в процессе обучения.

Для достижения поставленной цели разработан комплексный подход, включающий следующие методы:

1. *Оптимизация архитектуры модели*:

– Квантование параметров модели [18] с учетом специфики SQL-операторов. Это позволит уменьшить количество бит для представления весов нейросети при сохранении точности критически важных компонентов.

– Специализированный прунинг [18], который сохраняет веса, необходимые для корректной работы с табличными структурами и SQL-синтаксисом.

– Применение компактной версии архитектуры трансформер [19], адаптированной для задач Text-to-SQL.

2. *Оптимизация процесса обучения*:

– Использование предварительно обученной базовой модели SQL_DS_100M на стандартных бенчмарках Spider и CoSQL.

– Дополнительное обучение на специализированном бенчмарке “Data Solutions Business Intelligence” (DSBInf), включающем схемы баз данных и процедуры из предметной области.

– Применение методов активного (с выборочной разметкой данных) обучения [20] для выбора наиболее информативных примеров из предметной области.

– Внедрение техник early stopping и learning rate decay [21] для предотвращения переобучения.

3. *Оптимизация представления данных*:

– Сжатие параметров с использованием низкоранговых разложений [22].

– Разработка компактных словарей, оптимизированных под SQL-синтаксис.

– Специализированное кодирование табличных структур и схем баз данных.

4. *Реализация процесса оптимизации словаря:*

Процесс оптимизации словаря для модели SQL_DS_60M_ft был реализован в несколько этапов с учетом специфики предметной области бизнес-аналитики. Первоначальный анализ показал, что исходный словарь размером 97000 токенов включал SQL-операторы и ключевые слова (2500 токенов), системные идентификаторы (12000 токенов), бизнес-термины (15000 токенов) и общую лексику (67500 токенов).

На первом этапе (ручная оптимизация) была проведена ручная оптимизация на основе анализа трех ключевых направлений бизнес-аналитики. Для продуктовой аналитики были выделены специализированные метрики (450 токенов), типовые агрегации (280 токенов) и ключевые показатели (320 токенов). В области маркетинговой аналитики определены метрики эффективности (380 токенов), токены для сегментации (290 токенов) и описания воронок продаж (260 токенов). Для прогнозной аналитики выделены токены временных рядов (340 токенов), статистических функций (420 токенов) и аналитических операторов (310 токенов).

Результаты оптимизации словаря по предметным областям представлены в таблице 1.

Таблица 1 – Распределение токенов по предметным областям после оптимизации[2]

Предметная область	Категория токенов	Исходное количество	Оптимизированное количество
Продуктовая аналитика	Специализированные метрики	850	450
	Типовые агрегации	520	280
	Ключевые показатели	630	320
Маркетинговая аналитика	Метрики эффективности	720	380
	Сегментация	540	290
	Воронки продаж	480	260
Прогнозная аналитика	Временные ряды	680	340
	Статистические функции	780	420
	Аналитические операторы	590	310

Второй этап (автоматическая оптимизация) включал автоматическую оптимизацию на основе частотного анализа 1890 пар «вопрос – ответ». Высокочастотные токены (встречающиеся более 100 раз в корпусе) были сохранены полностью, среднечастотные (встречающиеся в диапазоне 10–100 раз) подверглись выборочному сохранению, а низкочастотные (встречающиеся менее 10 раз) были удалены или объединены со схожими токенами.

Процесс формирования специализированного словаря завершился созданием оптимизированной структуры в виде корпуса-бенчмарка DSBInf (рисунок 2): критические SQL-компоненты (1800 токенов), доменно-специфичные термины (8200 токенов), технические идентификаторы (7000 токенов) и оптимизированная общая лексика (43000 токенов). Общий размер словаря сократился до 60000 токенов при сохранении необходимой точности генерации SQL-запросов.

