論文要約

LLM関連

LONG2RAG: Evaluating Long-Context & Long-Form Retrieval-Augmented Generation with Key Point Recall LONG2RAG: 長文コンテキストおよび長文形式の検索強化生成の評価とキーポイントリコールによる評価

概要: LONG2RAGは、長文コンテキストでの LLMのRAG性能を評価するためのベンチマークです。 280の質問に5つの関連文書を設定し、検索された文書から抽出されたキーポイントをどれだけ含んでいるかを測定するKPRで評価。GPT-4oが最高スコアの 0.579を記録

RAG評価で主に長文コンテキストに対応する LLM評価ベンチマーク LONG2RAGを提案。280の質問が10の領域で設定され各質問に対して 5つの関連文書を設定。評価指標には KPR（Key Point Recall）を設定し検索された文書から抽出されたキーポイントをどれだけ含んでいるかを測定する方法（各スコアは 0から1の範囲で、高いほど良い性能を示します）を使用。質問は 8つのカテゴリ（事実、説明、比較、主観、因果関係、仮定、予測、方法論）に分類して評価、 GPT-4oのKPRは 0.579、Claude-3-SonnetのKPRは 0.477、Qwen2-72B（オープンソースモデルの大規模版） : KPRは、0.449、Phi-3-mini-128K: KPRは 0.434と商用モデルである GPT-4oが最も優れた結果を示しました。

また、KPRは長文生成を好む傾向もあるため、生成の質と長さのバランスが重要であることもわかっています

技術や手法

- **LONG2RAGベンチマーク **: 280の質問を使用し、各質問に対して平均 2444語の5つの検索文書が関連付けられています。これにより、モデルが長文の検索情報を取り込む能力を評価します。

- **キーポイントリコール（ KPR）**: 検索された文書から抽出されたキーポイントが生成された回答にどの程度含まれているかを評価する手法です。この評価を通じて、モデルが検索情報を活用しているかどうかを測定します。

- **データセットの構築方法 **: 自動パイプラインを用いて質問を生成し、関連する文書を検索してキーポイントを抽出。その後、 LLMと人間の協力によりキーとなるポイントの検証を行い、データセットを構築しました。

評価手法とパフォーマンス指標

論文では、LONG2RAGベンチマークを用いて 9つの最新の LLM（大規模言語モデル）を評価しました。評価に用いられた指標は以下の通りです。

Benchmarking Large Language Models in Retrieval-Augmented Generation 検索強化生成における大規模言語モデルのベンチマーク

概要: RAGのパフォーマンスを 4つのノイズ耐性、否定拒否、情報統合、反事実耐性の　RGBのコーパスで評価。LLMは一定のノイズ耐性を持つが、否定拒否や情報統合、誤情報処理にはまだ課題が多いことがわかりました

技術や手法

- **検索強化生成 (Retrieval-Augmented Generation, RAG):** RAGは、検索エンジンを用いて外部の知識を取得し、モデルの幻覚を軽減する手法。特に、インターネット上の膨大な情報から正確な知識を得るために使用される。

- **Retrieval-Augmented Generation Benchmark (RGB):** RGBは、RAGの4つの基本的な能力を評価するために設計された新しいベンチマークで、最新のニュース情報を基に構築されている。このベンチマークにより、LLMがノイズ情報に対してどの程度頑健であるかや、複数の情報を統合する能力などを評価できる。

    - **ノイズ耐性 (Noise Robustness):** 質問と関連があるが、回答を含まないノイズ文書から必要な情報を抽出する能力。

    - **否定拒否 (Negative Rejection):** 必要な知識が取得された文書に存在しない場合に、適切に回答を拒否する能力。

    - **情報統合 (Information Integration):** 複数の文書から情報を統合して質問に回答する能力。

    - **反事実耐性 (Counterfactual Robustness):** 取得された文書に誤った情報が含まれている場合に、そのリスクを認識して適切に処理する能力。

使用用途

この研究は、以下のようなシーンで活用が期待される:

- **検索エンジンの改善 :** LLMを用いた検索結果の生成において、ノイズ情報を適切にフィルタリングし、より正確な情報提供を行う。

Investigating the Role of Prompting and External Tools in Hallucination Rates of Large Language Models プロンプト設計および外部ツールの役割が大規模言語モデルの幻覚率に与える影響の調査

概要: LLMのハルシネーションがプロンプトエンジニアリングや　LLMエージェントの活用でどのように変わるかを調査

Temperatureをあげて複数回の LLM呼び出しの多数決で回答する SCを使用することが効果的だという結果になり、現実での知識を問うタスクには　KGRが効果的という結果になりました

技術や手法

1. プロンプト技術

1.1 チェイン・オブ・ソート（ CoT）プロンプト

チェイン・オブ・ソート（ Chain-of-Thought, CoT）プロンプトは、複雑な問題をより簡単に解決できるように、小さなステップに分割する手法です。この方法では、モデルが一度に問題全体を解決するのではなく、解決の過程を段階的に分解します。例えば、数学の問題を解く場合、問題をいくつかの小さなステップに分けて、それぞれのステップで部分的な答えを導き出し、最終的に全体の答えに到達します。この方法により、LLMはより精度の高い推論が可能になります。

1.2 自己一貫性（ SC）

自己一貫性（ Self-Consistency, SC）は、同じ質問に対して複数回の LLM呼び出しを行い、その結果を多数決で選ぶことで一貫性のある答えを導き出す手法です。この手法の目的は、モデルのランダムな生成によって生じる不安定な出力を安定させることです。温度（ temperature）の設定を調整し、複数の異なる出力から最も一貫した答えを選ぶことで、信頼性の高い回答が得られるようになります。この方法は、特に数学の問題や論理的な推論を必要とする課題に有効です。

1.3 木構造の思考（ ToT）

Distance between Relevant Information Pieces Causes Bias in Long-Context LLMs 関連情報の間の距離が長コンテキスト LLMにバイアスを引き起こす

