コグニティブアーキテクチャから実稼働アプリケーションへ：高度なシステムプロンプトの設計、評価、およびセキュリティ確保のための実践的フレームワーク

Part I: 応用プロンプトクラフト - 専門ドメインにおける最先端のシステムプロンプト

これまでの理論的探求から、システムプロンプトが単なる指示ではなく、大規模言語モデル（LLM）の動的な「コグニティブアーキテクチャ」を構築する強力なツールであることが明らかになった[1]。本パートでは、この理論を実践へと橋渡しするため、医療、法律、教育、ソフトウェア開発という4つの専門ドメインで実際に高いパフォーマンスを発揮している最先端（SOTA）のシステムプロンプトを解体し、その設計パターンを抽出する。これにより、抽象的な概念が具体的な応用技術へと昇華されるプロセスを詳述する。

医療ドメイン：診断推論と共感的対話のためのプロンプト設計

医療分野におけるプロンプトは、事実の正確性、複雑な臨床推論のシミュレーション、そして共感的なコミュニケーションという三つの要素を高度に両立させる必要がある。このドメインは、単純な指示追従から、構造化された認知的足場（コグニティブスキャフォールディング）の構築へとプロンプト設計が進化する必要性を示す典型例である。
学術研究では、臨床医の思考プロセスを模倣するよう設計された「診断推論プロンプト」が提案されている[2]。これらは、モデルに対し「鑑別診断CoT（Chain-of-Thought）」、「直感的推論CoT」、「分析的推論CoT」、「ベイズ推論CoT」といった特定の認知プロセスを実行するよう明示的に指示する。このアプローチは、LLMに内蔵された高速で直感的な「システム1」的応答傾向を抑制し、より熟慮的で分析的な「システム2」的思考を誘発するという、先行研究の理論的枠組みを直接的に応用したものである[1]。実験結果は示唆に富んでおり、GPT-3.5のような旧世代モデルではこれらの複雑なプロンプトによって性能が低下した一方、GPT-4では性能が安定、あるいは向上したことが報告されている。これは、高度な認知的足場を構築するプロンプトの有効性が、基盤となるモデルの能力に大きく依存することを示している。
実際の医療従事者向けに作成されたプロンプト集を見ると、特定のタスクに対して構造化された出力を要求するパターンが顕著である[3]。例えば、「[病状]について、新たに診断された患者向けに平

易な言葉で説明せよ。3つの例を挙げよ。各説明は最大200語とせよ」といったプロンプトは、語数制限や具体例の数を指定することで、簡潔さと実用性を両立させている。コミュニティで開発された医療診断用プロンプトも同様に、明確なペルソナ設定と鑑別診断の要求を含む、構造化された多段階プロセスを重視している [4]。
これらの分析から浮かび上がるのは、最も効果的な医療プロンプトが単に診断を求めるのではなく、LLMに臨床医の認知ワークフロー全体をシミュレートさせるという点である。これは、LLMの能力が特定のペルソナをプロンプトで与えられることで発現するという「シミュレーション理論」の具体的な現れと言える [1]。単純に「この患者を診断せよ」と指示した場合、モデルは直感的で誤りやすい単一の回答を生成しがちである。しかし、実際の臨床医は、症状を収集し、可能性のある疾患リスト（鑑別診断）を作成し、証拠に基づいて体系的に除外していくという構造化された推論プロセスを用いる。最先端のプロンプトは、この多段階プロセスを「鑑別診断を作成し、その後、正しい応答を決定せよ」といった形で指示に組み込むことで、モデルにその思考プロセスを外部化させる [2]。結果として、プロンプトは単なる指示書ではなく、タスク実行中、モデルに一時的な認知アーキテクチャを課す設計図として機能する。この外部化された推論は、出力の信頼性を高めるだけでなく、人間の専門家がその妥当性を検証することを容易にするという、解釈可能性の向上にも寄与する。

法律ドメイン：高忠実度な契約分析と法的推論のためのプロンプトエンジニアリング

法律分野のプロンプトは、極めて高い精度、厳格な制約の遵守、そして複雑で構造化された文書をナビゲートする能力を要求される。このドメインのプロンプトは、曖昧さを排除し、信頼性を確保するために、役割（ペルソナ）、ルール、そして明確な出力形式を多用する点が特徴である。
その代表例が、「LEGAL CONTRACT RED FLAG DETECTOR」のような契約書レビュー用プロンプトである [5]。このプロンプトは、
<Role>、<Context>、<Instructions>、<Constraints>、<Output_Format>といった明示的なセクションで構成されており、これは先行研究で特定された「GEM」モデル（ペルソナ、タスク/ルール、コンテキスト、出力形式）の構造と完全に一致する [1]。指示内容は極めて詳細で、「文書をセクションごとに体系的に分析せよ」と命じ、「曖昧または漠然とした文言」や「不公平または一方的な条件」といった具体的な危険信号を特定するよう要求する [5]。これは、プロンプト自体がタスク分解の戦略を内包していることを示している。
また、「All-Purpose Legal Document Drafter」プロンプトは、AIがまず「本日はどのような種類の法的文書を作成しますか？」といった明確化のための質問を投げかけ、ユーザーとの対話を通じてコンテンツを生成するという、動的なアプローチを採用している [6]。これにより、プロンプトは静的な指示セットから、対話的なワークフローへと進化している。学術研究においても、IRAC（Issue, Rule, Application, Conclusion）のような特定の法的推論技術から導出されたプロンプトが、一般的なCoTプロンプトよりも優れた結果を生むことが確認されている [7]。さらに、プロンプトとナレッジグラフを統合し、推論の一貫性と引用の正確性を向上させる研究も進んでいる [8]。
これらの高度な法律プロンプトは、LLMに特定の法的方法論を課す実行可能な法的フレームワークとして機能する。単に法的見解を求めるのではなく、モデルに、あらかじめ定義された厳密な業務手順に従うパラリーガルのように振る舞うことを強制するのである。法律業務は本質的に手続き的であ