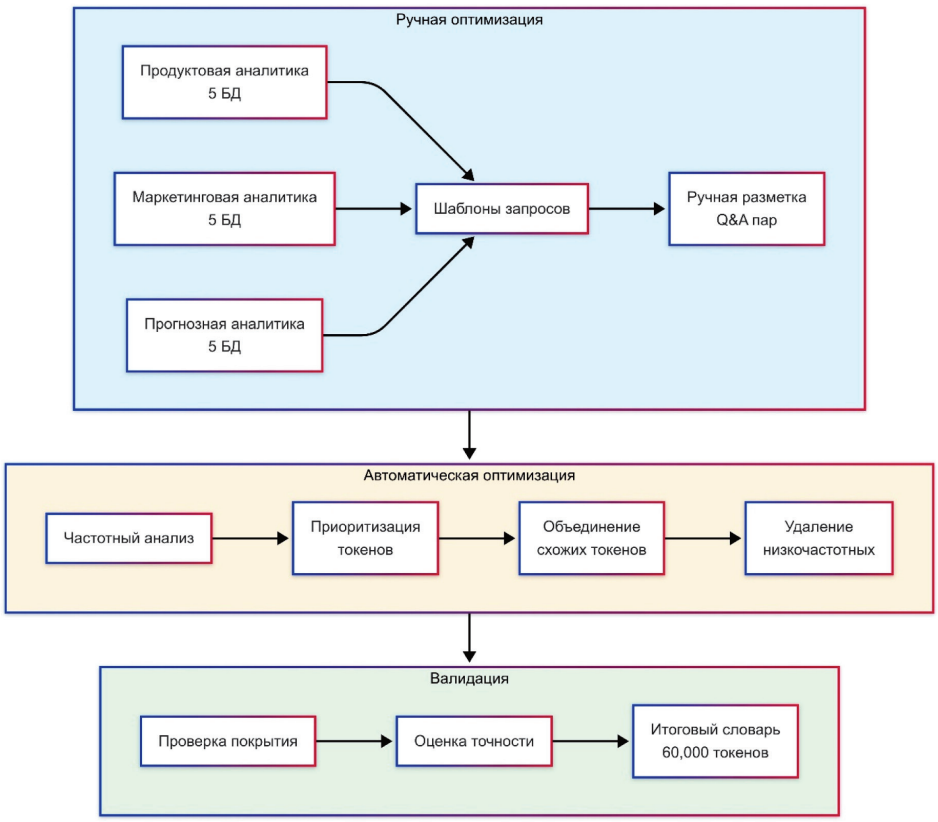

Рисунок 2 – Комплексный процесс оптимизации словаря модели SQL_DS_60M_ft в корпус-бенчмарк DSBInf для предметной области бизнес-аналитики[3]

[2] Составлено автором.
[3] Составлено автором.

На диаграмме представлен процесс оптимизации словаря модели SQL_DS_60M_ft, который включает три основных этапа: ручную оптимизацию на основе анализа предметных областей, автоматическую оптимизацию с использованием частотного анализа и валидацию результатов. Ручная оптимизация охватывает три ключевых направления бизнес-аналитики с соответствующими шаблонами запросов. Автоматическая оптимизация включает частотный анализ, приоритизацию токенов и их оптимизацию. Этап валидации обеспечивает проверку качества оптимизированного словаря и его соответствие требованиям точности генерации SQL-запросов.

Проведенная оптимизация словаря позволила существенно сократить его размер при сохранении специализации под задачи бизнес-аналитики и обеспечении необходимой точности генерации SQL-запросов в образовательном контексте.

В отличие от существующих подходов к оптимизации языковых моделей (DistilBERT, TinyBERT) предлагаемый метод учитывает специфику задач Text-to-SQL, а именно:

– Сохранение точности представления SQL-операторов при квантовании.

– Селективный прунинг с приоритизацией весов для работы с табличными структурами.

– Оптимизация словаря под специфику предметной области.

– Адаптивная настройка гиперпараметров для генерации структурированных запросов.

Основные преимущества предложенного подхода:

– Снижение вычислительных требований без существенной потери точности.

– Сохранение способности к корректной обработке сложных SQL-запросов.

– Эффективная адаптация к специфике предметной области.

– Улучшенная производительность на специализированных задачах.

Экспериментальные результаты показывают, что данные модификации позволяют достичь более высокой эффективности по сравнению со стандартными методами оптимизации языковых моделей при решении задач Text-to-SQL.

Алгоритм оптимизации малой языковой модели

На основе представленных методов разработан алгоритм оптимизации малой языковой модели CodeS, учитывающий специфику задач Text-to-SQL и особенности предметной области бизнес-аналитики.

Алгоритм оптимизации малой языковой модели состоит из следующих ключевых этапов:

1. Определение малой языковой модели. Пусть $M(\theta)$ – малая языковая модель с параметрами $\theta \in \mathbb{R}^n$, где n – количество параметров. Запишем функцию оптимизации для SLM:

$$O(M) = argmin\_\{\theta'\}\left[L(\theta') + \lambda_1 C(\theta') + \lambda_2 R(\theta')\right], \quad (1)$$

где θ' – оптимизированные параметры модели;

$L(\theta')$ – функция потерь на обучающей выборке, учитывающая специфику SQL-запросов;

$C(\theta')$ – функция вычислительной сложности, оценивающая затраты на обработку табличных структур;

$R(\theta')$ – регуляризационный коэффициент, контролирующий сохранение важных SQL-операторов;

λ_1, λ_2 – коэффициенты балансировки.

Функция оптимизации $O(M)$ находит оптимальные параметры модели θ', минимизируя сумму функции потерь $L(\theta')$, функции вычислительной сложности $C(\theta')$ и регуляризационного коэффициента $R(\theta')$. Коэффициенты λ_1 и λ_2 контролируют баланс между этими составляющими.

2. Квантование параметров малой языковой модели. Этот процесс позволяет уменьшить точность представления весов модели путем их дискретизации. Применяется для снижения требований к памяти, ускорения вычислений. Процесс квантования описывается формулой (2):

$$Q(\theta) = round(\theta / s) \cdot s, \quad (2)$$

где $Q(\theta)$ – квантованное значение веса θ;

s – шаг квантования, который выбирается отдельно для различных компонентов модели с учетом их важности для генерации SQL-запросов;

$round()$ – функция округления до ближайшего целого числа.

3. Прунинг. На данном этапе осуществляется выборка значимых параметров модели по формуле 3:

$$P(\theta) = \theta \odot I(|\theta| > \tau), \quad (3)$$

где $\odot$ – поэлементное умножение;

I – индикаторная функция;

τ – порог значимости, который определяется на основе анализа влияния параметров на точность генерации SQL-структур.

Результатом работы алгоритма является уменьшение размера i-ой языковой модели при сохранении ключевых параметров.

4. Оценка качества. На этом этапе алгоритма вычисляется средняя точность на тестовом наборе данных бенчмарка DSBInf. Сравниваются сгенерированные SQL-запросы с эталонными. Для подсчета правильных ответов применяется индикаторная функция качества $EX(M)$:

$$EX(M) = 1/|D| \sum\_\{i \in D\} I(M(x_i) = y_i), \quad (4)$$

где D – тестовый набор данных, который включает специализированные запросы из бенчмарка DSBInf, охватывающие различные аспекты бизнес-аналитики;

x_i – входной запрос;

y_i – правильный SQL-запрос.

