

コードスタイル変換と生成拡張検索を 組み合わせたコード検索

S2310157 谷口

背景

コード検索とは

- ・ クエリに対し、類似するコード断片を
コードベースから取得するタスク
- ・ GitHub等での大規模コードベースで活用
- ・ 開発の生産性を向上させる
- ・ コード生成との違い：信頼できるコード資産の再利用
コード検索

クエリ(入力)	Write a function to get the frequency of the elements in a list. (リスト内の要素の出現頻度を取得する関数を書け)
検索結果(出力)	<pre>import collections def freq_count(list1): freq_count = collections.Counter(list1) return freq_count</pre>



(出典:Li et al., ACL 2024)

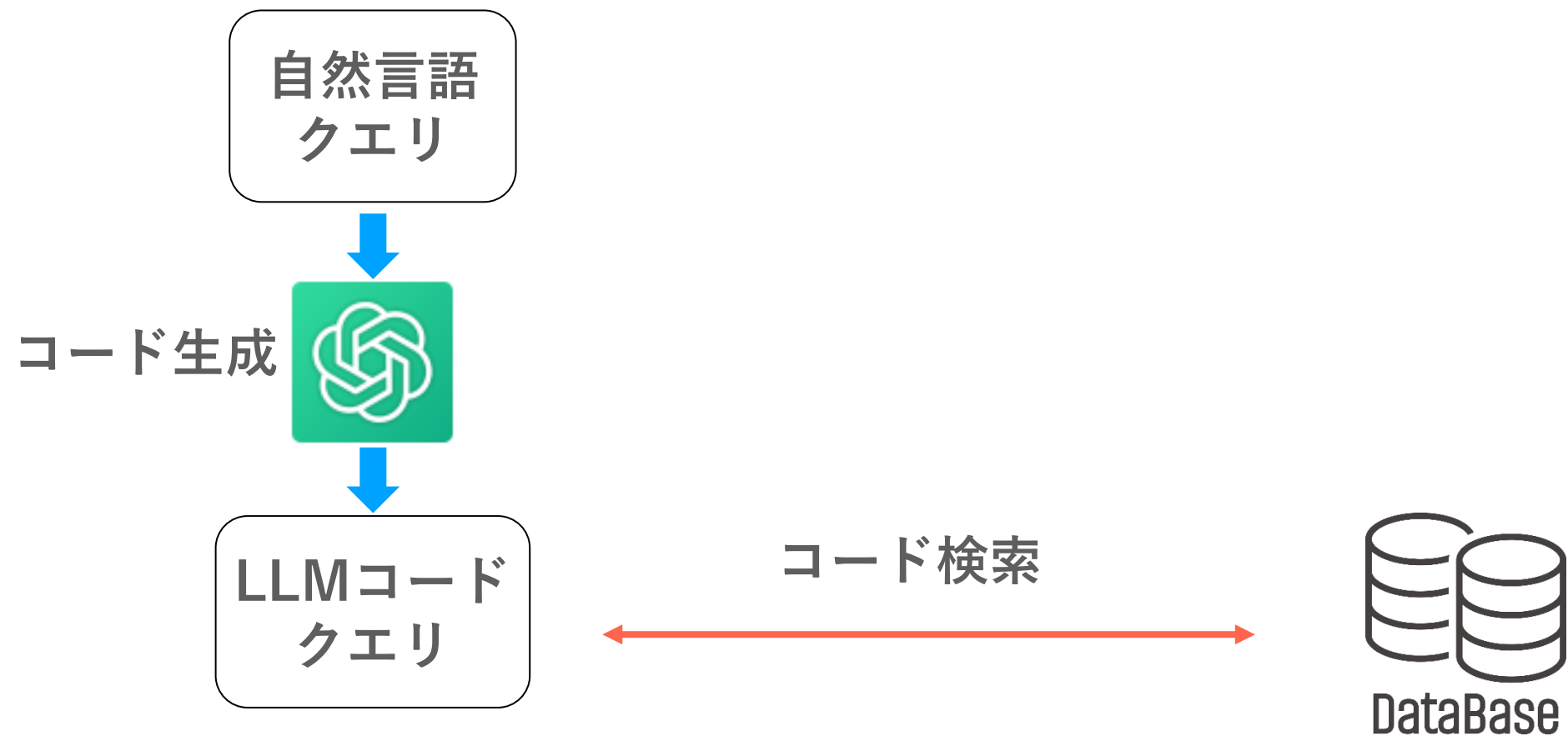
コード検索の課題

自然言語とコードの表現形式

- ・ 自然言語とコードの表現形式の差
 - 大きく異なる文法
 - 共通する語彙が少ない
- ・ コード検索の障害になる
 - 単語の一致で検索できない
 - LLMを用いる手法でもボトルネックに

既存のコード検索

GAR(生成拡張検索)



- ・ 自然言語クエリをLLMを用いてコードクエリに変換

既存のコード検索

GAR(生成拡張検索)

