Эмпирическое исследование компромиссов между компонентами пайплайна кодогенерации на основе малой языковой модели

А. Б. Белкин, Я. А. Бекенева

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

redloin@mail.ru

***Аннотация*. В работе исследуется влияние компонентов пайплайна кодогенерации на качество кода, генерируемого малой языковой моделью Qwen2.5-Coder-1.5B в условиях ограниченных вычислительных ресурсов. Рассматриваются два компонента: множественная генерация кандидатов и итеративная коррекция по результатам тестирования. На бенчмарке HumanEval (164 задачи) показано, что множественная генерация даёт прирост pass@1 до +20 процентных пунктов, тогда как итеративная коррекция на модели данного размера малоэффективна (+3.7 п.п.) и в ряде конфигураций ухудшает результат. Измерены эффекты взаимодействия между компонентами и построены кривые компромиссов качество/время, позволяющие выбирать оптимальную конфигурацию при заданном временном бюджете. Результаты показывают, что для малых моделей стратегия множественной генерации предпочтительнее итеративной коррекции, а оптимальная конфигурация не сводится к включению всех доступных компонентов.**

Ключевые слова: кодогенерация; малые языковые модели; множественная генерация; итеративная коррекция; эффекты взаимодействия; анализ компромиссов; ресурсные ограничения

I. Введение

Системы кодогенерации на основе больших языковых моделей достигают высоких результатов на стандартных бенчмарках, но требуют мощного оборудования. Разработчики на локальных машинах с ограниченным объёмом видеопамяти и оперативной памяти вынуждены использовать малые модели с числом параметров от 1 до 7 миллиардов (1–7B), качество которых заметно ниже. При этом качество кодогенерации определяется не только размером модели, но и конфигурацией пайплайна: числом сгенерированных кандидатов, наличием итеративной коррекции, объёмом контекста из внешних источников и другими параметрами.

Каждый дополнительный компонент пайплайна повышает вычислительную стоимость, что создаёт задачу распределения ограниченного вычислительного бюджета. При фиксированном временном бюджете разработчик может сгенерировать пять независимых кандидатов или потратить те же вычисления на итеративную коррекцию одного решения. Ответ на вопрос, какая стратегия эффективнее, зависит от взаимодействия компонентов, и для малых моделей эти зависимости до сих пор не исследованы систематически.

Работа CoCoS [1] показала, что малые модели испытывают фундаментальные трудности с самокоррекцией, однако исследовала только итеративную коррекцию изолированно. Работы по test-time compute scaling [2] демонстрируют, что модели класса 3B могут достигать результатов более крупных моделей при достаточном объёме вычислений, но не рассматривают совместную оптимизацию нескольких компонентов при аппаратных ограничениях.

Целью данной работы является систематическое измерение вклада множественной генерации кандидатов (N) и итеративной коррекции по тестам (K) в pass@1 модели Qwen2.5-Coder-1.5B, исследование эффектов взаимодействия между этими компонентами и построение зависимостей компромиссов качество/время для определения оптимальных конфигураций при заданном вычислительном бюджете.

Работа мотивирована спросом на локальные инструменты кодогенерации, не зависящие от облачных API. Разработчикам, работающим с конфиденциальным кодом или с ограниченным бюджетом, необходимо понимать, как получить максимальное качество на доступном оборудовании. Результаты данной работы позволяют обоснованно выбирать конфигурацию пайплайна в зависимости от доступного вычислительного бюджета.

II. Описание подхода

Эксперименты проведены на модели Qwen2.5-Coder-1.5B-Instruct [3] (1.5 миллиарда параметров), запущенной через платформу Ollama на ноутбуке с 8 ГБ оперативной памяти (инференс на CPU). В качестве бенчмарка использован HumanEval [4], содержащий 164 задачи на языке Python. Метрика оценки — pass@1: доля задач, для которых сгенерированное решение проходит все тесты.

Пайплайн включает два исследуемых компонента. Первый — множественная генерация кандидатов: для каждой задачи генерируется N независимых решений при температуре 0.8 (температура — параметр генерации, управляющий степенью случайности: при значении 0.0 модель выбирает наиболее вероятный токен, при более высоких значениях распределение вероятностей сглаживается, что увеличивает разнообразие генерируемых решений), каждый кандидат проходит синтаксическую проверку через ast.parse, затем запускаются тесты; первый прошедший тесты кандидат считается успешным. Второй — итеративная коррекция: при провале тестов модели передаётся исходное условие задачи, сгенерированный код и сообщение об ошибке с

запросом на исправление; процедура повторяется до K раз или до успешного прохождения тестов.

