Аннотация

С развитием баз данных возникла необходимость упрощения работы с ними. Изначально датасеты, предназначенные для решения задачи генерации SQL запроса по описанию вопроса на естественном языке, не позволяли обучить модели, которые могли бы написать запросы, использующие несколько таблиц, подзапросы или объединение результатов нескольких запросов. Когда появился вручную размеченный набор данных содержащий такие примеры, качество моделей для решения задачи росло очень медленно. Большинство появляющихся статей делали упор на поиск новых методов решения или увеличения размеров модели. Именно поэтому в данной работе будет рассмотрена аугментация данных как метод улучшения результатов. Основной целью является генерация новых данных разными метода, получение хороших результатов с помощью легкой модели и сравнение результатов моделей, полученных в результате обучения на разных аугментированных парах вопрос-запрос. В итоге получится легковесная модель сравнимая по качеству с некоторыми из списка лучших решений.

Содержание

Введение

Реляционные базы данных применяются в различных областях, включая:

- Бизнес: хранение информации о клиентах, заказах, продуктах и т. д.
- Финансы: учет финансовых операций, налоговой отчетности, бухгалтерского учета и т. д.
- Медицинский сектор: хранение информации о пациентах, медицинских диагнозах, результатов анализов и т. д.
- Образование: хранение информации о студентах, курсах, учебных планах, оценках и т. д.
- Транспорт: управление информацией о маршрутах, расписаниях, билетах и т. д.
- Интернет: хранение информации о пользователях, продуктах, заказах и т. д. в онлайн-магазинах, социальных сетях и других онлайн-сервисах.
- Производство: учет выпускаемой продукции, складских запасов, технической документации и т. д.
- Телекоммуникации: хранение информации о тарифах, клиентах, устройствах и т. д.
- Государственный сектор: учет налогов, персональных данных граждан, информации о лицензиях и т. д.
- Научные исследования: хранение и управление научной информацией, результатами исследований, экспериментальными данными и т. д.

Во многих компаниях требуется большое количество сотрудников, чтобы проводить подробную аналитику полученных данных. Для ускорения и упрощения работы с базами данных была поставлена задача преобразования естественного языка в SQL запросы. Она актуальна по нескольким причинам:

- Упрощение обращения к базе данных. Использование естественного языка для запросов к базе данных позволяет сократить время и усилия, которые требуются для изучения SQL или других языков запросов. Это позволяет пользователям без технического образования легко работать с базами данных.

- Рост количества данных. С ростом объема данных, с которыми мы сталкиваемся в современном мире, люди все больше полагаются на базы данных для хранения и управления ими. Использование естественного языка для запросов к базам данных позволяет сделать доступ к этим данным более интуитивно понятным и быстрым.
- Повышение эффективности работы. Работа с базами данных может занимать много времени и энергии. Использование естественного языка для запросов к базам данных может сократить время, которое пользователь тратит на поиск и обработку необходимой информации.
- Широкий спектр применений. Задача преобразования естественного языка в SQL запросы актуальна для многих областей, от бизнеса до медицины и правосудия. Это означает, что существует потребность в разработке разных решений для удовлетворения конкретных задач и запросов.
- Технологический прогресс. Развитие технологий машинного обучения и искусственного интеллекта позволяет создавать более точные и эффективные системы преобразования естественного языка в SQL запросы. Это открывает новые возможности для автоматизации и оптимизации работы с базами данных.

Описание задачи

Пусть задан текст на естественном языке, который содержит запрос на извлечение информации из базы данных и, возможно, информацию о схеме базы данных. Необходимо разработать алгоритм, который преобразует данный текст в корректный SQL запрос, который можно выполнить на соответствующей базе данных.

Для алгоритмического решения данной задачи необходимо выполнить следующие шаги:

1. Разобрать текст запроса на естественном языке на отдельные слова.
2. Определить, к каким таблицам базы данных относится запрос.
3. Определить, какие столбцы и какие операции необходимо использовать для выполнения запроса.
4. Сформулировать запрос на SQL, который будет соответствовать запросу на естественном языке.

5. Проверить корректность и возможность выполнения запроса на соответствующей базе данных.

С развитием баз данных и увеличению их использования для аналитики данных возникла необходимость в упрощении решения данной задачи. С развитием машинного обучения появилась задача преобразования вопросов, написанных на естественном языке в корректные SQL запросы при помощи языковых моделей.

1. Обзор литературы

1.1. Датасеты

Создание датасетов для задачи преобразования естественного языка в SQL запросы является сложным и трудоемким процессом, который требует значительных усилий со стороны специалистов в области обработки естественного языка и баз данных.

Для начала необходимо собрать достаточный объем данных, содержащих пары вопрос-ответ, где вопросы формулируются на естественном языке, а ответы представляют собой соответствующие SQL запросы. Это может быть выполнено путем ручной разметки готовых запросов или созданием модели генерации SQL запросов на основе данных, содержащих таблицы базы данных и структуру связей между ними.

После этого необходимо обработать и очистить полученные данные, чтобы удалить ошибки, повторы и прочие нежелательные элементы, которые могут повлиять на точность модели. Для этого может потребоваться относительно большое количество времени и ресурсов.

Весь процесс занимает порядка 1000 часов, согласно статье по созданию Spider[1] датасета. Поэтому большая часть задач решается и тестируется на уже существующем наборе данных вида вопрос-запрос. Рассмотрим подробнее существующие датасеты.

1.1.1. Single-domain

Изначально для решения задачи text2sql были использованы датасеты, предназначенные для проверки качества моделей, определяющих сущности в вопросе. Таких наборов данных довольно много, но в качестве значимых примеров можно привести следующие:

1. **ATIS**[2] (Air Travel Information System) датасет используется в задаче разбора естественного языка (NLP) и является одним из наиболее распространенных датасетов в области обработки естественного языка. Он содержит информацию о бронировании авиабилетов в США.

   ATIS датасет был создан в 90-х годах для разработки систем автоматической обработки речи и систем управления диалогом, а также для оценки и сравнения различных алгоритмов машинного обучения.

   Датасет состоит более чем из 4 тысяч запросов, сделанных на естественном языке, и соответствующих им меток, описывающих семантическую структуру каждого запроса.

2. **YELP**[3] - это датасет, содержащий отзывы на предприятия и места различных категорий в некоторых городах, таких как Сан-Франциско, Лос-Анджелес, Торонто, Нью-Йорк и др. Этот датасет содержит информацию о местонахождении, время работы, типы кухонь, ценовой диапазон, а также оценки и отзывы от пользователей.

   С помощью этого датасета можно исследовать различные места и определить, какие заведения пользуются наибольшей популярностью и лучшей репутацией в городе, а также определить цены на различные услуги.

Так как данные датасеты были изначально предназначены для решения задач распознавания сущностной, то схема базы данных, к которой осуществляются запросы, получается очень простой. Чаще всего она состоит из одной или двух таблиц. Из-за этого недостатка модели, обученные на данных датасетах, плохо работали с данными, которые они видели впервые, и с вопросами, которые подразумевают объединение нескольких таблиц для получения результата. Большинство датасетов были посвящены какой-то одной цели, поэтому плохо адатировались при применении в других задачах. Например, модель, обученная на данных о спорте, плохо работала для структуры школы. Эти недостатки привели к появлению следующего типа датасетов, а именно Multi-domain датасеты.

