論文要約

LLM関連

MUTUAL REASONING MAKES SMALLER LLMS STRONGER PROBLEM-SOLVERS 相互推論により、小型 LLMがより強力な問題解決能力を発揮

概要: 小型言語モデル(SLM)の結果を向上させる rStarを提案。

推論結果を自己生成と相互生成のプロセスに分離、 SLMがモンテカルロ木探索(MCTS)を使い推論ステップを自動生成、その経路から正しいものを別の SLMで再現するかで判断し、推論精度を高めます

1. モンテカルロ木探索(MCTS: Monte Carlo Tree Search)

rStarアプローチでは、MCTSを活用して SLMが推論軌跡を生成します。 MCTSは、特定の問題に対して複数の推論軌跡を探索し、その中から最適な軌跡を選択するための手法です。以下のステップで MCTS の詳細な動作を説明します。

1. **探索と利用のバランス **: MCTSでは、探索(未探索の軌跡を探索すること)と利用(既知の高品質な軌跡をさらに検討すること)のバランスを取るために、 UCT(Upper Confidence Bound for Trees)という指標が使用されます。この指標により、未知の領域を探索しつつ、既に高評価を得た軌跡を適切に活用することができます。

2. **木構造の構築 **: 問題が与えられた場合、 MCTSはまずその問題を根(ルート)ノードとして木構造を構築し始めます。各エッジは推論ステップ(アクション)を表し、ノードはそのステップの結果として生成された状態(推論の中間結果)を表します。ルートノードから葉ノード(推論が完了するノード)までの経路が一つの推論軌跡(候補解)となります。

3. **軌跡の評価 **: 各軌跡が生成された後、その正確性が評価されます。この評価は、軌跡がどれだけ正しい解に近づいているかを示す報酬(リワード)によって行われます。正しい解に到達する確率が高い軌跡ほど高い報酬が与えられ、次の探索で優先されるようになります。

4. **反復と改善 **: MCTSはこのプロセスを複数回繰り返し、次第に最も正確な推論軌跡を見つけ出すようにします。これにより、 SLMは多様な候補解の中から、より良い推論ステップを学習し、高品質な最終解を導き出すことが可能となります。

2. 相互一致の検証 (Mutual Consistency Verification)

rStarは、生成された推論軌跡が本当に正しいかどうかを確認するために、相互一致の検証プロセスを導入しています。このプロセスには、別の SLMが識別器(ディスクリミネーター)として参加します。

Bridging Domain Knowledge and Process Discovery Using Large Language Models ドメイン知識とプロセス発見の橋渡しを行うための大規模言語モデルの使用

概要: LLMを使用してドメイン知識を自動プロセス発見に使うための手法を提案

IMrフレームワークをベースにしてイベントログから直接フォローグラフを抽出し、ルールベースでプロセス構造を選択、　　LLMでプロセス記述をルールに変換しプロセス発見の時にドメイン知識を活用するようにします

1. IMrフレームワーク

IMrフレームワークは、プロセス発見に使用されるインダクティブマイニングの一種です。このフレームワークは、イベントログから「直接フォローグラフ（　　Directly Follows Graph, DFG）」を抽出し、ルールに基づいてプロセス構造を選択します。

手法の詳細 :

1. **直接フォローグラフ（ DFG）の抽出 **:

- IMrフレームワークでは、まずイベントログから　DFGが抽出されます。DFGは、一連の活動がどのように順番に発生するかを表すグラフであり、プロセスの流れを視覚化するために使用されます。

- DFGは、すべてのアクティビティ（イベントの種類）　Σとその間の直接の後続関係を示すエッジで構成されます。これにより、あるイベントが発生した後に次にどのイベントが続くかをグラフィカルに示します。

2. **バイナリカットの探索 **:

- 次に、IMrはΣ内のアクティビティを 2つの部分に分割するバイナリカットを探索します。これらのカットは、プロセスの構造を定義するもので、シーケンス、排他的選択、並列、ループなどのタイプに基づいて分類されます。

- IMrアルゴリズムは、すべての可能なカットを検討し、ルールセットによって違反のないカットのみを残して探索します。

3. **ルールに基づくカットのフィルタリング　**:

Pron vs Prompt: Can Large Language Models already Challenge a World-Class Fiction Author at Creative Text Writing? Pron対Prompt: 大規模言語モデルは世界クラスの小説家に創造的な文章で挑むこ　とができるか？

概要: 文章執筆を GPT-4とPatricio Pronというトップ小説家とで競うコンテストを実施。

30個の映画タイトルを提案し、互いにそのタイトルに基づいた短編小説のあらすじを執筆。

Bodenの創造性の定義（新規性、驚き、価値）に基づき、専門家が評価。

トップ作家には及ばないという結果に

1. コンテスト設計

目的: 大規模言語モデル（ LLM）がトップレベルの小説家とどれほど創造的な文章執筆能力を持っているかを比較するために設計されたコンテスト。

手順:

1. **タイトルの提案 :**
    - Patricio PronとGPT-4それぞれに、30個の映画タイトルを提案させます。これにより、全体で　60個の映画タイトルが集まります。
2. **シノプシスの作成 :**
    - それぞれの映画タイトルに対して、 600語程度のシノプシス（あらすじ）を作成します。シノプシスは、自分自身が提案したタイトルおよび相手が提案したタイトルの両方に対して作成されます。
    - 例えば、GPT-4が提案したタイトルに対しては、 Patricio PronとGPT-4の両者がシノプシスを作成し、逆もまた同様です。
3. **評価対象の作品 :**

