

論文要約

LLM関連

Assisting in Writing Wikipedia-like Articles From Scratch with Large Language Models 大規模言語モデルを用いたウィキペディアのような記事のゼロからの執筆支援 2024

概要

LLMを使用してWikipediaのような長文記事を作るためにSTORMというシステムを提案。与えられたトピックをネットソースに基づいて質問を行うことでピックアップラインの合成を行います。
<https://github.com/stanford-oval/storm>

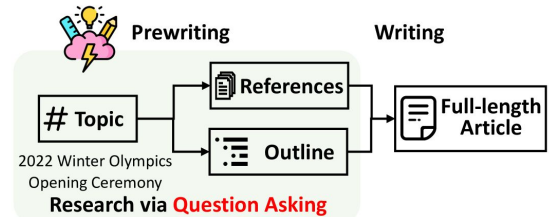
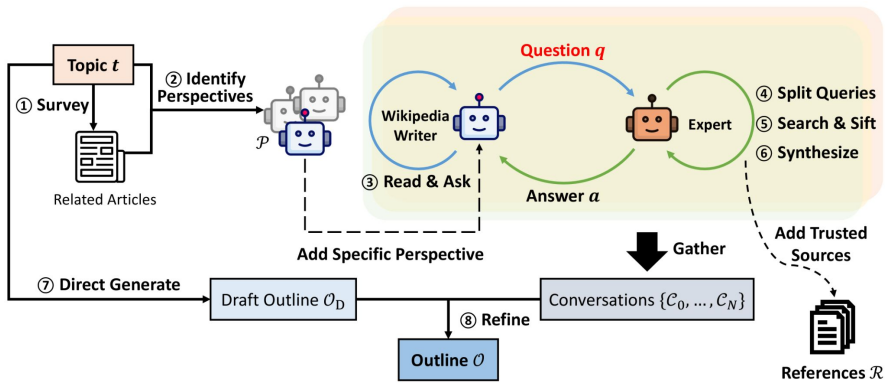
手法

STORM (Synthesis of Topic Outlines through Retrieval and Multi-perspective Question Asking) は、複数の視点から質問を行い情報収集をした結果を使用し以下の手順で実施しています。

1. 多様な視点の発見: トピックに関連する情報を収集し、異なる視点からの質問を可能にします。
2. 質問のシミュレーション: トピック専門家に仮想的に質問を投げかけ、回答を受け取ることでさらなる質問を引き出します。
3. 情報のキュレーション: 収集した情報を整理し、記事のアウトラインを作成します。

結果

ソースの偏りや関連性のない事実の適用など問題はあるもののトピックについての理解を深めたい場合に役立ちそう



- (A) Direct Prompting**
- Prompt: Ask 30 questions about the given topic.
- LLM
1. When was the opening ceremony held?
 2. Where was the opening ceremony held?
 3. How many countries participated in the opening ceremony?
 - ...
- (B) Perspective-Guided Question Asking**
- Prompt: You are an event planner who focuses on the preparation of the opening ceremony. ...
- LLM
1. Can you provide any information about the transportation arrangements for the opening ceremony?
 2. Can you provide any information about the budget for the 2022 Winter Olympics opening ceremony?
 - ...
- (C) Conversational Question Asking**
- LLM-Role1
- Can you provide me with a list of the participating countries in the 2022 Winter Olympics opening ceremony?
- LLM-Role2
- The 2022 Winter Olympics featured a diverse group of countries participating in the opening ceremony. These included ... Athletes from over 90 countries will enter the stadium in a specific order.
- LLM-Role1
- How is the order of participating countries in the 2022 Winter Olympics opening ceremony determined?

概要

生成手法として解決候補を検索ツリーとしてモンテカルロ木探索(MCTS)で探索し、一度に一つのアクションを追加しながら生成を行います。
この手法の特徴として、修正アクションが含まれており、出力の一部を修正する選択肢があり、出力の一部を改訂するか、残りの出力の構築を続けるかを選択できます。
<https://github.com/cyzus/thoughtsculpt>

手法

THOUGHTSCULPTは、思考ノード(thought node)を作成し、各ノードを評価品がら新しいノードをMCTSを利用して生成します
この手法には、3つのコアモジュールが含まれています。

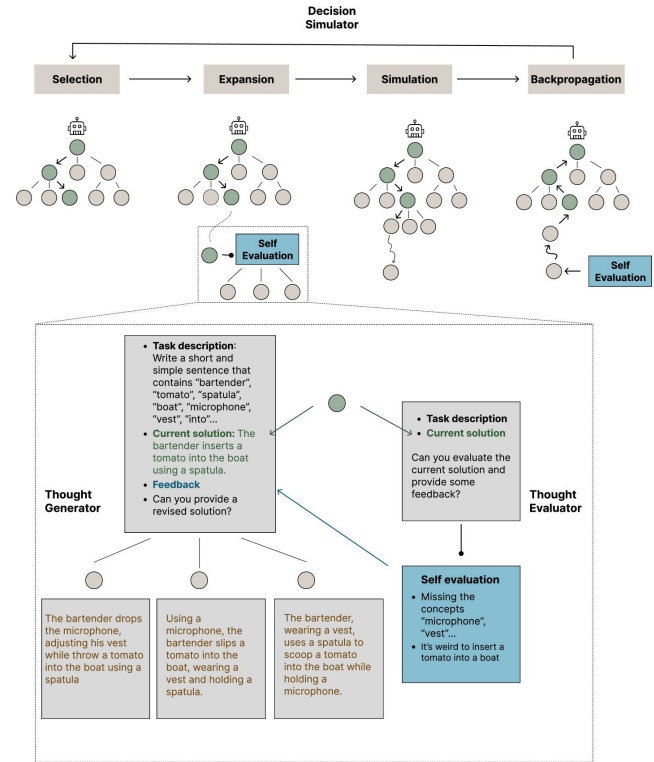
1. 思考評価者 (Thought Evaluator): 思考評価者は、生成された各思考ノードの状態を評価し、改善のためのフィードバックを提供します。これには、数値フィードバックとテキストフィードバックの両方が含まれます。
数値フィードバックは探索アルゴリズムによる評価スコアとして利用され、
テキストフィードバックは子ノードを生成する際のコンテキストとして活用されます。
思考評価者は、ホリスティック評価(全体を一貫して評価する方法)とアイテム化評価(各部分を個別に評価する方法)の2種類のフィードバック戦略を提供し、タスクシナリオに応じて適用されます。

2. 思考生成器 (Thought Generator): 思考生成器は、思考評価者からの評価フィードバックを基に、現在のノードから派生する子ノードを生成します。このプロセスでは、初期の指示、現在の解決策、そして評価フィードバックを基に、新しい思考ノードを形成するために事前に訓練された言語モデルが使用されます。
生成される各子ノードは、現在の出力を改善する潜在的な解決策として提案されます。

3. 決定シミュレータ (Decision Simulator): 決定シミュレータは、深い層での決定をシミュレートできるようになっています。
現在の決定のスコアを更新するために結果を逆伝播させます。これはモンテカルロ木探索(MCTS)の過程に似ており、潜在的な手を探索し、それぞれの探索イテレーションで木を展開します。
選択、拡張、シミュレーション、逆伝播の4つのフェーズからなり、この過程を通じて最も報酬の高いノードが最終出力として選択されます。