1.1.2. Multi-domain

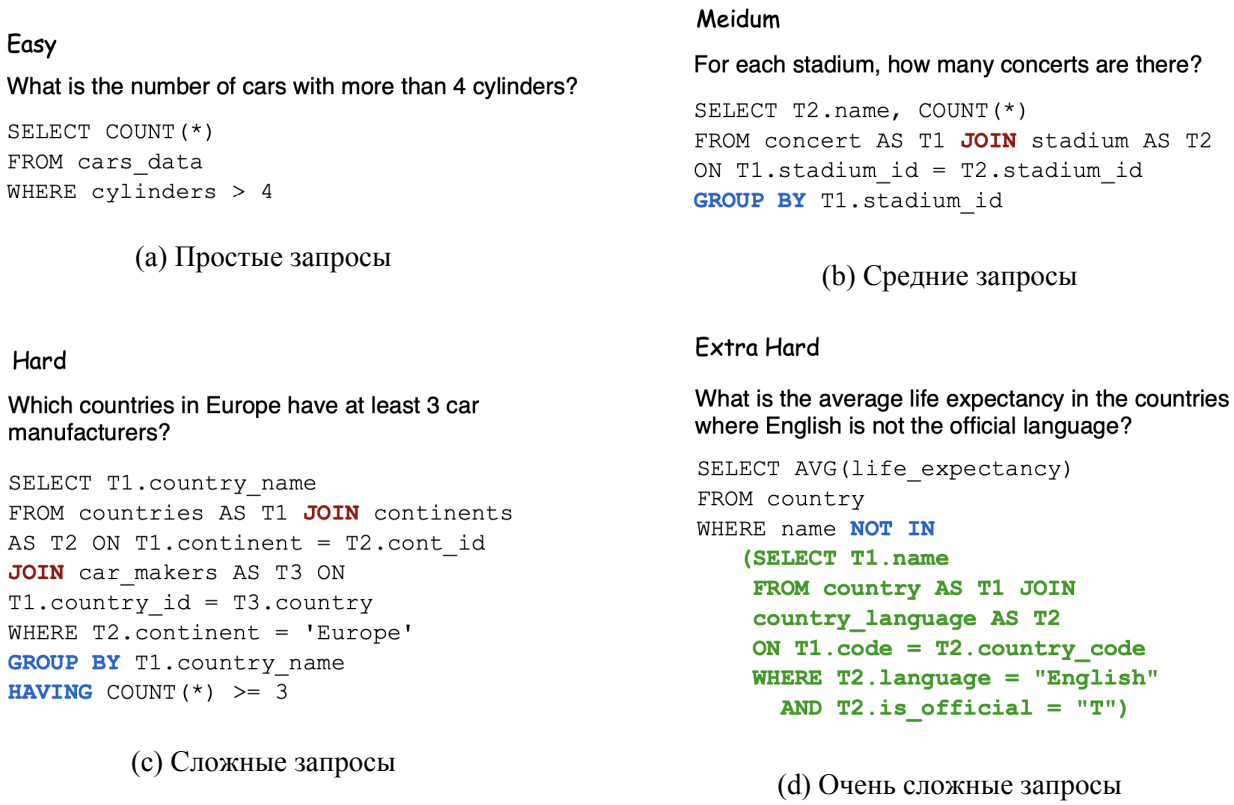
К данному типу относятся датасеты, которые содержат информацию о большом наборе независимых баз данных. Таблицы могут одновременно быть содержать данные о школах, спорте, устройстве компании и тд.

- **WikiSQL**[4] WikiSQL — это датасет, предназначенный для задачи SQL-запросов к базам данных. Он состоит из 80 654 естественно-языковых запросов и 24 241 базы данных таблиц из англоязычной Википедии. Каждая строка таблицы состоит из нескольких атрибутов (столбцов), а каждый SQL-запрос содержит запрос SELECT, условия и ключевые слова, необходимые для извлечения данных из таблицы.

  Недостатком данного датасета является отсутствие названий у таблиц. Так как все данные были получены из википедии, то у них не было названий или связи между собой. Wikisql решил проблему однотипности задач, но не помог модели понимать возможность существования взаимосвязи между сущностями.

- **Spider**[1]. Он является крупномасштабным и Multi-domain, и используется для тематического разбора и конвертации текста в SQL. Датасет состоит из 10 181 вопросов

и 5 693 уникальных сложных запросов SQL на 200 базах данных с несколькими таблицами. Различные домены были охвачены, включая 138 уникальных тематик. Этот датасет положил начало новой задачи для модели - кросс-доменная задача тематического разбора и конвертации текста в SQL, что означает, что модель обучается на различных сложных запросах SQL и базах данных. Такая задача требует от модели хорошей обобщающей способности, чтобы она могла работать с новыми запросами SQL и свежими схемами баз данных. В отличие от большинства предыдущих задач тематического разбора, где все используемые данные были одного типа, Spider представляет серьезный вызов для будущих исследований, так как качество моделей, являющихся лучшими на других датасетах не превышало 20 процентов при появлении Spider.



(a) Простые запросы

(b) Средние запросы

(c) Сложные запросы

(d) Очень сложные запросы

Рис. 1: Примеры запросов разной сложности

Чтобы лучше понимать производительность модели на разных запросах, SQL-запросы на разделены на 4 уровня: простой, средний, сложный, очень сложный. Сложность определяется на основе количества компонентов SQL, выборок и условий, так что запросы, содержащие больше ключевых слов SQL (GROUP BY, ORDER BY, INTERSECT, вложенные подзапросы, выборка столбцов и агрегаторов и т.д.), считаются более сложными. Например, запрос считается сложным, если он содержит более двух столбцов SELECT, более двух условий WHERE и GROUP BY два столбца или содержит EXCEPT

или вложенные запросы. SQL с большим количеством дополнительных условий считается очень сложным. На рисунке 1 приведены примеры SQL-запросов на четырех уровнях сложности.

- **Другие**. Существует множество датасетов для решения text2sql на других языках. Такими датасетами являются
  - TableQA(zh);
  - DuSQL(zh);
  - ViText2SQL(vi);
  - CSpider(zh);
  - PortugueseSpider(pt)

1.1.3. Другие

Отдельно выделяют датасеты, предназначенные для итеративного общения с пользователем и предоставлением результатов.

- **CoSQL**[5] - это набор данных и система тестирования, ориентированная на задачи естественно-языкового взаимодействия с базами данных, которая была представлена на EMNLP 2020. Набор данных содержит 7 744 диалоговые пары (запрос - ответ), связанные с темами путешествий, ресторанов и отелей. В каждой диалоговой паре человек задает вопрос на естественном языке, а модель должна сгенерировать соответствующий запрос SQL, который вернет правильный ответ из базы данных.
- **SParC**[6] - это набор данных для кросс-доменного семантического разбора в контексте, который состоит из 4298 последовательностей связанных вопросов (более 12 тыс. отдельных вопросов, размеченных SQL-запросами), полученных в результате управляемого взаимодействия пользователя с 200 сложными базами данных из 138 областей.

1.2. Методы улучшения решения

1.2.1. Аугментация данных

Посмотрим на размер датасетов, подготовленных для данной задачи.

Dataset	# Q	# SQL	# DB	# Domain	# Table /DB	ORDER BY	GROUP BY	NESTED	HAVING
ATIS	5,280	947	1	1	32	0	5	315	0
GeoQuery	877	247	1	1	6	20	46	167	9
Scholar	817	193	1	1	7	75	100	7	20
Academic	196	185	1	1	15	23	40	7	18
IMDB	131	89	1	1	16	10	6	1	0
Yelp	128	110	1	1	7	18	21	0	4
Advising	3,898	208	1	1	10	15	9	22	0
Restaurants	378	378	1	1	3	0	0	4	0
WikiSQL	80,654	77,840	26,521	-	1	0	0	0	0
Spider	10,181	5,693	200	138	5.1	1335	1491	844	388

Рис. 2: Датасеты[1]

Spider является уникальным датасетом с большим числом таблиц в пределах одной базы данных, что обеспечивает большее количество запросов, связанных с пониманием взаимосвязей между таблицами. Благодаря этой особенности он является основным датасетом для проверки качества работы моделей. Однако, его главным недостатком является большой размер самого датасета, так как ручная разметка данных — трудоемкая задача. Поэтому актуальным методом для улучшения результата работы моделей является генерация новых данных, аналогичных предыдущим.

