コード生成のための大規模言語モデルを用いた検索手法

中原 康宏　宮下 大輔 出口 淳
キオクシア株式会社
yasuhiro1.nakahara@kioxia.com

概要

Large Language Model によるプログラムコード生成は有用であるが，LLM が持つ情報の更新には膨大なコストが必要である．この課題に対応するために，関連する情報を検索し LLM への入力に追加する手法が提唱されており，コード生成の分野では BM25 などのベクトル検索を用いた手法が成果を挙げている．しかし，GitHub のような Web サイトをデータベースとする場合，全データを手元に置く必要のあるベクトル検索はデータ収集に膨大なコストを必要とし，常に情報を最新に保つことは困難である．よって，本論文では GitHub API がサポートしている既存の Web 検索と LLM を用いた検索手法を提案する．これにより，提案する手法は低コストで常に最新の情報にアクセスすることができる．本手法は Web 検索と LLM を組み合わせる際に生じるいくつかの典型的な課題にも対応している．提案手法を評価した結果，有用なリポジトリを見つけ出すことができたことを確認した．

1　はじめに

近年，Large Language Model (LLM) によるプログラムコード生成が大きな注目を集めている [1]．そのため，コード生成に向けた LLM が多数リリースされている [2, 3, 4]．しかし，これらのコード生成用 LLM は情報更新のために高いコストを必要とする点が課題である．プログラミングの世界では日々新規ライブラリの提案やライブラリの更新がなされている．コード生成用 LLM がこれらの更新を取り入れるためには，一般的には学習と呼ばれる時間的，電力的に高コストな処理が必要である [5]．よって，最新のライブラリに対応するために頻繁に LLM を更新することは難しい．

この課題を解決するために，LLM と検索を組み合わせた Retrieval Augmented Generation (RAG)[6] という手法が提唱されている．RAG は LLM の処理に関連する情報を検索し，その検索結果を LLM への入力に含めることで LLM に追加の情報を与える手法である．これにより，LLM は学習することなく最新の情報を反映したコードを生成することができる．先行研究 [7, 8, 9] では検索に BM25[10] や Dense Passage Retrieval (DPR)[11] などのベクトル検索を用いた RAG によってコード生成の性能向上を達成している．しかし，これらの手法では Web サイトをデータベースにすることは難しい．なぜなら，DPR や BM25 を実行するためにはデータベースの全情報を参照する必要があるためである．そのため，Web サイトをデータベースとする場合は検索のために全ての情報を一度ダウンロードしなければならない．これは多大な時間的コストを必要とし，情報を最新に保つために頻繁に行うことは現実的ではない．また，本行為は Web サイトの利用規約により禁止されている場合もある．GitHub や StackOverflow などプログラマにとって有用な情報源となる Web サイトは多くあるため，これは重要な課題である．

これを回避するための手法の一つとして，Web サイト側が提供している検索機能を利用する手法が挙げられる．例えば，GitHub は「LLM chat」などとキーワードを用いて検索するキーワード検索を Application Programming Interface (API) で提供している．キーワード検索を活用することで，最新の情報を低コストで取得できる．だが，キーワード検索にも大きく 2 つの課題が存在する．1 つは検索結果からデータを更に絞り込む必要がある点である．ベクトル検索では最も類似度の高いデータを選択すればよいが，キーワード検索では複数の検索結果から 1 つを選ぶ必要がある．この際，正しく有用な情報を選べなければ，生成コードの品質を向上させることはできない [12]．本論文ではこの課題に対して LLM を用いて有用なデータの絞り込みを行うことで対応する．この際に生じる 2 つ目の課題が，LLM に入力可能な文字数が限られている点である．この課題に対しては，絞り込みの際に以下の 3 つの工夫