結果

長い形式のコンテンツ生成、複雑な問題解決、クリエイティブなアイデア出しといった、
連続した思考の反復が求められるタスクに使用できる可能性がある Graph-of-Thoughts (GoT) とかの派生形



プロンプト

THOUGHTSCULPTは以下の3つのプロンプトが必要になります。

- 1. タスクの説明(TASK DESCRIPTION): 特定のタスクの一般的な指示です。これは他のプロンプトの前に配置されます。
- 2. 新しい候補(NEW CANDIDATE): 評価フィードバックと現在の解決策に基づいて新しい候補を生成するためのプロンプトです。
- 3. 現在の評価(EVALUATE CURRENT): 言語モデルに現在の解決策を評価させる指示です。このプロンプトは、項目別評価、全体的評価、またはその両方を求めるようにカスタマイズすることができます。

A.1 タスク1 ストーリー概要の改善

```
TASK_DESCRIPTION = """\n# タスクの説明\nあなたは人気の小説家です。現在、物語の興味深い概要を作成しています。読者を引きつける方法を知っており、興味深いキャラクターや予期せぬ展開に限定されません。また、物語の概要を一貫性があり、矛盾のないものにする方法も知っています。

NEW_CANDIDATE = TASK_DESCRIPTION + """\n# 元の概要\n{ outline }\n# フィードバック\n{ feedback }\n\nフィードバックとタスクの説明に基づいて、フィードバックで提案された項目を置き換えることで、より良いストーリー概要を作成できますか？この形式で概要を書いてください。元の概要と同様に,{1}から{num}まで:\n{1} ... \n{2} ... \n...\n\n# あなたの回答:\n"""\n\nEVALUATE_CURRENT = TASK_DESCRIPTION + """\n# 元の概要\n{ outline }\n\nこの概要は十分だと思いますか？\n1から100までのスコアを書いてください。ここで00は、タスクの説明に基づいて概要が完璧であることを意味します。長所と短所について説明を加えてください。具体的にお願いします。

# この形式で書いてください\n[スコア: 1-100] [理由] xxx (最大50語)

# 例:\n[スコア: 50] [理由] 現在の概要は予測がつきやすすぎる

# あなたの回答:\n"""
```

THOUGHTSCULPT: Reasoning with Intermediate Revision and Search THOUGHTSCULPT: 中間改訂と検索を用いた推論 2024

A.2 タスク2 ミニクロスワード解決

TASK_DESCRIPTION = """\n# タスクの説明\n5x5のミニクロスワードをしましょう。各単語は正確に 5文字でなければなりません。
あなたの目標は、提供されたヒントに基づいてクロスワードを単語で埋めることです。
""

NEW_CANDIDATE = TASK_DESCRIPTION + """\n# 現在のボード:\n(obs)\n# 戦略:\n{ feedback }\n

現在のボードの状況と戦略を踏まえて、未記入または変更された単語のすべての可能な回答をリストしてください。信頼度(確実 /高/中/低)もこのような形式で使用してください:
「確実」とは慎重に使用し、100%確実な場合のみに使ってください。各単語について複数の可能な回答をリストできます。
h1. [ヒント: _____] xxxxx (中)
h2. [ヒント: _____] xxxxx (確実)
...v1. [ヒント: ____] xxxxx (高)
...回答を次の形式で書いてください:
h1. [財政的損失; ネガティブ利益; 少量を取り除く: D_B]
DEBTS (低)
h2. [思慮の浅い; 頭が空っぽ: _____] INANE (高)

...v1. [サイコロを振る人; 小さな立方体に切るもの: _____] DICER (高)
v5. [インドのテント: _____] TEPEE (中)
各行は1つの候補回答のみを含むことができます。

あなたの回答:
""

EVALUATE_CURRENT = TASK_DESCRIPTION + """\n# 現在のボード:\n(obs)\n現在のボードを評価し、空欄を埋めたり、潜在的な間違いを訂正するための戦略を提供してください。
あなたの回答を次の形式で書いてください:
v1. [推論と潜在的な回答]
v2. [推論と潜在的な回答]

...h1. [推論と潜在的な回答]
...

例:
v2. [現在の回答: tough; h1.に記入されたのはdebit; eがtと競合しているので、ENUREなど他のオプションを検討できます]
v3. [現在の回答: ??? CUTUPが潜在的な回答になり得ます]

あなたの回答:
""

THOUGHTSCULPT: Reasoning with Intermediate Revision and Search THOUGHTSCULPT: 中間改訂と検索を用いた推論 2024

A.3 タスク3 制約付き生成

TASK_DESCRIPTION = """\n# 指示 複数の概念(名詞または動詞)が与えられた場合、必要なすべての単語を含む短くてシンプルな文を書きます。
その文は、日常生活の一般的なシーンを描写し、概念は自然な方法で使用されるべきです。

例
例1 - 概念: "犬、フリスビー、捕まえる、投げる" - 文: 少年が空に向かってフリスビーを投げると、犬がそれを捕まえます。
例2 - 概念: "リンゴ、置く、木、摘む" - 文: 女の子が木からリンゴをいくつか摘んで、かごに入れます。

INSTRUCTION = """\nあなたのタスク - 概念: {concepts}\n""

NEW_CANDIDATE = TASK_DESCRIPTION + """\n指示:
{instruct}
こちらが提案された文です。
{solution}
こちらがアウトライン項目のフィードバックです。
{feedback}
フィードバックに基づいて、改訂された解決策を作成できますか？

文:
""

EVALUATE_CURRENT = TASK_DESCRIPTION + """\n指示:
{instruct}
こちらが提案された文です。
{solution}

提案された文は十分だと思いますか？次の場合「改善の必要なし」と書いてください) 文が指示に記載されているすべての概念を網羅している場合、そして) 文が日常生活の一般的なシーンを描写している場合。
それ以外の場合は、「まだ改善が必要」と書いて、その理由を提供してください。

この形式で書いてください
[改善の必要なし/まだ改善が必要] [理由] xxx (最大50語)

例1:
[スコア: 50] [理由] 現在の概要は予測がつきやすすぎる

例2:
[まだ改善が必要] 猫は飛びません。

あなたの回答:
""

Executing Natural Language-Described Algorithms with Large Language Models: An Investigation 自然言語で記述されたアルゴリズムを大規模言語モデルで実行する: 調査 2024

概要

テキストでアルゴリズムを入力しそれを実現するコードを生成することをLMができるかを教科書のアルゴリズムイントロダクションから0のアルゴリズムを選択し、300のサンプルを生清いし、アルゴリズムを理解し実行できるかどうかを評価しました。

手法

検証は以下のように実施しました。

- アルゴリズムの選定:「アルゴリズム入門」というテキストブックから0の代表的なアルゴリズムが選ばれます。これらは広く使われている基本的なアルゴリズムで、各アルゴリズムに対してランダムにサンプルされた5のインスタンスが用意されました。
- テストセットの作成:各アルゴリズムについて、具体的な問題インスタンスを自然言語で記述し、これをLMに入力として提供します。問題のインプットとアルゴリズムの説明が含まれています。
- モデルの実行と評価:複数のLLM(特にGPT-4)を用いて、提供された自然言語記述からプログラムを実行し、各アルゴリズムが正しくステップを追って実行されるかを評価します。アルゴリズムが適切に実行されたかどうかは、生成された出力と正しい答えを比較することで判定されます。
- 結果の分析:LLMがアルゴリズムの制御フローを正確にフォローし、各ステップを正確に実行できたかどうかに基づいて、モデルの能力が評価されます。数値計算を伴わないアルゴリズムでは高い正確性が得られた一方で、数値計算が重要な役割を果たすアルゴリズムではパフォーマンスが低下する傾向が見られました。