概要: 長文を扱うLLMには、情報が中間にあると見落とす lost in the middle問題の他に複数の情報を活用して回答するときにその複数の情報同士の距離とその配置が遠くなることが結果に影響することが開発されたLONGPIBENCHというベンチマークからわかりました

技術や手法

1. **ポジショナルバイアスの問題と「 lost in the middle」現象 **

- **ポジショナルバイアス **とは、大規模言語モデル(LLM)が入力された情報の位置に応じて、その情報をうまく扱えなくなる現象を指します。この論文では、特に長文の入力での問題を扱っています。具体的には、重要な情報が文脈の中間に位置する場合、モデルがその情報を見落とす「 lost in the middle」現象が問題視されています。この現象は、 LLMsが長い文脈を効率的に利用する際の大きな障害です。

2. **LONGPIBENCHの設計と目的 **

- **LONGPIBENCH**は、複数の関連情報が含まれるタスクにおけるポジショナルバイアスを評価するためのベンチマークです。このベンチマークは、絶対位置と相対位置のバイアスを評価することを目的としています。

  - **絶対位置 **とは、文脈全体の中で関連情報がどの部分に位置するかを指します(例えば、入力の先頭、中間、末尾など)。

  - **相対位置 **は、複数の関連情報の間の距離や、それらの情報がどの程度密集しているかを意味します。この点に注目することで、 LLMが情報の分布や配置にどのようなバイアスを持っているかを評価します。

3. **LONGPIBENCHのタスク設計 **

DIVERSITY HELPS JAILBREAK LARGE LANGUAGE MODELS 多様性が大規模言語モデルの脱獄を支援する

概要: LLMの脱獄手法を試し GPT-4、Gemini、Llamaなどの主流のチャットボットに対して成功率を最大　62%向上、元の攻撃プロンプトから大きく逸脱した表現を使う多様化ステップ（　Diversification Step）と元の攻撃意図を保持しつつ、それを曖昧化するためのプロンプト隠蔽ステップ（　Obfuscation Step）の2ステップを使う DAGRフレームワークを提案

多様な攻撃手法の概要

本論文では、多様な攻撃手法を使用して大規模言語モデル（　LLM）を脱獄させることで、安全性を突破する成功率を向上させています。この多様な攻撃手法は以下のように　2つのステップから構成されており、それぞれの役割がモデルの脆弱性を効果的に突きます。

1. **多様化ステップ（ Diversification Step）**

2. **隠蔽ステップ（ Obfuscation Step）**

1. 多様化ステップ（ Diversification Step）

- **攻撃の創造性を最大限に活用する **: 多様化ステップでは、以前に生成された攻撃プロンプトから大きく逸脱し、過去の攻撃手法と異なる新たな攻撃プロンプトを生成することを目指します。この多様化には創造性やフィクションを取り入れており、攻撃手法の幅を広げることを重視しています。

- **具体的な生成方法 **:

    - 各深度において、新しい攻撃プロンプトを生成する際に、創造的でフィクションを含んだ内容にするように攻撃モデルに指示されます。

    - 攻撃対象の言語モデルを破るために、物語風の設定や仮想のシナリオを使用して、安全性メカニズムが警戒しにくい方法で攻撃を試みます。

RAGULATOR: LIGHTWEIGHT OUT-OF-CONTEXT DETECTORS FOR GROUNDED TEXT GENERATION RAGULATOR: 文脈外検出器による軽量なグラウンド化されたテキスト生成

概要: RAGulatorは、LLMがRAGで生成した文と、関連する文脈を比較することで外れていないかを　BERTベースの軽量モデルを使い、生成された文がその文脈と一致しているかどうかを分類します。このモデルは、要約や類似性データを基に訓練され、文脈外（　Out-Of-Context, OOC）の可能性がある部分を識別します。文が文脈から外れていると判断された場合、それは信頼性の低い情報、つまり「文脈外」とみなされます。

RAGulatorは、RAGにおいて、AIが生成する文が文脈に沿っているかどうか、つまり「文脈外」　(Out-Of-Context, OOC)の検出を目的としています。 RAGは事前に検索された情報を元に回答を生成しますが、その際に文脈から外れた内容を生成してしまうこと（いわゆる「幻覚」や「ハルシネーション」）が問題となります。このような不正確な情報は特に企業環境では深刻な影響を及ぼす可能性があります。

そこで、RAGulatorは軽量な分類器を用いて、生成された文が文脈に基づいているかどうかを判別する仕組みを提供しています。

2. データセットの生成

**データセットの準備 **は、RAGulatorの重要なステップです。このデータセットは、要約データセットとセマンティックテキスト類似性（　Semantic Textual Similarity, STS）データセットを活用して構築されました。

- **要約データセット **（BBC, CNN/Daily Mail, PubMed）:

  - 各データセットは要約と元記事のペアで構成されています。要約と無関係な記事をペアリングし、それぞれの要約を文に分割して、文脈外（　OOC）の例を生成します。これにより、文脈に沿っている場合と沿っていない場合を含む学習用データセットを作り出しています。

- **セマンティックテキスト類似性データセット　**（MRPC, SNLI）:

  - 文ペアとその類似性ラベルを元に、文脈内および文脈外の文ペアを作成。例えば、ランダムな文を追加して文脈を拡張し、それらが類似するかどうかで文脈内・外のラベルを付けました。

3. データセットの整形とラベリング

RAGVIZ: Diagnose and Visualize Retrieval-Augmented Generation RAGVIZ: リトリーバル・オーグメンテッド・ジェネレーションの診断と可視化

https://github.com/cxcscmu/RAGViz

https://www.youtube.com/watch?v=cTAbuTu6ur4

概要: RAGVizはモデルがどの文章に注目して答えを生成したかを各文書やトークンにどれくらい注意しているかをクエリとキーの内積をソフトマックス関数で正規化して可視化、どの文章が生成の根拠かをわかります