SYNTHEVAL: Hybrid Behavioral Testing of NLP Models with Synthetic CheckLists SYNTHEVAL: 合成チェックリストを用いた NLPモデルのハイブリッド行動テスト

概要: LLMを使用して自動的にテストタイプを生成し、注釈作業を減らすテストフレームワーク SYNTHEVALを提案。

LLMとタスクモデルを活用して多様で複雑なテストケースを自動生成し、 NLPモデルの潜在的な弱点や行動パターンを体系的に検出・評価することにより、静的なベンチマークでは見つけにくいモデルの脆弱性を見つけることが目的です

SYNTHEVALの手法は、主に NLPモデルの脆弱性や行動上の欠陥を特定するために、合成テストケースを生成し、モデルの性能を評価するための 3つの主要なステージから成り立っています。

1. **合成テストセット生成 (SynthTest)**

このステージでは、LLM(大規模言語モデル)を活用して多様なテストケースを生成します。生成されたテストケースは、モデルが予測に苦労するような複雑な言語構造や特定の表現を含むことを目的としています。以下の手順で進められます。

ステップ 1.1: 既存データセットからの単語のランダムサンプリング

まず、既存のデータセット(例 : IMDbやSST-2などの分類タスク用データセット)から単語をランダムにサンプリングします。この単語は生成するテスト文の基準となり、 LLMにこれを基にした文章を生成させます。たとえば、「 book(本)」や「 awful(ひどい)」といった単語をサンプリングします。

ステップ 1.2: LLMを使ったテキスト生成

ランダムサンプリングされた単語をプロンプトとして、 LLMに文章を生成させます。この際、 nucleusサンプリング(p=1.0)などの手法を用いて多様性の高いテキスト生成を行います。生成する文は、特定のタスクにおいてモデルが予測に苦労しやすいものとなるように設計されています。このプロセスで、最大 100,000文のテスト文を生成します。

Exploring the applicability of Large Language Models to citation context analysis 大型言語モデルの引用文脈分析への適用性の探究

概要: LLMと人間の注釈結果を比較して引用分析を行う　LLMは一貫性のでは優れているが、予測性能は低い。現状　LLMは人間の注釈者を完全に置き換えることはできないが、参考情報として使用できる。という結果を一貫性（ Cohen’s kappa）と予測性能（正確度、 F1スコア）で評価。

技術や手法

1. **データ収集 **: Nishikawa (2023)のデータセットを使用し、引用目的と引用感情の　2つのカテゴリに注釈を付ける。
    - **引用目的（ Citation Purpose）:**

        引用が行われる目的を 5つのクラスに分類（背景、比較、批判、証拠、使用）。

    - **引用感情（ Citation Sentiment）:**

        引用時の著者の精神的態度を 3つのクラスに分類（ポジティブ、ネガティブ、中立）。

2. **LLMの使用 **: OpenAPI Inc.のAPIを使用し、gpt3.5-turbo-0310モデルで注釈を行う。
3. **プロンプト設計 **: 人間の注釈マニュアルに基づき、複数のプロンプトパターンを設定。
4. **評価指標 **: 一貫性（ Cohen’s kappa）と予測性能（正確度、 F1スコア）で評価。

CRAG - Comprehensive RAG Benchmark CRAG - 包括的なRAGベンチマーク

概要: CRAGはRAG評価ベンチマークで 5つのリアリズム、豊富さ、洞察力信頼性長寿性の要素を考慮し、ウェブ情報の要約、それとナレッジグラフを組み合わせ回答を生成、評価を実施することでドキュメント内容の正確性を自動チェックができるようにしています

技術や手法 :

CRAGは、RAGシステムの評価に 5つの重要な要素を考慮しています :

1. **リアリズム **: 実際の使用ケースに近い現実的なシナリオを反映。

2. **豊富さ**: 多様で複雑な QAペアを含み、システムの限界を明らかにする。

3. **洞察力 **: データの異なるスライスに基づいて性能を評価し、システムの能力を把握する。

4. **信頼性 **: 評価結果が信頼でき、再現可能であること。

5. **長寿性 **: データが時を経ても陳腐化せず、定期的に更新されること。

CRAGは、5つのドメイン (金融、スポーツ、音楽、映画、オープンドメイン) と8つの質問タイプ (条件付き質問、比較質問、セット質問、マルチホップ質問、など) をカバーし、現実世界の質問に近づけるために多様な表現や時間的変動を含んでいます。また、モック APIを使用して、動的な情報検索を模擬します。

CRAG - Comprehensive RAG Benchmark CRAG - 包括的なRAGベンチマーク

概要: CRAGはRAG評価ベンチマークで 5つのリアリズム、豊富さ、洞察力信頼性長寿性の要素を考慮し、ウェブ情報の要約、それとナレッジグラフを組み合わせ回答を生成、評価を実施することでドキュメント内容の正確性を自動チェックができるようにしています

技術や手法 :

CRAGは、RAGシステムの評価に 5つの重要な要素を考慮しています :

1. **リアリズム **: 実際の使用ケースに近い現実的なシナリオを反映。

2. **豊富さ**: 多様で複雑な QAペアを含み、システムの限界を明らかにする。

3. **洞察力 **: データの異なるスライスに基づいて性能を評価し、システムの能力を把握する。

4. **信頼性 **: 評価結果が信頼でき、再現可能であること。

5. **長寿性 **: データが時を経ても陳腐化せず、定期的に更新されること。

CRAGは、5つのドメイン (金融、スポーツ、音楽、映画、オープンドメイン) と8つの質問タイプ (条件付き質問、比較質問、セット質問、マルチホップ質問、など) をカバーし、現実世界の質問に近づけるために多様な表現や時間的変動を含んでいます。また、モック APIを使用して、動的な情報検索を模擬します。