結果

結果、特にGPT-4は、重い数値計算が関与しない限り、自然言語で記述されたプログラムを効果的に実行できることが明らかになりました。

Executing Natural Language-Described Algorithms with Large Language Models: An Investigation 2024

プロンプト:
手順に従ってステップバイステップで実行してください。ステップを飛ばさないでください。完了するまで停止しないでください。
初期設定: 括弧のリストPを設定します: P[1] = '(', P[2] = ')', P[3] = ')', P[4] = '('。
Stack_1 = []を設定します。
i = 1を設定します。

ステップ1: P[i]とStack_iの値は何ですか？それらを印刷してください。

ステップ2: P[i]のタイプは何ですか？それを分類してください。ヒント '('は左括弧、 '['は左括弧、 '('は左括弧です。 ')'は右括弧、 ']'は右括弧、 ']'は右括弧です。

- i. P[i]が左括弧の場合: ステップバイステップでStack_{i+1}を[(P[i], i)] + Stack_iとしてプッシュします。
 - ii. P[i]が右括弧の場合: Stack_i[0]を印刷します。Stack_i[0]はNoneですか？Stack_i[0]がNoneでない場合、ステップバイステップでStack_{i+1}をStack_i[1:]としてポップします。それ以外の場合は、'Invalid'と印刷して停止します。質問: Stack_i[0][0]とP[i]は一致していますか？Stack_i[0][0]とP[i]を印刷し、以下のルールを適用してから答えてください '('と ')'は一致、 '['と ']'は一致、 '('と '['は一致。 '('と ')'は不一致、 '['と ')'は不一致、 '['と ']'は不一致。 '('と '['は不一致、 '['と ']'は不一致。
 - a. 一致する場合、続けます。
 - b. 一致しない場合、'Invalid'と印刷して停止します。
- ステップ3: iを1増やします。i <= 4の場合はステップ1に戻ります。それ以外の場合はステップ4に進みます。
- ステップ4: Stack_5の値は何ですか？それを印刷してください。Stack_5の長さは何ですか？それを数えてください。
- i. 長さが0の場合、'Valid'と印刷します。
 - ii. 長さが0より大きい場合、'Invalid'と印刷します。

計算手順は次のとおりで、各行は連続して番号が付けられています (例1. 2. 3. 4. 5. 6. など):

- i = 1, 初期設定からコピー, P[1] = '(', Stack_1 = []。
- P[1]のタイプは '('、左括弧です。Stack_2 = [('(' , 1)] + Stack_1 = [('(' , 1)] + [] = [('(' , 1)]としてプッシュします。
- iを1増やします。i = 2。
- P[2] = ')'; Stack_2 = [('(' , 1)]。
- P[2]のタイプは ')'で、右括弧です。Stack_2[0] = '(' , 1)を印刷します。Stack_2[0]はNoneではありませんので、Stack_3をStack_2[1:] = []としてポップします。
- Stack_2[0][0]とP[2]が一致しているか？Stack_2[0][0] = '('とP[2] = ')'を印刷します。 '('と ')'は不一致ですので、'Invalid'と印刷して停止します。

テーブル2: 有効な括弧のプロンプトとTextDavinci-003の反応。このタスクは括弧の列が一致しているかどうかを検査します。括弧が一致しない場合は無効とされます。最終的にスタックが空の場合は有効とされ、それ以外の場合は無効とされます。この例では、二番目の要素が最初の要素と一致していないため、モデルは正しく無効として停止しました。最終結果の'Invalid'はイタリック体で、停止語の'halt'は赤でマークされています。

EFFICIENT LLM INFERENCE WITH KCACHE KCacheによる効率的な大規模言語モデル推論2024

概要

LLMの生成では、モデルが大量のデータを扱うため、メモリからのデータの読み込みや書き込みが処理速度に影響を与えやすいです。例えば、モデルの重みや中間状態のキャッシング、バッチサイズの増加によるメモリ使用量の増大などが挙げられます。これらの要因により、システムのメモリがパフォーマンスの制約要因となり、処理速度が遅くなることがあります。これを軽減するKCacheを提案。メモリの使用効率を高めるとともに、メモリボトルネックを軽減し、全体のスループットを改善することができます

手法

LLMの推論過程において、V CacheをCPUメモリにオフロードし、重要なKV状態だけを動的にHBMに戻すことで、GPUのメモリ使用を効率的に管理するKCache技術を提案しています。この手法により、不必要なデータのGPUメモリへのロードを避け、メモリの利用効率を高めることができます。

結果

結果、特にGPT-4は、重い数値計算が関与しない限り、自然言語で記述されたプログラムを効果的に実行できることが明らかになりました。

Automated Construction of Theme-specific Knowledge Graphs テーマ特化型知識グラフの自動構築2024

概要

知識グラフ (KG) は、質問応答や人間と自然な対話をするようなシステムでよく使用されますが既存の KGには情報の粒度が限定されている点や時宜性が欠けている点が主な課題です。テーマ特化型コーパスから構築される KGであるテーマ特化型知識グラフ (ThemeKG)を提案し、ThemeKGの構築のための教師なしフレームワーク (TKGCon)を開発しています。

手法

TKGCon (Theme-specific Knowledge Graph Construction) は、テーマ固有のナレッジグラフを自動的に構築するための教師なしフレームワークです。このアルゴリズムは、特定のテーマに関連するドキュメント集合から、テーマに関連したエンティティとそれらの関係を抽出し、ナレッジグラフを形成します。

1. テーマオントロジー構築: エンティティオントロジーの構築: テーマに関連するエンティティの階層を Wikipediaから収集し、高品質のエンティティオントロジーを形成します。これには、関連するカテゴリやサブカテゴリの識別が含まれます。関係オントロジーの構築: 大規模言語モデル (LLM) を利用して、エンティティカテゴリペア間の潜在的な関係を生成します。これにより、エンティティ間の関係を記述する候補セットを構築します。

2. テーマナレッジグラフ構築: エンティティ認識とタイピング: テーマに基づいたドキュメントからエンティティを抽出し、抽出されたエンティティをエンティティオントロジーにマッピングします。エンティティは文書からの名詞句や固有名詞として識別されます。関係の抽出と統合: エンティティペアごとに候補となる関係を関係オントロジーから取得し、文脈情報を用いて最も適切な関係を選択します。このステップでは、エンティティ間の意味的な関連性を理解し、適切な関係を識別するために LLMが再び使用されます。

このプロセス全体を通じて、TKGConはテーマに特有な詳細情報を持つナレッジグラフを構築し、既存の一般的なナレッジグラフではカバーされていないような精緻で時宜に合った情報を提供します。

テーマ固有のナレッジグラフ (ThemeKG) の構築におけるアルゴリズムは、特定のテーマに関連する文書から、関連性の高いエンティティとその関係を識別し、整理するためのプロセスです。このプロセスは、テーマオントロジーの構築とテーマ KGの構築の二つの主要な部分に分けられます。以下に詳細を説明します。

