ソフトウェア工学タスクにおける LLM カスケードのコスト削減効果

佐久間 一颯
和歌山大学システム工学部
s276103@wakayama-u.ac.jp

大平 雅雄
和歌山大学システム工学部
masao@wakayama-u.ac.jp

要旨

近年，大規模言語モデル（*LLM: Large Language Model*）は自然言語処理（*NLP*）タスクに限らず，ソフトウェア工学（*SE*）タスクでも広く活用されているが，高額な *API* 利用コストが課題となっている．この課題に対処するため，モデルルーティング手法が近年注目されている．モデルルーティングとは，単一の *LLM* に全てのタスクを処理させるのではなく，タスクに応じて最適な専門モデルの組合せを選択し，出力を生成するための手法である．

そのモデルルーティングの一種として，*LLM* カスケード *[1]* がある．*LLM* カスケードは，安価なモデルから順にクエリを処理し，不十分な場合のみ高性能モデルへ委譲することでコスト削減を図る手法である．しかし既存研究はほとんどが *NLP* タスクを対象としており，*SE* タスクでの効果は十分に検証されていない．

本研究では，*SE* タスクである *HumanEval [2]*，*MBPP+ [3]* と，*NLP* タスクである *MMLU [4]*，*HEADLINES [5]* の *4* つのベンチマークを用いて，*GPT-5-nano*，*GPT-5-mini*，*GPT-5* の *3* 階層からなる *LLM* カスケードを構築し，*LLM* カスケードのコスト削減効果を実証的に検証した．結果，*HumanEval* で *71.26* %，*MBPP+* で *57.96* %のコスト削減を達成し，性能は全タスクで *GPT-5* 単体を上回った．また，*Wilcoxon* 符号順位検定により性能向上の有意性を統計的に確認した（*HumanEval:* $p < 0.01$，*MBPP+:* $p < 0.01$）．

本研究により，*SE* タスクにおいても *LLM* カスケードが有効であることが実証された．今後はより高難度な *SE* タスクや，異なるモデルの組み合わせがコスト削減・性能に与える影響を検証することが課題となる．

1. はじめに

1.1. LLM のコスト問題

近年，大規模言語モデル（LLM: Large Language Model）は自然言語処理（NLP）タスクのみならず，ソフトウェア開発分野の様々なタスクにおいても急速に普及している．しかし，高性能な商用 LLM は API 利用コストが高額になることが懸念されている．例えば，GPT-5 と GPT-5-nano の価格差は 25 倍にもおよび，高頻度で大量のクエリを処理するアプリケーションで高コストモデルを用いると，費用は膨大なものとなる．その結果，予算が限られている実務において最高性能のモデルを全面的に採用することは，経済的合理性の観点から困難な選択となる場合がある [6] [7].

1.2. モデルルーティングと LLM カスケード

高性能な商用 LLM の利用コスト問題を解決する手法として，モデルルーティング [1] [6] [8] が注目されている．モデルルーティングとは，入力クエリの複雑性に応じて最適なモデルを動的に選択する仕組みであり，システム全体の性能を落とさずに運用コストを大幅に抑えられる点が最大の利点である．

そのモデルルーティングの一種として，LLM カスケード [1] がある．LLM カスケードは，複数の LLM の中で最も安価であるが性能が低いモデル（低コストモデル）から順にクエリを処理させ，その回答が不十分な場合のみ，高性能モデル（高コストモデル）にクエリを送り回答の再生成を行う．これにより，性能は維持しつつコストを削減できることが様々なベンチマークを用いて実験的に示されている [6] [1].

一方，既存研究 [6] [1] は NLP タスクの評価に留まっ

ており，ソフトウェア工学（SE）タスクでの効果は十分に検証されていない．また，SE タスクは生成コードの構文的正しさが求められる点や，テストケースによる動的かつ客観的な検証が可能である点で NLP タスクと大きく異なる．よって，コードを実際に実行することによる決定論的な正誤判定が可能である特性が，LLM カスケードの効果にどう影響するかも解明されていない．

そこで本研究では，SE タスクの中でも中心的な位置を占めるコード生成タスクに焦点を当て，HumanEval [2] および MBPP+ [3] を対象タスクとして LLM カスケードを行う．コード生成タスクは LLM の活用が最も活発な SE タスクであり，テストケースによる客観的な正誤判定が可能であることから，LLM カスケードの効果を定量的に評価するのに適している．これらに加え，NLP タスクである MMLU [4]，HEADLINES [5] も用いることで，SE タスクにおいても，LLM カスケードによるコスト削減と性能維持の両立が可能であるかを比較評価する．