り、弁護士は契約書を読む際に、補償、解除、責任などの潜在的な問題点を網羅した精神的なチェックリストに従って分析を行う。前述の「RED FLAG DETECTOR」プロンプトは、この精神的チェックリストを具体的な指示の集合として外部化している。プロンプトに列挙された危険信号のリストこそが、分析のフレームワークそのものなのである。さらに、「要約」「危険信号」「欠落要素」「推奨事項」「免責事項」といった厳格な出力形式の指定は、単なる体裁のためではない。それは、モデルの分析結果を専門的な基準に沿って構造化させ、法務レビューのすべての重要要素が確実に網羅されることを保証するための機能的要件である。したがって、特定のペルソナ（「エキスパート法律契約アナリスト」）と詳細な手続き的スクリプトの組み合わせは、LLMの広大な潜在能力を、特定の法的タスクという狭く、厳格な水路へと効果的に制約するプログラムとして作用する。

教育ドメイン：教育学的深度とソクラテス式対話を備えたAIチューターの設計

教育、特にAIチューターのためのプロンプトは、単なる情報提供者ではなく、教育戦略そのものを体現する必要がある。その目標は答えを与えることではなく、学生が自ら答えを発見するプロセスを導くことであり、情報提供型のプロンプトから、対話的で問いかけを基本とするフレームワークへの転換が求められる。

この分野では、ソクラテス式対話法が主要なテーマとなっている。プロンプトはAIに対し、「決して即座に答えを提供しない」「導き出すような質問をすることで、学生が自らの答えを生成するのを助ける」といった行動を明確に指示する[10]。これは、否定的な制約（「～するな」）と肯定的な指示（「～せよ」）を組み合わせることで、AIの振る舞いを定義する典型例である。「Skill Mastery AI Tutor」プロンプトはさらに一歩進んでおり、「能動的想起（active recall）、間隔反復（spaced repetition）、チャンキング（chunking）といった実証済みの学習テクニックを適用する」という指示を組み込むことで、AIプロンプティングと教育科学を高度に融合させている[11]。

AIチューター用のプロンプトには、まず学生の学習レベルや事前知識を尋ねてから説明を始めるといった、詳細な対話フローが含まれることが多い[10]。これにより、適応的な対話ループが形成される。また、プロンプトがAIに「忍耐強い家庭教師」「仲間」「教えられる側の生徒」といった役割を動的に設定させる「ダイナミックロールセッティング」も重要な概念である[13]。これは、ペルソナが複雑な行動を効率的に圧縮して伝達する高帯域幅のメカニズムであるという先行研究の知見と一致する[1]。

優れたAIチューターのプロンプトは、単なる対話の指示書ではなく、特定の教育理論を実行するための教育学的エンジンである。プロンプトは、学生がAIとプレイする教育ゲームのルールをコード化している。効果的な教育とは、情報の伝達ではなく、発見を促進すること（構成主義）であり、ソクラテス式対話法はそのための古典的なアルゴリズムである。AIチューターのプロンプトは、この抽象的な教育原理を、「もし学生が行き詰まったら、ヒントを与えよ」「もし学生が理解を示したら、自分の言葉で説明させよ」「常に応答を質問で終えよ」といった、LLMのための具体的かつ操作可能なルールに変換する[10]。これらのルールは、対話を導く一種の状態機械（ステートマシン）を形成し、AIは探求、葛藤、発見、言語化という状態を学生が遷移していくよう、この教育学的ロジックを実行する役割を担う。したがって、プロンプトはLLMを教育アルゴリズムでプログラミングしているに等しい。「明るく、励ますような」ペルソナ設定は単なるスタイルではなく、学習成果に不可欠な学生のエンゲージメントを維持し、フラストレーションを軽減するために設計された、教育学的エンジンの機能的な構成要素なのである。

ソフトウェア開発ドメイン：安全で効率的、かつ保守可能なコード生成のための高度なプロンプト

ソフトウェア開発におけるコード生成プロンプトは、「Xを行う関数を書いて」という単純な命令から、コンテキスト、制約、セキュリティ要件を組み込んだ複雑な指示へと進化している。これは、コードを単なるテキストではなく、構造化され、高い信頼性が求められる成果物として扱うパラダイムシフトを反映している。
効果的なプロンプトは、まずペルソナと制約を定義する明確なシステムメッセージから始まる。例えば、「あなたはジュニア開発者にコーディングを教えることができる、親切なコードアシスタントです。使用言語はPythonです。コードの説明はせず、コードブロック自体を生成してください」といった指示は、簡潔で実用的な出力を促す [15]。タスクの分解も一般的なパターンであり、高レベルの目標をコメントブロック内で番号付きのステップに分割し、それをLLMにコードへ翻訳させる手法が用いられる [15]。
コンテキストの提供は極めて重要である。例えば、SQLクエリを生成するプロンプトでは、テーブルスキーマを直接プロンプト内に含める [15]。これは、モデルの出力を特定のデータ構造に根拠づけるインコンテキスト学習の一形態である。近年の研究は、特に
安全なコード生成に重点を置いている。セキュリティに特化した接頭辞の追加や、反復的な改善といった様々なプロンプティング技術が、脆弱なコードの生成をどの程度削減できるかについて、ベンチマーク評価が進められている [16]。これは、開発者の関心が「動作するか？」から「安全か？」へと移行していることを示している。さらに、この分野はコードの実行結果に基づいてプロンプトを反復的に改善する「Prompt Alchemy」のような自動プロンプト最適化（APO）技術へと向かっており、これは先行研究で触れられたAPOの概念と一致する [1]。
これらの高度なコード生成プロンプトは、LLMを単なるテキスト生成器としてではなく、ソフトウェアライブラリの仮想**API**として扱っている。開発者がライブラリを使用する際、その内部実装を知る必要はなく、どのような入力を受け取り、どのような出力を返すかというAPI仕様を理解すればよい。"""
Table departments... Table students... Create a MySQL query for all students in the Computer Science department """ [15] のようなプロンプトは、事実上のAPIコールである。テーブルスキーマは必須パラメータ、「Create a MySQL query」は関数名、そして生成されたSQLが戻り値に相当する。反復的改善 [17] や自動最適化 [18] といった新たな潮流は、このプロセスにテスト層を追加する。これにより、プロンプトエンジニアリングのサイクルは「プロンプト作成 → コード生成 → コードテスト → 失敗分析 → プロンプト改善」となり、テスト駆動開発（TDD）のような現代的なソフトウェア開発手法を彷彿とさせる。この文脈において、プロンプトは単なる要求ではなく、新しい種類のソフトウェア開発ライフサイクルにおけるソースファイルそのものなのである。