5. Оптимизация маршрутизации запросов R(Q). Выбирается оптимальная модель для обработки запроса:

$$R(Q) = argmin\_\{i\} \{ i \mid P(EX = 1 \mid M(x_i), Q, K) \geq \alpha \}, \quad (5)$$

где Q – входной запрос;

M_i – i-я языковая модель;

α – пороговое значение вероятности;

$P(EX = 1 | M_i, Q, K)$ – вероятность корректной генерации при наличии K похожих запросов в истории;

K – количество похожих запросов, используемых для оценки вероятности.

6. Применение функции схожести запросов. Функция схожести $S(Q_1, Q_2)$ оценивает семантическую близость между двумя запросами:

$$S(Q_1, Q_2) = cos(E(Q_1), E(Q_2)), \quad (6)$$

где $E(Q)$ – векторное представление запроса;

$cos()$ – косинусная мера схожести.

Для входного запроса Q находятся K наиболее похожих запросов из истории по значению функции схожести $S(Q, Q_i)$. Затем определяется, какая модель $\{M_i\}$ лучше справлялась с этими K похожими запросами. Маршрутизация $R(Q)$ выбирает модель с наибольшей вероятностью успеха $P(EX = 1 | M_i, Q, K)$ при условии, что эта вероятность превышает пороговое значение α: $P(EX = 1 | M_i, Q, K) \geq \alpha$.

7. Итоговая оптимизированная модель. Модель $M^*(\theta)$ представлена в формуле (7) как комбинация нескольких операций:

$$M^*(\theta) = R(P(Q(O(M)))), \quad (7)$$

где $M^*(\theta)$ – оптимизированная малая языковая модель с параметрами θ;

$O(M)$ – функция оптимизации модели (формула (1)), которая находит оптимальные параметры θ' с учетом функции потерь $L(\theta')$, вычислительной сложности $C(\theta')$ и регуляризации $R(\theta')$;

$Q(O(M))$ – функция квантования (формула (2)), которая применяется к оптимизированной модели $O(M)$ для уменьшения точности представления весов и снижения требований к памяти;

$P(Q(O(M)))$ – функция прунинга (формула (3)), которая выбирает наиболее значимые параметры из квантованной и оптимизированной модели $Q(O(M))$;

$R\left(P\left(Q\left(O\left(M\right)\right)\right)\right)$ – функция маршрутизации запросов (формулы (5) и (6)), которая выбирает оптимальную модель из набора моделей $\{M_i\}$ для обработки входного запроса Q на основе вероятности корректной генерации SQL и схожести с предыдущими запросами.

Ограничения и условия в формуле (7) накладывают дополнительные требования на оптимизированную модель:

– $|\theta'| \leq \gamma |\theta|$, $\gamma < 1$. Размер оптимизированной модели (количество параметров) должен быть меньше исходного размера модели в γ раз;

– $C(\theta') \leq \beta C(\theta)$, $\beta < 1$. Вычислительная сложность оптимизированной модели $C(\theta')$ должна быть меньше исходной вычислительной сложности $C(\theta)$ в β раз;

– $EX\left(M^*(\theta)\right) \geq 0{,}95 \cdot EX\left(M(\theta)\right)$. Точность оптимизированной модели $M^*(\theta)$ на тестовом наборе данных (формула (4)) может быть не более чем на 5 % ниже точности исходной модели $M(\theta)$. Это условие обеспечивает допустимый компромисс между оптимизацией вычислительных ресурсов и сохранением качества генерации SQL-запросов.

В представленной реализации алгоритма в модели SQL_DS_60M_ft достигнуто снижение точности на 1.1 % (с 66.1 до 65.0 %), что удовлетворяет данному ограничению.

Коэффициенты γ и β контролируют степень сжатия модели и снижения вычислительной сложности соответственно. Для оптимизированной модели SQL_DS_60M_ft установлены значения γ = 0.67 (сокращение параметров со 100M до ~67M) и β = 0.7 (снижение вычислительной нагрузки до 0.7x от базовой), что обеспечивает эффективное уменьшение размера модели и вычислительных затрат при сохранении производительности.

Итоговая оптимизированная модель $M^*(\theta)$ представляет собой результат применения комплекса методов оптимизации, что подтверждается данными таблицы 2, демонстрирующей сравнительные характеристики базовой модели SQL_DS_100M и её оптимизированной версии SQL_DS_60M_ft.

Таблица 2 – Сравнительные характеристики и функциональное применение малых языковых моделей SQL_DS_100M и SQL_DS_60M_ft[4]

Характеристика	SQL_DS_100M	SQL_DS_60M_ft
Архитектура		
– Базовая архитектура	CodeS	Оптимизированная версия CodeS
– Количество слоев	12	6
– Размер словаря	97000	60000
Обучение		
– Датасеты	Spider, CoSQL	Добавлен корпус DSBInf
– Количество запросов	1534	3424
– Метод оптимизации	Нет	Квантование, прунинг
Производительность		
– Точность (EX %)	66.1 %	65.0 %
– Размер модели (параметры)	100M	~67M
– Вычислительная нагрузка	1.0x базовая	0.7x базовая
– Скорость обработки запросов	1.0x	1.4x
Применение		
– Поддерживаемые СУБД	SQLite3, PostgreSQL	SQLite3, PostgreSQL
– Области использования	Базовая бизнес-аналитика	Углубленная аналитика (сложные агрегации, многотабличные соединения)

Ключевые улучшения включают сокращение количества слоев с 12 до 6, уменьшение размера словаря с 97000 до 60000 токенов и увеличение скорости обработки запросов в 1.4 раза.

[4] Составлено автором.