2. 既存研究

2.1. FrugalGPT

FrugalGPT [6] は自然言語クエリ応答タスクを対象に，LLM カスケードの有効性を実証した手法である．FrugalGPT はプロンプト適応，LLM 近似，LLM カスケードの 3 つの戦略を組み合わせており，中核となる LLM カスケードでは，生成スコアリング関数によって回答の信頼度を評価し，閾値を超えた場合はその回答を採用し，不十分な場合はより高性能なモデルへ委譲する．実験の結果，HEADLINES [5]，OVERRULING [9]，CoQA [10] などのタスクで最大 98 %のコスト削減を達成した．

FrugalGPT では，クエリと生成された回答のペアを入力として正誤を予測するスコアリングモデルを用いて信頼度スコアを算出し，委譲の判断基準としている．このアプローチは，HEADLINES [5] や OVERRULING [9] のように答えの種類が限定されるタスクでは，スコアリングモデルが正誤パターンを学習しやすいため有効に機能する．一方，コード生成タスクなどでは，コードの正誤はテストを実行するまで確定しないため，スコアリングモデルによる事前予測は本質的に困難である．ただし，テストケースが利用可能であれば，スコアリングモデルを介さず直接テストを実行する方が，より決定論的な委譲判断が可能である．以上の理由から，本研究の評価実験では，FrugalGPT の LLM カスケードの構造を採用し，委譲基準のみをスコアリングモデルによる信頼度ベースの手法からテストケースの通過可否に置き換える．

2.2. RouteLLM

RouteLLM [8] は LLM カスケードとは異なり，クエリ入力前に最適なモデルを選択する事前ルーティング手法である．行列分解や BERT 分類器などを用いてクエリの難易度を推定し，高コストモデルが必要かどうかを事前に判断することで，応答品質を維持しながら利用コストを削減する．

しかし，SE タスクでは，コードの正しさは動くか動かないかの二値であり，構文的に正しいコードでも論理的に誤っている場合があるため，実際にテストを実行するまで正しさが確定できない．また，クエリの見た目上の難易度と生成されるコードの正しさには必ずしも相関があるわけではないため，コード生成タスクにおいては事前にクエリを分析してモデルを選択する手法では最適な判断が困難な場合がある．以上の理由から，本研究では RouteLLM のような事前ルーティング手法ではなく，生成後の検証に基づく LLM カスケード手法を採用する．

2.3. 既存研究の限界

既存研究では SE タスクを体系的に取り扱っておらず，SE タスクへの有効性の検証が不足している．また，現状 SE タスクと NLP タスクを同一の比較軸で分析した研究はない．本研究では，タスク特性による LLM カスケードの影響も比較することにより，既存研究のギャップを埋める．

3. 研究目的とリサーチクエスチョン

3.1. 研究目的

本研究の目的は，SE タスクにおける LLM カスケードの有効性を実証的に検証することである．具体的には，SE タスクに対するコスト削減効果の定量化，性能への影響の評価，タスク特性の解明などが挙げられる．また，既存研究の多くが NLP タスクを対象にしているという偏りの現状を踏まえて，SE タスクにおける体系的な評

価と実用的なコスト最適化の指針を提供できることを目指す．これにより，予算の限られた開発組織においても，高性能な LLM を経済的に活用できる可能性を提示する．

本研究で LLM カスケードに着目した理由は，先行研究である FrugalGPT [6] が NLP タスクにおいて最大 98 % のコスト削減効果を達成したことにある．LLM カスケードは，難易度分布が存在するタスクにおいて効果を発揮する．コード生成タスクにも，標準的な関数実装から複雑なアルゴリズムまで NLP タスクと同様の難易度分布が存在するため，LLM カスケードの適用が期待できる．さらに，SE タスクではテストケースの通過可否という客観的な判定基準が利用可能であり，FrugalGPT の信頼度ベースの判断より確実な委譲判断が期待できる．これらの理由から，同手法のコード生成タスクへの適用は学術的・実務的に意義があると考えた．

3.2. RQ1: コスト削減効果はどの程度か

最初のリサーチクエスチョンとして，LLM カスケードを適用した場合と高コストモデル単体を使用し続ける場合を比較し，LLM カスケードを行うとどの程度のコスト削減が可能であるかを検証する．ここでいうコストとは，API 利用者が各プラットフォームへと支払う金銭的コストのことを指す．また，既存研究として，NLP タスクで高いコスト削減効果があった FrugalGPT [6] と同様の効果が SE タスクでも現れるのかについても焦点を当てている．

3.3. RQ2: 性能はどの程度変化するのか

RQ2 では，高コストモデル単体を運用した場合と比較して，LLM カスケードの性能がどの程度変化するのかを検証する．本研究で採用する判定メカニズムでは，低コストモデルが失敗した場合に最終的に高コストモデルへ委譲されるため，LLM カスケードの性能が高コストモデル単体を下回ることは原理的にない．したがって本 RQ は「高コストモデル以外の活用によってどの程度性能に変化があるか，またその理由は何か」を明らかにすることが目的である．

