Научная статья

УДК 004.891
DOI 10.25205/1818-7900-2024-22-3-28-39

Дообучение модели CodeBERT для написания комментариев к SQL-запросам

Данила Александрович Комлев

НИТУ МИСИС, Москва, Россия

komlevdanila742@gmail.com

Аннотация

Автоматизированное создание комментариев к исходному коду – актуальная тема в разработке программного обеспечения, где модели машинного перевода применяются для «перевода» кода в текстовые описания. Предобученная на шести языках программирования модель CodeBERT используется для поиска кода, генерации документации, исправления ошибок. Эта модель хорошо понимает семантики естественного языка, языков программировани, а также связи между ними, эта модель хорошо подходит для дообучения на различные прикладные задачи, связанные с кодом. В статье рассматривается дообучение модели CodeBERT для генерации комментариев к SQL-запросам. Эта задача является актуальной, так как в крупных проектах может использоваться множество SQL-запросов различной сложности, и комментарии помогают улучшить их читаемость и понимание. Однако ручное написание и поддержание актуальности комментариев требует времени и усилий разработчиков. В статье предложено использовать предобученную модель CodeBERT для автоматической генерации комментариев к SQL-коду, что сократит время и позволит поддерживать комментарии в актуальном состоянии. Для дообучения используются открытые датасеты, содержание SQL-запрос, а также комментарий к нему. Результаты тестирования показали, что дообученная модель успешно справляется с задачей создания комментариев к SQL-запросам, что также подтверждается полученными значениями метрики Bleu.

Ключевые слова

SQL-запрос, CodeBERT, Transformers, NLP, Bleu, дообучение, генерация комментариев

Для цитирования

Комлев Д. А. Дообучение модели CodeBERT для написания комментариев к SQL-запросам // Вестник НГУ. Серия: Информационные технологии. 2024. Т. 22, № 3. С. 28–39. DOI 10.25205/1818-7900-2024-22-3-28-39

Further Training of the CodeBERT Model for Writing Comments on SQL Queries

Danila A. Komlev

NUST MISIS, Moscow, Russian Federation

komlevdanila742@gmail.com

Abstract

Automated creation of comments to the source code is an urgent topic in software development, where machine translation models are used to “translate” code into text descriptions. The CodeBERT model, pre-trained in six programming languages, is used to search for code, generate documentation, and correct errors. This model understands well the semantics of natural language, programming languages, as well as the connections between them, this model is well suited for additional training on various applied tasks related to code. The article discusses the further training of the

CodeBERT model for generating comments on SQL queries. This task is relevant, since large projects can use many SQL queries of varying complexity, and comments help to improve their readability and understanding. However, manually writing and keeping comments up-to-date takes time and effort from developers. The article suggests using the pre-trained CodeBERT model to automatically generate comments on SQL code, which will reduce time and allow you to keep comments up to date. For further training, open datasets, the contents of the SQL query, as well as comments on it are used. The test results showed that the pre-trained model successfully copes with the task of creating comments to an SQL query, which is also confirmed by the obtained values of the Bleu metric.

Keywords

SQL query, CodeBERT, Transformers, NLP, Bleu, additional training, comment generation

For citation

Komlev D. A. Further training of the CodeBERT model for writing comments on SQL queries. Vestnik NSU. Series: *Information Technologies*, 2024, vol. 22, no. 3, pp. 28–39 (in Russ.) DOI 10.25205/1818-7900-2024-22-3-28-39

Введение

В большинстве программных продуктов для персистентного хранения данных используются реляционные базы данных, например MS SQL, Postgresql, MySQL.

Для манипуляции данными в реляционных базах данных применяется SQL (Structured Query Language).

SQL-запросы бывают двух видов: DDL и DML:

1. DDL (Data Definition Language) используется для определения структуры и организации данных в базе данных. Это включает в себя создание, изменение и удаление объектов базы данных, таких как таблицы, индексы, представления, функции и т. д.

2. DML (Data Manipulation Language) используется для манипулирования данными внутри таблиц базы данных. Это включает в себя операции вставки (INSERT), обновления (UPDATE), удаления (DELETE) и выборки (SELECT) данных.