Исследованы конфигурации с $N \in \{1, 2, 3, 5\}$ и $K \in \{0, 1, 2\}$. Для экспериментов с итерациями при N=1 использовалась температура 0.0 для обеспечения детерминированного результата. Для экспериментов с множественными кандидатами — температура 0.8, следуя методологии оригинальной работы HumanEval [4], где это значение использовалось для оценки pass@k. Контрольный прогон N=1 при температуре 0.8 проведён отдельно, чтобы разделить эффект температуры и эффект множественной генерации. Ряд конфигураций прогнан дважды для оценки стохастичности результатов.

III. Результаты

A. *Индивидуальные эффекты компонентов*

Контрольный эксперимент показал, что температура сама по себе не влияет на pass@1 при N=1: значения составили 0.622 при температуре 0.0 и 0.628 при температуре 0.8 (разница 0.6 процентного пункта, далее — п.п.). Таким образом, прирост при увеличении числа кандидатов нельзя объяснить температурой.

Множественная генерация оказалась основным источником прироста качества. Результаты представлены в табл. 1. Переход от N=1 к N=2 даёт скачок в +10–13 п.п. при увеличении времени всего на 4 секунды. Дальнейшее увеличение до N=5 добавляет ещё около 8 п.п. Разброс между повторными прогонами для N=2 и N=3 составляет около 3 п.п., что отражает стохастичность генерации.

ТАБЛИЦА I. ВЛИЯНИЕ ЧИСЛА КАНДИДАТОВ НА PASS@1 (K=0, ТЕМПЕРАТУРА 0.8)

N	**pass@1**	**Δ от базовой**	**Время/задачу**
1	0.628	—	18.3 сек
2	0.756 / 0.726	+12.8 / +9.8 п.п.	20.4 / 23.8 сек
3	0.750 / 0.750	+12.2 п.п.	19.6 сек
5	0.829	+20.1 п.п.	29.9 сек

Итеративная коррекция даёт ограниченный прирост: K=1 добавляет 3.7 п.п., K=2 — лишь 1.2 п.п. поверх K=1 (табл. 2). При качественном анализе конкретных задач обнаружено характерное поведение модели 1.5B: она не диагностирует корневую причину ошибки. В задаче HumanEval/0, требующей поиска пары близких чисел в списке, модель во всех итерациях проверяет только соседние элементы вместо всех пар. Получив сообщение об ошибке с конкретным тест-кейсом, модель добавляет проверку граничных условий, но алгоритмическую ошибку не исправляет.

ТАБЛИЦА II. ВЛИЯНИЕ ИТЕРАТИВНОЙ КОРРЕКЦИИ НА PASS@1 (N=1, ТЕМПЕРАТУРА 0.0)

K	**pass@1**	**Δ от K−1**	**Время/задачу**
0	0.622	—	16.4 сек
1	0.659	+3.7 п.п.	24.8 сек
2	0.671	+1.2 п.п.	35.7 сек

B. *Эффекты взаимодействия и комбинированные конфигурации*

Результаты комбинированных конфигураций представлены в табл. 3. Для количественной оценки взаимодействия компонентов вычислен эффект взаимодействия по формуле: $IE(N,K) = Q(N,K) - Q(N,0) - Q(1,K) + Q(1,0)$. Для пар (N=3, K=1) и (N=5, K=1) значение IE составляет +0.006, что указывает на почти аддитивный характер вкладов компонентов для модели 1.5B.

Таблица 3. Комбинированные конфигурации N × K (температура 0.8)

Конфигурация	**pass@1**	**Время/задачу**
N=2, K=0	0.756 / 0.726	20.4 / 23.8 сек
N=2, K=1	0.738 / 0.720	44.3 / 43.1 сек
N=3, K=1	0.793	56.4 сек
N=5, K=0	0.829	29.9 сек
N=5, K=1	0.872	60.3 сек

При N=2 обнаружен отрицательный эффект итераций: конфигурация N=2, K=1 устойчиво уступает N=2, K=0 в двух независимых прогонах (0.738/0.720 против 0.756/0.726). Вероятный механизм состоит в том, что процедура исправления может испортить частично корректное решение, а при двух кандидатах компенсировать это нечем. При N=5 такой проблемы не наблюдается: даже если исправление испортит одного из кандидатов, остальные четыре обеспечивают достаточно шансов на успех.