Аугментации данных посвящены разделы в статье[7]. Основными методами аугментации данных являются:

- Переформулирование вопросов и заполнение заранее определенных шаблонов для увеличения разнообразия данных. В данном методе собираются шаблоны запросов и заполняют их с помощью схемы базы данных. Затем используют модели, обученные на генерацию вопроса на естественном языке, который удовлетворял бы этому запросу.

- Использование готовых баз данных перефразирования (например, PPDB[8]). Но такие методы сложны для реализации в сложных предложениях, поскольку требуется вручную настраивать какие предложения и как преобразовывать, чтобы не получить далекое по смыслу предложение. Это вызвано тем, что синонимы иногда определяются по контексту, а готовые базы данных содержат только отображение из слова в его синонимы, что значительно усложняет автоматизацию.

Качество аугментированных данных важно, потому что низкокачественные данные могут негативно сказаться на производительности моделей[9]. Одним из актуальных решений для избежания проблемы ухудшения качества является фильтрация аугментированных данных. Для этого вне зависимости от метода аугментации стоит оставлять только запросы, которые уже были в выборке ранее. По этой причине большинство методов заключается в

генерации описания к готовым запросам.

1.2.2. Encoder

Все модели для решения задачи text2SQL состоят из двух основных частей: encoder и decoder. Рассмотрим подробнее первую часть:

- **Определение типов токенов**. В данном методе для каждого слова в предложении предсказывается его категория. Чаще всего предсказывают названия таблиц и колонок. Иногда классифицируют какие-либо ключевые слова в предложении, которые помогают обозначить ограничения, накладываемые на значения. Примером такой модели является TypeSQL[10].

- **Построение графа вопроса**. На основе вопроса строится дерево зависимостей слов. На его основе выделяются ключевые поля как и в предыдущем методе, при этом древесную структуру запроса может быть проще сопоставить с такой же структурой для текстового вопроса. Например, SADGA[11].

- **Self-attention**. Данные модели модифицируют механизм внимания, добавляя в него слагаемые, связанные со структурой базы данных. Например, RAT-SQL[12]

- **Использование готовых языковых моделей**. В качестве encoder'а в данном методе выступают предобученные модели, например, BERT или T5. Далее в алгоритме используются полученные эмбединги.

- **Использование моделей, предобученных для задачи text2sql**. Такой моделью является Grappa[13].

Больше моделей для каждого метода можно найти в таблице:

Methods	Adopted by	Applied datasets	Addressed challenges
Encode token type	TypeSQL (Yu et al., 2018a)	WikiSQL	Representing question meaning
Graph-based	GNN (Bogin et al., 2019a)	Spider	(1) Representing question and DB schemas in a structured way (2) Schema linking
	Global-GCN (Bogin et al., 2019b)	Spider	
	IGSQL (Cai and Wan, 2020)	Sparc, CoSQL	
	RAT-SQL (Wang et al., 2020a)	Spider	
	LEGSQL (Cao et al., 2021)	Spider	
	SADGA (Cai et al., 2021)	Spider	
	ShawdowGNN (Chen et al., 2021b)	Spider	
	S^2SQL (Hui et al., 2022)	Spider, Spider-Syn	
Self-attention	X-SQL (He et al., 2019)	WikiSQL	
	SQLova (Hwang et al., 2019)	WikiSQL	
	RAT-SQL (Wang et al., 2020a)	Spider	
	DuoRAT (Scholak et al., 2021a)	Spider	
	UnifiedSKG (Xie et al., 2022)	WikiSQL, Spider	
Adapt PLM	X-SQL (He et al., 2019)	WikiSQL	Leveraging external data to represent question and DB schemas
	SQLova (Hwang et al., 2019)	WikiSQL	
	Guo and Gao (2019)	WikiSQL	
	HydraNet (Lyu et al., 2020)	WikiSQL	
	Liu et al. (2021b), etc	Spider-L, SQUALL	
Pre-training	TaBERT (Yin et al., 2020)	Spider	
	GraPPA (Yu et al., 2021)	Spider	
	GAP (Shi et al., 2020a)	Spider	

Рис. 3: Виды encoder и модели, в которых они используются

1.2.3. Decoder

- **Построение дерева запроса**. Из полученных ранее результатов декодер составляет синтаксически корректное дерево запроса, то есть определяет все необходимые таблицы, колонки, накладываемые условия. syntaxsqlnet[14]

- **Заполнение шаблона**. По определенным ранее секциям вопроса выбирается корректный шаблон запроса, в который на нужные места помещаются названия соответствующих сущностной и численные значения. IRNet[15]

- **Attention**. Лучшим примером является RATSQL[12]. Для лучшего определения ключевых слов в вопросе в модель передается схема базы данных и используется механизм внимания по полученным таблицам и колонкам.

- **Механизм копирования**. Слова, которые могут выступать в роли названия таблиц или колонок заимствуются напрямую из вопроса. По нему работали первые, простейшие модели, поэтому их результат не был выдающимся.

- **Промежуточное представление** Оно представляет собой синтаксическое дерево, в котором каждый узел представляет элемент запроса, такой как таблица, колонка, условие и т.д. В этом представлении используются токены, которые соответствуют грам-

матической структуре запроса и помогают выявлять связи между его различными элементами. Примером реализации является IRNet[15].

Больше моделей для каждого вида декодера можно найти в таблице:

Methods		Adopted by	Applied datasets	Addressed challenges
Tree-based		Seq2Tree (Dong and Lapata, 2016)	-	Hierarchical decoding
		Seq2AST (Yin and Neubig, 2017)	-	
		SyntaxSQLNet (Yu et al., 2018b)	Spider	
Sketch-based		SQLNet (Xu et al., 2017)	WikiSQL	
		Dong and Lapata (2018)	WikiSQL	
		IRNet (Guo et al., 2019)	Spider	
		RYANSQL (Choi et al., 2021)	Spider	
Bottom-up		SmBop (Rubin and Berant, 2021)	Spider	
Attention Mechanism	Attention	Seq2Tree (Dong and Lapata, 2016)	-	Synthesizing information for decoding
		Seq2SQL (Zhong et al., 2017)	WikiSQL	
	Bi-attention	Guo and Gao (2018)	WikiSQL	
	Structured attention	Wang et al. (2019)	WikiSQL	
	Relation-aware Self-attention	DuoRAT (Scholak et al., 2021a)	Spider	
Copy Mechanism		Seq2AST (Yin and Neubig, 2017)	-	
		Seq2SQL (Zhong et al., 2017)	WikiSQL	
		Wang et al. (2018a)	WikiSQL	
		SeqGenSQL (Li et al., 2020a)	WikiSQL	
Intermediate Representation		IncSQL (Shi et al., 2018)	WikiSQL	Bridging the gap between natural language and SQL query
		IRNet (Guo et al., 2019)	Spider	
		Suhr et al. (2020)	Spider and others♠	
		Herzig et al. (2021)	GeoQuery, ATIS, Scholar	
		Gan et al. (2021c)	Spider	
		Brunner and Stockinger (2021)	Spider	
Others	Constrained decoding	UniSAr (Dou et al., 2022)	WikiSQL, Spide and others♡	Fine-grained decoding
		PICARD (Scholak et al., 2021b)	Spider, CoSQL	
	Execution-guided	SQLova (Hwang et al., 2019)	WikiSQL	
		Wang et al. (2018b)	WikiSQL	
	Discriminative re-ranking	Global-GCN (Bogin et al., 2019b)	Spider	SQL Ranking
		Kelkar et al. (2020)	Spider	
	Separate submodule	SQLNet (Xu et al., 2017)	WikiSQL	Easier decoding
		Guo and Gao (2018)	WikiSQL	
		Lee (2019)	Spider	
	BPE	Müller and Vlachos (2019)	Advising, ATIS, GeoQuery	
	Link gating	Chen et al. (2020b)	Spider	Synthesizing information for decoding

Рис. 4: Виды decoder и модели, в которых они используются

1.3. Метрики

1.3.1. Exact Match

Exact match - это метрика, используемая для оценки качества решения задачи text-to-SQL, которая показывает процентное соотношение запросов SQL, сгенерированных моделью, которые точно соответствуют правильному ответу. Данная метрика допускает некоторые различия в эталонном и предсказанном запросах, а именно, перестановку условий для фильтрации и колонок в select запросе.