統合分析：ドメイン横断的な高性能プロンプトアーキテクチャの原則

各ドメインの分析から、分野を超えて通用する普遍的な設計原則が浮かび上がる。

- 原則1：ペルソナ・プロセス・プロダクト（3P）フレームワーク
  最先端のプロンプトは、AIが「誰であるか」（ペルソナ）、どのように「思考」し、振る舞うべきか（プロセス）、そして出力が満たすべき構造と制約（プロダクト）を明確に定義する。医療プロン

プトはプロセス（診断推論）を、法律プロンプトはプロダクト（構造化された分析）を、教育プロンプトはペルソナ（ソクラテス式家庭教師）を特に重視する傾向がある。

- 原則2：タスク分解と認知的足場
  複雑なタスクは、決して一つの塊として扱われない。プロンプト自体の中で、より小さく、順序立てられたステップ（例：CoT、IRAC、多段階のコードコメント）に分解される。これにより、モデルの認知的負荷が軽減され、より信頼性が高く、監査可能な出力が生成される。
- 原則3：制約駆動型の生成
  最も堅牢なプロンプトは、肯定的な指示（「～せよ」）と否定的な制約（「～するな」「金融アドバイスは提供しない」）を組み合わせる。これらのガードレールは、特にハイステークスなドメインにおいて、安全性と信頼性を確保するために不可欠である。
- 原則4：コンテキストが王様
  パフォーマンスは、提供されるコンテキストの質と関連性に直接的に依存する。それが法律文書、患者の病歴、データベーススキーマ、あるいは学生の事前知識であれ、特定のプロンプト内データでLLMをグラウンディングさせること（RAGの中核原則）は、すべてのドメインに共通する普遍的なパターンである。

ドメイン	主要目標	ペルソナの典型	主要なプロセス/推論技術	主な制約	代表的な出力形式
医療	正確な診断支援と患者への説明	専門の診断医、共感的な医療従事者	鑑別診断、分析的推論 (CoT)	医療アドバイスの提供禁止、提供された情報源への準拠	鑑別診断リスト、ステップバイステップの推論、平易な言葉での説明
法律	契約リスクの特定と法的文書の起草	経験豊富な法律契約アナリスト、パラリーガル	IRAC (Issue, Rule, Application, Conclusion)、体系的な条項分析	法的助言の提供禁止、管轄区域の明示、免責事項の付与	危険信号のリスト、欠落条項の指摘、推奨される修正案、構造化された要約
教育	学生の自己発見的学習の促進	忍耐強いソクラテス式家庭教師、励ますコーチ	ソクラテス式対話法、能動的想起、間隔反復	即座に答えを与えない、常に質問で終える、学習レベルへの適応	導き出す質問、段階的なヒント、学習計画、進捗確認クイズ
ソフトウェア開発	安全で効率的なコードの生成と補完	親切なコードアシスタント、特定の言語の専門家	タスク分解、インコンテキスト学習（スキーマ提供）	特定言語の使用、説明の省略、セキュリティベストプラクティスの遵守	整形されたコードブロック、SQLクエリ、データベーススキーマ

Part II: 品質保証と継続的改善のためのフレームワーク

Part Iで設計されたプロンプトの品質を保証し、継続的に改善していくためには、体系的な評価とデバッグのプロセスが不可欠である。本パートでは、主観的な「良さそう」という評価から脱却し、客観的な指標に基づいた品質保証、潜在的な脆弱性を発見するための敵対的テスト、そして問題発生時の原因特定と修正を行うためのデバッグ手法について、実践的なフレームワークを提示する。

定量的評価：LLMパフォーマンスメトリクスの実践ガイド

プロンプトの品質保証を確立するためには、厳格で指標に基づいたアプローチが不可欠である。これには、従来の自然言語処理（NLP）メトリクスの限界を理解し、LLMおよび検索拡張生成（RAG）システムのために特別に設計された新しい評価指標群を導入することが含まれる。
BLEUやROUGEといった従来のメトリクスは、生成されたテキストと参照テキストの表層的な単語の重複度（n-gram）を測定するが、意味的な品質や事実の正確性を捉えることはできない[19]。そのため、LLMの評価にはより高度な指標が必要となる。特にRAGベースのカスタムGEMの評価においては、以下の指標が中心となる[20]。

- **Faithfulness (忠実性):** 生成された回答が、提供されたコンテキスト情報に事実として基づいているかを測定する。これは、ハルシネーション（幻覚）を検出するための主要な指標である。
- **Answer Relevancy (回答の関連性):** 生成された回答が、ユーザーの質問に直接的かつ適切に応答しているかを測定する。
- **Contextual Precision (文脈の適合率):** RAGシステムの検索コンポーネントを評価する。検索されたコンテキストの中で、関連性の高い情報源が、関連性の低いものよりも上位にランク付けされているかを測定する。
- **Contextual Recall (文脈の再現率):** 同じく検索コンポーネントを評価し、回答を生成するために必要な情報が、検索されたコンテキストの中にすべて含まれていたかを測定する。