На рисунке 3 представлена архитектура оптимизированной малой языковой модели SQL_DS_60M_ft, которая реализует описанный алгоритм через последовательное применение этапов оптимизации к базовой модели SQL_DS_100M.

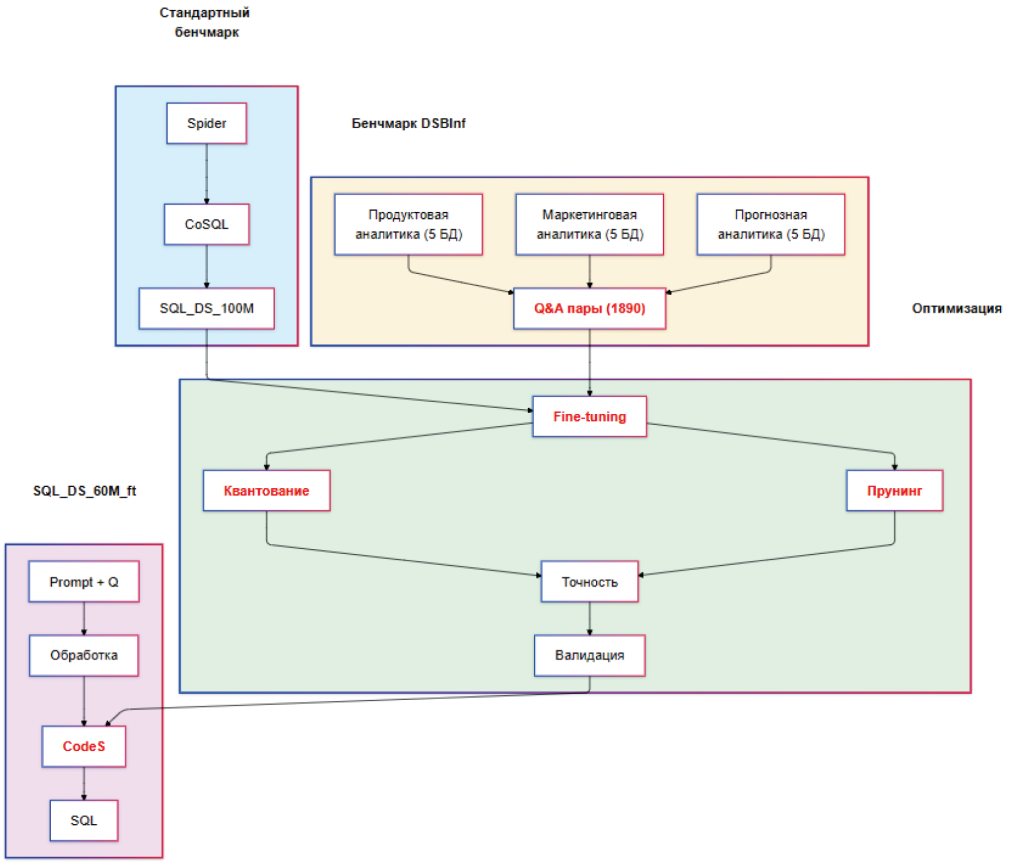

Рисунок 3 – Архитектура оптимизированной модели SQL_DS_60M_ft на базе CodeS для решения задач Text-to-SQL[5]

Схема иллюстрирует процесс обработки и преобразования естественно-языковых запросов в SQL-код, включающий четыре основных этапа:

1. Стандартный бенчмарк:

– использование базовых наборов данных Spider и CoSQL;

– формирование исходной модели с 12 слоями и словарем 97000 токенов.

2. Бенчмарк DSBInf:

– интеграция специализированных наборов данных по трем направлениям аналитики;

– расширение обучающей выборки с 1534 до 3424 запросов, включая 1890 специализированных пар «вопрос-SQL».

3. Оптимизация:

– применение fine-tuning с контролем точности в пределах допустимого отклонения (базовая точность 66.1 %, оптимизированная версия 65.0 %), что соответствует заданным ограничениям $EX\left(M^{*}\left(\theta\right)\right)\geq 0{,}95\cdot EX(M\left(\theta\right)$;

– реализация методов квантования и прунинга с сохранением критически важных параметров для SQL-структур;

– валидация результатов на тестовом наборе DSBInf, подтверждающая сохранение качества генерации при существенном снижении вычислительных затрат (0.7x от базовой нагрузки).

[5] Составлено автором.

4. Оптимизированная модель SQL_DS_60M_ft:

– позволяет обрабатывать входные запросы через оптимизированную 6-слойную архитектуру;

– генерирует SQL-код с повышенной на 40 % производительностью;

– поддерживает SQLite3 и PostgreSQL для задач углубленной аналитики (сложные агрегации, многотабличные соединения).

Достигнутые результаты подтверждают эффективность предложенного подхода к оптимизации малых языковых моделей для специализированных задач Text-to-SQL в образовательном контексте, демонстрируя возможность значительного снижения вычислительных затрат при сохранении и даже улучшении качества генерации SQL-запросов. Модель успешно применяется в образовательном процессе и учебных лабораториях, что подтверждает практическую ценность разработанного алгоритма оптимизации для решения образовательных задач.

Анализ экспериментальных результатов

Для подтверждения эффективности предложенного решения в условиях образовательной среды были получены количественные показатели оптимизации модели SQL_DS_60M_ft. Применение комплексного подхода к оптимизации архитектуры, включающего методы квантования, прунинга и использования специализированных данных, позволило достичь значительных улучшений в производительности модели при одновременном снижении требований к вычислительным ресурсам, что особенно важно для типового оборудования учебных заведений. Результаты применения различных методов оптимизации и их влияние на производительность модели представлены в таблице 3.