A Comparative Study on Large Language Models for Log Parsing ログ解析のための大規模言語モデルに関する比較研究

概要: 複数のLLMでシステムログの抽出性能を比較、オープンソース　LLMもよい結果を出し、特に CodeLlamaはGPT-3.5を上回る性能を発揮。

各モデルは、ログメッセージの正確な解析や構文的な類似度によって評価

技術や手法

1. **ログ解析手法 **:
   - ログ解析はログメッセージを構造化されたログテンプレートに変換するプロセスです。これにより、動的な変数と定数の部分を区別します。
   - 本研究では、ゼロショットプロンプトと少数ショットプロンプトという　2つの異なるプロンプト手法を設計し、 LLMが生成するテンプレートを評価しました。
2. **LLMの比較 **:
   - 6つのLLM（GPT-3.5、Claude 2.1、Llama 2、CodeLlama、Zephyr、CodeUp）を比較し、それぞれのモデルがシステムログに対してどのようにテンプレートを抽出するかを調査しました。
   - 各モデルは、ログメッセージの正確な解析（　Parsing Accuracy: PA）や構文的な類似度（編集距離や最長共通部分列　: ED, LCS）によって評価されました。
3. **評価手法 **:
   - 正確さと構文的類似性を評価するため、ログテンプレートの正確な抽出数と、生成されたテンプレートと正解との間の文字列類似度（編集距離や最長共通部分列）を計測しました。
   - 正確さの指標として、完全に一致するログテンプレートの数をカウントし、類似性の指標としては、　Levenshtein距離や最長共通部分列を用いて評価しました。

Attention Heads of Large Language Models: A Survey　大規模言語モデルにおけるアテンションヘッド　: サーベイ

概要: LLMの内部推論プロセス理解のためにアテンションヘッドに焦点を当て、推論メカニズムを分類・整理したサーベイ。

1. **人間の思考プロセスとアテンションヘッドの関連性　**

論文は、アテンションヘッドが　LLMの推論プロセスにどのように関与しているかを明確にするために、人間の思考プロセスを　　4つの段階に分け、その段階に対応するアテンションヘッドの機能を整理しています。これにより、　LLMの推論プロセスがより明確に理解できるように工夫されています。　　4つの段階とは以下の通りです。

1. **知識の呼び出し（ Knowledge Recalling, KR）**

　この段階では、過去に学習した知識が　LLMのパラメータから呼び出され、推論に活用されます。　LLMが保持する「パラメトリックな知識」と呼ばれる情報が、アテンションヘッドによって復元され、後の推論プロセスに提供されます。これにより、例えば、常識や専門知識が文脈に基づいて適切に活用されます。

2. **文脈の理解（ In-Context Identification, ICI）**

　次に、与えられた文脈を理解する段階です。アテンションヘッドは、文中の特定のトークンに焦点を当て、文脈内の情報を抽出し、識別します。特に、トークンの位置や構造、シンタックス（文法）やセマンティックス（意味）の情報を集約し、　LLMの推論に役立てます。

3. **潜在的な推論（ Latent Reasoning, LR）**

Planning In Natural Language Improves LLM Search For Code Generation　自然言語での計画がコード生成のための　LLM検索を改善する

概要: LLMがコード生成で似たような生成を繰り返すことを防ぐため、問題を解決するために役立つ洞察や手がかりを観察として生成し、それを組み合わせて解決策を導出する計画を作成する　PLANSEARCH を提案

HumanEval+、MBPP+、LiveCodeBenchで評価を行い、特に Claude 3.5 SonnetとPLANSEARCHでpass@200 = 77.0%というパフォーマンスでした

技術や手法

1. **PLANSEARCHのアルゴリズム **:

- 自然言語による観察を生成し、それらを組み合わせて解決策を導出する計画を作成します。各計画は多様なアイデアを反映し、その後のステップでその計画に基づいてコードを生成します

- PLANSEARCHは、1次および 2次の観察（アイデア）を生成し、これを組み合わせて問題を解決するための計画を作ります。

コード生成の問題に対して、モデルは最初に次のような「観察」を行います。

-「この問題ではハッシュマップを使うべきだ」

-「バイナリサーチを使うと効率的だ」

-「貪欲法で解決できる」

これらの観察は、モデルが問題に対してどうアプローチするかの高レベルな計画を示すものです。観察を組み合わせて問題解決のための計画を作成し、その計画に基づいて最終的なコードを生成します。

How to Align LLM for Teaching English? Designing and Developing LLM based-Chatbot for Teaching English Conversation in EFL, Findings and Limitations 英語教育のための LLMの調整方法は？ EFLにおける英会話教育用の LLMベースのチャットボットの設計と開発、発見と限界

概要: 英語を外国語（ EFL）として学ぶ環境で英会話を教えるための　LLMチャットボットの設計開発評価の紹介　ニーズ分析を行い、設計原則を確立、 CoTや指示と出力のペアを使用してモデルの動作を制御、GPT-4を使用して定量的、教師に聞き定性的に評価