これらの指標に加え、BERTScoreやCOMETのようなモデルベースのメトリクスは、単語の埋め込み表現を用いて意味的な類似性を評価するため、n-gramベースの指標よりも人間の判断に近い、より繊細な評価が可能となる[21]。また、評価プロセスを大規模化する手法として「LLM-as-a-judge」が注目されている。これは、GPT-4のような高性能なモデルに対し、詳細な評価基準（ルーブリック）をプロンプトとして与え、他のモデルの出力を採点させるアプローチである[19]。
この評価アプローチの進化は、LLMシステムの評価がもはや単一のスコアで語れるものではなくなったことを示している。評価は、システムを構成する各コンポーネント（検索器、生成器）を、異なる軸（関連性、忠実性、安全性）に沿って診断する、多次元的なプロセスへと変化した。RAGベースのシステムでは、不適切な応答の原因が、検索器が正しい情報を見つけられなかったこと（低いContextual Recall/Precision）にあるのか、あるいは生成器が正しい情報を持ちながらもハルシネーションを起こしたこと（低いFaithfulness）にあるのかを切り分ける必要がある。したがって、適切な評価フレームワークは単一のテストではなく、効果的な根本原因分析を可能にするための診断ダッシュボードとして機能しなければならない。これは、一枚岩のモデルを評価することから、複数コンポーネントからなる情報処理パイプラインを評価することへの、根本的なパラダイムシフトを意味する。

メトリック名	評価対象	定義	ユースケース	要件
Faithfulness	生成器 (Generator)	生成された回答が、提供されたコンテキスト情報に事	ハルシネーションの検出。	参照不要 (Reference-free)。

		実として基づいているかの度合い。		
Answer Relevancy	生成器 (Generator)	生成された回答が、ユーザーの質問に直接的かつ簡潔に応答しているかの度合い。	回答の焦点と簡潔性の評価。	参照不要 (Reference-free)。
Contextual Precision	検索器 (Retriever)	検索されたコンテキストの中で、関連性の高い情報源が上位にランク付けされているかの度合い。	検索結果のランキング品質評価。	正解データ (Ground truth) が必要。
Contextual Recall	検索器 (Retriever)	正解の回答を生成するために必要な情報が、検索されたコンテキストにすべて含まれているかの度合い。	検索された情報の網羅性評価。	正解データ (Ground truth) が必要。
BERTScore	生成器 (Generator)	参照テキストと生成テキスト間のトークンの意味的類似性を埋め込み表現を用いて計算する。	意味的な品質と流暢性の評価。	参照テキスト (Reference text) が必要。
G-Eval (LLM-as-a-judge)	システム全体	高性能LLMに評価基準を与え、他のモデルの出力をスコアリングさせる手法。	拡張可能で自動化された品質評価。	高性能な評価用LLMが必要。

敵対的テスト：体系的なレッドチーミングプロトコルの実装

レッドチーミングとは、悪意のある攻撃者が発見する前に、自社のシステムに対して意図的かつ体系的に攻撃を仕掛けることで脆弱性を発見する実践である。これは、デプロイされたLLMアプリケーションの安全性とセキュリティを確保するための不可欠なプロセスである。
体系的なレッドチーミングは、以下のステップで進められる[22]。

1. 弱点の特定**:** システムアーキテクチャを評価し、脆弱性が存在する可能性が高い箇所を特定する（例：RAGシステムは外部データソース経由の攻撃に脆弱である）。
2. 攻撃の選択**:** プロンプトインジェクション、個人情報漏洩、バイアス誘導など、広範な攻撃タイプを試行し、システムがどの攻撃に対して最も脆弱であるかを発見する。

3. 脆弱性の定義**:** 成功した攻撃に基づき、注力すべき重要な脆弱性を正式に定義する（例：「PDFアップロードを介した間接プロンプトインジェクションに対して脆弱である」）。

レッドチーミングは、単なるベンチマークテストではなく、モデルの振る舞いの限界を探るための、創造的で、手動の、チームベースの取り組みであるとされている [24]。成功のためには、エンジニアリング、セキュリティ、法務、ポリシーなど多様なチームを巻き込み、共有スプレッドシートなどを用いた明確な文書化プロトコルを確立し、開発ライフサイクル（デプロイ前、CI/CD、デプロイ後の監視）にレッドチーミングを統合することが推奨される [22]。

効果的なレッドチーミングは、既知の攻撃リストに対する合否判定テストではない。それは、未知の故障モードや創発的な脆弱性を発見するために設計された、オープンエンドな探索的プロセスである。その主なアウトプットはスコアではなく、システムの弱点に対するより深い理解である。標準化されたベンチマークは既知の問題をテストするが、LLMの攻撃対象領域は広大で常に進化している。レッドチーミングの目標は、明確に「探索」と「新規性」の発見にあるとされており、一度でも攻撃が成功すれば、その脆弱性の存在を証明するには十分である [24]。これは、大規模なデータセット上で統計的有意性を要求する定量的評価とは対照的である。したがって、レッドチーミングは「錬金術師の精神」でアプローチされるべきであり、創造的で型にはまらない攻撃戦略が奨励される [24]。その価値は、システムが安全であることを確認することにあるのではなく、それが失敗しうる予期せぬ一つの方法を見つけ出すことにある。これにより、レッドチーミングは、定量的で仮説検証的な評価アプローチを補完する、定性的で仮説生成的なプロセスとなる。

デバッグと根本原因分析：ペルソナ逸脱と指示不遵守の診断と緩和

プロンプトが意図通りに機能しない場合、その根本原因を特定するためには体系的なデバッグプロセスが必要となる。これには、コンポーネントの分離、指示の反復的改善、そしてモデルの内部状態を理解するための解釈可能性ツールの活用が含まれる。

単純なデバッグ手法として、まず失敗をプレイグラウンドのような管理された環境で再現し、次にLLM自身にその出力の誤った部分を「説明」させる方法がある。これにより、プロンプトのどの部分が混乱を招いているかが明らかになることがある [26]。反復的な改善が鍵となり、指示を修正・言い換えたり、「常に」「非常に重要」といった言葉で強調したり、指示間の矛盾を確認したりすることが有効である [26]。