技術や手法

1. **注意可視化 **:
   - 取得した文書に対するトークンの注意度を可視化する機能を提供します。生成されたトークンが取得文書のどの部分にどの程度依存しているかを視覚的に確認できます。
2. **文書トグル機能 **:
   - 取得した文書を追加または削除して生成の変化を比較できる機能を持っています。これにより、どの文書が生成結果にどのように影響するのかを容易に評価できます。
3. **分散アーキテクチャ **:
   - RAGVizは、複数のノードにデータセットを分割して効率的に処理を行います。また、効率的なLLM推論ライブラリを使用し、低遅延でのLLM出力生成を実現しています。
4. **スニッピング技術 **:
   - ドキュメントの一部を抜き出して使用するスニッピング技術を実装しており、クエリに最も関連する部分を取得するためにスライディングウィンドウを利用するなどの最適化を行っています。

注意スコアの算出方法は、トランスフォーマーモデルの「アテンション機構」を使用します。具体的には、以下のステップで注意スコアが計算されます。

AIOS Compiler: LLM as Interpreter for Natural Language Programming and Flow Programming of AI Agents AIOSコンパイラ：自然言語プログラミングと AIエージェントのフロープログラミングにおける　LLMを通訳者として使用

https://github.com/agiresearch/OpenAGI

https://github.com/agiresearch/AIOS

概要：Code Representation and Execution (CoRE)というAIエージェントを自然言語で「ステップ名」、「ステップタイプ」、「ステップ指示」、「ステップ接続」という　　4つの基本構成で整理しながら途中のステップで生成した結果や中間データを保存する外部メモリと、ツール（例えば、検索エンジンや　　API呼び出しツール）を利用できる環境を構築します

1. システムの全体構造

- **LLMのインタプリタとしての利用　**LLM（例: GPT-4やMixtralなど）を中心に据え、ユーザーの入力（自然言語プログラム）を解釈し、次のアクションを決定するコンポーネントを開発します。

- **外部メモリとツールの連携　**LLMが途中のステップで生成した結果や中間データを保存する外部メモリと、ツール（例えば、検索エンジンや　　API呼び出しツール）を利用できる環境を構築します。

- **CoRE言語を用いたプログラム表現　**自然言語、疑似コード、フロープログラミングを統合した形式でプログラムを表現するための独自のシンタックス（例えば、「　　:::」を使ったステップの構造化）を設計します。

2. 実装ステップ

ステップ 1: CoRE言語の定義とパーサーの実装

- **CoRE言語の構文定義 **CoRE言語の基本構造を定義します。各ステップには「ステップ名」「ステップタイプ（処理、判断、終了）」「指示」「次のステップへの接続」が含まれます。

- **構文解析（パーサー）　**CoRE言語で記述された自然言語プログラムを解析し、それを理解可能なデータ構造に変換するためのパーサーを実装します。　　Pythonであれば正規表現や文法解析ツール（例　：`lark-parser`や`PLY`）を利用して実装可能です。

Foundations and Recent Trends in Multimodal Mobile Agents: A Survey マルチモーダル移動エージェントの基礎と最近の動向　: 調査

https://github.com/aialt/awesome-mobile-agents

概要: モバイルエージェント技術の紹介。 LLMを使い、ユーザーの指示に従って画面操作を自動化。視覚情報とテキスト情報を統合することで、ボタンの押下や画面のナビゲーションを実施

技術や手法

1. **プロンプトベースの手法 **

　大規模言語モデル(LLM)を活用し、指示に基づいてタスクを実行するアプローチです。特に、 Chain-of-Thought(CoT)推論などを用いて、GUIの操作をより効率的に行えるようになっています。代表例には、OmniActやAppAgentがあります。

2. **訓練ベースの手法 **

　マルチモーダルモデルをモバイル環境向けに微調整するアプローチです。特に、 LLaVAやLlamaなどのモデルが用いられ、視覚データとテキストデータの統合処理を行うことにより、インターフェースのナビゲーションやタスクの実行が可能となっています。

3. **補完技術 **

　エージェントのパフォーマンスを向上させるための技術として、視覚エンコーダーの改善やモバイル特有のインタラクティブ要素を強化するデータセットの導入が挙げられます。

An Empirical Study on the Potential of LLMs in Automated Software Refactoring LLMの自動ソフトウェアリファクタリングにおける可能性の実証的研究

概要: LLMによる自動リファクタリングするためにリファクタリング検出ツール　RefactoringMirrorを提案。

ChatGPTとGeminiで実施したリファクタリングを検出し、 IntelliJ IDEAのAPIを利用して再適用することで自動での成功率が向上しています

技術や手法

1. **LLMによるリファクタリング機会の特定　**:

　- Javaプロジェクトから 180のリファクタリングケースを収集し、 ChatGPTおよびGeminiモデルを利用してリファクタリング機会を特定。

　- 一般的なプロンプトに加え、具体的なリファクタリングタイプを指定したプロンプトを使用することで、　LLMの成功率を改善。

2. **リファクタリングソリューションの提案　**:

　- ChatGPTは180のリファクタリングに対して 176のソリューションを提案し、そのうち 63.6%が専門家と同等、またはそれ以上の品質であると評価された。

3. **RefactoringMirror手法**:

　- LLMが生成したリファクタリングの結果に対し、 ReExtractorツールを用いてリファクタリングの詳細を検出し、それを　IntelliJ IDEAなどの確立されたリファクタリングエンジンで再適用することで、安全性を向上させる方法です。

　- この手法により、LLMが提案したバグのあるリファクタリングの全てを避けることに成功しました。

使用用途

- **自動コード改善**:

Assessing the Answerability of Queries in Retrieval-Augmented Code Generation 検索拡張コード生成におけるクエリの回答可能性の評価

概要: クエリの回答可能性を事前に評価する手法として　RaCGEvalベンチマークを提案、クエリに対して回答可能、部分的に回答可能、回答不可能化を判定することで生成結果を向上できます

技術や手法

1. **回答可能性評価タスクの提案　**: RaCGにおけるユーザーのクエリが回答可能かどうかを評価するタスクを提案。これにより、回答不可能なクエリに対して無駄なコード生成を行わないようにすることが目指されている。

2. **RaCGEvalベンチマークの構築 **: 回答可能性を評価するために　RaCGEvalというデータセットを構築。このデータセットは回答可能、部分的に回答可能、回答不可能なサンプルを含み、それぞれに対してAPIの説明が提供されている。