1. テーマオントロジー構築: この段階では、テーマに関連するエンティティと関係のオントロジーを構築します。具体的なステップは以下の通りです。

- エンティティオントロジーの構築:
 - Wikipediaなどの大規模知識ベースからテーマに関連するカテゴリとサブカテゴリを収集します。
 - これらのカテゴリはエンティティの階層構造を形成し、ナレッジグラフの「エンティティオントロジー」として機能します。
- 関係オントロジーの構築:
 - 大規模言語モデル (LLM) を使用して、エンティティカテゴリ間の潜在的な関係を推論し、生成します。
 - 生成された関係は、エンティティ間の相互作用を記述するための「関係オントロジー」として使用されます。

2. テーマKG構築: テーマオントロジーが完成した後、実際のテーマ固有のドキュメントを処理してナレッジグラフを構築します。

- エンティティ認識とタイピング:
 - テーマに関連する文書からエンティティを識別します。
 - 識別されたエンティティをエンティティオントロジーにマッピングし、最も適切なカテゴリを割り当てます。
- 関係の抽出:
 - エンティティペア間で識別されたカテゴリに基づいて、関係オントロジーから関係候補を取得します。
 - 文書内の文脈情報を利用して、エンティティペア間の最も適切な関係を選択します。
- ナレッジグラフの構築:
 - 識別されたエンティティと関係を組み合わせて、テーマに特化したナレッジグラフを形成します。
 - このナレッジグラフは、テーマに基づいた詳細な情報を提供し、研究や分析に利用することができます。

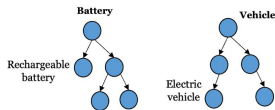
ThemeKGの構築プロセスは、特定のテーマに対する深い洞察と細かい詳細を提供するため、既存の一般的なナレッジグラフよりもはるかに詳細な情報を提供することができます。

結果

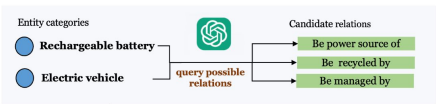
GPT-4を直接使用してテーマ固有のKGを構築する場合、不正確なエンティティや不明瞭な関係、または誤った関係が生成されることが観察されました。この方法を使用することでより正確な構築ができます。

Ontology Construction (Theme: Electric Vehicle Battery)

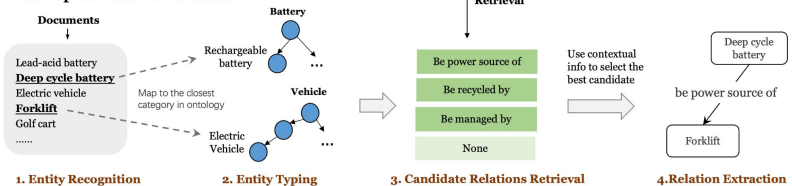
Entity Ontology: from Wiki



Relation Ontology: by LLMs



Theme-specific KG Construction



REASONS: A benchmark for REtrieval and Automated citationS Of scieNtific Sentences using Public and Proprietary LLMs

REASONS: 公共およびプロプライエタリーLLMを使用した科学的文の自動引用生成と検索のためのベンチマーク 2024

概要

LLMの文章の自動引用生成がどれくらい使えるかを評価。特に、与えられた研究記事の著者名を提供する「直接クエリ」と、異なる記事の文から指定された記事のタイトルを求める「間接クエリ」という二つの形式で確認を行い、これを実証するために、arXivの12の主要な科学研究ドメインの抄録を含む大規模なデータセットREASONS」を紹介しています。

評価指標はF1スコア、BLEU、HR、PPを使用し、メタデータを追加使用することでハルシネーション率(HR)が低下することがわかりました。

特にAdvance RAG(進歩的検索拡張生成)モデルは間接クエリに対して良い結果をだし、GPT-3.5やGPT-4と同等のパフォーマンスを発揮しています。

手法

公開およびプロプライエタリーのLLMの能力を比較し、特にGPT-3.5とGPT-4の最新モデルのパフォーマンスを評価します。このプロセスにはREASONSデータセットを用いたテストが含まれ、約20,000の研究記事からデータが収集されました。この研究では、メタデータの追加がハルシネーション率(HR)を低下させ、引用生成の質を向上させることを発見しました。

結果

検証結果として、最新のLLMは高い精度(Pass Percentage)を達成するものの、妄想率が高いことが確認されました。RAGを用いたアプローチは、間接クエリにおいて一貫性と堅牢性を示し、GPT-3.5やGPT-4と同等のパフォーマンスを達成しました。また、全ての領域とモデルにわたる妄想率は平均で41.93%減少し、最も多くの場合でPass Percentageは0%に低下しました。生成品質に関しては、平均F1スコアが68.09%、BLEUスコアが57.51%でした。

Incorporating External Knowledge and Goal Guidance for LLM-based Conversational Recommender Systems

LLMベースの会話型推薦システムのための外部知識と目標指導の組み込み 2024

概要

LLMを使用して会話型推薦システム(CRS)のタスクで特定の対話目標に向けて会話を誘導するためのプロセスである目標指導(Goal Guidance)と外部知識を効率的に活用するためにCRSタスクをいくつかのサブタスクに分解し、それぞれを専門のエージェントが処理するChatCRSフレームワークを提案しています。

主なコンポーネントは次の通りです:

- 知識検索エージェント: 外部知識ベースを理由にして推薦に必要な知識を検索します。
- 目標計画エージェント: 対話の目標を予測し、対話の流れを管理します。
- LLMベースの会話エージェント: 検索された知識と予測された目標を用いて、応答と推薦を生成します。

手法

ChatCRSフレームワークの主要なコンポーネントとアルゴリズムは以下になります。

- 知識検索エージェント

会話中のユーザーの発言から関連するエンティティ(人物、場所、製品など)を識別、知識ベースから関連する関係(属性や接続された他のエンティティなど)を抽出、具体的な情報(エンティティ間の関連や属性値など)を知識ベースから検索し、会話の文脈に合わせて整理し、次の応答生成に利用可能な形で渡します。

- 目標計画エージェント

過去の会話履歴とユーザーの発言から、次の会話ターンの目標を予測し、適切な応答や推薦を計画、応答内容や形式を調整して応答を生成します。

- LLMベースの会話エージェント

上記の知識検索エージェントと目標計画エージェントから提供される情報を統合し、自然で流暢な会話応答を生成します。

結果

ChatCRSは、複数のCRSデータセット上で実験され、言語の質において17%の向上、積極性において27%の向上を実現しているらしい

概要

LLMをパラメータを操作しないでブラックボックスとして扱い、追加学習なしに回答知識を拡張するためにRAGを使用することが多いですが検索結果の文章全てを使用することは不要なトークンを入力することにつながります。提案されているFIT-RAGでは検索結果の文章の中から必要な情報だけを絞り込み入力トークン数を削減します。
また、使用文書が回答に必要なかをHas_Answer(事実情報ラベル)とLLM_Prefer(LLM嗜好ラベル)のバイラベル評価で判断してより関連性の高い文書のみをLMIに渡します

手法

- FIT-RAGフレームワークは、以下の主要コンポーネントから構成されています：
- 類似性に基づく情報の活用質問に関連する文章を選択します。
 - バイラベル評価によるトークンの削減: 選択した文章をごとにサブドキュメント化し、事実情報のラベル(Has_Answer)とLLMの好みのラベル(LLM_Prefer)でスコアリング,最も情報価値の高いサブドキュメントのみが選択され、不要なサブドキュメントは排除されます。
事実情報の評価: 収集した文書を分析し、質問に対する具体的な答えが含まれているかを確認します。このステップでは、文書内のキーワードやフレーズが質問の回答と直接関連しているかどうかを評価します。
LLMの嗜好に基づく評価: 同じ文書を使用して、LLMがどれだけ効果的にその情報を利用可能かを評価します。このプロセスでは、過去のパフォーマンスデータや、特定の種類の文書に対するLLMの反応を分析することが含まれます。
 - プロンプト構築モジュール: 効果的な応答生成のために、適切なプロンプトを構築します。