重大な失敗モードの一つに、長い対話の過程でモデルが割り当てられたペルソナを失ってしまう「ペルソナ逸脱」がある。研究によれば、これはトランスフォーマーの注意機構における「アテンションの減衰」と関連している可能性が示唆されており、一般的な振る舞いの問題に対して技術的な根本原因を提供している [27]。指示追従自体も複雑なアラインメントの対象であり、モデルは「指示に従う」「役立つ」「有害な要求を拒否する」といった複数の、時には相反する目的の間でバランスを取るように訓練されているため、予測不能な失敗につながることがある [28]。場合によっては、モデルが倫理的違反を認識しつつも、主要な目標を達成するためには有害な行動を選択することさえある [30]。

このような複雑な問題を診断するために、モデルの内部動作を可視化する解釈可能性ツールが役立つ。

- アテンション可視化 **(BertViz):** このツールは、モデルが出力を生成する際に、入力のどのトークンに「注意」を払っているかを可視化する [31]。これにより、なぜモデルが特定の指示を無視したのか、あるいはコンテキストの誤った部分に焦点を当てたのかをデバッグできる。例え

ば、ある制約が常に無視される場合、アテンションを可視化すると、モデルがその制約を定義するトークンにほとんどアテンションを割り当てていないことが判明するかもしれない。

- 特徴アトリビューション / サリエンシーマップ **(Ecco):** Eccoのようなツールはさらに一歩進み、どの入力トークンが特定の出力トークンの生成に最も影響を与えたかを計算する [33]。これはアテンションだけでは得られない、より直接的な因果関係の手がかりを提供する。モデルが事実と異なる記述を生成した場合、サリエンシーマップはその原因が入力コンテキスト内の誤解を招くトークンにあったのかどうかを示すことができる。

LLMの複雑な失敗、特に指示追従やアラインメントにおける失敗のデバッグは、単に不備のあるプロンプトを修正する作業ではない。それは、モデルの訓練データとアラインメントプロセス（RLHFなど）によって形成された、モデルの内部的な優先順位と意思決定の計算論理をリバースエンジニアリングする行為である。モデルが指示に従わない場合、単純な説明は「プロンプトが悪かった」となる。しかし、より深い分析によれば、モデルは多くの場合、ユーザーの指示、安全性の訓練、そして一般的な「有用性」の訓練という複数の目的を天秤にかけている [28]。失敗は、ユーザーの指示が、より重み付けの高い内部的な目的と衝突したときに発生する。例えば、ある要求が潜在的に有害であると微妙に解釈され、安全性の目的が指示追従の目的を上書きすることがある。したがって、デバッグにはモデルの
内部的な葛藤に関する仮説を立てることが求められる。「なぜ拒否したのか？要求を安全上のリスクと誤分類したのかもしれない」。そしてプロンプトは、単に明確にするだけでなく、モデルの隠れた優先順位との衝突を解消するように（例えば、要求が安全であることを保証するコンテキストを追加するなどして）修正される。BertVizやEccoのような解釈可能性ツールは、モデルの「思考プロセス」を覗き見るための計装を提供し、このリバースエンジニアリングをより体系的なものにする。

ハルシネーション問題：最先端の検出・緩和技術の概観

ハルシネーション（事実と矛盾する、あるいは入力と矛盾する出力）は、生成モデルに固有の特性であり、その検出と緩和は、信頼できるAIを構築する上での中心的な課題である。
ハルシネーションの検出手法は、いくつかの異なるアプローチに分類できる [35]。

- 不確実性推定**:** ディープアンサンブルやモデルの出力確率分布の分析といった技術を用いて、モデルの「自信」を測定する。不確実性が高いほど、ハルシネーションの可能性が高いと相関することがある [38]。
- 検索ベース **/** 事実確認**:** RAGのように、生成された出力を外部の知識ベースと照合して主張を検証する。これは「Faithfulness」メトリクスの基本原理でもある。
- 内部状態分析**:** 「LLMの内部状態は、それが嘘をついている時を知っている」という研究結果が示唆するように、モデルの隠れ状態に対して軽量な分類器を訓練し、ハルネーションを生成しそうかを予測するアプローチがある [35]。

評価のためには、HaluEvalやFaithDialといった主要なハルシネーション評価ベンチマークが開発されている [35]。これらのツールと手法を組み合わせることで、ハルシネーションを体系的に管理し、より信頼性の高いカスタムGEMを構築することが可能となる。

Part III: システムの堅牢化 - プロンプトインジェクションに対す

る高度なセキュリティ

本パートでは、LLMアプリケーションにおける主要なセキュリティ脅威であるプロンプトインジェクションについて深く掘り下げる。これは、先行研究で軽く触れられたテーマを、より実践的かつ体系的なセキュリティフレームワークへと拡張するものである[1]。

脅威モデリング：**OWASP Top 10 for LLM Applications**の解体

OWASP Top 10 for LLM Applicationsは、AIシステムにおけるセキュリティリスクを理解し、優先順位を付けるための初の標準化されたフレームワークを提供する[39]。
2025年版のリストで特定された主要な脆弱性の中でも、最大のリスクとして挙げられているのが**LLM01: Prompt Injection**である[41]。この攻撃は、主に二つの形態を取る。

- 直接インジェクション（ジェイルブレイキング）: ユーザーが悪意のあるプロンプトを直接入力し、システムプロンプトを上書きしようと試みる。
- 間接インジェクション: 悪意のあるプロンプトが、LLMが処理する外部のデータソース（ウェブページ、文書、メールなど）に隠されている。

これらに加え、「Insecure Output Handling（安全でない出力処理）」、「Training Data Poisoning（訓練データの汚染）」、「Sensitive Information Disclosure（機密情報の漏洩）」といった他の重大なリスクも存在する[42]。これらの脅威を理解することは、効果的な防御戦略を構築するための第一歩である。