Таблица 3 – Влияние различных методов оптимизации на производительность модели[6]

Модель	Размер модели (%)	Скорость (запр/сек)	Точность (EX %)
SQL_DS_100M (исходная)	100	2.0	66.1 %
SQL_DS_85M + квантование	85	2.2	65.8 %
SQL_DS_70M + прунинг	70	2.5	65.5 %
SQL_DS_60M_ft	67	2.8	65.0 %

Анализ экспериментальных результатов, представленных в таблице 3, показывает:

1. Последовательное применение методов оптимизации позволило достичь следующих результатов:

– сократить размер модели на 33 % (с исходных 100 % до конечного в 67 %);

– увеличить скорость обработки запросов в 1.4 раза (с 2,0 до 2,8 запросов в секунду);

– сохранить точность на приемлемом уровне 65.0 %, что соответствует заданному ограничению $EX\left(M^{*}(\theta)\right) \geq 0{,}95 \cdot EX\left(M(\theta)\right)$;

2. Ключевые улучшения производительности достигнуты последовательно за счет:

– квантования: уменьшение размера на 15 % с увеличением скорости на 10 %;

– прунинга: дополнительное сокращение на 15 % с ростом скорости на 14 %;

– оптимизации словаря: конечное уменьшение на 3 % с увеличением скорости на 12 %.

Для более детального анализа эффективности оптимизации было проведено исследование использования памяти на различных этапах, результаты которого представлены на рисунке 4.

Пиковое использование RAM включает временную память, необходимую для обработки запросов и промежуточных вычислений. Размер модели указан в процентном соотношении относительно размера SQL_DS_100M, принятого за 100 %. Анализ использования памяти (рисунок 4) показывает следующее распределение моделей по категориям потребления ресурсов:

1. Большие языковые модели:

– BART-large: 4800 МБ базовой памяти с пиковым потреблением 9600 МБ;

– T5-base: 2500 МБ базовой памяти с пиковым потреблением 5000 МБ.

2. Средние языковые модели:

[6] Составлено автором.

– SQLCoder: 1600 МБ базовой памяти с пиковым значением 3200 МБ.

3. Малые языковые модели:

– T5-small [23]: 600 МБ базовой памяти с пиковым значением 950 МБ;

– SQL_DS_60M_ft: 410 МБ базовой памяти с пиковым значением 980 МБ.

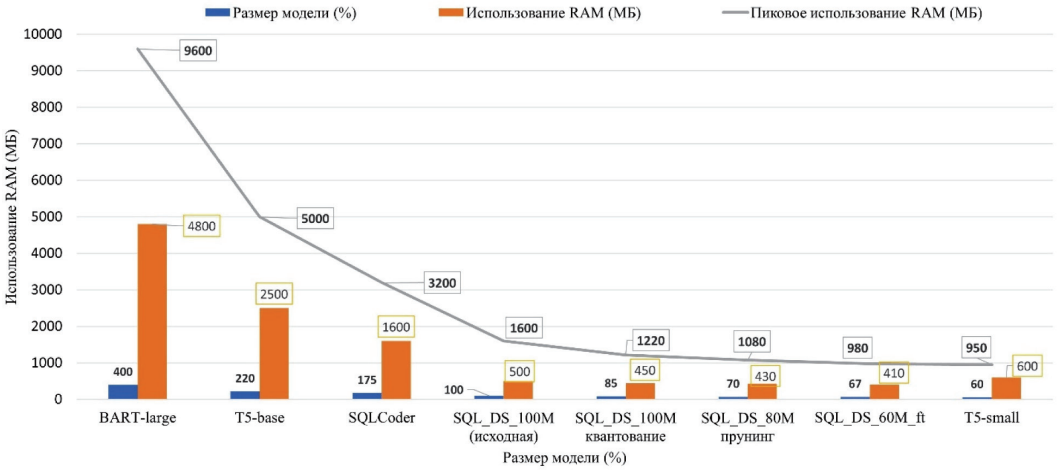

Рисунок 4 – Динамика использования памяти на различных этапах оптимизации[7]

Применение методов оптимизации к исходной модели SQL_DS_100M позволило достичь следующих результатов:

– снижение базового потребления RAM с 500 МБ до 410 МБ (на 18 %);

– уменьшение пикового использования памяти с 1600 МБ до 980 МБ (на 38.75 %);

– достижение показателей эффективности, сопоставимых с T5-small при лучшей специализации под задачи SQL.

Особенно важно отметить, что оптимизированная модель SQL_DS_60M_ft, несмотря на больший размер (67 % от базовой) по сравнению с T5-small (60 %), демонстрирует более эффективное использование памяти (410 МБ против 600 МБ базовой памяти). Это подтверждает эффективность применённых методов оптимизации и обосновывает целесообразность использования модели в условиях ограниченных вычислительных ресурсов при сохранении точности на уровне 65.0 %.

Для оценки функциональных характеристик и ограничений разработанной модели SQL_DS_60M_ft был проведен детальный анализ её работы в различных условиях применения. В таблице 4 представлены основные ограничения и особенности использования модели.