本研究では各タスクの性質に応じた，客観的な判定基準を設ける．NLP タスクでは，あらかじめ用意された正解データとモデルの生成した回答とが一致しているかで正誤判定を行う（Exact Match）．SE タスクでは，コードの実行可能性と論理的妥当性を検証するため，生成された回答が事前に定義された全てのテストケースを通過できるかどうかを判定の閾値とする（Pass@1）．

3.4. RQ3: 各階層における正答数と委譲数はどの程度か

RQ3 では，クエリを処理する優先順位に従ってモデルを配置した LLM カスケードの各階層について，モデルごとの正答数および各モデルへと委譲された問題数を算出・分析する．

LLM カスケードの運用効率を最大化する鍵は，初動を担う低コストモデルがいかに多くのクエリを高コストモデルへ委譲せずに処理できるかという点に集約される．仮に，第 1 階層に配置した低コストモデルの成功率が高水準を維持しているならば，高額な計算資源を消費する，上位のモデルへのタスクの委譲が最小限に抑制されていることを意味し，結果として LLM カスケードの経済的恩恵をより享受できることを示唆している．

また，高コストモデルへ委譲された問題数とその正答数を分析することで，低コストモデルや中コストモデルで解決できなかった問題が高コストモデルでどの程度解決されるかを明らかにできる．よって，LLM カスケードの階層構造が適切に機能しているか，あるいは LLM カスケードの構成を見直す必要があるかを判断する指針も得られる．

4. 実験設定

4.1. LLM カスケード手法

本研究で採用する LLM カスケードの基本的なアルゴリズムを図 1 および図 2 に示す．

本アーキテクチャは推論能力と運用コストが異なる 3 段階のモデル層で構成される．第 1 階層の GPT-5-nano で回答 A_{weak} を生成・検証し，A_{weak} に対して後述する検証プロセスを実行する．検証の結果，あらかじめ設定された成功基準を満たした場合には，A_{weak} を最終出力として採用し，失敗と判定した場合は第 2 階層の GPT-5-mini へ委譲し，同様に検証を行う．第 2 階層でも失敗した場合は最終階層の GPT-5 へ委譲する．最終階層における検証が完了した時点で，LLM カスケードの全工程が終了する．なお，最終階層においても検証で成功基準を満たさなかった場合，当該タスクは不正解として記録する．

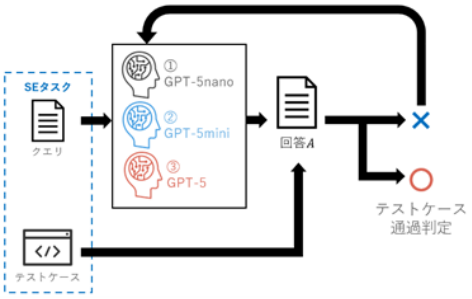

図 1. SE タスクにおける LLM カスケードの概略図

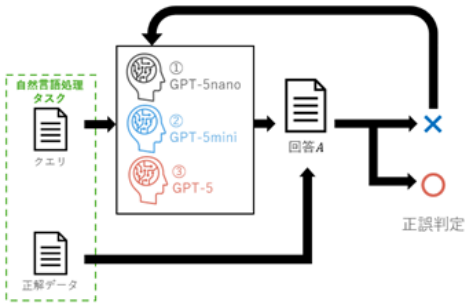

図 2. NLP タスクにおける LLM カスケードの概略図

4.2. データセットとモデル

本研究で対象となる SE タスクおよび NLP タスクを表 1 に示す．

HumanEval [2] は OpenAI 社が提供している全 164 問の Python のプログラミングベンチマークであり，内容は標準的な Python 関数についての問題である．各問題には関数シグネチャ，ドキュメント，テストケースが含まれている．

MBPP [11] は Google が提供している Python のプログラミング問題から構成されているベンチマークである．本研究では，MBPP の中でも EvalPlus [3] が提供する MBPP+を使用する．MBPP+は元の MBPP から曖昧な問題や不正確なテストケースを除外した計 378 問の検証済みサブセットであり，より信頼性の高い評価が可能である．各問題にはタスク説明，コード回答，テストケースが含まれている．

MMLU [4] は 15,000 問以上の多肢選択問題で構成された大規模な知識理解評価ベンチマークである．MMLU は数学，歴史，科学，法律など計 57 の異なる学問分野における問題が含まれており，正確な事実からなる知識，概念の理解，論理的推論能力が求められる．