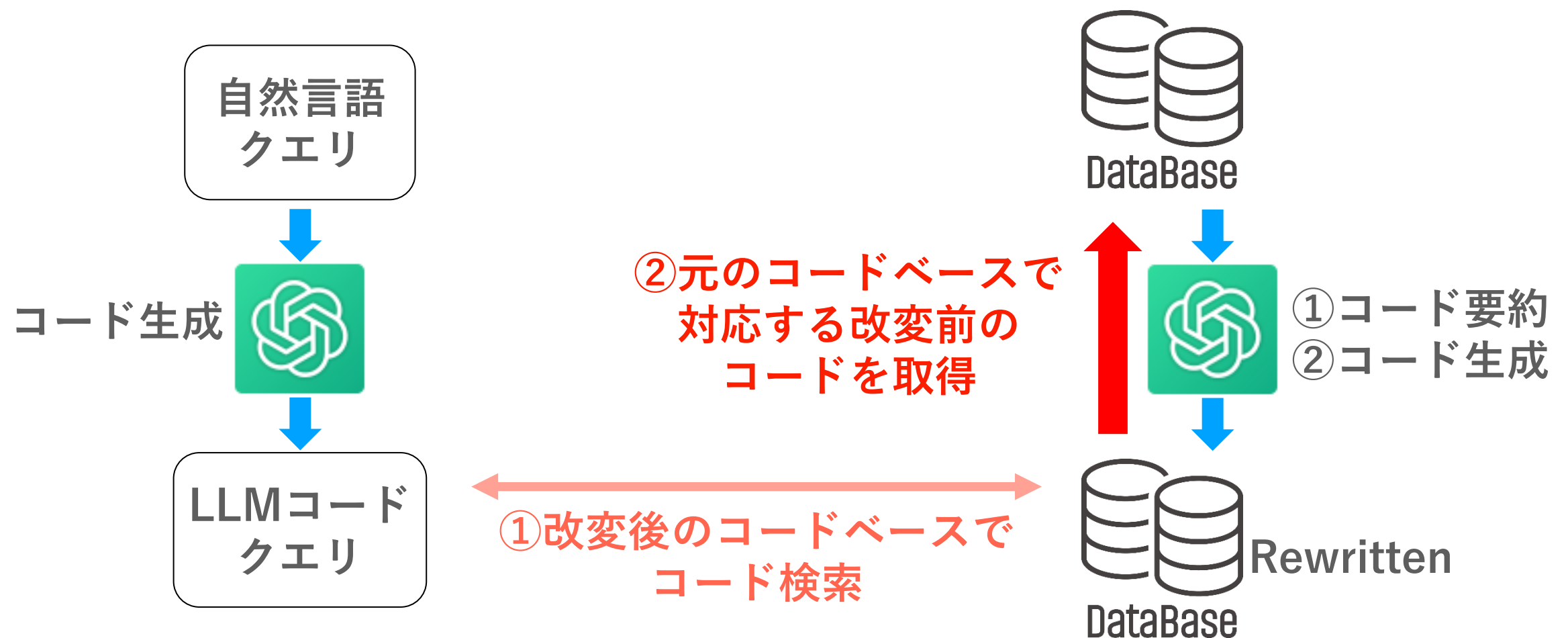
LLMクエリとデータベースの
コードスタイルの差が新たな障害に



- ・ 自然言語クエリをLLMを用いてコードクエリに変換

既存のコード検索

ReCo



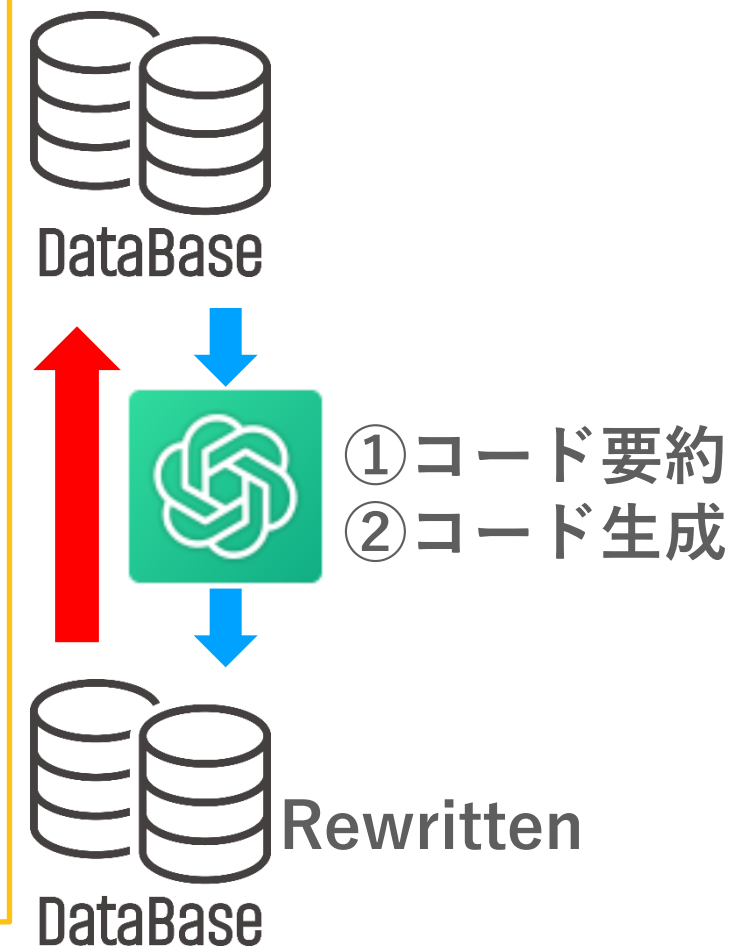
- ・ LLMを用いてデータベース側のコードを書き直す
- ・ クエリとデータベース両方のコードスタイルが統一され精度向上

参考: Rewriting the Code: A Simple Method for Large Language Model Augmented Code Search (Li et al., ACL 2024)

既存のコード検索

ReCo

コードベース全体を
LLMに書き直させる処理は
時間的・金銭的に高コスト



- ・ LLMを用いてデータベース側のコードを書き直す
- ・ クエリとデータベース両方のコードスタイルが統一され精度向上

参考: Rewriting the Code: A Simple Method for Large Language Model Augmented Code Search (Li et al., ACL 2024)