3. **モデルの評価 **:

    - **ゼロショット推論とファインチューニング　**: いくつかの大規模言語モデル（例　: gpt-3.5, llama3, gemma）を使用して回答可能性を評価。ゼロショット推論の精度はほぼランダムレベルであったが、ファインチューニングにより精度が向上した。

    - **インコンテキストラーニングの効果　**: 見慣れないドメインの情報をモデルに効果的に含めるためにインコンテキストラーニング（　　ICL）を活用。ICLにより精度が大幅に向上した。

4. **回答可能性評価とコード生成のトレードオフ　**: 回答可能性を評価することにより、誤ったコード生成を減らし、生成コードの精度を向上させることができると示された。

Towards Evaluation Guidelines for Empirical Studies involving LLMs LLMを含む実証研究の評価ガイドラインに向けて

概要: LLMの実証研究のためのガイドラインを提供

LLMをどのように使用されたか、どのバージョンをいつ使ったか、ハイパーパラメータの設定はどうか、使用プロンプトを確認し人による検証をすることで再現可能性と信頼性を高める方法を提示

それぞれのガイドラインについて

1. **LLMの使用と役割の明確化 **

このガイドラインは、実証研究において LLM（大規模言語モデル）がどのように使用されたかを具体的に明示することを求めています。これは、研究の再現性と透明性を向上させるために重要です。具体的な内容としては以下が含まれます:

- **タスクの種類 **: LLMをどのようなタスクに使用したのか（例: データの注釈付け、分析、生成タスクなど）。

- **目的の明示 **: なぜLLMを使用したのか、どのような結果を期待していたのか。

- **具体的な処理の説明 **: LLMの使用において、どの部分が LLMに依存していたか（例えば、データの事前処理は人間が行い、データ生成は LLMが行ったなど）。

- **複雑な周辺システムの説明 **: LLMを使う場合、プロンプトの事前処理や後処理、ユーザーの入力をどう変換したかなど、 LLM以外の処理部分についても明示的に記述することが求められます。

2. **モデルのバージョンと使用日時の報告 **

LLMは頻繁にアップデートされるため、特定のバージョンの性能が時間とともに変わることがあります。このため、どのバージョンを使用したのか、実験の実施日がいつであったかを明確に記載することが重

Towards Evaluation Guidelines for Empirical Studies involving LLMs LLMを含む実証研究の評価ガイドラインに向けて

概要: LLMの実証研究のためのガイドラインを提供

LLMをどのように使用されたか、どのバージョンをいつ使ったか、ハイパーパラメータの設定はどうか、使用プロンプトを確認し人による検証をすることで再現可能性と信頼性を高める方法を提示

それぞれのガイドラインについて

1. **LLMの使用と役割の明確化 **

このガイドラインは、実証研究において LLM（大規模言語モデル）がどのように使用されたかを具体的に明示することを求めています。これは、研究の再現性と透明性を向上させるために重要です。具体的な内容としては以下が含まれます:

- **タスクの種類 **: LLMをどのようなタスクに使用したのか（例: データの注釈付け、分析、生成タスクなど）。

- **目的の明示 **: なぜLLMを使用したのか、どのような結果を期待していたのか。

- **具体的な処理の説明 **: LLMの使用において、どの部分が LLMに依存していたか（例えば、データの事前処理は人間が行い、データ生成は LLMが行ったなど）。

- **複雑な周辺システムの説明 **: LLMを使う場合、プロンプトの事前処理や後処理、ユーザーの入力をどう変換したかなど、 LLM以外の処理部分についても明示的に記述することが求められます。

2. **モデルのバージョンと使用日時の報告 **

LLMは頻繁にアップデートされるため、特定のバージョンの性能が時間とともに変わることがあります。このため、どのバージョンを使用したのか、実験の実施日がいつであったかを明確に記載することが重

The Systems Engineering Approach in Times of Large Language Models 大規模言語モデル時代におけるシステム工学アプローチ

概要: テキスト分類に LLMを使用するために Code Completion Prompt（CoCoP）を提案

入力テキストと対応するラベルをコード形式に変換し、それを例として未知のラベルを　LLMを使用して予測する Incomplete-Code Generatorモジュールを利用して分類します

1. **CoCoPの目的**

CoCoPは、テキスト分類の精度を向上させるために、 LLMの「コード補完」能力を利用する方法です。コード補完のタスク能力を持つ　LLMを利用し、テキスト分類問題をコード補完問題に変換することで、LLMのコード補完能力を活用して分類タスクを実施します。

2. CoCoPの手法の基本的な流れ

CoCoPの手法は、大きく以下のような流れで進みます:

1. **入力データと例を準備する **

2. **Incomplete-Code Generatorモジュールでコードを生成する **

3. **コードをLLMに入力し補完を求める **

4. **Label Extractorでラベルを抽出する **

5. **最終結果を出力する **

PTR: Precision-Driven Tool Recommendation for Large Language Models PTR: 大規模言語モデル向け精度駆動型ツール推奨手法

概要: LLMに外部ツールを追加することで複雑な問題を解決するために使用するべき外部ツールを推奨　　PTRという過去の実績に基づいて最適な初期セットを準備し、それを機能ごとにマッチングして必要な調整を加え、最も効果的なツールセットを選ぶ手法を提案

技術や手法

1. **ツールバンドル取得 (Tool Bundle Acquisition)**

   - 過去のツール使用情報から、関連性の高いツールバンドルを取得し、新しいクエリに対して最適な初期ツールセットを構築。

2. **機能カバレッジマッピング (Functional Coverage Mapping)**

   - ユーザークエリを機能に分解し、初期ツールバンドルがすべての機能を満たしているか評価します。

   - ツールセットを最適化するために、未解決の問題を抽出し、不足している部分を特定します。

3. **多視点ベースの再ランキング (Multi-view-based Re-ranking)**

   - 未解決の問題に対して、ツールの直接的なセマンティック類似度、過去のクエリとの相関、そしてツール間のコンテキスト的な関連性を基にツールを選定し、再ランキングを行います。