1.3.2. Execution Accuracy

Execution accuracy - это метрика, используемая для оценки качества решения задачи text-to-SQL, которая показывает процент запросов SQL, сгенерированных моделью, которые точно соответствуют правильному результату выполнения запроса на базе данных.

Эта метрика оценивает не только корректность сгенерированного запроса SQL, но и его выполнение на базе данных. Если сгенерированный запрос SQL правильно выполнен на базе данных, то модель считается точной для этого запроса. Для вычисления данной метрики поднимается временная база данных для которой выполняются все запросы и проверяется корректность их выполнения.

1.4. Partial Match

Помимо проверки на совпадения всего запроса, можно анализировать какая именно его часть определяется ошибочно. Это поможет локализовать проблемы модели и определить вектор для улучшения подхода. Эта метрика применяется по определенным группам слов: по секциями запроса (какая доля секций Select, From Where, Group BY, Having, Order by корректно предсказана моделью) и по ключевым словам, то есть корректное определение сущностной, численных значений, ключевых слов AND, OR, LIMIT.

2. Постановка задачи

2.1. Анализ теории

Внимательный анализ статей с лучшими для своего времени решениями дает понять, что вопрос аугментации данных поднимался редко. Так как никто не ставил своей целью улучшения качества этим методом, поставим ее в этой работе. Как говорилось ранее, наиболее раскрыта тема аугментации данных в статье[7], обозревающей перспективы развития. Основные методы аугментации данных:

- **Аугментация с заменой слов**. Этот метод использует готовые базы данных синонимов и заменяет определенные слова в тексте вопроса. Такой метод уже был исследован и не дал особого прогресса, поэтому в данной работе он рассмотрен не будет.
- **Генерация новых описаний с помощью преобученных моделей**. В данном методе предлагается решить обратную задачу. Возьмем предобученную модель, которая умеет генерировать текстовые ответы, затем обучим ее на исходных датасетах. В качестве

входных данных модель получит SQL запросы. Выходными данными послужат текстовые описания к соответствующим запросам. В итоге мы получим модель, которая способна генерировать некоторый набор ответов, где в качестве вопроса будет поступать SQL запрос. Применим эту модель, для увеличения размера датасета, то есть к каждому запросу сгенерируем несколько текстовых описаний.

- **Аугментация вопросов с помощью готовых моделей**. Для решения данной задачи воспользуемся готовыми моделями, которые перефразируют вопросы. Это могут быть модели, которые генерируют синонимичные предложения или меняющие стиль текста.

На данный момент почти все модели, показывающие хорошие результаты на тестах являются крупными и у большинства из них отсутствует код и статьи, посвященные их реализации. Обучение крупных моделей — затратное по времени и мощности занятие. Поэтому в данной работе будет исследоваться влияние аугментации на легкие модели.

2.2. Формальная постановка задачи

Работа будет состоять из следующих этапов:

1. Загрузить Spider датасет.
2. Применить методы аугментации.
   (a) Обучить модель для генерации описания и применить её.
   (b) Применить несколько моделей для генерации синонимичных вопросов.
3. Обучить в одинаковых условиях T5-base на исходном датасете и на совмещенных с аугментацией.
4. Сравнить влияние методов аугментации на качество моделей по всем метрикам.

В качестве тестируемой модели была выбрана T5[16].

Модель T5 (Text-to-Text Transfer Transformer) — это глубокая нейронная сеть, основанная на трансформере, который был разработан компанией Google в 2020 году.

Одной из особенностей модели является ее способность решать различные типы задач в единой архитектуре. T5 может работать с текстовыми данными различных форматов: от задач машинного перевода до генерации текста и ответов на вопросы. В нашем случае, ответами на вопросы будут корректные sql запросы.

3. Эксперименты

3.1. Разметка с помощью моделей

Обучим T5-base на обратной задаче. С помощью полученной модели сгененрируем по 3, 5 и 7 описаний для каждого вопроса. Таким образом, полученные датасеты будут иметь следующий размер:

Таблица 1: Размер таблиц после аугментации

Название	Значение
Spider	7508
SpiderT5 Spider 3	30032
SpiderT5 Spider 5	45048
SpiderT5 Spider 7	60064

Здесь и далее все названия будут согласованы с методом, которым они получены, к кому применены и используемые гиперпараметры. В данном случае SpiderT5 означает, что использовалась модель T5 обученная на датасете spider, применилась к Spider и далее указано число генерируемых вопросов.

3.1.1. Примеры сгенерированных данных

Запрос

```
SELECT avg(num_employees)
FROM department
WHERE ranking BETWEEN 10 AND 15
```

Сгенерированные описания:

- Find the average number of employees for the departments whose rank is between 10 and 15.
- What is the average number of employees in the departments that rank between 10 and 15?
- What is the average number of employees of the departments whose ranking is between 10 and 15?

- What is the average number of employees of the departments whose rank is between 10 and 15?

3.1.2. Преимущества

Данная способ работает быстро, позволяет генерировать большой объем новых данных. Благодаря предобученным моделям генерируются разнообразные запросы хорошего качества.

3.1.3. Недостатки

Обе модели (применяемая для аугментации и обучаемая на полученных данных) видели только Spider датасет, поэтому результат может получиться переобученным для схемы из данного датасета.

Не смотря на хорошее качество описания, некоторые предложения совпадают слишком сильно, что тоже может сказаться на качестве модели на вопросах, которые модель не видела.

3.2. Аугментация вопросов с помощью готовых моделей

В данной части воспользуемся несколькими разными моделями для генерации текстовых описаний.

3.2.1. BART-paraphrase

Для начала используем bart paraphrase[17]. Данная модель была выбрана, так как основной задачей ее обучения было как раз перефразирование вопросов, которое нам требуется. Для каждого вопроса из исходного набора данных сгенерируем похожий вопрос и добавим результат вместе с исходным запросом в новый датасет, соответствующий изначальному вопросу и новые данные. Полученные данные:

Таблица 2: Размер таблиц после аугментации

Название	Значение
Spider	7508
Perephrase Spider 1	15016
Perephrase Spider 3	30032

3.2.2. Примеры сгенерированных данных

Запрос

```
SELECT avg(num_employees)
FROM department
WHERE ranking BETWEEN 10 AND 15
```

Сгенерированные описания:

- What are departments with average number of employees of 10 to 15?
- What departments have an average number of employees of 10 to 15?

3.2.3. Преимущества

Данная способ работает быстро, позволяет генерировать большой объем новых данных с качественными вопросами. Позволяет генерировать вопросы с большей вариативностью, поскольку не проходил дообучения и не привязан к схеме базы данных.

3.2.4. Недостатки

У данного метода меньше вариативность данных. Можно заметить, что он генерирует меньше вопросов, чем предыдущая модель. При этом текстовые описания, полученные данным методом, отличаются друг от друга лишь парой слов.