目的・アプローチ

目的

- ReCoのコスト問題の解消

- コードベース全体の書き直し
- 精度改善 ⇔ 全体を書き直すコスト
- 大規模プロジェクトでは現実的ではない

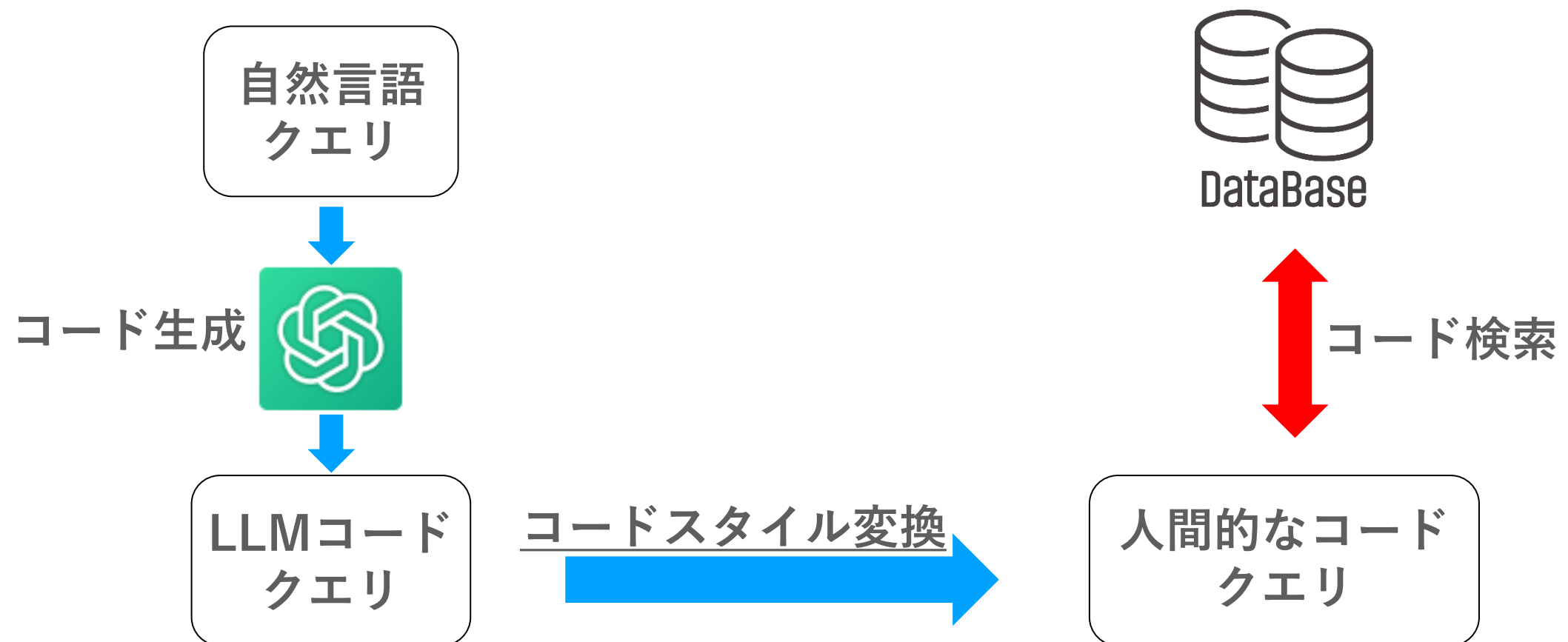
アプローチ

- クエリ側のみの変換でスタイルを統一

- コードベースへの操作を最小限に
- クエリ側の操作だけで解消

提案手法

コードスタイル変換を使用したコード検索



- ・ LLMと人間のコードスタイルの差による検索精度の限界
その差をコードスタイルを変換して埋める

参考: Rewriting the Code: A Simple Method for Large Language Model Augmented Code Search (Li et al., ACL 2024)

コードスタイル変換

人間的なコードへの変換

- LLMコード識別モデルが人間だと判定するようにコード変換
- 変換前後で機能は変化させない
- **MCTS (モンテカルロ木探索)^[1]**
 - 36種のコード変換の組み合わせ
 - 識別モデルの反応からより効果的な組み合わせを探索
- **MIST (多目的識別子および構造変換)^[2]**
 - 突然変異と交叉を用いて探索する遺伝的アルゴリズム
 - 変化の「少なさ」「自然さ」も重視した探索

[1]:Erwin Quiring, Alwin Maier, and Konrad Rieck. 2019. Misleading authorship attribution of source code using adversarial learning. In Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19). USENIX Association, USA, 479–496.