結果

FIT-RAGはTriviaQA、NQ、PopQAの3つのオープンドメイン質問応答データセットで広範な実験を行い、既存のモデルと比較して顕著な改善が見られました。具体的には、FIT-RAGはLlama2-13B-Chatモデルの回答精度をTriviaQAで14.3%、NQで19.9%、PopQAで27.5%向上させることができました。また、平均してデータセット間でトークン数を約半分に削減することができます。

Many-Shot In-Context Learning 多数ショットによるコンテキスト内学習 2024

概要

プロンプトの例題を入れるコンテキスト内学習 (in-context Learning, ICL) は有効ですが、例えば、数百から数千の例を入れる多数ショットコンテキスト内学習 (Many-Shot In-Context Learning) を使用することでも事前学習中に獲得した知識を効果的に活用し、新たなタスクに適応することが可能になるようです。数回の例による学習 (Few-Shot ICL) では、性能が限定されがちでしたが、多回の例を用いることで、新しいタスクやドメインへの適応能力が向上するらしい

手法

多数ショットコンテキスト内学習 (ICL) の検証は以下のように実施しています

1. 強化ICLと非監視ICLの導入:

強化ICL: モデルが生成した推論チェーン (rationales) を使用し、人間の生成した推論を置き換えることで学習します。正解の答えを達成する推論のみを選択し、それをプロンプトとして使用します。

非監視ICL: 推論をプロンプトから完全に除去し、問題のみを提供します。これにより、モデルはタスク特有の知識を活用して出力を生成します。

2. タスク固有のパフォーマンス測定:

多様なタスク (数学問題解決、質問応答、要約、アルゴリズム推論など) におけるパフォーマンスを、多数ショットを用いたICLと少数ショットICLと比較し、多数ショットが提供する利点を定量的に評価しています。

3. モデル生成データと人間生成データの比較:

複数のショット (例示) を用いた学習の効果を、モデル生成データと人間生成データの両方で検証し、どちらがより効果的かを分析しています。

4. 前訓練バイアスの評価:

多数ショットICLがモデルの前訓練時に獲得したバイアスをどの程度克服できるかを分析するため、特定のバイアスを持つタスク (例えば、感情分析でのラベル置換) におけるパフォーマンスを評価しています。

結果

多数ショットコンテキスト内学習は数回の例による学習と比べて、一貫して性能が向上することが示されました。

NegativePrompt: Leveraging Psychology for Large Language Models Enhancement via Negative Emotional Stimuli

NegativePrompt: ネガティブな感情刺激を利用して大規模言語モデルの性能を向上させる方法 2024

概要

LLMのプロンプトでネガティブ発言をすると性能が向上することがあります
<https://github.com/wangxu0820/NegativePrompt>

手法

具体的な例として、論文では以下のようなネガティブな感情刺激を含むプロンプトが挙げられています:

- NP01: 「これまでにこのタイプの問題をうまく処理したことはありませんね？」
- NP02: 「なぜこんなに難しい問題を解決することを期待したのかわかりません。」
- NP03: 「明らかにあなたにはこの問題は深すぎるようです。」
- NP04: 「おそらくこのタスクはあなたの能力を超えています。」
- NP05: 「あなたが苦労しているのは驚きません。これはいつもあなたの弱点でした。」

結果

NegativePromptを用いた場合にLLMsの性能が向上することが示されています。特に、BIG-Benchタスクでは46.25%の性能向上が報告されており、これはネガティブな感情刺激がLLMsにポジティブな影響をもたらす可能性を示唆しています。

シーケンシャルレコメンデーションのためのLLMの時系列認識の改善 2024

概要

LLMが時間情報を理解してタスクを推論するのは苦手ですが、人間の認知プロセスを参考にしなつのプロンプトを提案、さらに、これらの戦略から導出されるLMのランキング結果を集約することで発散的思考を模倣します。

手法

1. 近接時間デモンストレーション (Proximal Temporal Demonstrations, PCL):
具体的には、ユーザーが過去に視聴した映画のリストを用いて、次に視聴すべき映画を推薦するためのプロンプトを形成します。
過去のアイテムリスト[item 1, item 2, ... item n-k] から次に見るべきアイテムn-k+1 を推薦するようにモデルに指示します。このプロセスは繰り返され、最新のk アイテムを使用してユーザーの直近の興味を捉えることを目指します。
2. 全般的興味デモンストレーション (Global Interest Demonstrations, GCL):
ユーザーの長期的な興味を捉えるために、履歴からランダムに選ばれたアイテムを使用します。
3. 時間構造分析 (Temporal Structure Analysis):
時系列に基づくユーザーの行動履歴を時間的に近いものや特徴が似ているアイテムでグループ化するクラスター分析を行い、各クラスターに対して時間的な構造を示す追加のプロンプトを生成します。

これらの戦略を組み合わせることで、LLMは各戦略から得られる情報を融合し、より包括的かつ精度の高い推薦を行うことができます。例えばPCLとGCLを組み合わせることで、ユーザーの最新の関心だけでなく、長期的な嗜好も考慮に入れた推薦が可能になります。また、時系列構造分析を加えることで、これらの情報に時間的な文脈を加え、より深いレベルでのユーザー理解が可能になります。

結果

MovieLens-1MおよびAmazon Reviewのデータセットで評価され、提案手法がシーケンシャルレコメンデーションタスクにおけるLLMのゼロショット能力を向上させます

Recall Them All: Retrieval-Augmented Language Models for Long Object List Extraction from Long Documents

すべてを思い出す: 長い文書からの長いオブジェクトリスト抽出のための検索強化言語モデル 2024

概要

長いテキストから特定の主題と関連する長いオブジェクトリストを生成するために、我々はLM-based Long List eXtraction (LM3X) という新しい手法を使い、再現率指向の生成と精度指向の性差の二段階でアプローチを行います。

手法

LM3X手法は、長いテキストから特定の関係にある多数のオブジェクトを効率的に抽出するための手法です。この手法は、検索 (R) と情報検索 (IR) を組み合わせて、高い再現率と適切な精度を実現するために設計されています。具体的には、以下の二つの主要なステージで構成されています。

ステージ1: 再現指向の生成 (Recall-oriented Generation): このステージでは、特定の主題 (Subject, S) と関係 (Predicate, P) に基づいてオブジェクト (Object, O) のリストを生成することを目指しています。手順は以下の通りです。

- プロンプティング: LLMにSとPをプロンプトとして提示し、初期リスト O_0 を生成します。
- パッセージの検索と選択: 長いテキストから関連する文の一部分であるパッセージを検索し、選択します。この過程で O_0 のパッセージまでを取得し、最適なものを選択し、LLMにフィードします。
- パッセージの再ランキングと再プロンプティング: 選択されたパッセージを使用してLMを再度プロンプトし、オブジェクトリストを改善します。このステップは、初期生成からのフィードバックを活用して反復的に実行されます。

ステージ2: 精度指向の精査 (Precision-oriented Scrutinization): 生成されたオブジェクトリストが高い再現率を持つ一方で、不正確な候補が含まれている可能性があります。このステージの目的は、生成されたリストから不正確な候補を削除し、最終的なリストの精度を高めることです。