技術や手法

1. **設計と開発の研究（ DDR）**: ニーズ分析、設計原則の確立、プロトタイプの改良を通じて、教育技術の開発と評価を行う。

2. **LLMの整合方法 **:

   - **インストラクションチューニング　**: 指示と出力のペアを使用してモデルの動作を制御。

   - **プロンプト戦略 **: タスク固有の例をプロンプトに提供し、モデルの出力をガイド。

3. **評価方法 **:

   - **定量的評価 **: GPT-4を使用して他の LLMの性能を評価。

   - **定性的評価 **: 英語教師とのインタビューを通じて、チャットボットの実用性と教育的価値を評価。

An Empirical Study on Self-correcting Large Language Models for Data Science Code Generation データサイエンスのコード生成における自己修正型大規模言語モデルの実証研究

概要: CoTを使用して生成したコードを段階的に改善する　CoT-SelfEvolveを提案。

StackOverflowからの知識ベースを利用し、 CoTでガイダンスを生成し初期コードを生成

これを構文チェックしエラーが無い場合ユニットテストを実行

実行時に発生したエラーを基にさらに　CoTプロンプトを生成コードを改良しつつ正確なコードを生成するまで繰り返します

1. CoT-SelfEvolveフレームワークの詳細

CoT-SelfEvolveは、LLMを利用してコードを生成し、そのコードを反復的に自己修正するフレームワークです。このフレームワークは、以下の　　3つのステージから成り立っています。

ステージ1: 外部知識の取得と初期コード生成

まず、CoT-SelfEvolveは問題の説明を受け取り、問題の内容に応じて外部の知識を取得します。この外部知識として主に活用されるのは、開発者のディスカッションが集まった　　StackOverflowのようなプラットフォームです。ここでの手順は次の通りです:

1. **外部知識の取得 **

Beyond designer’s knowledge: Generating materials design hypotheses via large language models 設計者の知識を超えて：大規模言語モデルによる材料設計仮説の生成

概要：LLMで材料設計を設計者の知識を超えるようなアイディアを生み出すためにプロンプトエンジニアリングを論文の収集、そこからプロセッシング（　　P）、メカニズム（M）、構造（S）、プロパティ（P）の関係を抽出。P-M-S-M-Pの形式でシステムチャートを作成し組み合わせて仮説生成します

**技術や手法 **

1. **大規模言語モデル（ LLM）の活用：**

   - **モデルの選択 :** GPT-4-1106-previewモデルを使用。

   - **プロンプトエンジニアリング :** 具体的な指示や制約を与えることで、モデルの出力をコントロールする。

2. **システムチャートの作成と情報抽出 :**

   - **ステップ I: 論文の収集 **

      - 一般的なキーワード（例 : "cryogenic high entropy alloy"や"high entropy alloy"）を用いて、多数の関連論文を収集。

      - 24本の論文を選定し、 LLMの知識カットオフ日（ 2023年4月）以前に公開されたものを対象とする。

   - **ステップ II: 情報の抽出と整理 **

      - 各論文から、プロセッシング（ P）、メカニズム（M）、構造（S）、プロパティ（P）の関係を抽出。

      - P-M-S-M-Pの形式でシステムチャートを作成し、 LLMが効率的に情報を処理できるようにする。

3. **仮説の生成 :**

   - **ステップ III: LLMによる仮説生成 **

      - 異なる論文からのメカニズムを組み合わせ、シナジー効果を生むような新しい仮説を生成。

Think Together and Work Better: Combining Humans' and LLMs' Think-Aloud Outcomes for Effective Text Evaluation 一緒に考え、より良い仕事を：効果的なテキスト評価のための人間と　LLMの考えながらの結果を組み合わせる

概要：InteractEvalは、人間の専門知識と LLMを考えながら法で統合し、チェックリストベースのテキスト評価のための属性を生成する手法を使い、評価性能を向上させる。より広範な関連属性を生成するためにTA（Think-Aloud, TA）法を用いて人間と LLMのアイデアを統合します

**技術や手法 **

1. **問題設定と目的 :**

　- テキスト生成の評価において、従来の　LLMベースの評価手法はタスク固有のプロンプトに依存し、信頼性が低下する問題がありました。また、チェックリストベースの手法では、人間が手動でキーコンポーネントを作成するため、バイアスや不十分な詳細性が懸念されました。

　- 本研究の目的は、人間の柔軟な思考と　LLMの一貫性を組み合わせ、バイアスを軽減しつつ詳細な評価を可能にする新しいフレームワーク「　　InteractEval」を提案することです。

2. **InteractEvalの構成 :**

　- **考えながら法（ TA）の適用 :** 人間の専門家と LLMが、それぞれテキスト評価時に考えるべき属性を考えながら法を用いて生成します。

　- **属性の統合とチェックリストの作成　:**

　　- **コンポーネント抽出 :** 収集した属性から主要な評価項目（コンポーネント）を抽出します。

　　- **属性のクラスタリング　:** 属性を対応するコンポーネントごとにグループ化します。

　　- **質問の生成 :** 各コンポーネントに対して　Yes/No形式の評価質問を作成します。

　　- **質問の検証 :** 質問の明確性や関連性を　LLMで検証し、冗長性を排除します。

WINDOWSAGENTARENA: EVALUATING MULTI-MODAL OS AGENTS AT SCALE Windowsエージェントアリーナ：マルチモーダル OSエージェントの大規模評価

概要: WindowsAgentArena（WAA）は、Windows OS上でエージェントが自由に操作できる環境を提供

WAAは実際の Windows OS環境でエージェントが自由に操作します、操作は　OCR、アイコン検出、UIAツリー解析を組み合わせて要素を解析しタスクを実行します