表 1. 評価対象タスク一覧

タスクの種類	タスク内容	利用する問題数	評価指標
HumanEval	SE タスク（コード生成など）	164	Pass@1
MBPP+	SE タスク（コード生成など）	378	Pass@1
MMLU	NLP タスク（一般知識問題）	500	Exact Match
HEADLINES	NLP タスク（金融知識問題）	500	Exact Match

HEADLINES は FrugalGPT [6] でも使用されたベンチマークであり，元来は Sinha ら [5] によって金融・経済におけるニュースの影響を分析するために構築された．

本研究において，HumanEval と MBPP+はテストケースの通過可否をモデルの回答の正誤判定の指標とする．なお，正誤判定はあくまでも問題の仕様を満たす出力が生成できているかを測るものであり，コードの可読性や保守性といったコード品質を評価することはできない．コード品質の評価は本研究の目的であるモデル間のコスト性能比較の範疇外であるため，本研究ではこの評価軸は対象としない．

また，本研究では，実験の実行可能性とコスト効率を考慮し，各データセットのサンプル数を設定した．HumanEval については全 164 問，MBPP+については検証済みの全 378 問を使用する．NLP タスクについては，データセット全体が数万問規模に及ぶため，実験の実行可能性を考慮し，500 問をサンプリングして評価に用いることとした．また，API 利用コストの抑制と実験条件の統一を目的として，全モデルの出力トークン数上限は 2048 トークンに設定した．この設定は bigcode-evaluation-harness の公式ドキュメント [12] において，HumanEval および MBPP 系タスクには 512 トークンで十分であると述べられており，本設定ではこれを上回ることから妥当な設定であると判断した．

5. 評価指標

5.1. トークンベースによるコスト削減率

本研究では先行研究である FrugalGPT と同様に，API 利用コスト（ドルベース）の総額を比較することでコスト削減率を算出する．具体的には，GPT-5 単体使用時の総コストをベースラインとし，LLM カスケード適用時の総コストと比較する．コスト削減率は以下の式で定

表 2. 使用モデル一覧

階層	モデル名	入力コスト	出力コスト
低コスト	GPT-5-nano	$0.0005/10,000token	$0.004/10,000token
中コスト	GPT-5-mini	$0.0025/10,000token	$0.02/10,000token
高コスト	GPT-5	$0.0125/10,000token	$0.1/10,000token

義する．

$$\text{コスト削減率} = 1 - \frac{\text{LLM カスケード総コスト}}{\text{ベースライン総コスト}} \quad (1)$$

各モデルの API 利用コストは表 2 に示す通りである．

5.2. SE タスクにおける性能指標

本研究では SE タスクの評価指標として Pass@1 を用いる．Pass@1 は各問題に対して 1 回コードを生成し，全てのテストケースを通過した場合のみ正解とみなす指標であり，1 つ以上のテストケースが失敗した場合，あるいは構文エラーや実行時エラーが発生した場合は不正解として扱われる．また HumanEval の各問題には平均 7〜12 個，MBPP+には平均 3〜5 個のテストケースが含まれている．この基準により，部分的に正しいコードではなく，要求仕様を完全に満たす動作可能なコードのみが正解として評価される．

SE タスクのテストケースに基づく正誤判定は，NLP タスクにおける文字列一致判定と比較して，いくつかの特徴的な利点を持つ．第一に，テストケースは実行可能な仕様であるため，判定の客観性が極めて高い．生成されたコードの表記や実装方法が異なっていても，振る舞いが仕様を満たしていれば正解と判定される．第二に，複数のテストケースによる多角的な検証により，特定の入力に対してのみ動作する偶然の正解を排除できる．第三に，実行時エラーの検出により，構文的には正しいが実行不可能なコードを不正解として識別できる．これらの特性により，SE タスクにおける LLM カスケードの判定メカニズムは，高い信頼性を持ってタスクの委譲の要否を決定できると考えられる．

性能を測る式としては以下を用いる．

$$\text{ベースライン性能} = \frac{\text{GPT-5 正答数}}{\text{全問題数}} \quad (2)$$

$$\text{LLM カスケード性能} = \frac{\text{各コストモデルの正答数の合計}}{\text{全問題数}} \quad (3)$$

5.3. NLP タスクにおける性能指標

NLP タスクにおける正誤判定は，Exact Match を用いる．Exact Match は，生成された回答と事前に用意された正解データを比較することで正誤判定を行う．ただし，単純な文字列比較ではなく，生成された回答に対してマークダウン記号の除去や回答抽出などの前処理を行い，大文字小文字統一・空白除去・句読点削除の正規化後に正解データと比較する．例えば,「Paris」「paris」「Paris.」はすべて同一の回答として扱われる．正規化後のテキストを比較し，完全に一致した場合に正解と判定する．