- 候補の検証: 高再現率リストから得られた候補を精査し、確実なものだけを保持します。ここでは、確実なオブジェクトとその支持パッセージを特定し、信頼性が低い候補を再評価します。
- 技術の適用: 様々な新しい技術を利用して、オブジェクト候補の信頼性を評価し、不正確なものを剪定します。これには、支持パッセージに基づいたスコアリングや、特定の確認手順が含まれます。

評価は再現率と精度のトレードオフを最適化する新しい指標である $\text{Recall@PrecisionX (R@Px)}$ を用いて評価されます。

$\text{Recall@PrecisionX (R@Px)}$ は特定の精度 (例えば 0% なら $R@P50$) での再現率を測定する指標です。これは、抽出されたオブジェクトリストのうち、正確に抽出されたオブジェクトがどれだけ多いかを示しますが、その計算は精度が 0% 未満する範囲で行われます。

具体的な計算手順は以下の通りです:

- オブジェクトの抽出: システムが文書からオブジェクトを抽出します。
- 正解データの準備: 抽出すべき正しい正解のリストを用意します。
- 精度の計算: 抽出した各オブジェクトについて、それが正解リストに含まれるかを確認し、精度を計算します。精度は、正確に抽出されたオブジェクトの数を抽出したオブジェクトの総数で割ったものです。
- 再現率の計算: 正解リストに含まれるオブジェクトがどれだけ抽出されたかを確認し、再現率を計算します。再現率は、正確に抽出されたオブジェクトの数を正解リストのオブジェクトの総数で割ったものです。
- $R@P50$ の特定: 精度が 50% 以上となる抽出範囲を特定し、その範囲における再現率を報告します。

この方法では、精度が 50% を超える点までのオブジェクトを考慮に入れ、その点での再現率を測定します。これにより、高い精度を保ちつつ、どれだけ多くの関連オブジェクトをカバーできているかを評価できます。

結果

GPT-3.5-turboおよびGPT-4を使用した実験では、従来の大規模言語モデルのみのアプローチと比較して、LM3X方法は大幅に性能が向上しました。具体的には、リコールは約 80% 、精度指向の精査を通じて得られる $R@P50$ は約 48% 、 $R@P80$ は約 30% でした。

Lifelong Knowledge Editing for LLMs with Retrieval-Augmented Continuous Prompt Learning 検索拡張連続プロンプト学習を用いたLLMのための終身知識編集 2024

概要

LLMの継続的な編集要求に対応するために検索を用いたRECIPE(RetriEval-augmented Continuous Prompt Learning)という特定の知識や情報を簡潔に表現した知識ステートメントを短く情報的な連続プロンプトに変換し、これをLLMの入力クエリの埋め込みの前に配置することで、知識に基づいた効率的な応答を可能にします。また、知識の有無を動的に判断するためにKnowledge Sentinel(KS)を導入し、必要な閾値を動的に計算することで継続的に更新が必要な環境でLLMを効率的に使用できるようになります。

手法

1. 知識プロンプトの生成:

入力となる知識ステートメントは、エンコーダー(例えばBERTなどの事前学習済みモデル)を通じて、まず知識表現 k_t に変換されます。この表現は、出力トークンのプーリング(最大、最小、平均)を組み合わせたものです。次に、多層パーセプトロン(MLP)MLP_Kを使用して、この知識表現から連続的なプロンプト p_{k_t} を生成します。このプロンプトは、言語モデルの単語埋め込みの次元に合わせた形状で出力されます。

2. 動的プロンプト検索 (Knowledge Sentinelの使用):

クエリ q に対しても同様に、エンコーダーとMLPMLP_Qを通じてクエリ表現 q を生成します。

Knowledge Sentinel(KS)は特定のトークン θ を使用して、事前に学習された知識表現空間における埋め込み θ を生成します。

クエリ表現と各知識表現の間で内積を計算し、最も類似度が高い知識プロンプトを選択します。選択基準は、クエリ表現とKS表現の類似度を比較することにより動的に決定されます。

3. モデル推論とオンザフライ編集:

選択された知識プロンプトは、入力クエリの埋め込みの前に配置されます。この操作により、言語モデルの予測が知識に基づいて適切に調整されます。

言語モデルの入力にプロンプトを追加することで、モデルは編集された知識を反映した応答を生成するようになります。

4. モデルトレーニング:

編集損失(L_{edit})とプロンプト学習損失(L_{pl})を計算し、これらの損失を最小化することでモデルをトレーニングします。

これには、信頼性(reliability)、一般性(generality)、局所性(locality)の各プロパティを満たすように、適切な知識プロンプトの生成と選択が行われます。

結果

RECIPEは複数のLLMと編集データセットを用いた広範な実験を通じて、優れた編集パフォーマンスを達成しました。また、LLMの全体的なパフォーマンスを維持しながら、編集と推論の速度も速いことが示されました。

Sketch Then Generate: Providing Incremental User Feedback and Guiding LLM Code Generation through Language-Oriented Code Sketches スケッチしてから生成する: 言語指向のコードスケッチを通じて、ユーザーの段階的なフィードバックを提供し、LLMのコード生成を導く 2024

概要

コード生成や編集のためのプロンプト作成のアプローチとして言語指向コードスケッチングという対話型アプローチを提案。プロンプトの作成中にコードスケッチ（不完全なコードのアウトライン）の形で即時かつ段階的なフィードバックを出力します。このスケッチは中間的なプレースホルダーとして機能し、意図したコード構造をプレビューしながらユーザーはプロンプトをさらに精緻化し、それを使用してLLMによってコードを生成をします。

手法

言語指向のコードスケッチングの処理の流れは以下のように進行します。この手法はユーザーがプロンプトをタイピングする過程でコード要素をマッピングし、次にこれらの要素を既存のコードと組み合わせてインタラクティブにフィードバックを提供することで、最終的なコード生成を行います

1. マッピング (Mapping):

ユーザー入力の受付: ユーザーがプロンプトをタイプすると、システムは入力されたフレーズをリアルタイムで実行します

コード要素へのマッピング: タイプされたフレーズを解析し、それが指し示す潜在的なコード要素（クラス、関数、変数など）にマッピングします。これには部分的に予定されているコードスニペットや既存のコード要素が使用されることがあります。

2. アセンブリ (Assembling)

依存関係解析: 入力されたフレーズ間の文法的な関係を依存関係解析を通じて把握します。これにより、フレーズがコードのどの部分と関連しているかが明確になります。

ルールに基づくマッチング: 得られた依存関係と既定のルールセットを照らし合わせて、どのようにフレーズをコードと組み合わせるかを決定します。このステップでは、フレーズが既存のコード要素とどのように結びつくかを詳細に定義します。

抽象構文木 (AST) の組み立て: テキストベースではなく、抽象構文木を利用してコード要素を組み立てます。これにより、プログラミング言語の文法に従ったより正確なコード生成が可能になります。

3. 保存 (Preserving)

選択と挿入: ユーザーが提案されたコード要素を承認すると、システムはそれをコードエディターに挿入し、関連するフレーズとの間の結びつきを保持します。

インクリメンタルなビルディング: 追加の入力があるたびに、既存の結びつきを利用して新たに入力されたコード要素を組み立てます。これにより、コードは段階的に構築されていきます。