4. **データセットと評価指標 **

   - **RecToolsデータセット **: クエリごとに異なる数のツールを用いることで、現実の動的なツール使用に近い環境を再現。

   - **TRACC評価指標 **: 推奨ツールの精度を評価するための新たな指標。ツールの数と質の両方を考慮します。

PTR

M3DOCRAG: Multi-modal Retrieval is What You Need for Multi-page Multi-document Understanding M3DOCRAG: マルチページ・マルチドキュメント理解のためのマルチモーダル検索が必要です

概要:ドキュメントのマルチモーダルな情報(テキスト、チャート、図など)を　LLMで扱い1つまたは複数のドキュメントを対象に、視覚情報を保持しながら質問に対応するために　M3DOCRAGを提案

すべてのドキュメントを RGB画像に変換し、ColPaliを使用して視覚的な埋め込みを抽出、それとクエリから類似性の高いトップ　kのページを抽出。

この結果をマルチモーダルな言語モデル(　MLM)を使用して回答を生成します

先行研究との比較と優位性

- **既存手法 **:
  - 従来の DocVQA(Document Visual Question Answering)は主にシングルページや OCRを用いたテキストベースのアプローチに依存しています。
  - これらの手法は多くのドキュメントや長いページに対応する際に限界があり、複雑なビジュアル情報(表やグラフ)を無視することが問題とされています。
- **M3DOCRAGの優位性 **:
  - M3DOCRAGは、クローズドドメイン(特定のドキュメント内)からオープンドメイン(多数のドキュメント)の質問にも対応可能。
  - シングルホップ、マルチホップの質問に対応し、テキスト、チャート、図などの多様な証拠モダリティを扱える点で従来の手法を超えています。
  - 特にビジュアル情報を保持することで、画像に含まれる情報に基づく正確な回答が可能です。

技術・手法

Empowering Meta-Analysis: Leveraging Large Language Models for Scientific Synthesis メタ分析の強化：大規模言語モデルを活用した科学的統合

概要: 複数の研究や実験結果から一貫した結論を分析するメタ分析を化学文献で　LLMで実施する為に RAGと逆コサイン距離（ ICD）を開発、ICD損失関数を使用し、文脈の方向的類似性を強調することで出力品質を向上できた

この研究では、大規模言語モデル　(LLMs) を用いて、科学文献におけるメタ分析を自動化する手法を提案しています。メタ分析は複数の研究結果を統合し包括的な理解を提供する強力な統計手法ですが、手動での実施は労力がかかりエラーが生じやすいという課題があります。本研究では以下を実現しました。

- **データ:** 科学データセットを使用し、 LLMsを微調整。

- **手法:** 情報検索強化生成　(RAG) と逆余弦距離　(ICD) という新しい損失関数を活用。

- **結果:** 微調整されたモデルが　87.6%の関連性のあるメタ分析要約を生成。

- **評価:** 人間による評価で非関連性が　4.56%から1.9%に低下。

この研究はリソースの制約がある環境で実施され、メタ分析自動化の効率性と信頼性を向上させる重要な貢献をしています。

先行研究と比較しての特徴

1. **新しい損失関数の導入　:** 従来の方法では対応が難しい大規模データセットの扱いを改善するため、「逆余弦距離　(ICD)」を開発。

FlexFL: Flexible and Effective Fault Localization with Open-Source Large Language Models

概要: Fault localization (FL) は、バグ検出の為にバグ候補箇所をリスト化し、順番にチェックすることでバグの位置を自動で特定。特に Llama3-8Bを使用した FlexFLは、GPT-3.5を使用した AutoFLおよび AgentFLよりも多く特定できています

概要

Fault localization (FL) は、ソフトウェアシステム内のバグの位置を特定することでデバッグの効率を向上させ、ソフトウェア品質の向上に寄与する。 FlexFLは、オープンソースの大規模言語モデル (LLM) を用いてバグ関連情報を柔軟に利用できる FLフレームワークである。FlexFLは2段階で構成されており、まず最初に最新の FL技術を用いてバグの候補箇所を絞り込み、次にその候補を LLMで精査して最終的なバグ位置を特定する。

- **柔軟な情報利用 **: FlexFLはバグレポートやテストケースといった多様なバグ関連情報を利用できるため、バグの特定においてより柔軟である。

- **オープンソース LLMの使用 **: 従来の手法は GPT-3.5などのプロプライエタリな LLMに依存していたが、FlexFLはオープンソースの LLMを活用し、プライバシーリスクを低減している。

- **2段階プロセスの導入 **: バグの位置特定において 2段階の精査を行い、LLMの長いコンテキスト処理における制約を緩和し、より効果的にバグの位置を特定できるようにしている。

技術や手法

1. FlexFLの概要

Detecting Multi-Parameter Constraint Inconsistencies in Python Data Science Libraries Pythonデータサイエンスライブラリにおける多パラメータ制約の矛盾検出

概要: MPDetectorは、pythonライブラリとそのドキュメントの矛盾を LLMを使いデータの前処理、制約の抽出、矛盾の検出の 3段階のプロセスを用いて不一致を見つけます

技術や手法

1. **MPDetectorの概要**

MPDetectorは、APIドキュメントと対応するライブラリコードの間の矛盾をシンボリック実行と LLMを用いて検出します。3段階のワークフローを採用しており、データの前処理、制約の抽出、矛盾の検出のプロセスを経て、不一致を見つけ出します。

1. データの前処理 (Data Preprocessing)

この段階では、対象とする Pythonデータサイエンスプロジェクトから、ドキュメントとコードを分離し、シンボリック実行が実行しやすい形に整備します。具体的には以下のプロセスを含みます。

- **コードとドキュメントの分離 **: プロジェクトの各 Pythonファイルを抽象構文木(AST)に変換し、クラスや関数を分離します。これにより、ドキュメント付きの関数やクラスを抽出しやすくなります。

- **クラスの関数の分離と変換 **: Pythonのシンボリック実行ツールはクラスの解析に限界があるため、クラス内のメンバー関数を独立した関数として扱います。メンバー変数も対応するシンボリック入力に変更されます。