先行研究と比較しての優位性

1. **汎用性とスケーラビリティ **: これまでのエージェントベンチマークは、特定のドメインやモダリティに制限されているが、　WAAは多様なWindowsアプリケーションやウェブドメインに対応しており、エージェントがリアルな Windows環境で作業できる。

2. **高速な評価 **: Azureでの並列化により、従来のベンチマークが数日かかる評価を、　20分程度に短縮できる。

3. **新しいエージェントアプローチ **: 「Navi」というエージェントが、新しいセット・オブ・マーク（　SoM）プロンプト方式を用いてタスクを実行し、従来のエージェントに比べて優れたパフォーマンスを示している。

技術や手法

1. **環境の再現性 **: WAAは実際の Windows OS環境を使用し、エージェントが自由に操作できる再現性の高いプラットフォームを提供。これにより、人間と同様のタスクを実行できる。

2. **タスクの設計と評価 **: WAAは150以上のタスクを含んでおり、 LibreOffice、Microsoft Edge、Visual Studio Codeなどの一般的な Windowsアプリケーションにわたるタスクを含む。評価スクリプトは、エージェントのタスク完了を自動的に評価し、特定のエピソードの終わりに報酬を生成する。

3. **Azureでの並列実行 **: WAAはAzureクラウド上で複数の VMを使用し、ベンチマーク評価を並列化。ローカルとクラウドの両方で安全でスケーラブルな実行が可能。

4. **Naviエージェント **: Naviは、チェーン・オブ・ソート・プロンプト（ chain-of-thought prompting）を使用し、画面理解やツールの使用を行う。また、　OCR、アイコン検出、UIAツリー解析などの技術を組み合わせて、スクリーン上の要素を識別する「セット・オブ・マーク」を生成し、タスクを実行する。

Contri(e)ve: Context + Retrieve for Scholarly Question Answering Contri(e)ve: 学術的質問応答のための文脈と情報抽出

概要: 情報の抽出を、複数の知識グラフ(DBLP、SemOpenAlex)と非構造化データ(Wikipedia)を組み合わせるというハイブリッドアプローチを採用し、質問に答えるためのハイブリッドな質問応答システムを提案

また、プロンプトエンジニアリングで LLMで文脈の適切な抽出と回答生成を試し Fスコア40%を達成したが、いくつか異常な応答もあった

技術や手法 :

1. **データセット **:
   - Scholarly-QALDデータセットを使用し、 5000の質問とそれに対する回答ペアが含まれます。
   - 質問には、DBLPの著者 IDが付与されており、正確な回答を得るために複数のデータソースから情報を抽出する必要があります。
2. **コンテキスト抽出 **:
   - DBLP知識グラフから著者情報を SPARQLクエリで抽出し、ORCIDを用いて SemOpenAlex知識グラフからも追加情報を取得します。
   - Wikipediaからは著者や関連機関に関する情報を抽出しますが、テキストの要約やキーワード検索を用いて文脈を精査しています。
3. **プロンプトエンジニアリング **:
   - プロンプトは、質問、簡潔な質問、文脈、出力指示の 4つの部分から構成されています。文脈情報は著者、関連機関、出版物に関する情報に分かれて提示されます。
   - Llama 3.1モデルを使用して質問に答えるため、プロンプトの長さや内容を調整して精度を最適化しています。

GenCRF: Generative Clustering and Reformulation Framework for Enhanced Intent-Driven Information Retrieval GenCRF: 意図駆動型情報検索を強化する生成クラスタリングと再定式化フレームワーク

概要: クエリの意図を多様に捉えた情報検索の成功率を向上させるために、 GenCRF(生成クラスタリングと再定式化フレームワーク)を提案します。 GenCRFは、LLMを使用して多様なクエリを生成し、それらをクラスタリングすることで、多様なユーザー意図を効果的に反映させることを目指しています。また、重み付け集約戦略やフィードバックループを組み込んで、検索性能を最適化します。 BEIRベンチマークでの実験により、GenCRFは既存の手法を上回る性能を達成しました。

技術や手法 :

1. **生成とクラスタリング **:

   - クエリの意図を広げるために、 LLMを使って3種類のカスタマイズされたプロンプト(「コンテクスト拡張」「詳細指向」「アスペクト指向」)を用いて、クエリを多様に生成します。

   - 生成されたクエリをクラスタリングして、冗長な情報を排除し、異なる意図を明確に表現します。

2. **重み付け集約戦略 **:

   - **類似性ベースの動的重み付け **: 初期クエリとの類似性に基づいて、生成クエリを動的に重み付けします。

   - **スコアベースの動的重み付け **: 生成クエリの質を 5つの基準(関連性、特異性、明確性、包括性、有用性)で評価し、動的に重み付けします。

3. **クエリ評価報酬モデル(QERM)**:

   - 生成されたクエリの質を評価し、再生成や再クラスタリングが必要かどうかを判断します。これにより、最適なクエリを生成します。

4. **ファインチューニング **:

   - **ScoreDW-FT**というファインチューニング手法を用いて、 LLMのクエリ評価能力を向上させ、生成クエリの質を最適化します。

TART: An Open-Source Tool-Augmented Framework for Explainable Table-based Reasoning TART: 説明可能なテーブルベースの推論のためのオープンソース・ツール強化フレームワーク