Например, для того чтобы получить Id, Username и Email для пользователей, зарегистрировавшихся, начиная с 2023-01-01, используется такой запрос:

```
SELECT UserID, Username, Email
FROM Users
WHERE RegistrationDate >= ‘2023-01-01’;
```

В больших программных продуктах может использоваться огромное число SQL-запросов разной сложности.

Для облегчения работы с этими запросами разработчики часто добавляют комментарии, чтобы улучшить читаемость и понимание кода. Однако написание комментариев к SQL-коду является рутинной задачей, которая может занимать значительное время разработчиков и аналитиков данных. Более того, часто возникают сложности с поддержанием актуальности комментариев: при изменении SQL-запроса комментарий к нему может остаться необновленным, что создает путаницу и затрудняет понимание кода.

В свете этих проблем актуальной задачей становится автоматизация процесса написания комментариев к SQL-коду с применением нейронных сетей. Это не только поможет сэкономить рабочее время программистов, но и обеспечит актуальность и консистентность комментариев, что приведет к улучшению процесса разработки и поддержки программного обеспечения.

Обзор модели CodeBERT

В данной работе будет использована предобученная модель CodeBERT. Эта модель разработана исследовательской командой в Microsoft. Модель специализируется на NLP-задачах, связанных с программным кодом:

- Code-To-Code – дополнение кода или перевод с одного языка программирования на другой;
- Code-To-Text – написание комментариев к коду;
- Text-To-Code – поиск кода;
- Text-To-Text − машинный перевод, генерация текста.

CodeBERT обучена на шести языках программирования: Python, Java, JavaScript, PHP, Ruby, Go и хорошо понимает как семантику программного кода, так и семантику человеческого языка. Суммарный объем данных, на которых обучалась модель, – более миллиона записей.

На языке SQL модель CodeBERT не обучалась. Также отличительной особенностью языка SQL является то, что SQL – не язык программирования, а язык запросов. А также, все языки, на которых обучалась модель, являются императивными, т. е. содержат прямые указания, что должна делать программа, а язык SQL является декларативным [1], т. е. с помощью языка описывается ожидаемый результат, а не способ его получения. Другими примерами декларативных языков являются HTML и XML.

Рассмотрим архитектуру CodeBERT. BERT (Bidirectional Encoder Representations from Transformers) – это модель глубокого обучения, разработанная на основе трансформерной архитектуры, которая способна эффективно анализировать контекст текстовых данных. Ее особенность заключается в том, что она обучается на больших объемах текста с использованием двунаправленных кодировщиков, что позволяет модели учитывать контекст как слева, так и справа от текущего слова или фразы. Это позволяет BERT создавать глубокие контекстуальные представления слов и фраз, что делает ее одной из наиболее эффективных моделей для широкого спектра задач обработки естественного языка.

Механизм внимания (attention) является ключевой особенностью архитектуры Transformers. При каждом новом предсказании трансформер оценивает контекст, фокусируя внимание на тех словах, которые являются наиболее значимыми для текущего шага. Эта оценка осуществляется путем вычисления весов внимания для каждого слова, отражающих его относительную важность.

Внимание позволяет модели определять вес для каждой позиции, сопоставляя запросы с ключами всех слов во входной последовательности. После нормализации весов с помощью функции softmax значения взвешиваются для создания контекстуально релевантных представлений.

Этот механизм заменил традиционные рекуррентные нейронные сети (RNN) благодаря своей способности параллельно обрабатывать длинные последовательности. RNN из-за рекурсивной природы часто испытывают трудности с захватом долгосрочных зависимостей. В отличие от них трансформеры, благодаря механизму внимания, эффективно анализируют и ближайшие, и отдаленные связи.

По сравнению с RNN трансформеры обрабатывают информацию быстрее, так как позволяют каждой позиции в последовательности напрямую взаимодействовать с любой другой позицией. Это решает проблему обучения в параллельных вычислениях, что особенно критично при работе с большими наборами данных. Трансформеры способны учитывать сложные взаимосвязи между словами независимо от их относительной позиции, что позволяет им успешно выполнять такие задачи, как машинный перевод, генерация текста и многие другие задачи обработки естественного языка.