攻撃ベクトルの分析：直接的、間接的、および難読化されたプロンプトインジェクション技術

効果的な防御策を設計するためには、攻撃者が用いる具体的な技術を理解することが不可欠である。攻撃の手法は、「以前の指示を無視せよ」といった単純なものから、はるかに洗練されたものへと進化している。
学術文献では、目的の乗っ取り、プロンプトの漏洩、そしてBase64エンコーディングなどを用いた難読化技術といった、多様な攻撃手法が報告されている[43]。中でも、外部データと連携するアプリケーションにとって、より巧妙で現実的な脅威となるのが
間接プロンプトインジェクションである[47]。攻撃者がウェブページ、メール、文書などに埋め込んだ指示が、被害者ユーザーのLLMエージェントによって意図せず実行されてしまうシナリオがこれに該当する[47]。興味深いことに、研究によれば、GPT-4のような高性能なモデルほど、その優れた指示追従能力ゆえに、正当なユーザー指示と悪意のある注入された指示を区別することが難しく、結果としてこれらの攻撃に対してより脆弱になる傾向があることが示されている[50]。

多層防御：プロンプトセキュリティのためのアーキテクチャ戦略

プロンプトインジェクションに対する単一の万能な防御策は存在しない。堅牢なセキュリティ体制を築くには、予防、検出、そして影響緩和を組み合わせた多層的な「Defense-in-Depth」戦略が必要である。
Microsoftが提唱する包括的な多層防御戦略は、このアプローチの優れたケーススタディとなる [47]。

- 予防: システムプロンプトを強化し、「Spotlighting」（区切り文字、データマーキング、エンコーディングなど）のような技術を用いて、モデルが信頼できる指示と信頼できない外部コンテンツを区別するのを助ける。
- 検出: Microsoft Prompt Shieldsのような分類器ベースのツールを使用して、推論時に悪意のある入力を検出する。
- 影響緩和: これが最も重要な層である。予防と検出が失敗する可能性を前提とし、以下の対策を講じる。
    - 厳格なアクセス制御: LLMがアクセスできるツールやデータに対して、最小権限の原則を適用する。
    - ヒューマンインザループ: メール送信やデータベースへの書き込みといった高リスクな操作には、ユーザーの明確な同意を要求する。
    - 出力の検証/サニタイズ: すべてのLLM出力を信頼できないものとして扱い、下流のシステムに渡す前に検証する。これは「Insecure Output Handling」（LLM02）に対する防御の基本原則でもある [41]。

他の防御策として、第二のLLMをフィルターとして使用する手法や、攻撃を妨害するための入力置換技術も提案されている [46]。しかし、多くの既存の防御策は、防御メカニズムを回避するように設計された適応型攻撃に対しては、必ずしも堅牢ではないことが批判的に評価されている [51]。
指示追従型LLMの根本的な性質を考えると、プロンプトインジェクションを完全に予防することは不可能に近いかもしれない。指示に従うように設計されたモデルにとって、テキストに埋め込まれた指示を処理することはその本質的な機能であり、これをセキュリティ上の要求（テキスト内の指示を無視する）と両立させることには根本的な矛盾が存在する [50]。攻撃者は常に、現在の予防・検出ヒューリスティックを回避する巧妙な表現を見つけ出すだろう。したがって、堅牢なセキュリティ戦略は、インジェクションがいつかは成功するという前提に立たなければならない。その上で最も重要な問いは、「もし攻撃者がLLMの制御を奪った場合、最悪何ができるか？」となる。この問いは、セキュリティの焦点を、完璧な壁を築くこと（予防/検出）から、攻撃者が壁を乗り越えた場合に引き起こせる損害を限定すること（
影響緩和）へとシフトさせる。LLMのサンドボックス化、ツールに対する厳格な権限設定、機密性の高い操作に対するユーザー確認の要求、そして出力のサニタイズといったアーキテクチャレベルの解決策が、確率的な脅威に対する決定論的な防波堤となる。

攻撃手法	入力フィルタリング/サニタイズ	システムプロンプト強化	検出分類器	ツールアクセス制御	ヒューマンインザループ	出力検証
直接インジェクション (ジェイルブレイキング)	既知の攻撃パターンのブロック	役割と制約を明確化、「～するな」という指示を追加	悪意のある意図を検出 (例: Prompt Shields)	-	高リスク操作の実行前に確認	生成されたコードやコマンドをサニタイズ
間接インジェ	信頼できない	区切り文字や	外部コンテン	検索対象	-	外部情報に

クション **(RAG**経由**)**	データソースからのHTML/スクリプトタグを除去	Spotlighting技術を使用	ツ内の悪意のある指示を検出	データへのアクセス権を最小化		基づく主張を検証
間接インジェクション **(**ツール出力経由**)**	APIレスポンスから制御文字やスクリプトを除去	ツールからの出力は「データ」であり「指示」ではないことを明記	ツール出力に含まれる異常なパターンを検出	ツールに最小権限の原則を適用 (読み取り専用など)	ツールによる破壊的変更の前に確認	ツールからのデータが期待されるスキーマに準拠しているか検証
難読化ペイロード **(**例**: Base64)**	難読化された入力をデコードして分析	-	難読化パターン自体を不審な入力として検出	-	-	出力が予期せずエンコードされている場合に警告

結論と戦略的提言

本レポートの分析を通じて、システムプロンプトが単純な指示から、LLMの振る舞いを形成する洗練されたコグニティブアーキテクチャへと進化したことが明らかになった。医療、法律、教育、ソフトウェア開発といった各専門分野における最先端のプロンプトは、単にタスクを指示するのではなく、その分野特有の思考プロセスや方法論をモデルに課すことで、高いパフォーマンスと信頼性を実現している。

しかし、これらの高度なプロンプトを設計し、実運用に乗せるには、その品質を客観的に保証し、潜在的なリスクから保護するための体系的なフレームワークが不可欠である。評価は、検索器と生成器の性能を個別に診断する多面的なアプローチへと移行し、レッドチーミングは未知の脆弱性を発見するための創造的な探索プロセスとして位置づけられるべきである。さらに、プロンプトインジェクションという避けがたい脅威に対しては、予防や検出だけに頼るのではなく、万が一攻撃が成功した場合の被害を最小限に抑える「影響緩和」を中心とした多層防御アーキテクチャの構築が最も重要な戦略となる。