[2]:Xiaodan Xu, Chao Ni, Xinrong Guo, Shaoxuan Liu, Xiaoya Wang, Kui Liu, and Xiaohu Yang. 2025. Distinguishing LLM-Generated from Human-Written Code by Contrastive Learning. ACM Trans. Softw. Eng. Methodol. 34, 4, Article 91 (May 2025),

31 pages. <https://doi.org/10.1145/3705300>

実験

MBPP

- 基本的なPythonプログラムの課題セット974問
 - 訓練用500問、テスト用374問、検証用100問
- 訓練用500問を使用
- データベース内に正解コード500個
- 問題文から対応する正解コードを検索してこれるか
- 4種の方法を比較
 - ベクトル検索のみ、GAR、ReCo、**GAR+コードスタイル変換**

実験結果（GPT-3.5）

GARに対してほぼ変化なし

Claude Codeによる実験結果

ほぼ同じ
スコア

	キーワード	Dense Retrieval (ベクトル検索)		
	BM25	UniXcoder	Contriever	CodeBERT(finetuned)
通常	0.1046	0.6501	0.4259	0.7602
GAR	0.3538	0.7668	0.5758	0.7803
+ MCTS	0.3243	0.7704	0.5710	0.7806
+ MIST	0.3136	0.7683	0.5817	0.7790
ReCO	0.5975	0.8508	0.7616	0.8475

ReCoの元論文での結果

元論文の実験は
大体再現
できていそう

	BM25	UniXcoder	Contriever	CodeBERT(finetuned)
通常	0.126	0.693	0.553	0.796
GAR	0.351	0.797	0.656	0.816
ReCO	0.708	0.862	0.795	0.835

結果考察

精度は改善しなかった

- キーワードマッチング(BM25)ではわずかに精度劣化
 - 変数名置換のコード変換による影響
- ベクトル検索ではほぼ変化なし

GARの検索失敗のボトルネック調査

- GARの失敗ケースを手動で確認
- 具体的な問題から検索のボトルネックを探る

失敗ケースの調査

スタイル以外のボトルネックの特定

- 正解コードが検索候補の2位以降に出てくるケースを調べる
- データセット: MBPP
- 検索手法: GAR(コード生成によるクエリ拡張)
ベクトル検索
- 失敗ケース165件
- 現状50件ほど

失敗の要因

類似タスクの存在

- ・ 部分的に命名や処理が一致する別コード
 - リストのリストから各要素の頻度を出すコード(正解)
リストから各要素の頻度を出すコード (誤答)
 - 文字列リストを文頭でソートするコード(正解)
タプルリストを2要素目でソートするコード(誤答)
 - インタフェースや条件が違う似たタスクと間違えやすい
 - 「細かな条件」は関数名に反映されないことが多く
埋め込み上で区別されにくい可能性

失敗の要因：MBPP

問題が曖昧

- ・ 問題文の誤字 (Woodall数 → Woodball数)
- ・ 曖昧な指示 (符号化の問題なのに、復号化とも取れる指示)

正解コードが不可解

- ・ 不明瞭な命名 (正解 / GAR)
 - 約数の総和を数える関数 : `sum` / `sum_common_divisors`
 - タプルの特定要素の出現数を数える関数 : `count_X` / `count_occurrences`
- ・ バグや不備がある

正解と酷似したコード

- ・ 実装や命名が違っているが機能的に等価なコード
- ・ 同じコードが別idの問題の答えとして登録されている

まとめ

提案手法

- GARのコードクエリにコードスタイル変換を組み合わせる手法
 - コードスタイルの差の問題を
従来手法(ReCo)より低コストで解決したい
- コードスタイル変換
 - LLMコード識別モデルに人間だと判定されるように変換
 - 機能的に等価なコード変換の組み合わせ
 - 探索方式：現状MIST, MCTSの二種

実験結果

- GARとほぼ変わらない結果

今後の展望

現状考えていること

- コードスタイル変換の手法を変える
 - 目星はたっていない
- データセットを変える
 - HumanEval, SWE-bench?
 - ReCoの元論文で未使用のデータセット
- モデルを変える
 - 現在 : GPT-3.5
- 具体的な失敗ケースを見る
 - あまりこれといった原因や解決策が見えてこない
- テーマを変える

相談

希望が見えない

- 現状の方法以外のコードスタイル変換の方法が分からない
 - うまい変換方法があってそれを見つけてないだけか
 - コードスタイルの抽出すら分からない
- コードベース内に複数のスタイルがあったら厳しいかも
- コード生成単体にコストとパフォーマンスで負ける
 - 安全性が検証されたデータを使用できるという利点はある
 - MBPPではLLMが十分なコードを生成できるためその利点は示せない