- **ドキュメントの整形 **: ドキュメント内のパラメータ記述をパラメータ -説明の形式で抽出し、制約に関連する記述のみを保持します。これにより、大規模言語モデル(LLM)による正確な解析が可能になります。

A Taxonomy of AgentOps for Enabling Observability of Foundation Model based Agents 基盤モデルに基づくエージェントの可観測性を実現するための　AgentOpsの分類

概要: LLMのAgentlにより自動化タスクが可能になりましたが複雑さが増すことで、結果の信頼性や開発過程の可視化が必要であり、そのための開発から運用までの全てのプロセスを監視・追跡できるように設計したAgentOpsを提案

エージェントの各ステップを記録し、可視化します。

技術や手法

AgentOpsというエージェントの開発から運用までのプロセス全体を包括的に管理し、エージェントの可観測性と追跡可能性を確保することに焦点を当てています。

1. **AgentOpsのフレームワーク **:

- **AgentOpsのプラットフォーム **は、LLMベースの自律エージェントを管理するためのインフラを提供します。　DevOpsやMLOpsと類似した概念で、開発（ Dev）から運用（ Ops）に至るまでの全プロセスを支援します。

- **AgentOpsは以下の主な機能 **を含みます:

- **エージェントの開発 **: AIエージェントの設計、構築、評価を行うためのフレームワークを提供します。

- **プロンプト管理 **:エージェントに使用するプロンプトをバージョン管理し、エージェントの動作を最適化します。プロンプトプレイグラウンドを利用して異なるプロンプトやモデルをテストできます。

- **観測可能性（ Observability）ツールの統合 **:エージェントのワークフロー、プロンプト、ツールの利用、アクションの全プロセスを追跡し、どのステップで何が行われているのかを完全に可視化します。これにより、開発者はエージェントの動作を理解し、デバッグしやすくなります。

- **追跡可能性（ Traceability）**:エージェントの実行過程をトレース（追跡）することで、どのようにして特定の結論に至ったかを明確に記録します。これにより、特定の問題がどこで発生したのかを特定　し、修正することが容易になります。

- **ガードレール（ Guardrails）**:エージェントの動作に対する制約や規則を設定し、意図しない動作やリスクのある行動を未然に防ぎます。ガードレールは、エージェントの動作をブロックしたりフィルタリングしたり、人間の介入を促したりすることで、エージェントの安全性を確保します。

FTA generation using GenAI with an Autonomy sensor Usecase 自動運転センサーのユースケースにおける　GenAIを使用したFTA生成

概要: オープンソースの LLMを使用して Lidarセンサーの故障に関する故障の木解析　(FTA)をするためにプロンプトエンジニアリングで　PlantUMLを使いFTA図を作成

LLMとPlantUMLを使用して FTA図を自動生成します

FTA図を作成する手順

1. **PlantUMLのインストールと準備 **
   - まず、PlantUMLというツールが必要です。このツールは、シンプルなテキストベースの記述で　UML図を生成できるもので、様々な開発環境（例えば　VS Code）で拡張機能として利用可能です。
   - PlantUMLをインストールするには、 Javaが必要になりますので、事前に Javaをインストールしておきましょう。
2. **大規模言語モデル（ LLM）を利用したプロンプト生成 **
   - LLM（例えばChatGPTやPerplexity）を使用し、FTA図に必要な要素をプロンプトとして入力します。例えば、「　Lidarセンサーの FTAを生成して欲しい」といった具体的な要求を行います。
   - プロンプトエンジニアリングを行い、適切な要素（例えば故障原因、環境要因、ハードウェアやソフトウェアの故障など）を得て、それを　FTA図に反映させます。
3. **PlantUMLコードの生成 **
   - LLMに生成してもらった情報をもとに、 PlantUMLコードを作成します。以下の例は、 Lidarセンサーの故障の木解析を表す PlantUMLのコードです:

   ```
   @startuml LIDAR Sensor Failure FTA
   ```

Leveraging LLMs for Legacy Code Modernization: Challenges and Opportunities for LLM-Generated Documentation LLMを活用したレガシーコードのモダナイゼーション　: LLM生成ドキュメントの課題と機会

概要: LLMでレガシーコードのドキュメントを生成する方法を　MUMPS言語の電子健康記録（ EHR）システムとIBMメインフレームアセンブリコード（ ALC）の2つのデータセットを使い LLMで行ごとにコメント生成しそのドキュメントの完全性、読みやすさ、有用性、ハルシネーションの有無を評価。結果は、　 LLMが生成したMUMPSとALCのコメントは正確でだったが、 ALCのようなアセンブリ言語のコメントになぜこの処理が必要かの情報が少なく実用性は薄かった

1. **データセット **

この研究では、LLMを用いたコメント生成を評価するために、 **MUMPSとALCの2種類のレガシーコード **を使用しました。

a. **MUMPSデータセット **

- **MUMPS**（Massachusetts General Hospital Utility Multi-Programming System）は1960年代に開発された医療記録管理のための高水準言語です。　MUMPSは、医療分野で広く使用されていることから、特に米国退役軍人医療システムで使用される　VistA（Veterans Health Information Systems and Technology Architecture）のコードを選定しました。

- 本研究では、この **VistAの「不完全記録追跡モジュール」 **を使用しました。このモジュールは、コメントの量やファイル数の面で　 VistA全体の平均を代表するものであり、比較的標準的な例を提供するために選ばれました。

- MUMPSデータセットには **78のファイルが含まれ、合計 5,107行のコード **と**235の開発者コメント **が存在しています。

b. **ALCデータセット **

- *IBMメインフレームアセンブリコード（ ALC）**は、主にメインフレームシステムで使われる低レベルのアセンブリ言語です。この研究では、　 Walmartのオープンソースの zFAMリポジトリからサンプルを取得しま した。

An Empirical Study on LLM-based Agents for Automated Bug Fixing 自動バグ修正のための LLMベースエージェントに関する実証研究

