УКД 004.85

От объема к качеству: подготовка кодовых корпусов для генеративных моделей

Иванова П.А., Тюрин Е.Е. (МИЭМ НИУ ВШЭ)
Научный руководитель - кандидат технических наук, профессор Авдошин С.М. (МИЭМ НИУ ВШЭ)

Введение

В современном мире интерес к генерации программного кода привёл к появлению масштабных корпусов данных для обучения моделей, включая CodeSearchNet и The Stack [1,3]. Однако на ранних этапах подготовки данных исследователи часто делали акцент на объём, уделяя недостаточно времени качеству и чистоте. Практика показала, что дублирование усиливает переобучение, а наличие тестовых примеров в обучающей выборке (contamination) может привести к завышенным результатам оценки [4,5,9]. Для новых классов моделей, включая диффузионные, особенно важны стабильная токенизация и низкий уровень шума в данных [7,8].

Основная часть

В данной работе систематизированы этапы подготовки кодовых датасетов для генеративных моделей и предложен воспроизводимый пайплайн обработки. Основа обучающей выборки формируются из пар “docstring - функция” Python-срезов CodeSearchNet и CodeXGLUE [1,2], а популярные бенчмарки HumanEval и APPS рассматриваются для оценки и финального тестирования, чтобы исключить риск утечки данных [5,9].

Мы провели предварительный анализ данных (EDA) и определили измеримые критерии качества. Для структурного анализа использованы AST-метрики, применяемые в современных подходах к генерации и пониманию кода [6]. Анализ показал, что код из CodeSearchNet и CodeXGLUE почти полностью синтаксически корректен (parsable rate близок к 100%), а глубина AST умеренная (CodeSearchNet: median 10, p99 16; CodeXGLUE: median 9, p99 15). Это позволило нам использовать структурные ограничение как «мягкий» фильтр для отсечения аномально сложных фрагментов [6].

На основе EDA был построен пайплайн очистки. Мы выявили, что один из основных источников шума в корпусах – это закомментированный код, который увеличивает длину примеров и добавляет неисполняемые фрагменты, поэтому он удаляется на этапе предобработки. Помимо этого выполняются нормализация форматирования и контроль синтаксической корректности через ast.parse. Разбиение train/val/test фиксируется, а пересечения между сплитами устраняются, что снижает риск переобучения и делает сравнение моделей корректнее [4,5,9].

Особое место в работе занимает токенизация как основа дискретного представления для генеративной модели. Мы рассмотрены две стратегии: стандартный byte-level BPE и основной вариант структурной токенизации Python с явным кодированием переноса строки <nl> и отступов <indent>, <dedent> [7]. Для снижения вариативности словаря и OOV используется режим нормализации идентификаторов [7]. Качество токенизации оценивается метриками OOV rate, truncation rate и parsable rate после обратной детокенизации. Эксперименты показывают, что структурная токенизация обеспечивает parsable rate до 100% при приемлемом уровне OOV, что важно для дискретных и диффузионных генеративных моделей [7,8].

Выводы

Сегодня фокус в задачах генерации кода смещается от простого увеличения объёма данных к контролю и обеспечению их чистоты и структурной целостности.

Предложенный нами протокол, объединяющий EDA, очистку, контроль разбиения без contamination и структурно-ориентированную токенизацию, повышает устойчивость обучения и позволяет корректнее сравнивать авторегрессионные, GAN и диффузионные модели [4,5,8,9]. В дальнейшем планируется усиление алгоритмов поиска нечетких дублей (near-dedup) и внедрение единого протокола оценки функциональной корректности (на базе HumanEval, APPS) с дополнением AST-метриками для анализа структуры сгенерированного кода [5,6,9].

Благодарности

Исследование выполнено с использованием суперкомпьютерного комплекса НИУ ВШЭ.

Литература

1. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search [Электронный ресурс]. – arXiv:1909.09436. – URL: https://arxiv.org/abs/1909.09436 (дата обращения: 07.02.2026).
2. CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation [Электронный ресурс]. – arXiv:2102.04664. – URL: https://arxiv.org/abs/2102.04664 (дата обращения: 07.02.2026).
3. The Stack: 3 TB of Permissively Licensed Source Code [Электронный ресурс]. – arXiv:2211.15533. – URL: https://arxiv.org/abs/2211.15533 (дата обращения: 07.02.2026).
4. Deduplicating Training Data Makes Language Models Better [Электронный ресурс]. – arXiv:2107.06499. – URL: https://arxiv.org/abs/2107.06499 (дата обращения: 07.02.2026).
5. Data Contamination Quiz: A Tool to Detect and Estimate Contamination in Large Language Models [Электронный ресурс]. – arXiv:2311.06233. – URL: https://arxiv.org/abs/2311.06233 (дата обращения: 07.02.2026).
6. AST-T5: Structure-Aware Pretraining for Code Generation and Understanding [Электронный ресурс]. – arXiv:2401.03003. – URL: https://arxiv.org/abs/2401.03003 (дата обращения: 07.02.2026).
7. An Empirical Study of Tokenization Strategies for Code [Электронный ресурс]. – arXiv:2010.02534. – URL: https://arxiv.org/abs/2010.02534 (дата обращения: 07.02.2026).
8. DiffuCoder: Understanding and Improving Masked Diffusion Models for Code Generation [Электронный ресурс]. – arXiv:2506.20639. – URL: https://arxiv.org/abs/2506.20639 (дата обращения: 07.02.2026).
9. On Inter-dataset Code Duplication and Data Leakage in Large Language Models [Электронный ресурс]. – arXiv:2401.07930. – URL: https://arxiv.org/abs/2401.07930 (дата обращения: 07.02.2026).