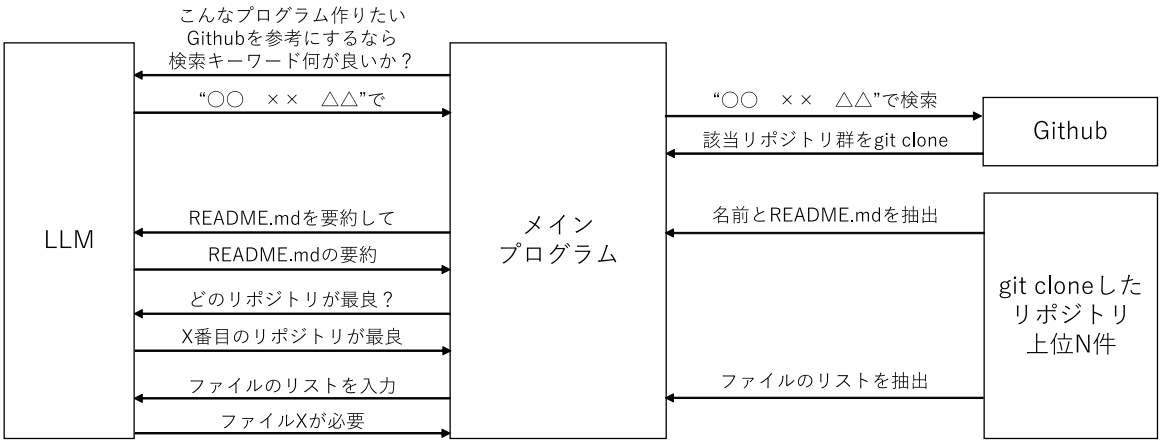

図 1　提案手法の処理の流れ

点をツールに組み込み，LLM に一度に入力する文字列の長さを削減することで対応する．

- 絞り込みを 2 段階に分割
- LLM で README.md を要約してから比較
- コードをプロンプトに含めずに絞り込む

RAG の有効性は先行研究 [7, 8] によって既に明らかにされているため，本稿では検索手法に注目して議論を行う．データベースとする Web サイトは最新のライブラリやコード例が多数投稿されている GitHub とする．LLM で生成した 10 個の要望対して本手法を用いて検索を行った結果，8 個の要望について有用なリポジトリを選択でき，6 個の要望について有用なコードを選択することができた．

2　提案手法

2.1　概要

本手法は GitHub API を用いて Web 検索を行った後に検索結果を絞り込み，コード生成に有用ないくつかの python ファイルを提案する手法である．Web 検索で使用するキーワードの生成や検索結果の絞り込みには LLM を用いる．python ファイルの提案までの工程は全自動で実行され，ユーザは要望を入力するだけで良い．

LLM に入力できるトークンの数には限りがあるため，本手法では絞り込みの際に以下に示す 3 点を考慮して絞り込みのアルゴリズムを構築している．1 点目は絞り込みを 2 段階に分けている点である．全ての情報を加味して一度に必要なデータを判断することが，判断の正確性という点では良い手法であると考えられる．しかし，トークン数の制限を考慮すると LLM に一度に与えるデータは少ないほうが良い．そのため，本手法ではリポジトリの絞り込みとファイルの絞り込みの 2 段階に分割して絞り込みを行う．

2 点目は LLM でリポジトリの絞り込みを行う際に，README.md 本体でなくその要約を入力に使用する点である．リポジトリの内容を理解するために README.md は有用であるが，その大きさはリポジトリごとに大きく異なる．よって，比較のために複数の README.md を読み込んだ場合，リポジトリによってはトークン数の制限を超えてしまう可能性がある．したがって，LLM を用いて README.md を短く要約することで，トークン数を削減している．

3 点目はプログラムコードを LLM に読み込ませないという点である．プログラムの規模はリポジトリによって大きく異なるため，大規模なリポジトリのコードを LLM に読み込ませるとトークン数の制限を超えてしまう可能性がある．よって，本手法ではコード本体ではなく README.md やファイル名などを絞り込みのための指標としている．

2.2　処理の流れ

本節では，提案手法の処理の流れを具体例を用いて述べる．例で用いるユーザの要望は「OpenCALM を用いたチャットボット」とする．要望の詳細は付録 A に示す．提案手法の処理の流れを図 1 に表す．

まず，提案ツールは LLM を用いて要望からシンプルな 3 つの検索キーワードを作成する．チャットボットの例では “chatbot”，“OpenCALM”，“log” の 3 つのキーワードが生成された．その後，GitHub API を用いてこれらのキーワードを組み合わせた 6 パターンの検索を行う．チャットボットの例では “chatbot”，“OpenCALM”，“log”，“chatbot

OpenCALM”，“chatbot log”，“OpenCALM log” を検索に用いる．キーワードを組み合わせるのは “app”，“list”，“log” など単体では意味をなさないキーワードを生成する場合があるためである．これらのキーワードで検索した結果からそれぞれ指定した件数のリポジトリを手元にダウンロードする．検索の際，リポジトリは python を使用言語としているものを指定して検索している．