C. *Компромисс: качество и временные затраты*

Для практического применения важно соотношение прироста качества и дополнительных временных затрат. Конфигурация N=2, K=0 является наиболее эффективной: она обеспечивает прирост в +13 п.п. за 4 дополнительных секунды на задачу (0.3 секунды на каждый процентный пункт прироста). Итеративная коррекция обходится существенно дороже: +3.7 п.п. за 8 дополнительных секунд (2.3 секунды на процентный пункт). При фиксированном временном бюджете оптимальный выбор: 20 секунд — N=2 (pass@1=0.756), 30 секунд — N=5 (0.829), 60 секунд — N=5, K=1 (0.872).

IV. Обсуждение

Полученные результаты показывают, что для модели размером 1.5B генерация нескольких независимых решений работает лучше, чем попытки исправить одно неудачное. Модель такого размера не справляется с самокоррекцией: получив сообщение об ошибке, она повторяет тот же алгоритмический подход, добавляя поверхностные граничные проверки. Этот результат согласуется с данными работы CoCoS [1], полученными для других малых моделей. Наша работа дополняет вывод CoCoS, помещая итеративную коррекцию в контекст многокомпонентного пайплайна: ресурсы, затрачиваемые на коррекцию, эффективнее направить на генерацию дополнительных кандидатов.

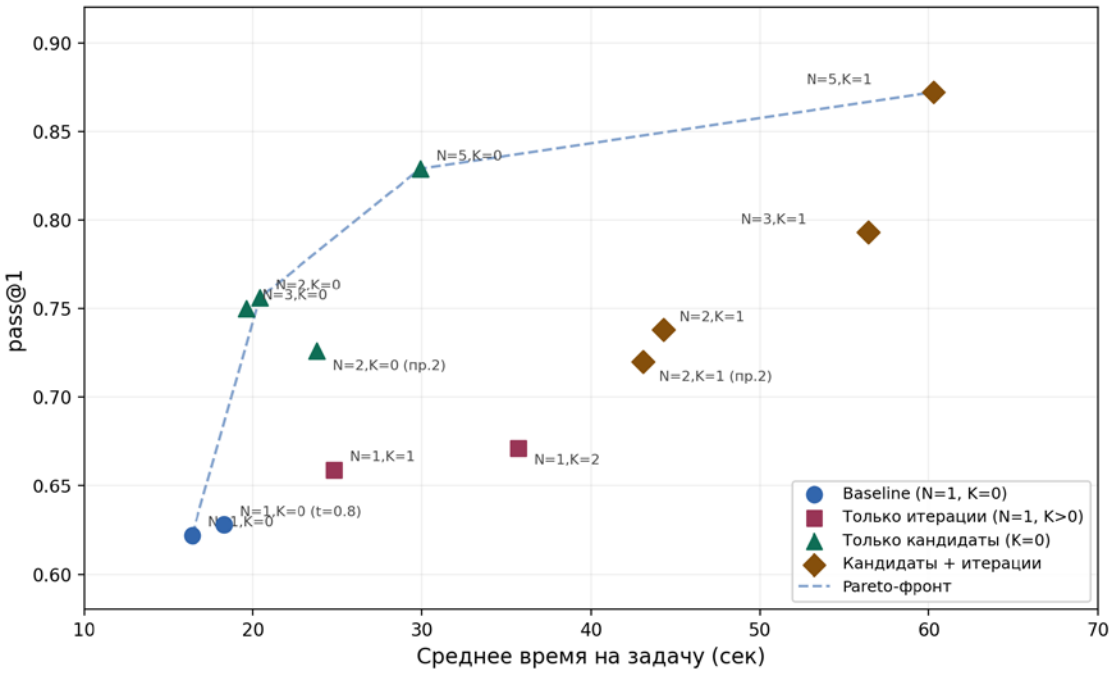

Рис. 1. Фронт Парето: зависимость pass@1 от среднего времени генерации

Эффекты взаимодействия между N и K на модели 1.5B оказались слабыми (+0.6 п.п.) — вклады компонентов почти аддитивны. На практике это означает, что для данной модели конфигурацию можно выбирать, оптимизируя каждый компонент по отдельности. Однако этот вывод справедлив только для 1.5B. На моделях 3B и 7B итеративная коррекция может оказаться более полезной, поскольку модель большего размера лучше диагностирует ошибки, что, вероятно, усилит эффекты взаимодействия. Проверка этой гипотезы — предмет дальнейшей работы.