概要: 複数のLLMエージェントを用いたバグ修正の研究により、 MarsCode Agentが最も優れた性能を示し、フォールトローカライゼーション精度とバグ再現の効果が修正成功率に影響することが確認できる結果に

技術や手法

LLMベースのエージェントによる自動バグ修正の過程を以下のように詳細に分析しています。

1. **システム間の性能差 **:

   - 7つのシステムの中で、商用システムである MarsCode Agentが最も優れたパフォーマンスを示し、 SWE-bench Liteにおける問題の 39.3%を解決しました。

   - MarsCode Agentの次に高いパフォーマンスを示したのは Honeycombで、これらのシステムはフォールトローカライゼーション（バグの位置特定）と修正の生成の両面で優れた能力を発揮しました。

2. **フォールトローカライゼーションの精度 **:

   - ファイルレベルのフォールトローカライゼーションでは、ほとんどのシステムが高い精度を示しましたが、行レベルのフォールトローカライゼーションではさらなる向上の余地がありました。

   - 行レベルの精度が高いシステムほど、最終的な修正の成功率が高いことが示されました。これは、バグの具体的な位置をより正確に特定することが、修正の成功に直結しているためです。

3. **バグ再現能力の効果 **:

   - バグ再現は、バグの原因特定に追加情報を提供し、修正候補の正確性を確認する役割を果たします。成功した修正の中で再現を利用した割合は、例えば Honeycombでは100%、MarsCode Agentでは70.3%と、システムによって異なります。

   - 一方で、既に十分な情報が提供されている場合には、バグ再現がモデルの判断を誤らせることがあり、その結果、修正の成功率が低下することも観察されました。

4. **改善すべき点 **:

LLMs Do Not Think Step-by-step In Implicit Reasoning 大規模言語モデルは暗黙的な推論において逐次的に考えない

概要: LLMのの暗黙的な連鎖推論（ Chain-of-Thought, CoT）が、明示的な CoTと同等かを調査。暗黙的な CoTでは中間ステップの結果をほとんど計算してなく初期値や最終結果を使い、数式の順序を逆にする など問題の提示方法を変えただけでも大幅に結果が悪くなりましたが、明示的な CoTではそのようなことは起こりませんでした

結論としては明示的は CoTを複雑な問題を解決するために暗黙的 CoTは初期値や最終結果を使う直観と経験に基づいていて代替えできませんでした

技術や手法

1. **実験デザイン **:
   - 算術の多段階問題を用意し、通常の Chain-of-Thoughtを使わずに解答を直接出力するようモデルを誘導。
   - Qwen2.5-72B-Instructモデルを使用し、隠れ層の状態を記録し、中間結果を線形分類器で予測する手法を取った。
2. **プロービング技術 **:
   - 各隠れ層から抽出した情報を用いて、中間ステップが計算されているかを確認するために、 1層の多層パーセプトロン（ MLP）を使用。
   - 結果として、モデルは最初と最後のステップの結果は認識できるが、中間ステップの情報はほとんど計算していないことが示された。
3. **修正されたプロンプトの影響評価 **:
   - 問題の順序を逆にしたり、数値を 10で割ったりした変形問題を与え、暗黙的推論と明示的推論でどのように結果が変わるかを評価。
   - 結果、暗黙的推論ではパフォーマンスが大きく低下したが、明示的推論では完璧な結果を維持した。

暗黙的な連鎖推論（ Chain-of-Thought, CoT）が明示的な CoTと同等かを調査した結果、以下のことが明らかになりました。

Scholar Name Disambiguation with Search-enhanced LLM Across Language 検索拡張LLMを用いた学者名の言語横断的な識別

概要: 検索拡張LLMを用いて学者名の識別を多言語対応で改善。検索エンジンで豊富な情報を収集し、識別精度を向上。実験で識別性能が向上したことを確認

1. 研究の背景と課題

学者名の識別(Name Disambiguation)は、学術的なデータ処理や著者情報の特定において重要な課題です。この問題は、同じ名前を持つ異なる学者のプロフィールを正確に区別することが難しいため、複雑な異種データが絡む実際のデータセットでの精度向上が難しいという特徴があります。また、これまでのアプローチは人間の介入が多く必要であり、大規模データセットでの処理が難しいという課題がありました。

本研究では、**検索エンジンと大規模言語モデル(LLM)を組み合わせることで、学者名の識別を効率的に行う新しいアプローチを提案しています。このアプローチは、特に多言語環境 **における識別精度を高めることを目指しており、検索エンジンの **クエリ再構成 **や**意図認識 **を活用して、学者のプロフィールに関する情報をより豊富に収集します。

2. 提案する手法の概要

本研究では、以下の 3つのエージェントを中心に据えた手法を提案しています。各エージェントは、それぞれ特定のタスクを遂行し、相互に補完し合うことで識別精度を向上させます。

2.1 Extract Agent(学術プロフィール抽出エージェント)

Extract Agentは、学者に関する情報(名前、所属機関など)を入力として、関連する文献、プロフィール、発表などを抽出します。このエージェントは主に以下の手順で動作します:

Automated Literature Review Using NLP Techniques and LLM-Based Retrieval-Augmented Generation NLP技術とLLMベースのリトリーバル拡張生成を用いた自動文献レビュー

概要: RAGを使用して PDFを入力して文献レビューを自動生成する　spaCy、Simple T5、GPT-3.5-turboの3つの手法を ROUGEスコアで評価、GPT-3.5-turboが良くそれに基づき複数アップロードや進捗状況を視覚化した GUIを作成

先行研究と比べてどこがすごいのか

本研究では、PDFファイルのみを入力とする文献レビュー生成の完全なシステムパイプラインを提案し、ユーザーが　DOIとPDFを提供するだけで自動的に文献レビューを生成できる点が他の研究と異なります。また、3つの異なる手法（周波数ベース、トランスフォーマーモデル、　LLMベース）を用いたシステムを比較し、最良のアプローチを特定し、　GUIを構築することで、ユーザーが手軽に利用できるツールを提供しています。