次に，検索結果から有用なリポジトリを絞り込む．最初に各リポジトリの README.md を 200 単語程度に LLM を用いて要約する．要約の際に README.md が LLM に入力可能なトークン数の上限を超える場合，トークン数が収まるように文末から文字列を削ることで対応する．その後，LLM を用いて有用なリポジトリを絞り込む．この際，LLM に入力する情報はリポジトリのタイトル及び要約のリストである．チャットボットの例では 3 つのリポジトリ [13, 14, 15] に絞り込まれた．これらのリポジトリはいずれも OpenCALM をダウンロードして推論までを実行するコードを含むリポジトリであるため，コード生成に有用である．コード生成は絞り込んだリポジトリそれぞれに対して行い，その中から最も要望に近い物を採用する．

最後に，それぞれのリポジトリについて有用なファイルを絞り込む．はじめに，各リポジトリから python ファイルを抽出し，それらの名称をリストアップする．リストアップの際は，[src/xxx/aaa.py, examples/bbb.ipynb] のようにリスト形式でファイルのパスを列挙する．その後，リポジトリ名，README.md ファイル，リストアップしたファイル名を LLM に入力し，最良のファイルを選択する．以上の手順により絞り込んだファイルを要望とともに LLM に入力することで，リポジトリの情報を加味したコード生成を行うことができる．

3 評価

3.1 評価条件

まず，先に述べたトークン数を削減するための 3 つの工夫点について，それらの有効性について評価する．評価のためのユーザの要望は LLM を用いて作成した．この時，恣意的な評価とならないよう要望の具体的な内容は指定しない．要望の詳細は付録 B に示す．この要望について検索までを行い，絞り込みの対象となったリポジトリの README.md や python コード，リポジトリ内に存在する python ファイルのリストについてのトークン数を評価する．

次に，提案ツールがどの程度要望から適切なコードを検索できるのかについて評価を行う．コード生成に対する RAG の有効性は既に示されているため [7, 8]，本評価では検索にのみ注目して評価を行う．トークン数の評価で使用した 10 個の要望に対して，有用なリポジトリをいくつ検索できたかを評価する．提案システムが選択したリポジトリ，ファイルについてそれらが要望を達成するために有用であるかを人間[1)]が有用，一部有用，使用不可の 3 段階で評価する．少しのコード修正で要望を達成可能なものを有用，要望達成のためにある程度機能の追加が必要なものを一部有用，要望の達成に使用できないものを使用不可とする．リポジトリとファイルの絞り込みにおいて，それぞれ最小 1 つ，最大 3 つのものを提案するように LLM に指示する．提案システムが一つの要望に対して複数のリポジトリやファイルを提案した場合，最も良い結果を最終結果とする．LLM は GPT-3.5 turbo（ver. 0613）を使用する．

3.2 評価結果と考察

リポジトリがもつデータ量についての評価結果を表 1 に表す．表 1 より，python ファイルのトークン数の中央値は 10,045 であった．今回使用する GPT-3.5 turbo が扱えるトークン数は 4,096 であるため，半数以上のリポジトリでコードを扱うことができない．詳細を確認した結果，入力トークン数が 4,096 以下のリポジトリは約 34%であった．出力トークン数も加味すると，python ファイルを処理可能なリポジトリの割合は更に減少すると考えられる．よって，絞り込みのために python コードを LLM に入力することは難しいと言える．

また，README.md のトークン数は中央値で 583 であった．仮に中央値である 25 個のリポジトリ間での比較を行った場合，トークン数は 14,575 となり上限を大きく超える．対して要約のトークン数は中央値で 100 と大きく削減されており，同様の比較を行った場合トークン数は 2,500 と処理可能な範囲内に収まる．このように，README.md を要約する手法はトークン数削減に有効であることが分かる．