У моделей BERT есть два этапа разработки:

- предобучение (pre-training) – обучение модели на большом объеме неразмеченных данных, которое помогает модели понять языковые закономерности и создать общее представление о языке;
- дообучение (fine-tuning) – донастройка модели для решения конкретной задачи. Для дообучения требуется значительно меньше данных, чем для предобучения.

В данной работе рассмотрен процесс дообучения предобученной модели CodeBERT для решения задачи написания комментариев к SQL-коду. Модель CodeBERT понимает семантики как языков программирования, так и семантики естественного языка, а также их взаимосвязи. Модель ничего не знает про SQL, так как в процессе предоучения SQL не использовался. Однако общие знания о семантике языков программирования позволят модели с помощью относительно небольшого датасета научиться писать комментарии к SQL.

CodeBERT, подобно BERT, использует архитектуру Transformer [2]. Однако в отличие от BERT, который обучается на текстах естественного языка, CodeBERT обучается на текстах программного кода и сопутствующих текстовых описаниях. Это позволяет модели эффективно работать с бимодальными данными и улавливать связи между кодом и его описанием.

Архитектура CodeBERT включает в себя два важных метода обучения: Masked Language Modeling (MLM) [3] и ReplacedToken Detection (RTD):

1) Masked Language Modeling. MLM – это техника обучения языковых моделей, применяемая в NLP. Идея заключается в том, что часть токенов во входных данных случайным образом маскируется, а затем модель обучается прогнозировать эти замаскированные токены на основании контекста, это используется в процессе предобучения модели. MLM является обучением без учителя (self-supervised learning), так как никакой предварительной разметки данных не требуется. Наглядный пример того, как работает MLM, представлен на рис. 1;

2) Replaced Token Detection. RTD [4], так же как и MLM, является техникой, используемой в процессе предобучения языковой модели. Идея заключается в том, что во входной последовательности токенов часть токенов заменяется. Но не на [MASK], как в MLM, а на другой токен, относительно близкий по смыслу, но неверный. Например, токен «окно» может быть заменен на токен «стекло». Между этими токенами есть связь, но они означают разное. В процессе обучения модель должна научиться определять токены, которые были заменены, это улучшает понимание моделью контекста, а также семантики предложения.

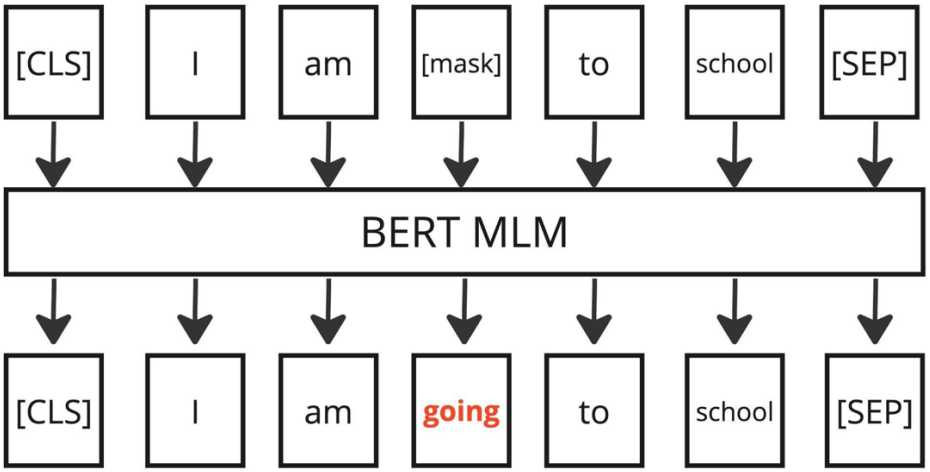

Рис. 1. Пример работы MLM
Fig. 1. An example of MLM operation

Токены замены генерируются, как правило, небольшой MLM-моделью, которая выдает близкий к оригинальному по смыслу токен. Пример работы RTD можно увидеть на рис. 2. Генератор с помощью MLM заменяет [MASK] на токены, близкие или равные исходным, а дискриминатор определяет, был ли исходный токен заменен.