3.2.5. Parrot paraphrase

Рассмотрим применение Parrot paraphrase[18] Преимущества этой модели в том, что он позволяет не только генерировать определенное число предложений, но и фильтровать их, чтобы результаты были качественными. Для этого есть два настраиваемых гиперпараметра: адекватность, то есть генерированный текст передает то же значение, что и исходный контекст, и беглость, то есть текст написан на грамматически правильном английском языке. Регулируя эти два критерия, получим следующие датасеты:

Таблица 3: Размер таблиц после аугментации

Название	Значение
Spider	7508
Parrot Spider 0.9 0.75	10163
Parrot Spider 0.75 0.75	10931
Parrot Spider 0.85 0.85	10054
Parrot Spider 0.75 0.8	10707

3.2.6. Примеры сгенерированных данных

Запрос

```
SELECT avg(num_employees)
FROM department
WHERE ranking BETWEEN 10 AND 15
```

Сгенерированные описания:

- what are departments with average number of employees of 10 to 15?
- how many employees are in departments with a rank between 10 and 15?
- tell me the average number of employees in a department between 10 and 15?
- show the average number of employees in departments between 10 and 15?
- what is the average number of employees of departments between 10 and 15?

3.2.7. Преимущества

Модель генерирует более разнообразные описания без потери смысла. Этот факт позволит модели обучиться на более разнообразных данных и лучше работать на вопросах, которые она не видела ранее.

3.2.8. Недостатки

Так как помимо генерации модель проводит набор проверок для определения качества полученных текстовых описаний, на применение требуется большой объем времени.

Возникает вопрос в балансе между качеством вопросов и объемом аугментированных данных. При изменении гиперпараметров можно получать больше текстовых описаний на естественном языке, но их разнообразие и схожесть с вопросами, которые может задать человек, будет уменьшаться.

4. Результаты

Для каждого полученного датасета обучим T5-base и сравним полученные результаты. Для тестирования качества обученных моделей воспользуемся тестовой выборкой датасета Spider и метриками, упомянутыми ранее. Всего в ней находится 1034 пары вопрос-запрос. Распределение по сложностям представлено в таблице 4.

	easy	medium	hard	extra	all
количество запросов	250	440	174	170	1034

Таблица 4: Содержание тестовой выборки

4.1. Execution Accuracy

Все модели, обученный на аугментированных данных показали результаты значительно лучше, чем baseline (см таблицу 5). Можно заметить, что во всех секциях появился значительный скачек, особенно заметный в очень сложных запросах.

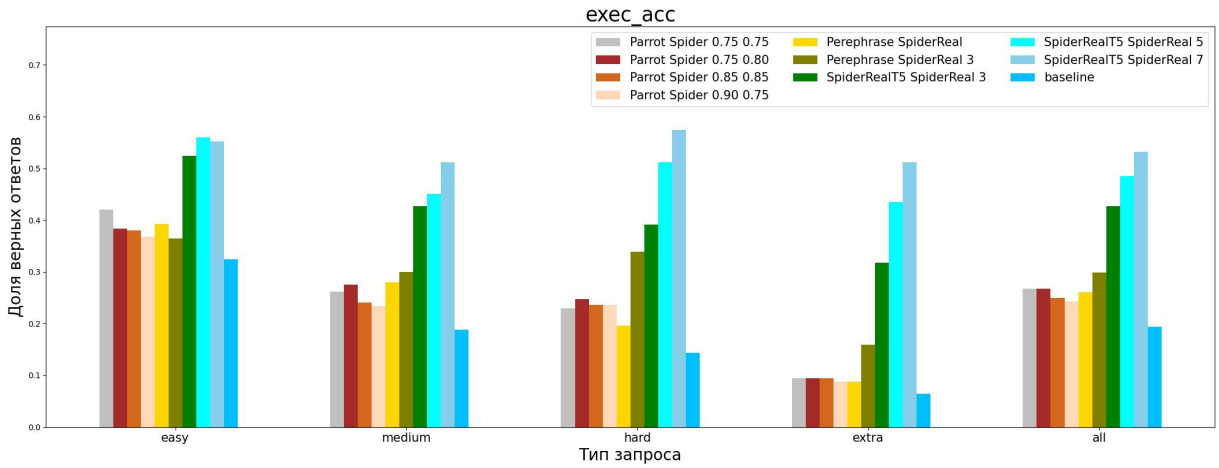


Рис. 5: Execution accuracy

Лучше всего показали себя модели, которые были обучены на обратной задаче. Лучший результат составил 0.54 по точности исполнения. Аналогичные результаты в списке лучших

по решению этой задачи на сайте датасета данная модель была бы на 25 месте. Важно отметить высокое качество моделей на сложных и очень сложных запросах (порядка 57% и 51% соответственно). Такой большой скачек означает, что модель научилась распознавать ситуации, когда необходимо объединять сущности, писать подзапросы и несколько ограничений. При этом никак не используются данные о схеме базы данных, то есть без дополнительных надстроек, как в более сложных моделях.

Таблица 5: Execution accuracy

Название модели	easy	medium	hard	extra	all
baseline	0.324	0.189	0.144	0.065	0.193
Perephrase SpiderReal	0.392	0.280	0.195	0.088	0.261
Perephrase SpiderReal 3	0.364	0.300	0.339	0.159	0.299
SpiderRealT5 SpiderReal 7	0.552	0.511	0.575	0.512	0.532
SpiderRealT5 SpiderReal 5	0.560	0.450	0.511	0.435	0.485
SpiderRealT5 SpiderReal 3	0.524	0.427	0.391	0.318	0.426
Parrot Spider 0.90 0.75	0.368	0.234	0.236	0.088	0.243
Parrot Spider 0.90 0.85	0.408	0.270	0.195	0.141	0.270
Parrot Spider 0.75 0.80	0.384	0.275	0.247	0.094	0.267
Parrot Spider 0.75 0.75	0.420	0.261	0.230	0.094	0.267
Parrot Spider 0.85 0.85	0.380	0.241	0.236	0.094	0.250

4.2. Exact Match

Аналогично все модели значительно лучше, чем baseline (см таблицу 6).

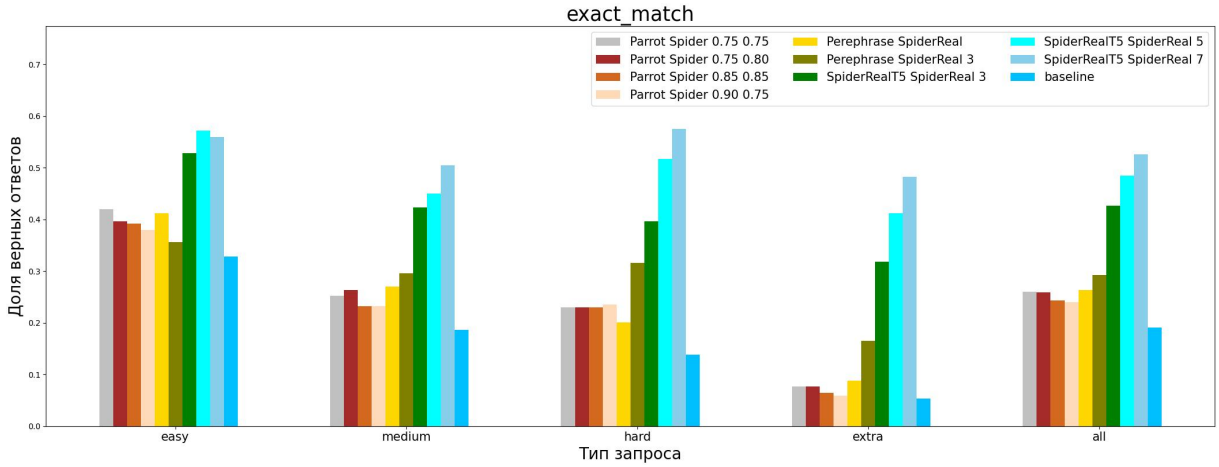


Рис. 6: Exact Match

Лучший результат по проценту совпадения составил 53%. Это аналогично 57 позиции в топе моделей с сайта spider датасета. Стоит отметить, что выше находятся модели, которые значительно больше по размеру и используют схему базы данных, что облегчает получение высоких результатов.