次に，ファイルリストのトークン数に注目すると，中央値が 72 と少ない．しかし，リポジトリの絞り込みとファイルの絞り込みを一度に行おうとする

1） 判定は筆頭著者の中原康宏が行った．

表 1　リポジトリがもつデータ量についての評価結果

	最小	平均	最大	中央値
要望当たりのリポジトリ数	18	25	30	25
リポジトリ当たりの python ファイル数	0	121	4,727	11
README.md のトークン数（文字数）	0 (0)	1,982 (6,578)	79,185 (190,638)	583 (2,026)
要約のトークン数（文字数）	7 (36)[a]	100 (524)	285 (1,362)	113 (588)
リポジトリ当たりの python コードのトークン数（文字数）	0 (0)[b]	205,736 (826,109)	8,069,997 (33,260,608)	11,045 (40,469)
ファイルリストのトークン数（文字数）	2 (1)[c]	1,679 (5,876)	94,556 (1,451,410)	72 (261)

[a] README.md が無い場合，要約でその旨を記述するため文字数が 0 ではない．
[b] 検索結果に python ファイルを持たないリポジトリが存在したため．
[c] python ファイルが存在しない場合，ファイルリストは [] と記述するため．

表 2　検索結果の有用性についての評価結果

アプリ名	リポジトリ	コード
記事推薦	一部有用	一部有用
時計	一部有用	一部有用
音楽	有用	使用不可
ニュース	有用	一部有用
料理レシピ	一部有用	使用不可
買い物リスト	使用不可	使用不可
予定管理	有用	一部有用
交通状況	使用不可	使用不可
翻訳	一部有用	一部有用
天気予報	有用	有用

と（README.md 要約＋ファイルリスト）×（リポジトリ数）のデータが必要となるため中央値で計算してもトークン数の上限に達してしまう．対して絞り込みを分けた場合，要約の代わりに README.md 本体を使用したとしてもトークン数に十分な余裕が生まれる．加えて，README.md にはファイルの絞り込みに有用なデータが含まれていることも多くあり，README.md 本体を使用することによる精度向上も見込める．そのため，絞り込みを 2 段階にすることはツールの安定実行に貢献すると考えられる．

本稿で提案した検索の有用性についての評価結果を表 2 に表す．リポジトリでは 8/10，コードでは 6/10 の割合でコード生成に有用な情報を取得することができた．一方で，一部の要望については有用な情報を取得することができなかった．これは，GitHub 上に求めるリポジトリが少ないことが原因の一つだと考えられる．例えば買い物リストアプリの場合，生成された検索キーワードは “shopping”, “list”, “application” と妥当なものであるが，検索結果はショッピングサイト向けのカートの実装が殆どであり求めるリポジトリは存在しない．交通状況アプリについても同様であった．

また，コード選択で誤ったものを選択しているケースもあった．本ケースでは example.py など使用例を示したコードを選択せず，func.py などプログラムの一部を指定した．このようなファイルは有用である場合もあるが使用例などよりは有用度は低い．

また，有用な情報を取得できないケースとして，大規模なリポジトリを選択したというケースもあった．プログラムが大規模である場合，一部のファイルを参考にしてもその活用に別のファイルの情報を必要とする場合が多い．よって，どのファイルを選択してもコード生成の参考にはならない．

これらの絞り込みに失敗したケースについては，要望と合致するリポジトリが検索結果に存在しないケースを除き，検索する範囲，LLM に与える情報，LLM へ入力するプロンプトの文言などを調整することで対応可能であると考えられる．

以上より，提案手法は 3 つの工夫点により LLM のトークン数制限を回避しながら情報の絞り込みができる事が分かった．また，本手法は要望に合致するリポジトリが多く存在する場合には，有用な情報にアクセスできると言える．

4　おわりに

本稿では RAG のための新たな検索手法を提案した．本手法は Web 検索と LLM を組み合わせた検索手法であり，先行研究の課題である情報を最新に保つことが困難であるという点を解決している．また，提案手法は Web 検索導入に伴ういくつかの課題に対応している．LLM を用いて生成した 10 種の問題に対して本手法を適用した結果，8/10 で適切なリポジトリを，6/10 で適切な python ファイルを選択することができた．よって，本手法は要望と合致したリポジトリが多く存在する場合において有望であると言える．また，今後取り組むべき課題として絞り込みが失敗するケースにいくつかのパターンがあることが確認できた．

参考文献

[1] Bei Chen, Daoguang Zan, Fengji Zhang, Dianjie Lu, Bingchao Wu, Bei Guan, Yongji Wang, and Jian-Guang Lou. Large language models meet nl2code: A survey. In **ACL 2023**, June 2023.