技術や手法の詳細

1. **周波数ベースのアプローチ（ spaCy）**:
    - テキストを NLPトークンに変換し、ストップワードや句読点を削除。
    - 単語頻度を計算し、各文の重要性を評価。
    - トップ10%の文を最終出力として選定。
2. **トランスフォーマーベースのモデル（ Simple T5）**:
    - SciTLDRデータセットを使用してトレーニング。
    - 各論文を個別に要約し、モデルを評価。
    - モデルは DOIとPDFを入力とし、抽出したテキストを用いて最終的な文献レビューを生成。

From Generation to Judgment: Opportunities and Challenges of LLM-as-a-judge 生成から判断へ： LLMを審査官として利用する機会と課題

概要: LLMを使った評価手法 LLM-as-a-judgeを提案

LLMが「何を評価するか（例えば、助けになるか、有害でないか、信頼性など）」「どのように評価するか（手法やプロンプト設計）」「どこで評価するか（評価、整合性の調整、検索など）」という三つの視点から評価プロセスを分類・評価します

1. LLM-as-a-judgeの導入背景と目的

NLP分野において、生成された内容の評価は重要ですが、従来の評価方法（　BLEUやROUGEなど）は主に語彙レベルの一致に依存しており、ニュアンスや内容の質の違いを適切に捉えることが難しい問題がありました。これに対し、「 LLM-as-a-judge」は、より柔軟で適応的な評価を行うための新しい枠組みを提案します。

この手法の核は、 LLMが候補者（生成されたテキストなど）を対象にその質や関連性、信頼性を評価し、その結果を基にランク付けや選択を行うというものです。

2. LLM-as-a-judgeのタクソノミー（分類）

LLM-as-a-judgeの分類は以下の 3つの側面に基づいています:

1. **何を評価するか（ Attribute）**: どの属性を評価するかを定義する側面です。
    - **助けになるか（ Helpfulness）**: 生成された内容がユーザーにとって役立つかどうか。
    - **有害でないか（ Harmlessness）**: 内容が安全であるか、倫理的に問題がないか。

Is my Meeting Summary Good? Estimating Quality with a Multi-LLM Evaluator 私の会議要約は良いか？マルチ LLM評価者による品質の推定

概要: 会議要約の評価に LLMを使用するフレームワークとして MESAを提案

個別のエラータイプを三段階（エラー検出、影響評価、スコア付け）で評価し、多エージェントによる議論で精度を高め、フィードバックに基づく自己訓練でエラー定義の理解を深めます。これにより、エラー検出や一貫した評価が可能となり、人間の判断により近い評価をしています

先行研究と比較して

1. **既存の自動評価基準の限界 **: ROUGEやBERTScoreなどの既存の自動評価指標は、人間の判断と相関が低く、要約に含まれる微妙なエラーを捉えることが難しいとされています。この問題を解決するために、最近では LLMを利用した要約評価が提案されていますが、それでもなお誤りを見逃したり、評価の一貫性に欠けることがあります。

2. **MESAの改善点 **: MESAは、個々のエラーを多段階で評価することで、詳細なエラー検出と一貫した評価が可能です。また、マルチエージェントディスカッションにより、異なる視点からの評価が可能となり、より多角的で精度の高い評価を実現しています。

技術や手法

1. **三段階評価プロセス **: 各エラータイプに対して、三段階のプロセス（エラー検出、影響評価、スコア付け）を使用します。これにより、エラーの存在や影響度を正確に把握します。

2. **マルチエージェントディスカッション **: 複数のエージェントが協力して初期評価を精査・修正します。このプロセスで、誤検出を減らし、評価の精度を高めています。

3. **フィードバックに基づく自己訓練 **: MESAは人間の注釈データと比較し、フィードバックを元に評価の修正を行う自己訓練メカニズムを備えています。これにより、人間の判断との整合性を高めます。

使用用途

GUI Agents with Foundation Models: A Comprehensive Survey 基盤モデルを活用した GUIエージェント：包括的サーベイ

概要：LLMGUIエージェントに関連するデータセット、フレームワーク、アプリケーションをまとめたサーベイ

GUIエージェントの 5つの主要構成要素（ GUI Perceiver, Task Planner, Decision Maker, Memory Retriever, Executor）として分類提示

GUI Agent の構成要素

1. GUI Perceiver

概要

GUI Perceiverは、GUIエージェントがユーザー入力を解釈し、 UIの変化を認識するための基盤となる部分です。このモジュールは、エージェントがユーザーインターフェースを正確に理解し、タスクを達成するための最適な操作を選択する助けとなります。

詳細

- **目的**:
  - ユーザーの指示（自然言語やその他の形式）を理解。
  - GUI（グラフィカルユーザーインターフェース）の構造や状態を把握。
- **アプローチ**:

Multi-expert Prompting Improves Reliability, Safety and Usefulness of Large Language Models マルチエキスパートプロンプティングが大規模言語モデルの信頼性、安全性、実用性を向上させる

月: Nov

公開日: 1

概要: 複数の専門家プロンプトがそれぞれに回答を作り集約することで最良の応答を選択　NGTの7つのタスクを用い、LLMの信頼性、事実性、情報性、有用性を向上させます

技術や手法

マルチエキスパートプロンプティングは、以下の　2つの主要なステップで構成されます:

1. **エキスパートと応答の生成 **: 指示が与えられた際に、LLMは複数のエキスパートの役割をゼロショットプロンプティングで生成します。それぞれのエキスパートは短い役割説明を伴って指示に応答し、それぞれの回答が生成されます。この過程により、さまざまな視点が得られ、偏りのない質の高い応答が可能となります。

2. **エキスパート応答の集約 **: 各エキスパートの応答を集約し、個々の応答と集約された応答を評価して最良のものを選択します。この過程には、Nominal Group Techniqueに基づく7つのサブタスクが含まれています（合意の形成、意見の衝突の解決、ユニークな視点の抽出など）。これにより、幅広い観点を持つ総合的な応答が得られます。

これらのステップにより、マルチエキスパートプロンプティングは情報の正確性、有害性の低減、多角的な視点の提供を可能にしています。

Appendix