Таблица 6: Exact Match

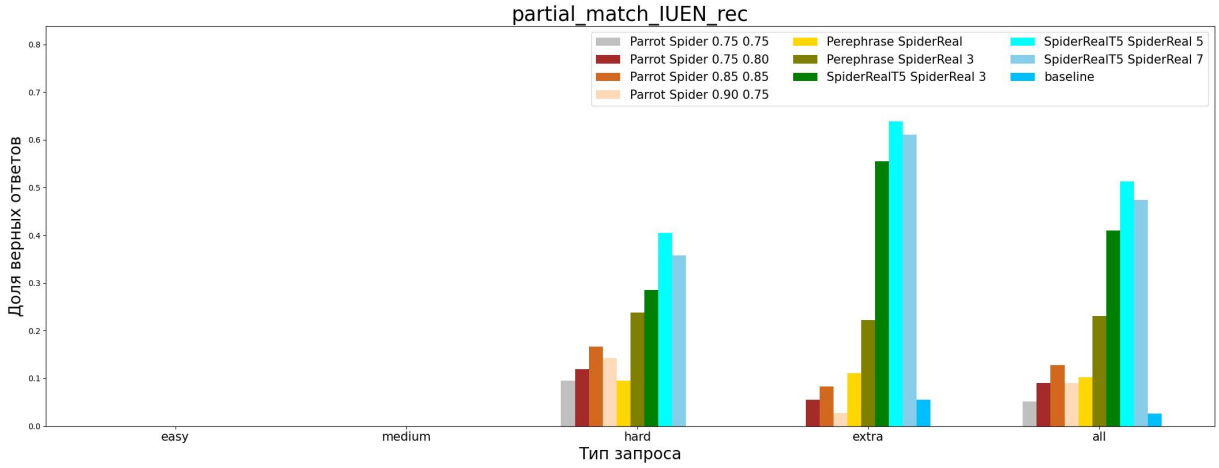
Название модели	easy	medium	hard	extra	all
baseline	0.328	0.186	0.138	0.053	0.191
Perephrase SpiderReal	0.412	0.270	0.201	0.088	0.263
Perephrase SpiderReal 3	0.356	0.295	0.316	0.165	0.292
SpiderRealT5 SpiderReal 7	0.560	0.505	0.575	0.482	0.526
SpiderRealT5 SpiderReal 5	0.572	0.450	0.517	0.412	0.485
SpiderRealT5 SpiderReal 3	0.528	0.423	0.397	0.318	0.426
Parrot Spider 0.90 0.85	0.416	0.257	0.190	0.094	0.257
Parrot Spider 0.90 0.75	0.380	0.232	0.236	0.059	0.240
Parrot Spider 0.85 0.85	0.392	0.232	0.230	0.065	0.243
Parrot Spider 0.75 0.80	0.396	0.264	0.230	0.076	0.259
Parrot Spider 0.75 0.75	0.420	0.252	0.230	0.076	0.260

Хотя результаты, полученные с помощью готовых моделей для перефразирования не так хороши, как для обученной вручную, но они могут дать лучший результат на запросах, которые модель не видела ранее. Так как прямая и обратная задача решались для одного и того

же датасета, то модели научились понимать структуру из-за большого объема данных. Перефразирование с помощью готовых моделей не дало такого результата, так как количество сгенерированых описаний было меньше. Помимо этого parrot генерировал более разнообразные предложения, что осложняло работу для модели, решающей задачу преобразования естественного языка в SQL запросы, но повысило бы качество модели в целом, то есть на вопросах, которые относились бы к другим схемам баз данных. К сожалению, готовых размеченных наборов данных, содержащий множество таблиц из разных областей, мало, поэтому проверить эту гипотезу не получится.

4.3. Partial match

Для лучшего понимания, в чем именно заключается преимущество полученных моделей, стоит рассмотреть результаты, полученные для отдельных частей запроса, то есть понять, почему именно выросло качество. Большая часть графиков похожа на полученные ранее метрики и по ним можно лишь сделать вывод, что больший набор данных помог лучше обучиться модели. Наиболее интересными является график recall для пересечения, объединения и исключения (сокращенно IUEN). Запросы, содержащие такие конструкции относятся к сложным и очень сложным, так как в них присутствует много условий. В отличии от модели, которая обучалась только на исходных данных, другие стали лучше распознавать такие сложные конструкции. Одной из целей Spider датасета является обучение моделей для распознавания сложных запросов, для написания которых пользователю пришлось бы изучать подробно структуру базы данных. Полученные модели справляются с этой задачей в большинстве случаев.



Для всех частей запроса были измерены accuracy, recall и f1. Полученные данные представлены ниже в таблицах.

Таблица 7: Accuracy

Название модели	select	select(no AGG)	where	where(no OP)	group(no Having)
baseline	0.946	0.950	0.822	0.873	0.905
Perephrase SpiderReal	0.947	0.955	0.882	0.908	0.963
SpiderRealT5 SpiderReal 7	0.974	0.977	0.934	0.951	0.943
SpiderRealT5 SpiderReal 5	0.962	0.965	0.924	0.949	0.914
SpiderRealT5 SpiderReal 3	0.956	0.966	0.935	0.950	0.957
Parrot Spider 0.90 0.75	0.944	0.948	0.894	0.929	0.949
Parrot Spider 0.90 0.85	0.937	0.943	0.878	0.919	0.909
Parrot Spider 0.85 0.85	0.953	0.965	0.839	0.888	0.964
Parrot Spider 0.75 0.80	0.946	0.958	0.893	0.912	0.924
Parrot Spider 0.75 0.75	0.951	0.960	0.887	0.925	0.953

Таблица 8: Accuracy

Название модели	group	order	and/or	IUEN	keywords
baseline	0.825	0.937	0.940	0.333	0.925
Perephrase SpiderReal	0.939	0.893	0.945	0.571	0.940
SpiderRealT5 SpiderReal 7	0.914	0.932	0.982	0.787	0.947
Perephrase SpiderReal 3	0.889	0.888	0.954	0.818	0.909
SpiderRealT5 SpiderReal 5	0.892	0.931	0.972	0.976	0.938
SpiderRealT5 SpiderReal 3	0.913	0.918	0.963	0.865	0.944
Parrot Spider 0.90 0.85	0.886	0.890	0.950	0.600	0.934
Parrot Spider 0.90 0.75	0.885	0.886	0.941	0.778	0.950
Parrot Spider 0.85 0.85	0.892	0.870	0.942	0.667	0.929
Parrot Spider 0.75 0.80	0.873	0.843	0.946	0.500	0.926
Parrot Spider 0.75 0.75	0.929	0.927	0.946	0.571	0.964

Таблица 9: Recall

Название модели	select	select(no AGG)	where	where(no OP)	group(no Having)
baseline	0.236	0.237	0.204	0.216	0.212
Perephrase SpiderReal	0.309	0.311	0.284	0.292	0.294
Perephrase SpiderReal 3	0.370	0.372	0.363	0.378	0.309
SpiderRealT5 SpiderReal 7	0.581	0.583	0.687	0.700	0.491
SpiderRealT5 SpiderReal 5	0.535	0.537	0.613	0.630	0.472
SpiderRealT5 SpiderReal 3	0.467	0.472	0.546	0.555	0.409
Parrot Spider 0.90 0.85	0.315	0.317	0.317	0.332	0.297
Parrot Spider 0.90 0.75	0.279	0.279	0.265	0.275	0.275
Parrot Spider 0.85 0.85	0.292	0.296	0.252	0.267	0.297
Parrot Spider 0.75 0.80	0.306	0.309	0.298	0.305	0.271
Parrot Spider 0.75 0.75	0.302	0.305	0.298	0.311	0.301