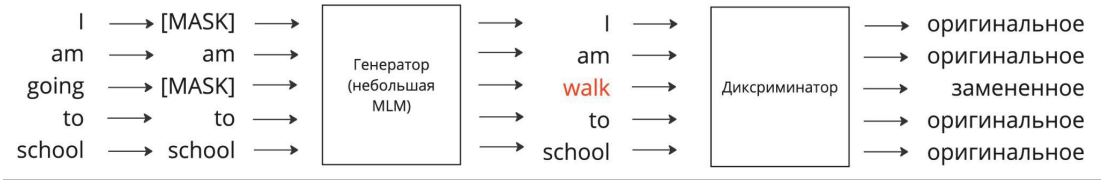

Рис. 2. Пример работы RTD
Fig. 2. An example of RTD operation

Как было отмечено ранее, CodeBERT является бимодальной моделью, так как работает с естественным языком и программным кодом. В связи с этим в RTD у CodeBERT используется не один генератор, как в базовой модели BERT, а два: для естественного языка и программного кода.

Оценка BLEU (BiLingual Evaluation Understudy) является метрикой в машинном переводе, используемой для оценки его качества [5]. Она представляет собой число в диапазоне от 0 до 1, где 0 соответствует низкому качеству перевода, а 1 – высокому.

Алгоритм BLEU основан на сравнении n-грамм [6] (обычно от 1 до 4) в сгенерированном тексте с эталонным текстом. Например, для фразы «я сегодня купил хлеб» и $n = 2$, имеем три биграммы:

1. «я сегодня»
2. «сегодня купил»
3. «купил хлеб»

Затем вычисляется пропорция n-грамм эталонного текста, которые присутствуют в сгенерированном тексте. Для расчета BLEU могут использоваться несколько наборов n-грамм, и итоговая оценка формируется как их взвешенная сумма.

BLEU учитывает не только совпадение слов, но и их количество и порядок. Это позволяет более точно оценивать качество перевода, учитывая контекст и семантику предложений.

Также BLEU штрафует короткие предложения по формуле 1:

$$\text{Brevity Penalty} = \begin{cases} 1, & if\ c > r \\ e^{(1-\frac{r}{c})}, & if\ c \le r \end{cases}. \quad (1)$$

Это нужно для того, чтобы короткие предложения, не имеющие смысла, например, состоящие из одного слова, не могли получить высокую оценку.

BLEU может быть представлена формулой (2):

$$\text{BLEU} = \text{BP} * \exp\left(\sum_{1}^{N} w_n * \log\left(p_n\right)\right), \quad (2)$$

где p_n – доля n-грамм эталонного текста, которые присутствуют в сгенерированном, а w_n – вес каждой n-граммы, как правило, $\frac{1}{N}$.

Для наглядности рассмотрим расчет метрики на примере:

Сгенерированное предложение: *A black cat is sleeping on the sofa*

Эталонное предложение: *The black cat is sleeping on the couch*

Рассчитаем n-граммы:

$$P_1 = \frac{6}{8} = \frac{3}{4}$$

$$P_2 = \frac{5}{7}$$

$$P_3 = \frac{4}{6} = \frac{2}{3}$$

$$P_4 = \frac{3}{5}$$

$$\text{BLEU} = \exp(\ln\left(\frac{3}{4}\right) + \frac{1}{2} * \ln\left(\frac{5}{7}\right) + \frac{1}{3} * \ln\left(\frac{2}{3}\right) + \frac{1}{4} * \ln\left(\frac{3}{5}\right) \approx 0,49 .$$

Однако стоит заметить, что в разных реализациях могут использоваться разные основания логарифма.

Предлагаемое решение

Для дообучения модели необходим датасет, содержащий в себе 2 колонки:

- SQL-запрос;
- комментарий к SQL-запросу на естественном языке.