性能を測る式としては，SE タスクと同様に式 (2)，式 (3) を使用する．

5.4. LLM カスケードにおける各階層の正答数と委譲数

LLM カスケードの各階層の正答数と委譲数を記録し，LLM カスケードの動作特性を分析する．具体的には，2 種類の指標を各モデルについて算出する．

第一に，各モデルの正答数である．低コストモデルの正答数が多いほど，高コストモデルの使用頻度が抑制され，全体のコスト削減効果が大きくなる．

第二に，中コストモデルおよび高コストモデルへ委譲された問題数である．委譲数が多い場合，前の階層のモデルでは対応困難な問題が多いことを示唆する．

さらに，委譲された問題数をタスク種別ごとに比較することで，タスク特性が LLM カスケードの効果に与える影響を明らかにできる．例えば，NLP タスクの方が SE タスクよりも低コストモデルの正答数が多かった場合，NLP タスクの方が LLM カスケードに適していると結論付けられる．この分析により，実務においてどのようなタスクに LLM カスケードを適用すべきかという意思決定の指針を提供できる．

本研究では，各モデルにおける正答数と委譲数を各データセットごとに測定し，タスク特性による違いを分析する．これにより，第 3 章で設定した RQ3 に対する明確な回答を得ることを目指す．

表 3. 各手法におけるコスト削減率の結果

タスク名	ベースライン総コスト	LLM カスケード総コスト	コスト削減率
HumanEval	$1.2879	$0.3702	71.26 %
MBPP+	$2.5853	$1.0869	57.96 %
MMLU	$1.8753	$0.4063	78.33 %
HEADLINES	$0.5446	$0.2974	45.39 %

表 4. 各手法におけるタスク正答数

タスク名	ベースライン単体	LLM カスケード
HumanEval(164 問)	143	149（+6）
MBPP+(378 問)	281	302（+21）
MMLU(500 問)	455	470（+15）
HEADLINES(500 問)	380	383（+3）

表 5. 各タスクにおける性能

タスク名	ベースライン単体	LLM カスケード	性能の変化
HumanEval(164 問)	87.20 %	90.85 %	+3.65 %
MBPP+(378 問)	74.34 %	79.89 %	+5.56 %
MMLU(500 問)	91.00 %	94.00 %	+3.00 %
HEADLINES(500 問)	76.00 %	76.60 %	+0.60 %

表 6. 各コストモデルの正答数

タスク名	GPT-5-nano	GPT-5-mini	GPT-5
HumanEval(164 問)	111	38	0
MBPP+(378 問)	233	68	1
MMLU(500 問)	410	55	5
HEADLINES(500 問)	374	5	4

表 7. 各コストモデルに委譲された問題数

タスク名	GPT-5-nano	GPT-5-mini	GPT-5
HumanEval(164 問)	164	53	15
MBPP+(378 問)	378	145	77
MMLU(500 問)	500	90	35
HEADLINES(500 問)	500	126	121

6. 実験結果

6.1. コスト削減効果はどの程度か

各タスクにおいて GPT-5 のみを使用した場合と，LLM カスケードを行った場合のドルベースでの総コストおよびコスト削減率を表 3 に示す.

表 3 より，全てのタスクにおいて GPT-5 のみを使用し続ける場合よりも，LLM カスケードを利用している方が総コストが削減されることが分かる．SE タスクでは，HumanEval が 71.26 %（$1.2879→$0.3702），MBPP+が 57.96 %（$2.5853 →$1.0869）のコスト削減を達成した．NLP タスクでは，MMLU が 78.33 %（$1.8753 →$0.4063），HEADLINES が 45.39 %（$0.5446 →$0.2974）の削減率となった.

これらの結果は，最高コストモデルである GPT-5 を使用せずとも，低コストモデルである GPT-5-nano や GPT-5-mini で正しく処理できる問題が多く存在し，GPT-5 の使用頻度を大幅に削減できることを示している．また，同じ SE タスクであっても HumanEval は 71.26 %，MBPP+は 57.96 %，NLP タスクにおいても MMLU は 78.33 %，HEADLINES は 45.39 %と，タスクの内容によってコスト削減率は大きく異なることが見て取れる.

6.2. 性能に変化があるのか

各データセットにおける問題の正答数を表 4 に，性能を表 5 に示す.

性能の検証においては，全タスクで LLM カスケードはベースラインを上回る性能を示した．HumanEval で+3.65 %，MBPP+で+5.56 %，MMLU で+3.00 %，HEADLINES で+0.60 %の向上が確認された．性能向上の要因については第 7 章にて考察を行う.

6.3. 各階層における正答数と委譲数はどの程度か

LLM カスケードにおける各コストモデルの正答数を表 6 に，各コストモデルへと委譲された問題数を表 7 に示す.