概要: TARTは、表構造を理解し、 クエリに応じてテーブル整形、適したツールを　pyrhon関数としてテーブル操作や数値推論を特定の計算ツールの作成、ツールを利用した推論の過程を人間に理解しやすい形で説明する説明生成の 3段階で構成で表形式に対応します

先行研究との比較

- 従来の **Chain-of-Thought (CoT)**推論は柔軟性があるものの、テーブル操作や数値推論の正確性に欠ける傾向がありました。

- *Program-based reasoning (PoT)**アプローチは正確なコード生成ができる一方で、推論の計画性に欠け、理解が難しいという欠点がありました。

- TARTは、ツールを外部から呼び出しながら推論を行う新たなアプローチを提供し、従来の方法の弱点を補完しています。

論文で説明している技術や手法

TARTの技術的な詳細について順番に説明します。

1. **テーブルフォーマッタ** (Table Formatter)**:

Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks 思考プログラムプロンプティング：数値推論タスクにおける計算と推論の分離

概要：Program of Thoughts（PoT）は言語モデルとプログラミング言語を用いた手法で言語モデルは推論のみを行い、計算は　Pythonインタープリタに委任します。

**Program of Thoughts（PoT）の技術や手法の詳細 **

1. PoTの概要

Program of Thoughts（PoT） は、数値推論タスクにおいて、言語モデルが推論ステップを生成し、計算部分を外部のプログラムインタープリタに委任する手法です。これは、言語モデルが推論と計算を同時に行う従来の **チェーン・オブ・ソーツ（ CoT）**の問題を克服するために提案されました。具体的には、　PoTでは、言語モデルが Pythonプログラムの形で推論のステップを生成し、そのプログラムを　Pythonインタープリタが実行します。

2. PoTとCoTの違い

CoTでは、言語モデルが推論ステップと計算を同時に行うため、以下の問題が発生します：

The title of the uploaded paper is "TO COT OR NOT TO COT? CHAIN-OF-THOUGHT HELPS MAINLY ON MATH AND SYMBOLIC REASONING The Japanese translation of the title is:「COTを使うか、使わないか？チェイン・オブ・ソートは主に数学と象徴的な推論で役立つ」 .

概要: CoTがどのタスクで効果的かを 100以上の論文を使用し、 20のデータセットと 14のモデルを評価しました。その結果、 CoTは主に数学や論理的なタスクで顕著な性能向上をもたらし、それ以外のタスクでは効果が少ないことがわかりました。

先行研究と比較して

CoTの適用が数学や記号操作に限られることを示すとともに、特に「 =」などの記号が含まれるタスクで効果があることを明確に示しています。これまでの研究では、 CoTの効果を広範囲にわたるタスクに適用していましたが、本研究はその有効範囲を特定し、 CoTが常に効果的ではないことを証明しています。

技術や手法:

- **メタアナリシス **: 100以上の CoTを用いた論文を分析し、 CoTがどのタスクに対して有効かを調査しました。

- **独自の評価 **: 20のデータセットと 14のLLMを使用し、数学や記号推論における CoTの有効性をテストしました。

- **タスク分類 **: タスクを記号操作、数学、論理推論、百科事典的知識などに分類し、それぞれのタスクに対する CoTの効果を比較しました。

使用用途:

- **数学および記号推論 **: 数学の問題や記号操作に関するタスクでは、 CoTが計算精度を向上させ、パフォーマンスの改善が見られる。

- **論理推論 **: 推論や計画立てのタスクでも、CoTが効果を発揮する。

Harnessing LLMs for API Interactions: A Framework for Classification and Synthetic Data Generation LLMを活用したAPIインタラクションのための分類と合成データ生成のフレームワーク

概要: LLMを使用して自然言語入力を対応する APIコールに分類し、API機能に特化したサンプルデータセットを自動生成するフレームワークの提案

GPT-4の分類精度が 0.996を達成

1. **API分類システム **

API分類システムは、自然言語で入力されたユーザーのクエリを LLM(大規模言語モデル)で解析し、適切な API呼び出しを自動的に行うためのフレームワークです。このシステムは次の 4つの主要なステージで構成されています。

1.1. **ユーザー入力(User Input)**

システムはユーザーから自然言語で入力されたクエリを受け取ります。このクエリは、事前に定義されたプロンプトと組み合わされ、 LLMに送信されます。プロンプトには APIの階層や応答の形式に関するルールが含まれています。これにより、システムが API構造に準拠した結果を返すことができるようになっています。入力されるクエリは、シンプルな質問から複雑な命令まで幅広く対応可能です。

1.2. **API分類(API Classification)**

ユーザーのクエリを受け取ると、統合された LLMがそれを適切な API機能にマッピングします。 LLMはクエリを解析し、API階層に基づいてカテゴリ(ラベル)を返します。このラベルには、必要な APIモジュールとその機能が含まれています。さらに、 API機能への入力に必要なキーワードも抽出されます。たとえば、天気 APIでは「日付」と「場所」が必要です。このラベルに基づいて、システムは最適な APIを呼び出します。

1.3. **API実行(API Execution)**

Harnessing LLMs for API Interactions: A Framework for Classification and Synthetic Data Generation LLMを活用したAPIインタラクションのための分類と合成データ生成のフレームワーク

概要: LLMを使用して自然言語入力を対応する APIコールに分類し、API機能に特化したサンプルデータセットを自動生成するフレームワークの提案

GPT-4の分類精度が 0.996を達成

1. **API分類システム **

API分類システムは、自然言語で入力されたユーザーのクエリを LLM(大規模言語モデル)で解析し、適切な API呼び出しを自動的に行うためのフレームワークです。このシステムは次の 4つの主要なステージで構成されています。