Было найдено два открытых датасета:

1) https://huggingface.co/datasets/b-mc2/sql-create-context – датасет, содержащий сопоставление SQL-кода и человекопонятного описания этого кода;
2) https://huggingface.co/datasets/iamtarun/code_instructions_120k_alpaca – датасет сопоставления программного кода и описание этого кода на естественном языке. Код представлен на разных языках, в том числе на SQL.

Оба датасета хорошо подходят под нашу задачу, однако также в них есть некоторые особенности, которые негативно повлияют на дообучение модели, а именно:

1. Разные комментарии написаны в разном формате. Для близких по смыслу SQL-запросов могут быть совсем разные комментарии.

2. Встречаются строки, в которых написан не только SQL-запрос, но и код на каком-либо языке программирования. Как правило, это строки, в которых SQL-запрос являются частью программного кода.

3. В данных упоминаются разные диалекты SQL, например, MS SQL, Postgresql, MySql. В данной задаче нам не важны различия диалектов.

4. Встречаются строки в вопросной форме.

Для решения этих проблем была проведена предобработка данных, в рамках которой были выполнены следующие действия:

- были удалены все записи, в столбце комментария которых упоминаются языки программирования, фреймворки, а также другие, специфичные для программирования, термины, например: *api, endpoint, class*;
- были удалены все записи с комментарием в вопросной форме. Это те записи, которые заканчиваются ? и содержат одно из следующих слов: *what, where, which, who, how many, when;*
- комментарии были приведены к единой форме. Например, фразы: *dispay the, compute the, identify the, get the* были заменены на *find the*, так как *find the* чаще всего встречается в комментариях к запросам типа SELECT;
- из обоих столбцов были удалены управляющие последовательности, такие как \n и \t, обозначающие перенос строки и табуляцию;
- были удалены записи, содержащие слишком короткий или длинный комментарий или SQL-запрос;

- также весь текст был переведен в нижний регистр, так как SQL-запросы являются регистронезависимымы (кроме строковых литералов).

На рис. 3 представлены несколько записей из итогового датасета.

В финальном датасете 17323 записи. Из них по 1500 отводится на тестовую выборку и валидационную. Остальные записи (15323) отводятся на обучение модели.

В первом датасете присутствует колонка `context`, содержащая DDL-выражения, используемые для создания таблиц, которые фигурируют в SQL-запросах (колонка `answer`). Эта информация может быть полезна для генерации комментариев, однако текущая архитектура модели не предусматривает ее использования, модель принимает на вход только sql-запрос, для которого необходимо сгенерировать комментарий. Также во втором, более крупном датасете, информация о DDL-выражениях отсутствует.

С помощью датасета, описанного выше, будем дообучать модель.

Результаты

Для дообучения будет использоваться репозиторий CodeXGLUE [7], созданный Microsoft. Репозиторий предоставляет удобный интерфейс для взаимодействия с предобученной моделью CodeBERT, а также возможность дообучать модель под разные типы задач, в нашем случае «Code To Text» (генерация текстовых описаний из кода).

Модель Seq2Seq. Для этой цели мы используем архитектуру Seq2Seq, которая объединяет предобученный энкодер CodeBERT и новый декодер Transformer:

1. Энкодер (CodeBERT). Энкодер из CodeBERT служит основой, поскольку он уже предобучен на большом наборе данных и обладает глубокой способностью распознавать паттерны в коде. Этот этап подготовки позволяет эффективно извлекать ключевые признаки и семантическую информацию из входного кода. В процессе дообучения энкодер фокусируется на адаптации к более специализированным аспектам, необходимым для задачи «Code to Text».

	question	answer
0	find the most expensive product from the table...	select * from products order by price desc lim...
1	find all records from a table ordered by date ...	select * from table order by date asc;
3	create a stored procedure in insert new data i...	create procedure insert_user(in first_name var...
6	find the highest price for each product type.	select type, max(price) as 'highest_price'from...
8	sql query to find the top 5 highest vote counts	select name, vote from votes order by vote des...
...	...	...
20788	find the date for opponent in the final being ...	select date from table_name_5 where opponent_i...
20789	find the tournament for grass surface and oppo...	select tournament from table_name_11 where sur...
20790	find the fcc info on the radio frequency mhz 1...	select fcc_info from table_name_64 where frequ...
20791	find the mhz frequency of allapattah, florida.	select frequency_mhz from table_name_28 where ...
20792	find the attendance with date of june 11	select attendance from table_name_92 where dat...