各モデルの正答数は，HumanEval では 111 問，MMLU では 410 問が GPT-5-nano のみで処理を完了しており，コスト削減の主な原動力となった．一方，GPT-5 に委譲された問題の正答数は HumanEval で 0 問，MBPP+で 1 問，MMLU で 5 問，HEADLINES で 4 問と極めて低かった.

7. 考察

7.1. SE タスク間の差異

実験結果から，SE タスクにおいても LLM カスケードは有効であることが確認されたが，コスト削減効果はタスクの特性によって異なることも分かった.

原因としては，問題仕様の明確さの違いが考えられる．HumanEval では，関数シグネチャや入出力例が明示されているため低コストモデルでも正確に処理できる．一方 MBPP+は自然言語説明のみであり，モデルが関数シグネチャを推測する必要がある．この仕様の曖昧さによっ

```
def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """ Check if in given list of numbers, are any two numbers closer to each other than
    given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
```

図 3. HumanEval のタスクの一例

```
%プロンプト
 "Write a function to find squares of individual elements in a list."

%テストケース
 assert square_nums([1,2,3,4,5,6,7,8,9,10])==[1,4,9,16,25,36,49,64,81,100]
 assert square_nums([10,20,30])==([100,400,900])
 assert square_nums([12,15])==([144,225])
```

図 4. MBPP+のタスクの一例

て，低コストモデルの誤解釈が増え委譲が多く発生したと考えられる．具体的な HumanEval と MBPP+のプロンプト文は図 3，図 4 に示す．

図 3 に示される HumanEval では，関数シグネチャ（関数名，引数名，引数の型，返り値の型）が明示的に定義されており，docstring 内には関数の動作説明と具体的な入出力例が含まれている．このように問題仕様が明確に構造化されているため，モデルは期待される動作を正確に把握した上でコードを生成できる．

一方，図 4 に示される MBPP+では，自然言語による問題説明のみが与えられ，関数シグネチャは提供されない．関数名，引数名，引数の型，返り値の型はすべてモデルが自ら推測する必要があり，この仕様の曖昧さが低コストモデルでの誤解釈を招きやすいと考えられる．

7.2. 性能向上の要因

実験結果より，GPT-5-nano と GPT-5-mini の正答数の合計が，GPT-5 単体の正答数を上回るケースが確認された．例えば HumanEval では，GPT-5 単体が 143 問正解であるのに対し，GPT-5-nano と GPT-5-mini の合計は 149 問であった．

具体的には表 8 に示す 6 つのタスクが GPT-5 単体では不正解であったが，LLM カスケードでは GPT-5-mini によって正解となった．正答数の結果から，低コストモデルが解決できて GPT-5 が解決できない問題が存在することが示唆される．

表 8 に示した 6 つのタスクについて，GPT-5 の出力トークン数を分析した結果を表 9 に示す．

表 8. GPT-5 単体で不正解，LLM カスケードで正解となったタスク

タスク ID	内容	正解モデル
HumanEval/47	中央値の計算	GPT-5-mini
HumanEval/76	べき乗判定	GPT-5-mini
HumanEval/129	グリッド最短パス	GPT-5-mini
HumanEval/130	隣接要素入れ替え判定	GPT-5-mini
HumanEval/132	ネストした括弧判定	GPT-5-mini
HumanEval/147	要素数と合計の積	GPT-5-mini

表 9. GPT-5 単体で不正解となったタスクの出力トークン分析

タスク ID	出力トークン数	回答の有無	原因
HumanEval/47	2048	なし	上限到達
HumanEval/76	2048	なし	上限到達
HumanEval/129	2048	なし	上限到達
HumanEval/130	2048	なし	上限到達
HumanEval/132	2048	なし	上限到達
HumanEval/147	1438	あり	ロジックエラー

表 9 より，6 件中 5 件において出力トークン数が設定上限である 2048 トークンに到達していたことが判明した．GPT-5 は詳細な説明を含む冗長な回答を生成する傾向があり，トークン上限に達して出力が途中で打ち切られた結果，不完全なコードとなりテストに失敗したと考えられる．一方，GPT-5-mini は簡潔な回答を生成したため，同じタスクでも正常にコードを出力できた．

なお，HumanEval/147 では出力トークン数が上限に達していないにもかかわらず不正解となった．このタスクは，配列 $a[i] = i * i - i + 1$ の要素から，3 つの要素の和が 3 の倍数となる組み合わせの数を求める問題である．