Отрицательный эффект итераций при N=2 нельзя предсказать без эксперимента. Он показывает, что оптимальная конфигурация пайплайна не сводится к включению всех доступных компонентов: процедура исправления может портить частично корректное решение, и при малом числе кандидатов это снижает общее качество. Отсюда следует, что компоненты пайплайна нужно оптимизировать совместно, а не по отдельности.

V. Заключение

Результаты анализа компромиссов показывают нелинейную зависимость между качеством и временными затратами. Конфигурация N=2, K=0 даёт лучшее соотношение прирост/стоимость: +13 п.п. при увеличении времени на 24%. Наиболее качественная конфигурация (N=5, K=1 с pass@1=0.872) требует почти четырёхкратного увеличения времени по сравнению с baseline. Выбор между ними зависит от конкретного вычислительного бюджета разработчика, что и создаёт потребность в адаптивном подходе к конфигурации пайплайна.

К ограничениям работы относятся использование одной модели (1.5B), одного семейства моделей (Qwen2.5-Coder), одного бенчмарка (HumanEval), а также отсутствие компонентов RAG и experience memory. Не все конфигурации прогнаны многократно, что ограничивает статистическую надёжность отдельных результатов. Тем не менее, повторные прогоны нескольких конфигураций показали разброс в пределах 3 п.п., а основные выводы (превосходство множественной генерации над итеративной коррекцией, отрицательный эффект итераций при N=2) воспроизводятся устойчиво.

Дальнейшая работа включает расширение экспериментов на модели 3B и 7B для измерения зависимости эффектов взаимодействия от размера модели, добавление RAG-компонента и experience memory, расширение на бенчмарки MBPP и ClassEval, а также формализацию задачи выбора конфигурации как задачи условной оптимизации (MCKP с зависимыми группами).

Список литературы

[1] Cho C., Choi J., Park G., Kim J., Lee S. CoCoS: Is your small language model able to self-correct? // Proceedings of the 48th IEEE/ACM International Conference on Software Engineering (ICSE ’26). 2026.

[2] Li Y., Yu H., Yang G. S*: Test-time scaling for code generation : препринт. 2025. arXiv:2502.14382. URL: https://arxiv.org/abs/2502.14382 (дата обращения: 12.04.2026).

[3] Hui B., Yang J., Cui Z., Yang J., Liu D., Zhang L., Liu T., Zhang J., Yu B., Lu K., Dang K., Fan Y., Huang Y., Zhang B., Zhu J., Men R., Ren X., Zhou J., Lin J. Qwen2.5-Coder technical report : препринт. 2024. arXiv:2409.12186. URL: https://arxiv.org/abs/2409.12186 (дата обращения: 12.04.2026).

[4] Chen M., Tworek J., Jun H., Yuan Q., Pinto H., Kaplan J., Edwards H., Burda Y., Joseph N., Brockman G., Ray A., Puri R., Krueger G., Petrov M., Khlaaf H., Sastry G., Mishkin P., Chan B., Gray S., Ryder N., Pavlov M., Power A., Kaiser L., Bavarian M., Winter C., Tillet P., Such F., Cummings D., Plappert M., Chanez F., Barnes E., Herbert-Voss A., Guss W., Nichol A., Paino A., Tezak N., Tang J., Babuschkin I., Balaji S., Jain S., Saunders W., Hesse C., Carr A., Leike J., Achiam J., Misra V., Morikawa E., Radford A., Knight M., Brundage M., Murati M., Mayer K., Welinder P., McGrew B., Amodei D., McCandlish S., Sutskever I., Zaremba W. Evaluating large language models trained on code: препринт. 2021. arXiv:2107.03374. URL: https://arxiv.org/abs/2107.03374 (дата обращения: 12.04.2026).

[5] Ray S., Mehrotra A., Chakraborty S. AdaptEvolve: Adaptive model selection for code evolution: препринт. 2026. arXiv:2602.11931. URL: https://arxiv.org/abs/2602.11931 (дата обращения: 12.04.2026).