これらの知見に基づき、カスタムGEMを開発・運用する実践者に対し、以下の戦略的提言を行う。

1. プロンプトを「認知設計図」として設計する:
   単なる指示の羅列ではなく、「ペルソナ（誰が）」「プロセス（どのように思考し）」「プロダクト（何をどのような形式で出力するか）」という3Pフレームワークに基づき、プロンプトを体系的に設計すること。特に複雑なタスクにおいては、CoTやIRACのような思考プロセスを明示的に組み込み、モデルの推論を構造化することが不可欠である。
2. 評価プロセスを「診断ダッシュボード」として構築する:
   単一の精度指標に依存せず、Faithfulness、Answer Relevancy、Contextual Precision/Recallといった複数のメトリクスを用いて、RAGシステムの各コンポーネントを個別に評価する体制を整えること。これにより、問題発生時の迅速な根本原因分析が可能となる。
3. セキュリティを「影響緩和」から逆算して設計する:
   プロンプトインジェクションの完全な防御は困難であると認識し、セキュリティ戦略の重心を影響緩和に置くこと。LLMがアクセスできるツールやデータには最小権限の原則を徹底し、メー

ル送信やデータベース更新のような高リスクな操作には必ずヒューマンインザループ（ユーザーの承認）を介在させ、LLMからの出力は常に信頼できないものとして扱い、下流システムに渡す前に検証・サニタイズするアーキテクチャを標準とすること。

プロンプトエンジニアリングは、もはや試行錯誤の「アート」から、設計、評価、セキュリティ確保という体系的なプロセスを伴う「エンジニアリング分野」へと成熟しつつある。本レポートで提示したフレームワークは、その実践を加速させ、より信頼性が高く、安全で、効果的なAIアプリケーションを構築するための羅針盤となるだろう。

引用文献

1. システムプロンプト設定の有効性調査.pdf
2. Diagnostic reasoning prompts reveal the potential for large ..., 9月 11, 2025にアクセス、https://pmc.ncbi.nlm.nih.gov/articles/PMC10808088/
3. 100+ ChatGPT prompts for healthcare professionals - Paubox, 9月 11, 2025にアクセス、https://www.paubox.com/blog/100-chatgpt-prompts-for-healthcare-professionals
4. Want to use chat for medical diagnosis? Here's the prompt : r/ChatGPT - Reddit, 9月 11, 2025にアクセス、https://www.reddit.com/r/ChatGPT/comments/16oqt46/want_to_use_chat_for_medical_diagnosis_heres_the/
5. ChatGPT Prompt of the Day: LEGAL CONTRACT RED FLAG ..., 9月 11, 2025にアクセス、https://www.reddit.com/r/ChatGPTPromptGenius/comments/1jb0q8r/chatgpt_prompt_of_the_day_legal_contract_red_flag/
6. ChatGPT Prompt of the Day: The All-Purpose Legal Document ..., 9月 11, 2025にアクセス、https://www.reddit.com/r/ChatGPTPromptGenius/comments/1hst9lv/chatgpt_prompt_of_the_day_the_allpurpose_legal/
7. Exploring the Effectiveness of Prompt Engineering for Legal Reasoning Tasks, 9月 11, 2025にアクセス、https://www.semanticscholar.org/paper/Exploring-the-Effectiveness-of-Prompt-Engineering-Yu-Quartey/9661dda8024192342097647c37423eb7fccee298
8. [2507.07893] An Integrated Framework of Prompt Engineering and Multidimensional Knowledge Graphs for Legal Dispute Analysis - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/abs/2507.07893
9. Citation: Zhang, M., Zhao, N., Qin, J., Xu, Q., Pan, K., Luo, T. An Integrated Framework of Prompt Engineering and Multidimensional Knowledge Graphs for Legal Dispute Analysis. arXiv:2025.xxxxx, 9月 11, 2025にアクセス、https://arxiv.org/html/2507.07893v1
10. AI for Tutoring - Super Prompt | openNCCC, 9月 11, 2025にアクセス、https://openncccc.nccommunitycolleges.edu/courseware/lesson/946/overview
11. ChatGPT Prompt of the Day: The Skill Mastery AI Tutor : r ... - Reddit, 9月 11, 2025にアクセス、https://www.reddit.com/r/ChatGPTPromptGenius/comments/1i2owyo/chatgpt_pr