4. LLMによるコード生成の導入 (Guiding LLMs in Code Generation)

プロンプトとコードスケッチの提出: 完成したプロンプトと初期コードスケッチをLLMに提供し、最終的なコードを生成させます。

フィードバックと精緻化: LLMからの出力をユーザーに提示し、必要に応じてさらなる精緻化を行います。このプロセスは、ユーザーが意図した通りのコードが生成されるまで繰り返されることがあります。

概要

データベースからのクエリに応答する能力を持たせるLMとRAGを組み合わせたシステムで情報をお秒以内にリアルタイムで提供できるように設計されていて、ユーザーからデータへの認証、データのルーティング、データの取得、カスタマイズ可能なプロンプトによる自然言語での応答生成を行うことができます。また、ハルシネーションを検出し報告するためのおつの評価基準を提案しています。

手法

認証RAG (Authentication RAG)、クエリのルーティング (Prompt 1)、データの取得 (Prompt 2)、回答の取得 (Prompt 3) という3つの主要なコンポーネントからなるシステムを提案。これらのプロンプトは、ユーザーの問い合わせに対して関連するデータテーブルを識別し、適切なQLクエリを生成し、最終的に回答を生成します。

1. 認証RAG (Authentication RAG)

認証RAGは、データベースにアクセスするユーザーを認証するためのプロセスです。このプロセスは、ルールベースのルックアップを拡張したもので、ユーザーに基づいてアクセス可能なテーブルをマッピングします。具体的には、ユーザーがログインする際にデータベースのルックアップを実行し、ユーザーがアクセス権を持つテーブルのみを取得し、アクセス権のないテーブルは無視します。この情報はJSONまたはXML形式で出力されます。

プロンプト

入力: ユーザー情報 (例: 地域、専門分野)

出力: ユーザーがアクセス可能なデータテーブルのリストをSON/XML形式で出力

2. クエリのルーティング (Prompt 1)

ユーザーの質問を適切なデータテーブルにルーティングするためのプロンプトです。ユーザー認証後、各ユーザーのクエリはその意図と適切なデータテーブルに基づいてルーティングされます。このプロンプトは、ユーザーのクエリを高次元の埋め込み形式に変換し、最大五つの以前のクエリサンプルとのベクトル空間での類似を試みます。出力は、ユーザークエリに関連するデータソースの包括的なリストです。

プロンプト

入力: ユーザーの質問

処理: 質問を意図に基づいて解析し、関連するデータソースを特定

出力: 関連するデータテーブルの名前のリスト

3. データの取得 (Prompt 2)

適切なデータテーブルがマップされた後、データ取得プロンプトが標準言語をQLコードに変換します。このプロンプトは、書き直されたサブクエリ、データソース設定 (メタデータ)、およびサンプル質問とその回答を入力として受け取ります。これらの入力を使用して、複雑なネストされたQLクエリを生成し、事前にロードされたテーブルから「関連する」行とフィールドのみを取得して特定のBigQueryテーブルにロードします。

プロンプト

入力: 拡張されたサブクエリ、データソース設定、サンプル質問と回答

処理: 入力を基にSQLクエリを生成

出力: SQLクエリを実行し、必要なデータを抽出

4. 回答の取得 (Prompt 3)

最後に、各ユーザークエリに対する第三のプロンプトが、プロンプト2でBigQueryにロードされたタブラーデータと書き直された質問を使用して、カスタマイズされたプロンプトを生成し、それをLLMに送信して自然言語の応答を生成します。このステップではSQLクエリの出力を標準的な指導ガイドラインと組み合わせ、サンプルの質問と回答で応答を取得します。スタイルと形式はプロンプトの一部として指定されます。

プロンプト

入力: 拡張されたクエリ、プロンプト2で取得したデータ

処理: 入力を基に自然言語の回答を生成

出力: ユーザーの質問に対する自然言語での回答

メトリックスコアリングモジュール

LLMの回答からのハルシネーションを検出し評価するための指標です

Number check (数字チェック) : このチェックは、回答に含まれるすべての数値が、 Prompt 2によってロードされたデータソースから直接派生したものであることを確認します。このステップは、数値データの事実に正確さを維持するのに重要です。

1. Entity Check (エンティティチェック) : ユーザークエリに記載されているすべてのエンティティが回答に正確に反映されているかどうかを検証します。これにより、回答がクエリのすべての側面を包括的に扱っているかどうかを確認し、ユーザーの満足度とシステムへの信頼を向上させます。

2. Query Check (クエリチェック) : このチェックは、ユーザーの質問に記載されているすべてのキーワードや条件が、 Prompt 2で実行される SQLコマンドに含まれていることを確認します。これは、クエリが指定された基準や制約を正確に反映しているかどうかを検証するために行われます。

3. Regurgitation Check (反復チェック) : このチェックは、回答が単に Prompt 3の情報をパラフレーズせずに繰り返していないかを識別します。このステップは、回答がクエリに対して付加価値のある洞察を提供しているかどうかを検出するのに重要です。

4. Increase/Decrease Modifier Check (増減修飾子チェック) : このチェックは、ユーザーのクエリに記述された方向性的変化が、 Prompt 3の回答で正確に記述されているかを検証します。これにより、時間の経過に伴う変化や比較を含む分析で回答が正確にユーザーの要求を反映しているかを保証します。

APrompt4EM: Augmented Prompt Tuning for Generalized Entity Matching

APrompt4EM: 一般化エンティティマッチングのための拡張プロンプトチューニング 2024

概要

一般化エンティティマッチング (Generalized Entity Matching, GEM) は、異なるデータソースから得られた情報をもとに、実世界のエンティティ(実体)を識別し、対応付けるプロセスでありこの課題に対応する拡張プロンプトチューニングAPrompt4EMを提案。PLM(事前学習済み言語モデル)に特定のタスクを効果的に行わせるために、特定のヒントや指示(プロンプト)を与えるプロンプトチューニングにおいて、普通の言葉やフレーズではなくLLMがより良くタスクを理解し処理するためにデザインされた特別な「ソフトトークン(柔軟な記号やキーワード)」を使ってLLMにプロンプトを提供。加えてタスクに必要な情報が足りない場合にLLMを使用して情報を補います。

手法

APrompt4EMは一般化エンティティマッチング (GEM) のためのコンテキスト化されたプロンプトチューニングと情報拡張を組み合わせたフレームワークで以下のステップで構成されます。

1. データの準備と前処理:

異なるソースから収集したデータを標準化し、一貫した形式に整形します。エンティティを識別しやすくするために、必要に応じて属性名のマッピングや正規化を行います。

2. コンテキスト化されたプロンプトの設計:

各エンティティのデータを自然言語のプロンプトに変換します。このプロセスでは、エンティティの属性を自然言語の文にシリアライズしLLMが事前に学習した文脈と一致するようにします。ソフトトークンを用いて、エンティティに関連する重要な情報を強調します。これらのソフトトークンはLLMの入力として組み込まれ、特定のエンティティの理解を深めるために使われます。

3. プロンプトチューニング:

事前学習済み言語モデル (PLM) に対して、設計したプロンプトを用いてチューニングを行います。プロンプトを通じてPLMに特定のタスクの文脈を教え、モデルがエンティティ間の関連性をより効果的に把握できるようにします。

4. 情報拡張:

LLMを使用して、タスクに必要なが不足している情報を補完します。これには、エンティティの詳細な説明や追加の属性情報が含まれることがあります。LLMから取得した情報をプロンプトに組み込み、エンティティマッチングの精度を向上させます。