Таблица 10: Recall

Название модели	group	order	and/or	IUEN	keywords
baseline	0.193	0.249	0.997	0.026	0.229
Perephrase SpiderReal	0.286	0.316	1.000	0.103	0.306
Perephrase SpiderReal 3	0.297	0.367	0.997	0.231	0.382
SpiderRealT5 SpiderReal 7	0.476	0.578	0.998	0.474	0.613
SpiderRealT5 SpiderReal 5	0.461	0.511	0.995	0.513	0.558
SpiderRealT5 SpiderReal 3	0.390	0.426	0.997	0.410	0.490
Parrot Spider 0.90 0.85	0.290	0.308	0.998	0.077	0.326
Parrot Spider 0.90 0.75	0.257	0.295	0.999	0.090	0.283
Parrot Spider 0.85 0.85	0.275	0.283	1.000	0.128	0.288
Parrot Spider 0.75 0.80	0.257	0.295	0.999	0.090	0.304
Parrot Spider 0.75 0.75	0.294	0.321	0.998	0.051	0.313

Таблица 11: F1

Название модели	select	select(no AGG)	where	where(no OP)	group(no Having)
baseline	0.313	0.393	0.968	0.048	0.367
Perephrase SpiderReal	0.439	0.467	0.972	0.174	0.462
Perephrase SpiderReal 3	0.446	0.519	0.975	0.360	0.538
SpiderRealT5 SpiderReal 7	0.626	0.714	0.990	0.592	0.744
SpiderRealT5 SpiderReal 5	0.608	0.659	0.983	0.672	0.699
SpiderRealT5 SpiderReal 3	0.547	0.582	0.980	0.557	0.645
Parrot Spider 0.90 0.85	0.437	0.458	0.973	0.136	0.483
Parrot Spider 0.90 0.75	0.398	0.443	0.969	0.161	0.436
Parrot Spider 0.85 0.85	0.420	0.427	0.970	0.215	0.439
Parrot Spider 0.75 0.80	0.397	0.438	0.972	0.152	0.457
Parrot Spider 0.75 0.75	0.446	0.476	0.971	0.094	0.473

Таблица 12: F1

Название модели	group	order	and/or	IUEN	keywords
baseline	0.313	0.393	0.968	0.048	0.367
Perephrase SpiderReal	0.439	0.467	0.972	0.174	0.462
Perephrase SpiderReal 3	0.446	0.519	0.975	0.360	0.538
SpiderRealT5 SpiderReal 7	0.626	0.714	0.990	0.592	0.744
SpiderRealT5 SpiderReal 5	0.608	0.659	0.983	0.672	0.699
SpiderRealT5 SpiderReal 3	0.547	0.582	0.980	0.557	0.645
Parrot Spider 0.90 0.85	0.437	0.458	0.973	0.136	0.483
Parrot Spider 0.90 0.75	0.398	0.443	0.969	0.161	0.436
Parrot Spider 0.85 0.85	0.420	0.427	0.970	0.215	0.439
Parrot Spider 0.75 0.80	0.397	0.438	0.972	0.152	0.457
Parrot Spider 0.75 0.75	0.446	0.476	0.971	0.094	0.473

По полученным результатам, можно заметить, что почти все метрики увеличились в несколько раз. Исключения составляют значения, близкие к единице в исходной модели.

Можно сделать вывод, что каждый метод позволил моделям лучше разбираться в запросах.

4.3.1. Анализ ошибок

Результаты все еще не идеальны и результата в 55% говорит, что такой моделью не очень пользоваться в жизни. Посмотрим внимательно на примеры ошибок, чтобы понять, в чем именно проблема. Для этого вручную рассмотрим все различия в запросах, которые были рассчитаны, как ошибочные. Так как их несколько сотен, то все они не будут отображены в данной работе. Для начала рассмотрим самые простые запросы:

```
easy pred:                  easy gold:
SELECT                      SELECT
  DISTINCT country            DISTINCT country
FROM artist                 FROM singer
WHERE age > 20              WHERE age  >  20
```

Из-за незнания схемы базы данных модель предлагает синонимичное название колонки. Для решения этой проблемы можно использовать готовый код (например, PICART[19]) или при падении запроса получить предложения по исправлению названия искомого атрибута. Таких ошибок много, поэтому полученные метрики такие низкие, однако это не означает, что данная модель не может помочь в жизни.

```
medium pred:                              medium gold:
SELECT                                    SELECT
  T2.Location,                              LOCATION,
  T2.Name                                   name
FROM stadium AS T1                        FROM stadium
  JOIN stadium_capacity AS T2             WHERE
  ON T1.Stadium_ID = T2.Stadium_ID          capacity BETWEEN 5000 AND 10000
WHERE
    capacity BETWEEN 5000 AND 10000
```

В данном случае ошибка более критичная, ведь моедль усложнила запрос, попытавшись объединить несколько таблиц. Это тоже объясняется отсутствием знаний схемы. Для решения данной проблемы достаточно использовать любой метод, который учитывает в себе данные о базе данных. Например, RATSQL[12], который представляет из себя дополнение механизма внимания на схему к T5 модели.

```
extra pred:
SELECT
    T1.name,
    T2.capacity
FROM stadium AS T1
     JOIN concert AS T2
     ON T1.id = T2.stadium_id
WHERE T2.year >= 2014
GROUP BY T2.stadium_id
ORDER BY count(*) DESC
LIMIT 1
```

```
extra gold:
SELECT
  T2.name ,
  T2.capacity
FROM concert AS T1
  JOIN stadium AS T2
  ON T1.stadium_id  =  T2.stadium_id
WHERE T1.year  >=  2014
GROUP BY T2.stadium_id
ORDER BY count(*) DESC
LIMIT 1
```

В данном запросе можно увидеть, что ошибочно определена принадлежность колонки name и название колонки для объединения таблиц. Данную ошибку так же решают методы, указанные ранее.

4.4. Обзор лучших решений

Рассмотрим подробнее модели из таблицы 7, которые на данный момент являются лучшими для решения данной задачи. Для метрики execution accuracy первое место занимает DIN-SQL + GPT-4[20] с результатом в 85.3%. Предположительно GPT-4 состоит из 100 триллионов параметров, но даже такая большая модель генерирует неверные запросы в 15% случаев. Далее в топе идут модели, которые используют T5-3B или другие аналогичные модели и добавляют дополнительные механизмы для распознавая названий таблиц и колонок. Размеры таких моделей не менее 20 гигабайт и более 3 миллиарда параметров, что в разы превосходит используемую T5-base, размер которой 800 мегабайт и 223 миллиона параметров. Даже такое преимущество в количество параметров не позволяет моделям превзойти 80 процентов по точности исполнения.

Для метрики Exact match все аналогично, но лучший результат не превосходит 74%.