1.1. **ユーザー入力(User Input)**

システムはユーザーから自然言語で入力されたクエリを受け取ります。このクエリは、事前に定義されたプロンプトと組み合わされ、 LLMに送信されます。プロンプトには APIの階層や応答の形式に関するルールが含まれています。これにより、システムが API構造に準拠した結果を返すことができるようになっています。入力されるクエリは、シンプルな質問から複雑な命令まで幅広く対応可能です。

1.2. **API分類(API Classification)**

ユーザーのクエリを受け取ると、統合された LLMがそれを適切な API機能にマッピングします。 LLMはクエリを解析し、API階層に基づいてカテゴリ(ラベル)を返します。このラベルには、必要な APIモジュールとその機能が含まれています。さらに、 API機能への入力に必要なキーワードも抽出されます。たとえば、天気 APIでは「日付」と「場所」が必要です。このラベルに基づいて、システムは最適な APIを呼び出します。

1.3. **API実行(API Execution)**

RUIE: Retrieval-based Unified Information Extraction using Large Language Model RUIE: 大規模言語モデルを用いた検索ベースの統一情報抽出

概要: 名前付きエンティティ認識(NER)、関係抽出(RE)、イベント抽出(EE))を統一した統一情報抽出(UIE)を行うために検索の手法を活用した RUIEを提案

LLMの員コンテキストを使用し、未経験のタスクに適応できます

1. **UIE(統一情報抽出 : Unified Information Extraction)の概要 **

UIEは、異なる種類の情報抽出タスク(例:名前付きエンティティ認識(NER)、関係抽出(RE)、イベント抽出(EE))を統一された 1つのモデルまたはフレームワークで処理する技術です。従来の情報抽出(IE)は、特定のタスクごとに個別のモデルを必要とするため、タスク間の一般化能力が低く、 UIEはこれに対する解決策を提供します。

2. **RUIEの基本的な考え方 **

RUIE(Retrieval-based Unified Information Extraction)は、大規模言語モデル(LLM)の**インコンテキスト学習 **を活用し、未経験のタスクに迅速に適応することを目的とした検索ベースのフレームワークです。従来の UIE手法は指示チューニング(instruction tuning)によって LLMを微調整するものが多いですが、これには大きな計算コストがかかり、未経験のタスクへの一般化が難しいという課題がありました。 RUIEはこれらの課題に対処し、効率的に UIEを実現します。

2.1 **RUIEの特徴**

RUIEは、以下の 2つの主要な技術的特徴を持っています:

- **デモンストレーション選択 **:LLMの嗜好(preference)を取り入れて、検索されたデモンストレーションをランク付けします。

Improving LLM Reasoning with Multi-Agent Tree-of-Thought Validator Agent マルチエージェント・ツリーオブソート・バリデーターエージェントによる　LLM推論の改善

概要: Tree of Thoughts（ToT）でも誤った生成をする場合があり、この問題を解決するために、　ToTベースの Reasonerエージェントと Thought Validatorエージェントを組み合わせた新しいアプローチを提案

複数の Reasonerエージェントが多様な推論経路を探索し、 Thought Validatorがその妥当性を検証します。この方法は誤った推論を排除します

- マルチエージェントシステムにおいて、各エージェントが特定の役割を担うことで推論の質を向上させるというアプローチは新しいものではありませんが、従来の　Reasonerエージェントは浅い推論探索にとどまりがちでした。本論文では、 Tree of Thoughts（ToT）を利用して、Reasonerエージェントが多様な推論経路を深く探索できるようにし、推論の信頼性を高めています。

- さらに、Thought Validatorエージェントが推論経路を評価し、誤った推論を排除することで、信頼性の向上を実現しています。

技術や手法の詳細 :

1. **Reasonerエージェント **:

    - 複数の Reasonerエージェントが並列に動作し、それぞれが　ToTを用いて異なる推論経路を探索します。各推論経路はツリー構造として表現され、次のステップでの推論を生成します。

    - 推論ステップごとに、 State Evaluationエージェントが推論の質を評価し、次のステップへと進む経路を選定します。このプロセスが最終段階まで続き、最も高評価の推論が選ばれます。

2. **Thought Validatorエージェント **:

    - Thought Validatorは、Reasonerエージェントが生成した推論経路を評価し、論理的整合性や事実の正確性をチェックします。推論に誤りがある場合はその経路を無効とし、妥当な推論のみが最終決定に影響を与えるようにします。

3. **コンセンサス投票メカニズム **:

Towards Fair RAG: On the Impact of Fair Ranking in Retrieval-Augmented Generation フェアRAGに向けて：リトリーバル強化生成におけるフェアランキングの影響について」

概要: Fair RAGはユーザーが行った検索や質問に対して一部のコンテンツが過剰に選択されることを防ぎすべての関連アイテムが公平に出現するよう確率的なランキングを使い、関連アイテムを均等に露出の公平性と生成品質を両立しようとします

技術や手法

論文で説明されている主な技術や手法は、 **フェアランキングを RAGシステムに統合 **し、アイテム露出の公平性を高める方法に焦点を当てています。以下、順番に詳細を説明します。