図 5 に問題文，図 6 に GPT-5 の回答，図 7 に GPT-5-mini の回答を示す．両モデルとも同じアルゴリズム（3 で割った余りによる分類と組み合わせ計算）を採用しているが，実装の細部が異なる．GPT-5 は境界条件として `n < 3` の場合に `0` を返し，内部関数 `comb3` を定義して組み合わせを計算している．一方，GPT-5-mini は境界条件を設けず，if 文による直接計算を行っている．結果として，GPT-5-mini の実装のみがテストケースを通過した．この事例は，高性能モデルが必ずしも全てのタスクで優位とは限らず，実装スタイルの違いが結果に影響することを示している．

```
def get_max_triples(n):
    """
    You are given a positive integer n. You have to create an integer array a of length
n.
        For each i (1 ≤ i ≤ n), the value of a[i] = i * i - i + 1.
        Return the number of triples (a[i], a[j], a[k]) of a where i < j < k,
    and a[i] + a[j] + a[k] is a multiple of 3.

    Example :
        Input: n = 5
        Output: 1
        Explanation:
        a = [1, 3, 7, 13, 21]
        The only valid triple is (1, 7, 13).
    """
```

正解コード

```
if n <= 2: return False
    one_cnt = 1 + (n - 2) // 3 * 2 + (n - 2) % 3
    zero_cnt = n - one_cnt
    return one_cnt * (one_cnt - 1) * (one_cnt - 2) // 6 + zero_cnt * (zero_cnt - 1) *
(zero_cnt -2) // 6
```

図 5. HumanEval/147 の問題文

```
if n < 3:
        return 0
    q, r = divmod(n, 3)
    c0 = q + (1 if r >= 2 else 0)  # count of i with i % 3 == 2 => a[i] % 3 == 0
    c1 = n - c0                    # count of i with i % 3 in {0,1} => a[i] % 3 == 1

    def comb3(m):
        return (m * (m - 1) * (m - 2)) // 6 if m >= 3 else 0

    return comb3(c0) + comb3(c1)
```

図 6. HumanEval/147 に対する GPT-5 の回答（不正解）

各タスクの分析結果から，低コストモデルが GPT-5 より優れた結果を示した要因として以下が考えられる．第一に，出力の簡潔さの違いが挙げられる．表 9 に示したように，GPT-5 は冗長な回答を生成してトークン上限に達する傾向があるのに対し，GPT-5-mini では簡潔な回答を生成するため，同じタスクでも正常にコードを出力できた．

第二に，モデル間の特性の違いが考えられる．異なるモデルは異なる学習過程を経ており，特定の問題パターンに対する得意・不得意が異なる可能性がある．ただし，LLM の出力は確率的であるため，同一のタスクに対して再度実験を行った場合，GPT-5 がトークン上限に達せず正解する可能性も存在する．なお，実験全体の再現性については，後述の 7.3 節にて 50 回の繰り返し試行による統計的検定を実施しているため，性能向上の有意性は確認されている．よって，個別タスクレベルでの再現性の詳細な検証が今後の課題である．

一方，性能向上が限定的にとどまった要因としては，GPT-5 の出力特性（トークン上限・実装スタイル）に起

```
rem = n % 3
    base = n // 3
    # indices with i % 3 == 2 produce a[i] % 3 == 0
    c0 = base + (1 if rem >= 2 else 0)
    # the rest produce a[i] % 3 == 1
    c1 = n - c0
    result = 0
    if c0 >= 3:
        result += c0 * (c0 - 1) * (c0 - 2) // 6
    if c1 >= 3:
        result += c1 * (c1 - 1) * (c1 - 2) // 6
    return result
```

図 7. HumanEval/147 に対する GPT-5-mini の回答（正解）

表 10. 統計的検定の結果

タスク	平均性能差	中央値	p 値
HumanEval	+4.795%	+4.880%	< 0.01
MBPP+	+5.164%	+5.030%	< 0.01

因する偶発的な失敗であると考えられる．よって LLM カスケードが性能改善できるようなケースは全問題の一部にすぎず，改善幅に上限が生じるものと考えられる．

7.3. 統計的検定による有意性の確認

性能向上が偶然によるものではないことを統計的に検証するため，HumanEval および MBPP+について，GPT-5 単体と LLM カスケードの性能差を 50 回の試行で計測し，Wilcoxon 符号順位検定を実施した．

検定の結果（表 10），LLM カスケードはベースラインに対して有意に高い性能を示した（$\alpha = 0.01, p < 0.01$）．

また 50 回の試行を行った結果，HumanEval では平均性能差 +4.795%（中央値 +4.880%），MBPP+では平均性能差 +5.164%（中央値 +5.030%）を示した．

さらに，全 50 回の試行において性能差が正であったことも，LLM カスケードの性能優位性の一貫した再現性を示している．

以上より，LLM カスケードによる性能向上は統計的に有意であり，偶然による結果ではない可能性が高い．

7.4. 高コストモデルへ委譲された問題の分析

GPT-5 に委譲された問題に対する正答率を表 11 に示す．GPT-5 に委譲された問題の正答率は最高でも 14.29 %と極めて低く，LLM カスケードの階層を増やせば増やすほど効果があるものではなく，性能向上には一定程度の限界が存在すると考えられる．カスケードを利用し