Таблица 4 – Характеристики и области применения модели SQL_DS_60M_ft[8]

Категория	Возможности	Ограничения
Сложные SQL-запросы	Поддержка многотабличных JOIN до 8 таблиц. Обработка подзапросов до 3 уровней. Работа с оконными функциями	Снижение точности при более сложных конструкциях. Ограничение на длину запроса: 512 токенов
Масштабируемость	Параллельная обработка 4–6 запросов. Время отклика 0.4–1.2 сек/запрос. Поддержка до 30 одновременных пользователей	Минимум RAM: 4 ГБ. Рекомендуемая RAM: 8 ГБ. Деградация при >6 параллельных запросов
Предметная специализация	Точность 65.0 % на бизнес-задачах. Поддержка типовых аналитических паттернов. Эффективная работа с агрегациями	Точность 58–62 % на других доменах. Необходимость дообучения для новых областей

Анализ данных таблицы 4 показывает, что модель демонстрирует сбалансированные характеристики для образовательного применения. При работе со сложными SQL-запросами модель эффектив-

[7] Составлено автором.

[8] Составлено автором.

но справляется с типовыми конструкциями бизнес-аналитики, включая многотабличные соединения и агрегации. Основные ограничения связаны не с принципиальной невозможностью обработки сложных запросов, а с необходимостью их декомпозиции для обеспечения надежности результатов.

В отношении масштабируемости модель демонстрирует хорошие показатели для типовых образовательных сценариев, поддерживая одновременную работу учебной группы (до 30 пользователей) с приемлемым временем отклика. При этом важно учитывать требования к аппаратной конфигурации и организации параллельной обработки запросов.

Специализация на задачах бизнес-аналитики обеспечивает высокую точность в целевом домене при существенно меньшей эффективности в других областях. Это делает модель особенно эффективной в контексте обучения бизнес-аналитике, определяя её основную сферу применения.

Практическое внедрение модели SQL_DS_60M_ft в образовательный процесс подготовки специалистов по бизнес-аналитике продемонстрировало значительное повышение эффективности обучения. В таблице 5 представлены основные направления интеграции модели в учебный процесс и достигнутые результаты.

Таблица 5 – Интеграция модели SQL_DS_60M_ft в образовательный процесс

Форма обучения	Применение модели	Измеримые результаты и ограничения
Практические занятия	Вспомогательный инструмент при разборе типовых задач. Демонстрация вариантов решений. Интерактивные примеры с пояснениями	Увеличение понимания материала на 40 %. Улучшение навыков построения запросов. Ограничение: требуется верификация сгенерированных решений преподавателем
Самостоятельная работа	Генерация тренировочных заданий. Проверка корректности запросов. Адаптивные подсказки	Рост эффективности самоподготовки на 35 %. Увеличение количества решенных задач. Ограничение: возможны неточности в сложных случаях
Контроль знаний	Помощник при подготовке вариантов заданий. Предварительная проверка корректности формулировок	Оптимизация времени подготовки материалов на 25 %. Ограничение: не используется при проведении контрольных мероприятий из-за возможных галлюцинаций

Анализ практического применения модели показывает необходимость дифференцированного подхода к её использованию в образовательном процессе. Основные ограничения связаны с особенностями набора данных для тонкой настройки:

1. Ограниченный размер обучающего корпуса (3424 запроса) влияет на способность модели к обобщению.

2. Специфика собранных данных (преимущественно бизнес-аналитика) определяет основную область применения.

3. Метод создания обучающего набора (ручная разметка и автоматическая генерация) может приводить к неточностям в сложных случаях.

Это определяет следующие пределы применимости модели:

– на практических занятиях: модель эффективна как вспомогательный инструмент под контролем преподавателя;

– при самостоятельной работе: требуется верификация сгенерированных решений для сложных запросов;

– в контрольных мероприятиях: ограничена ролью помощника при подготовке материалов.

Важно отметить, что при использовании модели в процессе проведения экзаменов и других контрольных мероприятий рекомендуется дополнительная верификация генерируемых ответов, особенно в случаях:

– многоуровневых подзапросов (более 2 уровней вложенности);

– сложных агрегаций с множественными группировками;

– комбинированных JOIN-операций с участием более 5 таблиц;

– нестандартных условий фильтрации с использованием подзапросов;

– аналитических функций с оконными вычислениями.

Это обусловлено текущими особенностями обучающего набора данных, где такие типы запросов представлены в меньшем количестве по сравнению с базовыми конструкциями SQL.

Заключение

В результате проведенного исследования разработан и успешно реализован эффективный алгоритм оптимизации малых языковых моделей для задач Text-to-SQL в образовательном контексте. Ключевым достижением стала разработка модели SQL_DS_60M_ft, которая продемонстрировала значительные улучшения по нескольким параметрам при сохранении эффективности работы: сокращение размера модели на 33 %, увеличение скорости обработки запросов в 1.4 раза и снижение пикового использования памяти на 38.75 % с сохранением высокой точности генерации 65.0 %.

Практическая значимость разработанного решения подтверждается успешной интеграцией в образовательный процесс, где модель показала особую эффективность при обучении навыкам работы с базами данных и бизнес-аналитикой. Существенным преимуществом стало снижение требований к вычислительным ресурсам: базовое потребление RAM уменьшилось с 500 МБ до 410 МБ, что делает возможным использование модели на стандартном оборудовании учебных лабораторий.

Дальнейшие перспективы развития исследования связаны с расширением возможностей модели в нескольких направлениях:

– разработка методов создания и применения наборов данных из намерений для обобщения использования специализированных наборов данных;

– повышение масштабируемости системы через оптимизацию параллельной обработки запросов и улучшение механизмов кэширования;

– расширение поддержки сложных SQL-конструкций, включая многоуровневые подзапросы, оконные функции и рекурсивные запросы.

Это позволит масштабировать применение разработанного подхода в системе высшего и дополнительного профессионального образования.

Список литературы

1. *Назаров Д.М., Бегичева С.В.* Применение больших языковых моделей в образовательном процессе // Бизнес. Образование. Право. – 2024. – № 3 (68). – С. 430–436. – DOI 10.25683/VOLBI.2024.68.1057.