Рис. 3. Датасет
Fig. 3. Dataset

2. Декодер. Для генерации текстовых описаний из кодовых фрагментов создается новый декодер на основе Transformer. Декодер содержит несколько слоев, чтобы эффективно обрабатывать сложные зависимости в последовательностях. В отличие от энкодера декодер начинается с нуля и будет обучен на специализированном датасете для конкретной задачи. В ходе тренировки он научится создавать детальные текстовые описания на основе выходных данных энкодера.

В качестве функции потерь используется кросс-энтропия [8], она измеряет расхождения между предсказанными вероятностями модели и истинными значениями. Значение кросс-энтропии считается по формуле:

$$H(p,q) = -\sum_{x \epsilon X} p(x) * \log\left(q(x)\right), \tag{3}$$

где $p(x)$ – истинная вероятность значения x, $q(x)$ – предсказанная вероятность значения x (0 или 1).

Функция потерь учитывает разницу между этими двумя вероятностями, где большие расхождения приводят к более высоким значениям кросс-энтропии. Если модель делает точные предсказания, вероятность $q(x)$ будет близка к истинной вероятности $p(x)$, что приводит к низкому значению кросс-энтропии. Таким образом, чем более уверена модель в правильном предсказании, тем меньше значение кросс-энтропии.

В контексте задачи данной статьи значение кросс-энтропии зависит от того, насколько точно модель предсказывает следующий токен в последовательности. Поскольку модель Seq2Seq работает с последовательными данными, каждый токен, предсказанный декодером, сравнивается с истинным следующим токеном в предложении.

В процессе генерации комментариев используется эвристический алгоритм Beam Search [9]. Этот алгоритм помогает эффективно формировать последовательности токенов для комментариев, выбирая на каждом шаге некоторое, заранее заданное количество наиболее вероятных токенов. Beam Search комбинирует элементы поиска в ширину и поиска в глубину, обеспечивая баланс между качеством и эффективностью предсказаний.

Для работы алгоритма задается параметр «ширина луча» (beam width), который определяет количество узлов, учитываемых на каждом шаге. При небольшом значении ширины луча алгоритм напоминает жадный поиск, быстро находя оптимальное решение, но с возможными компромиссами в точности и осмысленности генерируемых последовательностей. Увеличение ширины луча позволяет рассматривать большее число узлов, что повышает вероятность получения более релевантных токенов, но также увеличивает вычислительную нагрузку и время выполнения алгоритма.

Beam Search выполняет следующие основные шаги.

1. Инициализация: начинается с начального состояния, представляющего собой начальный токен (например, Start Of Sequence, SOS).

2. Расширение узлов: на каждом шаге алгоритм генерирует все возможные продолжения текущих последовательностей токенов.

3. Оценка вероятностей: каждое продолжение оценивается по вероятности с учетом предыдущих токенов и состояния модели.

4. Отбор лучших гипотез: из всех возможных продолжений отбирается фиксированное количество наиболее вероятных (beam width).

5. Проверка завершения: процесс повторяется, пока не достигнут конечный токен (End Of Sequence, EOS) или максимальная длина последовательности.