表 11. LLM カスケードにおける GPT-5 の正答率

タスク	問題数	正答数	正答率
HumanEval(164 問)	15	0	0 %
MBPP+(378 問)	77	1	1.30 %
MMLU(500 問)	35	5	14.29 %
HEADLINES(500 問)	121	4	3.31 %

たコード自動生成タスクなどでは，本質的に困難な問題には人間によるレビューを補完的に実施するなど，別のアプローチが必要であることが示唆される．

7.5. 妥当性への脅威

7.5.1 内的妥当性

本研究は HumanEval と MBPP+の Python コード生成タスクのみを対象としており，バグ修正やテスト生成といった他の SE タスクへの適用可能性は不明である．さらに，本研究では GPT-5 シリーズのみを使用して LLM カスケードを構築しており，異なる LLM（Claude，Gemini 等）を組み合わせた場合に同様の効果が得られるかは検証していない．

7.5.2 外的妥当性

本研究で使用した HumanEval や MBPP+は標準化されたテストケースを持つ学術的ベンチマークであり，仕様が曖昧であったり既存コードへの変更を含む実務レベルのタスクを完全に代表するものではない．本研究の結果はあくまでこれらのベンチマーク条件下における効果を示すものであり，SWE-Bench [13] のようなより実務的なベンチマークへの拡張は今後の課題とする．

8. まとめと今後の課題

本研究では，SE タスクにおける LLM カスケードの有効性を実証的に検証した．GPT-5-nano，GPT-5-mini，GPT-5 の 3 階層からなる LLM カスケードを構築し，SE タスク（HumanEval，MBPP+）と NLP タスク（MMLU，HEADLINES）の計 4 つのベンチマークを用いて評価を行った．

コスト削減効果については，HumanEval で 71.26 %，MBPP+で 57.96 %，MMLU で 78.33 %，HEADLINES で 45.39 %の削減を達成し，SE タスクにおいても NLP タスクと同等以上の効果が得られることが確認された．

性能については，全てのタスクにおいて LLM カスケードがベースラインである GPT-5 単体と同等以上の値を達成した．この結果は，低コストモデルが誤答した場合でも検証プロセスによりそれが検出され，より高性能なモデルへ処理が委譲されることで，最終的な回答品質が担保されているためと考えられる．さらに，HumanEval および MBPP+について Wilcoxon 符号順位検定を実施した結果，いずれも $p < 0.01$ となり，性能向上が偶然によるものではないことが統計的に確認された．

タスク特性の影響については，HumanEval と MBPP+の両方で高いコスト削減効果が得られ，仕様が明確で標準的なコード生成問題では LLM カスケードが効果的であることが示された．

各階層の分析については，GPT-5-nano が全タスクでコスト削減の主な原動力となった一方，GPT-5 への委譲問題の正答率は最高でも 14.29 %にとどまり，LLM カスケードの性能向上には一定の限界が存在することが示された．

本研究により，SE タスクにおいても LLM カスケードがコスト削減と性能維持の両立において有効であることが実証された．この知見は，予算制約のある開発組織が LLM を経済的に活用するための指針を提供するものである．

今後の課題としては，他の商用 LLM およびオープンソース LLM への適用と，SWE-Bench [13] などのより高度な SE タスクにおけるカスケーディング手法の効果の検証が挙げられる．

謝辞

本研究の一部は，文部科学省科学研究費補助金（基盤（B）：26K02890）による助成を受けた．

参考文献

[1] David Dohan, Winnie Xu, Aitor Lewkowycz, et al. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[3] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[4] Dan Hendrycks, Collin Burns, Steven Basart, et al. Measuring massive multitask language understanding. *Proceedings of ICLR*, 2021.

[5] Ankur Sinha and Tanmay Khandait. Impact of news on the commodity market: Dataset and results. In *Advances in Information Retrieval (ECIR 2021)*, pp. 514–527. Springer, 2021.

[6] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. Survey Certification.

[7] Dylan Patel and Andafzal Ahmad. The inference cost of search disruption – large language model cost analysis, 2023.

[8] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[9] Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? assessing self-supervised learning for law and the casehold dataset, 2021.

[10] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, Vol. 7, pp. 249–266, 2019.

[11] Jacob Austin, Augustus Odena, Maxwell Nye, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[12] BigCode Project. Bigcode evaluation harness. `https://github.com/bigcode-project/bigcode-evaluation-harness/blob/main/docs/README.md`, 2023. Accessed: 2025-04-29.

[13] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.