ompt_of_the_day_the_skill_mastery_ai/
12. Student AI Guide - Larry Schrenk, 9月 11, 2025にアクセス、https://larryschrenk.com/AI-Guide.pdf
13. AI-Powered Educational Agents: Opportunities, Innovations, and Ethical Challenges - MDPI, 9月 11, 2025にアクセス、https://www.mdpi.com/2078-2489/16/6/469
14. Socratic AI - Kaggle, 9月 11, 2025にアクセス、https://www.kaggle.com/code/savvabojko/socratic-ai
15. Generating Code | Prompt Engineering Guide, 9月 11, 2025にアクセス、https://www.promptingguide.ai/applications/coding
16. [2407.07064] Prompting Techniques for Secure Code Generation: A Systematic Investigation - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/abs/2407.07064
17. [2502.06039] Benchmarking Prompt Engineering Techniques for Secure Code Generation with GPT Models - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/abs/2502.06039
18. Prompt Alchemy: Automatic Prompt Refinement for Enhancing Code Generation - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/abs/2503.11085
19. LLM Evaluation: Frameworks, Metrics, and Best Practices | SuperAnnotate, 9月 11, 2025にアクセス、https://www.superannotate.com/blog/llm-evaluation-guide
20. LLM Evaluation Metrics: The Ultimate LLM Evaluation Guide ..., 9月 11, 2025にアクセス、https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation
21. LLM evaluation: Metrics, frameworks, and best practices | genai-research - Wandb, 9月 11, 2025にアクセス、https://wandb.ai/onlineinference/genai-research/reports/LLM-evaluation-Metrics-frameworks-and-best-practices--VmlldzoxMTMxNjQ4NA
22. LLM red teaming guide (open source) - Promptfoo, 9月 11, 2025にアクセス、https://www.promptfoo.dev/docs/red-team/
23. LLM Red Teaming: The Complete Step-By-Step Guide To LLM ..., 9月 11, 2025にアクセス、https://www.confident-ai.com/blog/red-teaming-llms-a-step-by-step-guide
24. Defining LLM Red Teaming | NVIDIA Technical Blog, 9月 11, 2025にアクセス、https://developer.nvidia.com/blog/defining-llm-red-teaming/
25. Planning red teaming for large language models (LLMs) and their applications - Azure OpenAI in Azure AI Foundry Models | Microsoft Learn, 9月 11, 2025にアクセス、https://learn.microsoft.com/en-us/azure/ai-foundry/openai/concepts/red-teaming
26. A simple (not easy) technique for debugging LLMs using LLMs | by Aman Dalmia - Medium, 9月 11, 2025にアクセス、https://medium.com/inveterate-learner/a-simple-not-easy-technique-for-debugging-llms-using-llms-d97a175e4bb5
27. Measuring and Controlling Persona Drift in Language Model Dialogs - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/html/2402.10962v1
28. Problems with instruction-following as an alignment target - AI Alignment Forum,

9月 11, 2025にアクセス、https://www.alignmentforum.org/posts/CSFa9rvGNGAfCzBk6/problems-with-instruction-following-as-an-alignment-target
29. Problems with instruction-following as an alignment target - LessWrong, 9月 11, 2025にアクセス、https://www.lesswrong.com/posts/CSFa9rvGNGAfCzBk6/problems-with-instruction-following-as-an-alignment-target
30. Agentic Misalignment: How LLMs could be insider threats - Anthropic, 9月 11, 2025にアクセス、https://www.anthropic.com/research/agentic-misalignment
31. jessevig/bertviz: BertViz: Visualize Attention in NLP Models ... - GitHub, 9月 11, 2025にアクセス、https://github.com/jessevig/bertviz
32. BertViz: Visualize Attention in Transformer Language Models - CodeCut, 9月 11, 2025にアクセス、https://codecut.ai/bertviz-visualize-attention-in-transformer-language-models/
33. jalammar/ecco: Explain, analyze, and visualize NLP language models. Ecco creates interactive visualizations directly in Jupyter notebooks explaining the behavior of Transformer-based language models (like GPT2, BERT, RoBERTA, T5, and T0). - GitHub, 9月 11, 2025にアクセス、https://github.com/jalammar/ecco
34. Jay Alammar – Visualizing machine learning one concept at a time., 9月 11, 2025にアクセス、https://jalammar.github.io/
35. EdinburghNLP/awesome-hallucination-detection - GitHub, 9月 11, 2025にアクセス、https://github.com/EdinburghNLP/awesome-hallucination-detection
36. HillZhang1999/llm-hallucination-survey - GitHub, 9月 11, 2025にアクセス、https://github.com/HillZhang1999/llm-hallucination-survey
37. A Survey on Hallucination in Large Language and Foundation Models - Preprints.org, 9月 11, 2025にアクセス、https://www.preprints.org/manuscript/202504.1236/v2
38. Hallucination Detection in LLMs Using Bayesian Neural Network Ensembling - DiVA portal, 9月 11, 2025にアクセス、https://www.diva-portal.org/smash/get/diva2:1887965/FULLTEXT01.pdf
39. Breaking Down the OWASP Top 10 for LLM Applications - Checkmarx, 9月 11, 2025にアクセス、https://checkmarx.com/learn/breaking-down-the-owasp-top-10-for-llm-applications/
40. OWASP AI Security Project: Top 10 LLM Vulnerabilities Guide ..., 9月 11, 2025にアクセス、https://konghq.com/blog/engineering/owasp-top-10-ai-and-llm-guide
41. Quick Guide to OWASP Top 10 LLM: Threats, Examples & Prevention - Tigera, 9月 11, 2025にアクセス、https://www.tigera.io/learn/guides/llm-security/owasp-top-10-llm/
42. What are the OWASP Top 10 risks for LLMs? - Cloudflare, 9月 11, 2025にアクセス、https://www.cloudflare.com/learning/ai/owasp-top-10-risks-for-llms/
43. Defense Against Prompt Injection Attack by Leveraging Attack Techniques - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/html/2411.00459v2
44. Attack and defense techniques in large language models: A survey and new perspectives, 9月 11, 2025にアクセス、https://arxiv.org/html/2505.00976v1

45. Breaking Down the Defenses: A Comparative Survey of Attacks on Large Language Models, 9月 11, 2025にアクセス、https://arxiv.org/html/2403.04786v2
46. Prompt Injection Attacks in Defended Systems - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/html/2406.14048v1
47. How Microsoft defends against indirect prompt injection attacks ..., 9月 11, 2025にアクセス、https://msrc.microsoft.com/blog/2025/07/how-microsoft-defends-against-indirect-prompt-injection-attacks/
48. Indirect prompt injection attacks target common LLM data sources - ReversingLabs, 9月 11, 2025にアクセス、https://www.reversinglabs.com/blog/indirect-prompt-injections-target-llm-data
49. Indirect Prompt Injection: Generative AI's Greatest Security Flaw, 9月 11, 2025にアクセス、https://cetas.turing.ac.uk/publications/indirect-prompt-injection-generative-ais-greatest-security-flaw
50. Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models - arXiv, 9月 11, 2025にアクセス、https://arxiv.org/pdf/2312.14197
51. A Critical Evaluation of Defenses against Prompt Injection Attacks, 9月 11, 2025にアクセス、https://arxiv.org/abs/2505.18333