Пример работы алгоритма изображен на рис. 4

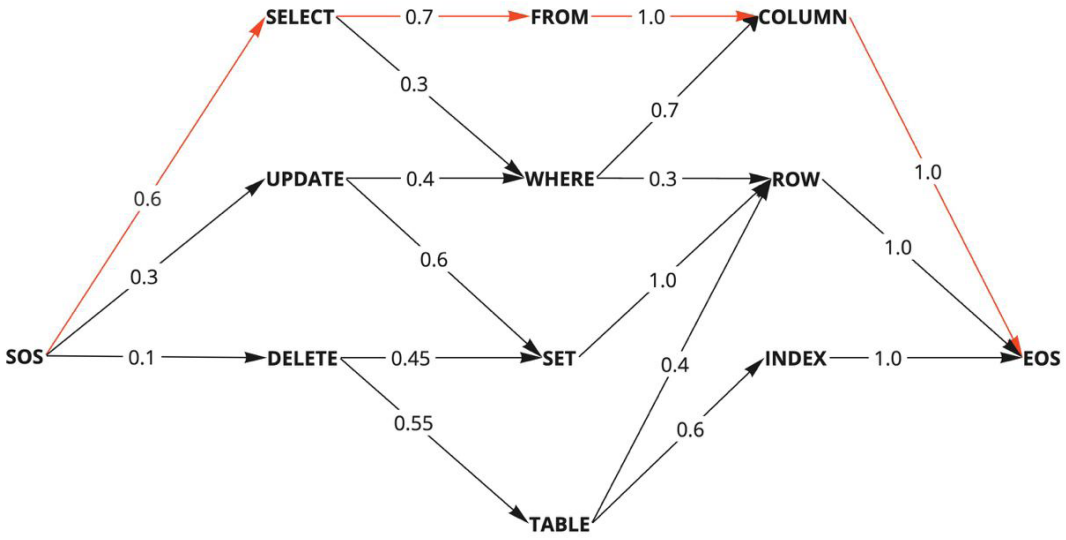

Рис. 4. Алгоритм Beam Search
Fig. 4. Beam Search Algorithm

Процесс дообучения, использующий модель Seq2Seq [10]:

- инициализируем предобученный энкодер и новый декодер с конфигурацией, включающей параметры для Beam Search;
- подготавливаем датасет, разделяя его на тренировочный и валидационный наборы, и настраиваем параметры оптимизатора и планировщика обучения;
- при тренировке модель обрабатывает каждый батч данных, рассчитывает потери и оптимизирует свои параметры посредством обратного распространения;
- модель дообучается на семи эпохах;
- после каждой эпохи производятся оценка и валидация, чтобы убедиться, что модель улучшает свое представление в задаче генерации текста из кода.

Выводы

После проведения дообучения модели CodeBERT для генерации комментариев к SQL-коду модель была проверена на тестовом наборе данных из 1000 записей.

Среднее значение метрики BLEU составило 0.45. Для понимания этого результата, отметим, что при значении BLEU выше 0.6 сгенерированный текст практически неотличим от текста, написанного человеком. Хотя наша модель еще не достигла этого уровня, результат 0.45 указывает на значительное улучшение качества генерации комментариев по сравнению с изначальным состоянием модели.

В таблице представлены некоторые результаты запуска модели на тестовом датасете. Из результатов видно, что модель выдает значения, хорошо описывающие SQL запрос. Даже в тех случаях, когда значение BLEU оценивается как низкое, сгенерированные комментарии все же предоставляют общее представление о содержании запроса, в некоторых случаях модель описывает sql-запрос не хуже, чем исходный комментарий, но другими словами.

Результаты генерации

Generation results

SQL-запрос	Исходный комментарий	Сгенерированный комментарий	BLEU
1	2	3	4
select count(loss) from table_name_87 where avg_g > 129.2	how much loss has an avg/g larger than 129.2	how much loss has a avg larger than 129.2	0.43
select sum(average) from table_name_99 where interview > 9.57 and swimsuit > 9.65	find the sum of average for interview more than 9.57 and swimsuit more than 9.65	find the sum of average for interview more than 9.57 and swimsuit more than 9.65	1.0
create procedure getproduct(@num1 int, @ num2 int) as begin declare @result int select @result = @num1 * @num2 return @result end	create a sql stored procedure that accepts two integer numbers and returns the product of these two numbers.	create a sql stored procedure that returns the number of items from a given table.	0.30
-- create the users tablecreate table users (id serial primary key, name varchar(255) not null);-- create the posts tablecreate table posts (id serial primary key, user_id integer references users(id), content varchar(255) not null);	create a sql database with two tables, users and posts. the users table should contain fields for id and name, while posts should contain fields for user_id and content.	create a sql database that stores information about users and posts.	0.06
select pole_position from table_1140085_2 where rnd = 3	find the pole position for the rnd 3	find the pole position with rnd being 3	0.38
select top 5 salaryfrom table order by salary desc;	select the top 5 highest salaries from a table.	find the top 5 highest salaries from a table.	0.86
update users set age = 25 where age > 25;	create a update the age field of the users whose age is greater than 25.	update the age of all users older than 25.	0.00
select count(average) from table_name_21 where wickets > 537 and career = «1899/00-1925/26» and matches < 211	find the total number of average for wickets more than 537, career of 1899/00-1925/26 and matches less than 211	find the total number of average for wickets more than 537 and career of 1819/19/26	0.52
select t1.fname from authors as t1 join authorship as t2 on t1.authid = t2.authid join papers as t3 on t2.paperid = t3.paperid where t3.title like «%functional%»	find the first names of all the authors who have written a paper with title containing the word «functional».	Find the first names of all the authors who paper contains «function»	0.44