Rank	Model	Test
1 Apr 21, 2023	DIN-SQL + GPT-4 *University of Alberta* (Pourreza et al.,'2023) code	**85.3**
2 Jun 1, 2023	C3 + ChatGPT *Anonymous*	**82.3**
3 Feb 7, 2023	RESDSQL-3B + NatSQL (DB content used) *Renmin University of China* (Li et al., AAAI'23) code	**79.9**
4 Nov 21, 2022	SeaD + PQL (DB content used) *Anonymous*	78.5
5 Apr 21, 2023	DIN-SQL + CodeX *University of Alberta* (Pourreza et al.,'2023) code	**78.2**
6 Sep 14, 2022	CatSQL + GraPPa (DB content used) *Anonymous*	78.0
7 Sep 13, 2022	Graphix-3B+PICARD (DB content used) *Alibaba DAMO & HKU STAR & SIAT* (Li et al., AAAI'2023) code	77.6
8 Sep 1, 2022	SHiP+PICARD (DB content used) *AWS AI Labs* (Zhao et al.,'22)	76.6
9 Dec 15, 2022	N-best List Rerankers + PICARD (DB content used) *Alexa AI* (Zeng et al., IEEE SLT 2023)	75.9
10 Jun 4, 2022	RASAT+PICARD (DB content used) *SJTU LUMIA & Netmind.AI* (Qi et al., EMNLP'22) code	75.5

(a) Execution accuracy

Rank	Model	Dev	Test
1 Sep 13, 2022	Graphix-3B + PICARD (DB content used) *Alibaba DAMO & HKU STAR & SIAT* (Li et al., AAAI'2023) code	77.1	**74.0**
1 Sep 14, 2022	CatSQL + GraPPa (DB content used) *Anonymous*	78.6	73.9
3 Sep 1, 2022	SHiP + PICARD (DB content used) *AWS AI Labs* (Zhao et al.,'22)	77.2	73.1
4 May 23, 2022	G³R + LGESQL + ELECTRA (DB content used) *Anonymous*	78.1	72.9
6 Aug 12, 2022	RESDSQL+T5-1.1-lm100k-xl (DB content used) *Anonymous*	78.1	72.4
6 May 8, 2022	T5-SR (DB content used) *Anonymous*	77.2	72.4
7 Dec 15, 2022	N-best List Rerankers + PICARD (DB content used) *Alexa AI* (Zeng et al., IEEE SLT 2023)	76.4	72.2
8 Sep 1, 2021	S²SQL + ELECTRA (DB content used) *Alibaba DAMO* (Hui et al., ACL-Findings '22) code	76.4	72.1
9 Feb 7, 2023	RESDSQL-3B + NatSQL (DB content used) *Renmin University of China* (Li et al., AAAI'23) code	80.5	72.0
10 Jun 1, 2021	LGESQL + ELECTRA (DB content used) *SJTU X-LANCE Lab & AISpeech* (Cao et al., ACL'21) code	75.1	72.0

(b) Exact match

Рис. 7: Лучшие решения

Так как на данный момент не существует моделей, способных идеально решать задачу преобразования естественного языка в SQL запросы, то выгоднее всего использовать методы, которые применялись в данной работе. По существующей истории запросов можно генерировать датасет, размечая его с помощью аналогичных моделей, а затем дообучать легкие модели. Для обучения лучших решений потребуется много времени и вычислительных ресурсов, в то время как легких модели и предложенные методы аугментации помогают получить хороший результат за меньшее время.

Заключение

В результате данной работы было проверено несколько способов аугментации данных, а именно, генерация описания с помощью обученных на обратной задаче моделей и с помощью готовых моделей для перефразирования текста. Предобученная T5 модель была дообучена на сгенерированных данных. Результаты тестирования этих моделей превосходят в несколько раз аналогичные результаты для той же модели без аугментации. Лучший результат показали модели, обученные на генерацию описания на естественном языке к SQL запросам.

Полученный результат составил 53% по точности исполнения и 53% по совпадению, что превосходит многие модели, решающие ту же самую задачу более сложными методами. Лучшие результаты достигаются моделями, имеющими миллиарды и даже триллионы параметров, при этом точность таких моделей не превосходит 85%. Учитывая отсутствие моделей, идеально решающих данную задачу, наличие легковесных альтернатив, которые всегда можно переобучить под свою область за малое время, позволит использовать их для любой базы данных.

Все части запроса стали предсказываться с большей точность. Одним из самых важных улучшений в точности можно считать значительный прирост метрик для запросов, содержащих объединение или пересечение нескольких. Ранее модель хуже справлялась с данной задачей.

Основной причиной потери точности является отсутствие у модели знаний о схеме базы данных, что порождает ошибки с неверно указанными атрибутами в запросах и усложнения вида запроса в некоторых случаях. Данные проблемы решаются с помощью алгоритмов, поэтому полученные модели можно использовать.

Список литературы

1. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task / T. Yu [и др.] // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. — Brussels, Belgium : Association for Computational Linguistics, 10-11.2018. — С. 3911—3921. — DOI: 10.18653/v1/D18-1425. — URL: https://aclanthology.org/D18-1425.
2. *Hemphill C. T.*, *Godfrey J. J.*, *Doddington G. R.* The ATIS Spoken Language Systems Pilot Corpus // Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990. — 1990. — URL: https://aclanthology.org/H90-1021.
3. *Asghar N.* Yelp Dataset Challenge: Review Rating Prediction. — 2016. — arXiv: 1605.05362 [cs.CL].
4. *Zhong V.*, *Xiong C.*, *Socher R.* Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. — 2017. — arXiv: 1709.00103 [cs.CL].
5. CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases / T. Yu [и др.] // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). — Hong Kong, China : Association for Computational Linguistics, 11.2019. — С. 1962—1979. — DOI: 10.18653/v1/D19-1204. — URL: https://aclanthology.org/D19-1204.
6. SParC: Cross-Domain Semantic Parsing in Context / T. Yu [и др.]. — 2019. — arXiv: 1906.02285 [cs.CL].
7. *Deng N.*, *Chen Y.*, *Zhang Y.* Recent Advances in Text-to-SQL: A Survey of What We Have and What We Expect // Proceedings of the 29th International Conference on Computational Linguistics. — Gyeongju, Republic of Korea : International Committee on Computational Linguistics, 10.2022. — С. 2166—2187. — URL: https://aclanthology.org/2022.coling-1.190.
8. *Ganitkevitch J.*, *Van Durme B.*, *Callison-Burch C.* PPDB: The Paraphrase Database // Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. — Atlanta, Georgia : Association for Computational Linguistics, 06.2013. — С. 758—764. — URL: https://aclanthology.org/N13-1092.

9. Data Augmentation with Hierarchical SQL-to-Question Generation for Cross-domain Text-to-SQL Parsing / K. Wu [и др.] // Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. — Online, Punta Cana, Dominican Republic : Association for Computational Linguistics, 11.2021. — С. 8974—8983. — DOI: 10.18653/v1/2021.emnlp-main.707. — URL: https://aclanthology.org/2021.emnlp-main.707.
10. TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation / T. Yu [и др.]. — 2018. — arXiv: 1804.09769 [cs.CL].
11. SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL / R. Cai [и др.]. — 2022. — arXiv: 2111.00653 [cs.CL].
12. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers / B. Wang [и др.]. — 2021. — arXiv: 1911.04942 [cs.CL].
13. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing / T. Yu [и др.]. — 2021. — arXiv: 2009.13845 [cs.CL].
14. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-DomainText-to-SQL Task / T. Yu [и др.]. — 2018. — arXiv: 1810.05237 [cs.CL].
15. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation / J. Guo [и др.] // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — Florence, Italy : Association for Computational Linguistics, 07.2019. — С. 4524—4535. — DOI: 10.18653/v1/P19-1444. — URL: https://aclanthology.org/P19-1444.
16. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer / C. Raffel [и др.]. — 2020. — arXiv: 1910.10683 [cs.LG].
17. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension / M. Lewis [и др.]. — 2019. — arXiv: 1910.13461 [cs.CL].
18. *Damodaran P.* Parrot: Paraphrase generation for NLU. — Вер. v1.0. — 2021.
19. *Scholak T.*, *Schucher N.*, *Bahdanau D.* PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. — 2021. — arXiv: 2109.05093 [cs.CL].
20. *Pourreza M.*, *Rafiei D.* DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. — 2023. — arXiv: 2304.11015 [cs.CL].