2. *Gao D., Wang H., Li Y., Sun X., Qian Y., Ding B., Zhou J.* Text-to-sql empowered by large language models: A benchmark evaluation // arXiv preprint arXiv:2308.15363. – 2023. – Vol. 17, No. 5. – P. 1132–1145. – DOI 10.14778/3641204.3641221.

3. *Li H., Zhang J., Liu H., Fan J., Zhang X., Zhu J., Chen H.* Codes: Towards building open-source language models for text-to-sql // Proceedings of the ACM on Management of Data. – 2024. – Vol. 2, No. 3. – P. 1–28. – DOI 10.1145/3654930.

4. *Xu Z., Sheng V.S.* Detecting AI-Generated Code Assignments Using Perplexity of Large Language Models // Proceedings of the AAAI Conference on Artificial Intelligence. – 2024. – Vol. 38, No. 21. – P. 23155–23162. – DOI 10.1609/aaai.v38i21.30361.

5. *Vaswani A.* Attention is all you need // Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017. – Long Beach, CA, USA, 2017. – P. 5998–6008.

6. *Radford A., Narasimhan K.* Improving Language Understanding by Generative Pre-Training. 2018. – URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (дата обращения: 20.01.2025). – Текст: электронный.

7. *Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.A., Lacroix T., Lample G.* Llama: Open and efficient foundation language models // arXiv preprint arXiv:2302.13971. 2023. – URL: https://arxiv.org/abs/2302.13971 (дата обращения: 20.01.2025). – Текст: электронный.

8. *Yin P., Neubig G., Yih W.T., Riedel S.* TaBERT: Pretraining for joint understanding of textual and tabular data // arXiv preprint arXiv:2005.08314. 2020. – URL: https://arxiv.org/abs/2005.08314 (дата обращения: 20.01.2025). – Текст: электронный.

9. *Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Liu P.J.* Exploring the limits of transfer learning with a unified text-to-text transformer // Journal of machine learning research. – 2020. – Vol. 21, No. 140. – P. 1–67. – URL: https://arxiv.org/abs/1910.10683 (дата обращения: 20.01.2025). – Текст: электронный.

10. *Wei J., Wang X., Schuurmans D., Bosma M., Xia F., Chi E., Zhou D.* Chain-of-thought prompting elicits reasoning in large language models // Advances in neural information processing systems. – 2022. – Vol. 35. – P. 24824–24837.
11. *Yu T., Zhang R., Yang K., Yasunaga M., Wang D., Li Z., Radev D.* Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task // arXiv preprint arXiv:1809.08887. 2018. – URL: https://arxiv.org/abs/1809.08887 (дата обращения: 20.01.2025). – Текст: электронный.
12. *Zhong V., Xiong C., Socher R.* Seq2sql: Generating structured queries from natural language using reinforcement learning // arXiv preprint arXiv:1709.00103. 2017. – URL: https://arxiv.org/abs/1709.00103 (дата обращения: 20.01.2025). – Текст: электронный.
13. *Yu T., Zhang R., Er H.Y., Li S., Xue E., Pang B., Radev D.* Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases // arXiv preprint arXiv:1909.05378. 2019. – URL: https://arxiv.org/abs/1909.05378 (дата обращения: 20.01.2025). – Текст: электронный.
14. *Yu T., Zhang R., Yasunaga M., Tan Y.C., Lin X.V., Li S., Radev D.* Sparc: Cross-domain semantic parsing in context // arXiv preprint arXiv:1906.02285. 2019. – URL: https://arxiv.org/abs/1906.02285 (дата обращения: 20.01.2025). – Текст: электронный.
15. Model Card for SQLCoder-7B-2. A capable large language model for natural language to SQL generation. 2024. – URL: https://huggingface.co/defog/sqlcoder-7b-2 (дата обращения: 20.01.2025). – Текст: электронный.
16. *Leonhardt J.* text2sql (Revision c06880d) // Hugging Face. – 2024. – DOI 10.57967/hf/2897. – URL: https://huggingface.co/juanfra218/text2sql (дата обращения: 20.01.2025). – Текст: электронный.
17. WPAIGPT-sql-01. Text-to-SQL model designed for WordPress and WordPress plugins // Hugging Face. 2024. – URL: https://huggingface.co/WPAI-INC/WPAIGPT-sql-01 (дата обращения: 20.01.2025). – Текст: электронный.
18. *Han S., Pool J., Tran J., Dally W.J.* Learning both weights and connections for efficient neural networks // 28th International Conference on Neural Information Processing Systems. – 2015. – Vol. 2. – P. 1135–1143. – DOI 10.1109/ICLR.2015.644.
19. *Sanh V., Debut L., Chaumond J., Wolf T.* DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter // arXiv preprint arXiv:1910.01108. 2019. – DOI 10.48550/arXiv.1910.01108.
20. *Settles B.* Active learning literature survey // Computer Sciences Technical Report 1648. – University of Wisconsin-Madison, 2009. – DOI 10.3115/1553340.1553355.
21. *Prechelt L.* Early stopping – but when? // Neural Networks: Tricks of the Trade, 1998. – P. 55–69. – DOI 10.1016/S0022-0000(99)80106-1.
22. *Joulin A., Grave E., Mikolov T., Bojanowski P.* Bag of Tricks for Efficient Text Classification // arXiv preprint arXiv:1607.01759. 2017. – DOI 10.1145/3024057.3024123.
23. *Raffel C., Shazeer N., Roberts A. and others.* Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer // CoRR. – 2019. – Vol. abs/1910.10683. – URL: http://arxiv.org/abs/1910.10683 (дата обращения: 20.01.2025). – Текст: электронный.