1. フェアランキングと RAGシステム

従来の RAGシステムでは、ユーザーのクエリに対して関連性の高いアイテムを固定的にランク付けし、その中から上位 k個を選択して生成プロセスに渡す手法が一般的です。しかし、この方法では一部のアイテムが過剰に露出され、他の同じくらい関連性のあるアイテムが無視されることがありました。本論文では、この固定的なランキングを **確率的なランキング（ stochastic retriever）**に置き換え、露出の偏りを減らし、関連するすべてのアイテムが均等に表示されるようにします。

2. 確率的ランキング

確率的ランキングでは、アイテムの選択がランダム要素を含むため、同じクエリに対しても異なるアイテムが繰り返し露出されるようになります。具体的には、 **Plackett-Luceサンプリング **を用いてランキングを生成します。これにより、ランキングにランダム性が導入され、露出の公平性を確保します。

**Plackett-Luceサンプリングの式 **:

MAGICORE: Multi-Agent, Iterative, Coarse-to-Fine Refinement for Reasoning MAGICORE: 多エージェント、反復的、粗から細への推論の改良手法

概要: LLMの予測結果は複数の解答を生成しその中で信頼性の高いものを選ぶとよい結果になるが、多く生成しても効果は薄くなっていきます改良はフィードバックを使用し質をあげること、しかし過剰な改良、エラーを局所化し修正できない、改良の不十分さという　3つの課題があります。 この課題に対して問題を「簡単」「難しい」と分類し、難しい問題には詳細かつ反復的な多エージェント改良を適用するフレームワークMAGICOREを提案しています

先行研究と比べてどこがすごいのか　:

MAGICOREは、先行する自己改良や集約手法に比べて以下の点で優れている:

1. **動的リソース割り当て **: 問題の難易度に応じてリソースを効率的に配分し、簡単な問題には最低限の集約、難しい問題には詳細な改良を行う。

2. **外部の報酬モデルの活用 **: モデルが自己改良できる限界を認識し、外部の報酬モデルを使用してエラーの局所化と修正を行う。

3. **多エージェントシステム **: Solver、Reviewer、Refinerの3エージェントが協力し、エラーの修正と反復的改良を行う。

4. **改良の反復性 **: 1回の改良ではなく、複数回の改良を繰り返すことで、精度を高める。

論文で説明している技術や手法　:

MAGICOREは、多ステップ推論における性能向上を目的とした　**自適応型フレームワーク **であり、以下の要素を含む:

Retrieval Augmented Generation (RAG) and Beyond: A Comprehensive Survey on How to Make your LLMs use External Data More Wisely Retrieval Augmented Generation（RAG）とその先：LLMに外部データをより賢く活用させるための包括的な調査

概要：RAGを使用する際の難易度をユーザークエリから　4つに分類、それぞれの課題と最適手法を整理

レベル1：データ内に直接答えが存在するシンプルな質問、レベル　2：複数の情報を組み合わせて答えを導く必要がある質問、レベル　3：専門的な手順やガイドラインを理解して答える必要がある質問、レベル4：データ内に明示されていない専門知識や経験に基づく推論が必要な質問

先行研究と比べてどこがすごいのか　:

この論文の卓越性は、LLMと外部データの統合に関する課題を包括的に分類し、それぞれのレベルに適した解決策を提示している点にあります。従来の研究は特定の技術やタスクに焦点を当てることが多かったのに対し、本論文はユーザークエリを　4つのレベルに分類し、それぞれのレベルでの課題と最適な手法を体系的に整理しています。これにより、開発者が　　LLMを専門分野で効果的に活用するための実用的なガイドラインを提供しています。

技術や手法 :

**レベル1：明示的事実クエリ **

概要：

レベル1のクエリは、外部データ内に直接答えが存在するシンプルな質問です。例えば、「　　2024年のオリンピック開催地はどこですか？」のようなものです。

最適手法：

COMPOSITIONAL HARDNESS OF CODE IN LARGE LANGUAGE MODELS - A PROBABILISTIC PERSPECTIVE 大規模言語モデルにおけるコードの合成的困難性　- 確率的視点から

概要: LLMが同じコンテキスト内で複数のサブタスクを実行するときに回の長さと比例して指数関数的に合成困難性が増えます

この場合、問題を複数のエージェントに分散させると効率的になるということを生成複雑性　N(P,x)を問題xに対する正解 yの逆数1/P(y|x)として定義しました

**技術や手法 **

本研究では、以下の技術と手法を詳細に説明しています。

1. **生成複雑性メトリックの定義 **: 生成複雑性 N(P,x)N(P, x)N(P,x) を、問題 xxx に対する正しい解 yyy の確率の逆数として定義します。 N(P,x)=P(y∣x)1

$$N(P,x)=1P(y∣x)N(P, x) = \frac{1}{P(y|x)}$$

これは、少なくとも 1つの正しい解を得るために必要なモデルの生成回数を示します。

2. **合成問題のモデル化 **: 合成問題を 2つの単純なコード問題の組み合わせとしてモデル化し、その解が各サブ問題の解の連結で表せると仮定します。

3. **スクリーン効果の導入 **: オートレグレッシブな LLMが合成問題を解く際、以前に生成されたトークンが後続のトークン生成にノイズを導入し、正しい解の生成確率を減少させる「スクリーン効果」があると仮定します。

4. **理論的証明 **: 確率論的手法と濃度不等式を用いて、合成問題の生成複雑性が各サブ問題の生成複雑性の積よりも指数関数的に大きくなることを証明します。具体的には、以下の不等式を導出します。N(P,x)≥N(P,x1)N(P,x2)⋅eΔ(L1+L2)/4

Appendix