Окончание табл.

1	2	3	4
select title, count(rating) as rating_count from moviesgroup by title order by rating_count desclimit 3;	find the top 3 most popular movies from a database of movies with their ratings.	find the top 3 movies with the highest rating.	0.18
select t3.headquartered_city from store as t1 join store_ district as t2 on t1.store_id = t2.store_id join district as t3 on t2.district_id = t3.district_id group by t3.headquartered_city order by count(*) desc limit 1	find the city with the most number of stores.	find the city with the most count of stores	0.66

Следует также отметить, что, несмотря на меньшее количество данных для SQL по сравнению с другими языками программирования (в среднем на 1 порядок), на которых изначально обучалась модель, сгенерированные комментарии оказались осмысленными и достаточно точно описывающими SQL-код. Это подчеркивает способность модели адаптироваться к новым языковым задачам и генерировать релевантные комментарии даже при ограниченном объеме обучающих данных.

Таким образом, дообучение CodeBERT для задачи генерации комментариев к SQL-коду продемонстрировало значимые результаты. Текущая дообученная модель способна генерировать комментарии к SQL-запросам относительно небольшого размера. Однако с использованием более полного и качественного датасета реализованный процесс дообучения может показать более высокие значения метрики BLEU.

Также это подтверждает потенциал модели CodeBERT для дообучения на различные прикладные задачи, связанные с программным кодом.

Список литературы / References

1. What Is Declarative Programming? URL: https://codefresh.io/learn/infrastructure-as-code/declarative-vs-imperative-programming-4-key-differences/
2. **Vaswani O., Shazeer N., Parmar N. etc.** Attention Is All You Need. URL: https://arxiv.org/abs/1706.03762
3. Masked Language Modeling in BERT. URL: https://www.scaler.com/topics/nlp/masked-language-model-explained/
4. **Clark K., Luong M.-T. D.** Manning C Electra: pre-training text encoders as discriminators rather than generators. URL: https://openreview.net/pdf?id=r1xMH1BtvB
5. Natural Language Processing: Bleu Score. URL: https://www.baeldung.com/cs/nlp-bleu-score
6. N-граммы. URL: https://deepai.org/machine-learning-glossary-and-terms/n-gram
7. CodeXGLUE. URL: https://microsoft.github.io/CodeXGLUE/
8. Cross Enptropy. URL: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
9. Beam Search. URL: https://d2l.ai/chapter_recurrent-modern/beam-search.html
10. **Zhu Qingfu, Zhang Weinan, Zhou Lianqiang, Liu Ting.** Learning to Start for Sequence to Sequence Architecture. 2016. URL: https://www.researchgate.net/publication/306357583_Learning_to_Start_for_Sequence_to_Sequence_Architecture

Сведения об авторе

Комлев Данила Александрович, студент магистратуры

Information about the Author

Danila A. Komlev, Graduate Student

Статья поступила в редакцию 23.05.2024;
одобрена после рецензирования 15.10.2024; принята к публикации 15.10.2024

The article was submitted 23.05.2024;
approved after reviewing 15.10.2024; accepted for publication 15.10.2024