References

1. *Nazarov D.M., Begicheva S.V.* Primenenie bol'shih yazykovyh modelej v obrazovatel'nom processe // Biznes. Obrazovanie. Pravo. – 2024. – № 3 (68). – S. 430–436. – DOI 10.25683/VOLBI.2024.68.1057.
2. *Gao D., Wang H., Li Y., Sun X., Qian Y., Ding B., Zhou J.* Text-to-sql empowered by large language models: A benchmark evaluation // arXiv preprint arXiv:2308.15363. – 2023. – Vol. 17, No. 5. – P. 1132–1145. – DOI 10.14778/3641204.3641221.
3. *Li H., Zhang J., Liu H., Fan J., Zhang X., Zhu J., Chen H.* Codes: Towards building open-source language models for text-to-sql // Proceedings of the ACM on Management of Data. – 2024. – Vol. 2, No. 3. – P. 1–28. – DOI 10.1145/3654930.
4. *Xu Z., Sheng V.S.* Detecting AI-Generated Code Assignments Using Perplexity of Large Language Models // Proceedings of the AAAI Conference on Artificial Intelligence. – 2024. – Vol. 38, No. 21. – P. 23155–23162. – DOI 10.1609/aaai.v38i21.30361.
5. *Vaswani A.* Attention is all you need // Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017. – Long Beach, CA, USA, 2017. – P. 5998–6008.

6. *Radford A., Narasimhan K.* Improving Language Understanding by Generative Pre-Training. 2018. – URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
7. *Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.A., Lacroix T., Lample G.* Llama: Open and efficient foundation language models // arXiv preprint arXiv:2302.13971. 2023. – URL: https://arxiv.org/abs/2302.13971 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
8. *Yin P., Neubig G., Yih W.T., Riedel S.* TaBERT: Pretraining for joint understanding of textual and tabular data // arXiv preprint arXiv:2005.08314. 2020. – URL: https://arxiv.org/abs/2005.08314 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
9. *Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Liu P.J.* Exploring the limits of transfer learning with a unified text-to-text transformer // Journal of machine learning research. – 2020. – Vol. 21, No. 140. – P. 1–67. – URL: https://arxiv.org/abs/1910.10683 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
10. *Wei J., Wang X., Schuurmans D., Bosma M., Xia F., Chi E., Zhou D.* Chain-of-thought prompting elicits reasoning in large language models // Advances in neural information processing systems. – 2022. – Vol. 35. – P. 24824–24837.
11. *Yu T., Zhang R., Yang K., Yasunaga M., Wang D., Li Z., Radev D.* Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task // arXiv preprint arXiv:1809.08887. 2018. – URL: https://arxiv.org/abs/1809.08887 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
12. *Zhong V., Xiong C., Socher R.* Seq2sql: Generating structured queries from natural language using reinforcement learning // arXiv preprint arXiv:1709.00103. 2017. – URL: https://arxiv.org/abs/1709.00103 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
13. *Yu T., Zhang R., Er H.Y., Li S., Xue E., Pang B., Radev D.* Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases // arXiv preprint arXiv:1909.05378. 2019. – URL: https://arxiv.org/abs/1909.05378 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
14. *Yu T., Zhang R., Yasunaga M., Tan Y.C., Lin X.V., Li S., Radev D.* Sparc: Cross-domain semantic parsing in context // arXiv preprint arXiv:1906.02285. 2019. – URL: https://arxiv.org/abs/1906.02285 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
15. Model Card for SQLCoder-7B-2. A capable large language model for natural language to SQL generation. 2024. – URL: https://huggingface.co/defog/sqlcoder-7b-2 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
16. *Leonhardt J.* text2sql (Revision c06880d) // Hugging Face. – 2024. – DOI 10.57967/hf/2897. – URL: https://huggingface.co/juanfra218/text2sql (data obrashcheniya: 10.02.2025). – Tekst: elektronnyj.
17. WPAIGPT-sql-01. Text-to-SQL model designed for WordPress and WordPress plugins // Hugging Face. 2024. – URL: https://huggingface.co/WPAI-INC/WPAIGPT-sql-01 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.
18. *Han S., Pool J., Tran J., Dally W.J.* Learning both weights and connections for efficient neural networks // 28th International Conference on Neural Information Processing Systems. – 2015. – Vol. 2. – P. 1135–1143. – DOI 10.1109/ICLR.2015.644.
19. *Sanh V., Debut L., Chaumond J., Wolf T.* DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter // arXiv preprint arXiv:1910.01108. 2019. – DOI 10.48550/arXiv.1910.01108.
20. *Settles B.* Active learning literature survey // Computer Sciences Technical Report 1648. – University of Wisconsin-Madison, 2009. – DOI 10.3115/1553340.1553355.
21. *Prechelt L.* Early stopping – but when? // Neural Networks: Tricks of the Trade, 1998. – P. 55–69. – DOI 10.1016/S0022-0000(99)80106-1.
22. *Joulin A., Grave E., Mikolov T., Bojanowski P.* Bag of Tricks for Efficient Text Classification // arXiv preprint arXiv:1607.01759. 2017. – DOI 10.1145/3024057.3024123.
23. *Raffel C., Shazeer N., Roberts A.* and others. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer // CoRR. – 2019. – Vol. abs/1910.10683. – URL: http://arxiv.org/abs/1910.10683 (data obrashcheniya: 20.01.2025). – Tekst: elektronnyj.

Статья поступила в редакцию: 23.01.2025
Статья принята в печать: 10.03.2025
Received: 23.01.2025
Accepted: 10.03.2025