[2] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D.Q. Bui, Junnan Li, and Steven C. H. Hoi. Codet5+: Open code large language models for code understanding and generation. **arXiv preprint**, 2023.

[3] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. **arXiv preprint arXiv:2306.08568**, 2023.

[4] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2023.

[5] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview, 2020.

[6] Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. A survey of knowledge enhanced pre-trained language models. **IEEE Transactions on Knowledge and Data Engineering**, pp. 1–19, 2023.

[7] Md Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. Retrieval augmented code generation and summarization. **In Findings of the Association for Computational Linguistics: EMNLP 2021**, pp. 2719–2734, 2021.

[8] Shuyan Zhou, Uri Alon, Frank F. Xu, Zhiruo Wang, Zhengbao Jiang, and Graham Neubig. Docprompting: Generating code by retrieving the docs. In **International Conference on Learning Representations (ICLR)**, Kigali, Rwanda, May 2023.

[9] Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung won Hwang, and Alexey Svyatkovskiy. Reacc: A retrieval-augmented code completion framework, 2022.

[10] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. **Found. Trends Inf. Retr.**, Vol. 3, No. 4, p. 333–389, apr 2009.

[11] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.

[12] Shangwen Wang, Mingyang Geng, Bo Lin, Zhensu Sun, Ming Wen, Yepang Liu, Li Li, Tegawendé F. Bissyandé, and Xiaoguang Mao. Natural language to code: How far are we? In **Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering**, ESEC/FSE 2023, p. 375–387, New York, NY, USA, 2023. Association for Computing Machinery.

[13] Github: naltoma/open-calm-finetuning, （2023-12 閲 覧）. https://github.com/naltoma/open-calm-finetuning.

[14] Github: miyakei1225/OpenCALM-7B-SandBox, （2023-12 閲 覧）. https://github.com/miyakei1225/OpenCALM-7B-SandBox.

[15] Github: hirok-to-i/CALM_CA_LLM_CPU_API, （2023-12 閲覧）. https://github.com/hirok-to-i/CALM_CA_LLM_CPU_API.

A　処理の流れで用いた要望

2.2 節で例として使用した要望を以下に示す．

Create a chatbot using OpenCALM.
OpenCALM is a large language model.
The program flow is as follows
1. wait for input from the user.
2. input the user input to OpenCALM and print the OpenCALM's answer.
3. return to 1.

Add the previous conversation to the input from the user before entering it into OpenCALM so that OpenCALM can remember the previous conversation. Save the conversation in log.txt.

B　評価に用いた要望

評価で用いた 10 個の要望は GPT-3.5 turbo（ver. 0613）に以下のプロンプトを与えて作成した．“生徒にプログラムを作成させる問題を作りたい。ユーザがこういうプログラムを作りたいと要望している風の問題をいくつか作ってほしい。解かせるためだけのプログラムではなく実用的なプログラムが良い。”

作成した 10 個の要望を表 3 に示す．生成された要望は日本語であったが，トークン数削減のため英語に翻訳している．

表 3　LLM を用いて生成したユーザの要望

アプリ名	本文
記事推薦	Article recommendation application: Create an application that uses the API to present articles of interest to the user. Allow users to select topics and genres of interest.
時計	Clock app: Create an app with the basic functions required for a clock app, such as digital clock, alarm, stopwatch, timer, etc.
音楽	Music Player: Create a player application that allows users to play, stop, skip, and create playlists of music. If necessary, artist information and album art will also be displayed.
ニュース	News App: Create an app that uses RSS feeds to provide users with the latest news articles.
料理レシピ	Recipe App: Create an app that allows users to register their own recipes and upload names, ingredients, processes, and photos.
買い物リスト	Shopping List: Create an application that allows users to create a list of products to be purchased and add prices and quantities.
予定管理	Task Manager: Create an application that can track tasks created by the user. Task priorities, due dates, status, etc. will be displayed.”
交通状況	Traffic jam forecasting application: Create an application that acquires real-time traffic information and provides users with locations and times when traffic jams are expected.
翻訳	Translation App: Create an application that provides online translation services to users. The text entered by the user can be translated into the specified language.
天気予報	Weather apps: Create apps that provide weather information. This includes today's weather forecast, weekly weather, local weather, etc.

記載されている社名・商品名・サービス名などは，それぞれ各社が商標として使用している場合があります．