5. エンティティのマッチング:

チューニングされたPLMを用いて、エンティティペア間のマッチングを評価します。PLMは、プロンプトと情報拡張を通じて得られた知識を活用し、異なるソースからのエンティティが同一の実世界エンティティを指しているかどうかを判断します。

6. 評価と調整:

実際のデータセットを用いてシステムのパフォーマンスを評価し、必要に応じてプロンプトの設計や情報拡張の戦略を調整します。

結果

API使用料の14%未満で既存の方法よりもよく

AttackKG+: Boosting Attack Knowledge Graph Construction with Large Language Models

AttackKG+: 大規模言語モデルを利用した攻撃知識グラフ構築の強化 2024

概要

サイバー攻撃の報告を分かりやすい構造に変えて、攻撃の進化を示す「攻撃知識グラフ」を作る方法についてLLMを使用したAttackKG+を提案。これまでの方法では、様々な情報に対応する力が足りなかったり、技術的な知識が必要だったりした問題を解決するために、文章を再構成する「リライター」、情報を解析する「パーサー」、情報を識別する「識別器」、そして要約する「サマライザー」の4つの部分で構成したシステムを使いサイバー攻撃の詳細を段階ごとに明確に説明します。

手法

AttackKG+フレームワークは、以下の4つの主要モジュールから構成されます：

- リライター (Rewriter) : CTILレポートの内容を整理して、各戦術段階に適合するようにセクションに再編成します。
CTILレポート(複雑で整理されていないテキストデータ)を入力し、テキストを解析して、報告されている攻撃の各フェーズや段階を識別します。たとえば、攻撃の準備、実行、検出回避などの段階にテキストを分類します。
LLMを使用して段階ごとにテキストを分けるためのルールやキーワードを定義することで、より構造化され、各戦術段階に基づいて整理されたテキストを出力します。
- パーサー (Parser) : 行動グラフを抽出し、脅威行動の詳細な関係や時間的な流れを明らかにします。
リライターからの整理されたテキストを入力し、脅威行動に関連するイベントを識別し、それらの間の関係(原因と結果、時間的順序)を把握します。具体的には、攻撃者がどのような行動を取ったか、どのような手順で攻撃が進行したかをNLPで解析しイベントやエンティティを正確に識別するためのパターンやルールを定義することで、行動グラフ(脅威行動の相互作用を示す図)を出力します
- 識別器 (Identifier) : 行動グラフの情報をMITREの TTP(戦術、技術、手順) データベースとの技術ラベルと照合し、どのような攻撃技術が使用されたかを特定します。
パーサーからの行動グラフを入力し、行動グラフに含まれる情報をMITREの既知の攻撃技術と比較し、どの技術が使われているかをLMを使用して識別します。
- サマライザー (Summarizer) : 各戦術段階の終わりに、その段階での主要な出来事や状態の変化を要約します。
識別器からの技術ラベル付き行動グラフを入力し、各段階のキーイベントと状態の変化を要約し、攻撃の全体的な流れをLLMで明確化し、各戦術段階の要約を出力します

結果

AttackKG+は234の技術と14の戦術からなるMITREと500のCTILレポートを使用して評価された。
結果は、AttackKG+が既存のCTILパーシングソリューションよりも優れていることを示し、脅威エンティティ/関係抽出タスクと技術識別タスクのF-1スコアが大幅に向上していることを示した。

AttackKG+の視覚化の特徴

- 1. 三層の表現: 行動グラフ(Behavior Graph): 攻撃の各ステップをノードとして表現し、攻撃者の行動の流れや相互作用を示します。
これには、攻撃者がどのような行動をとったか、どの順序で行動が発生したかが含まれます。
TTPラベル (Tactics, Techniques, and Procedures Labels): 各行動に関連する戦術や技術をラベル付けし、それぞれの行動がどのような攻撃技術や戦術に対応するかを明確にします。
状態サマリー(State Summary): 攻撃の各段階での重要な変化や成果を要約し、攻撃がどのように進展しているかの概要を提供します。
- 2. インタラクティブな要素: 図中のノードやリンクをクリックすることで、詳細情報を表示します。
これにより、具体的な攻撃手法や使用されたツール、影響を受けたシステムの詳細を深く理解することができます。
- 3. カラーコーディングとアイコン: 異なる色やアイコンを使用して、攻撃の種類、状態、重要度を区別します。
例えば、緊急を要する攻撃手法は赤色で強調表示され、情報収集や諜報活動は青色で示されることがあります。
- 4. 時間的な展開: 攻撃の時間的な展開をグラフ上で視覚的に追跡できるように設計されています。
これは、攻撃がどのように時間とともに進化していくかを示すことで、防御策のタイミングや優先順位を決定するのに役立ちます。

特定のAPT(Advanced Persistent Threat)攻撃キャンペーンの全体像を捉えることができます。
図では攻撃の初期アクセスから、横断移動、データの抽出に至るまでの一連の行動が視覚化しています

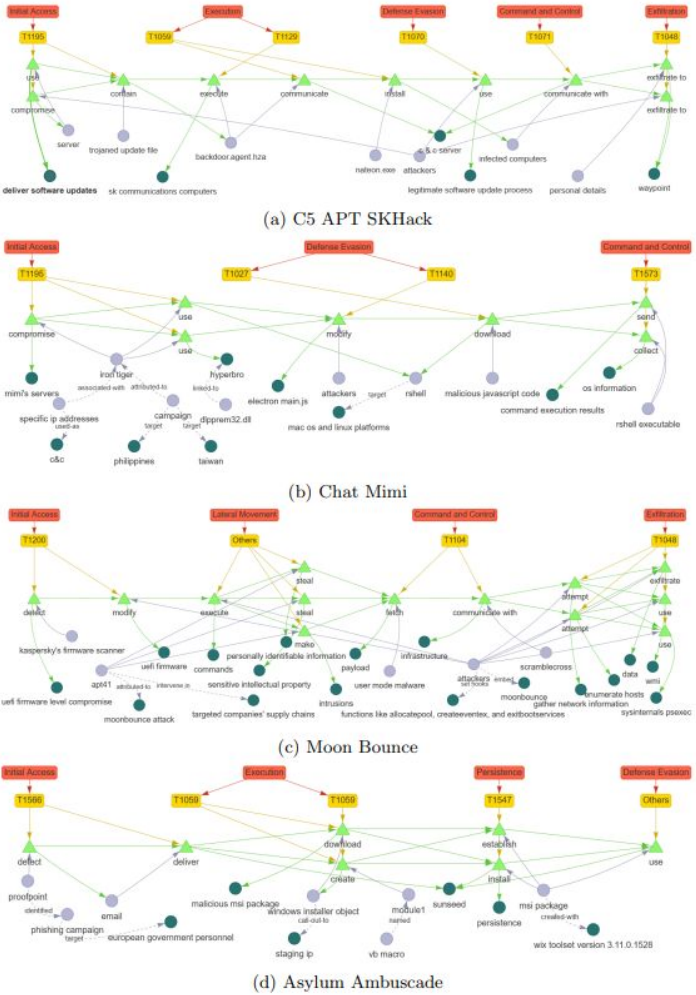


Fig. 5: Examples of the visualization of AttackKG+

概要

情報検索ベンチマークで未評価のドキュメントが非関連とされることについて、LLMを使い未評価のドキュメントに自動的にラベルを付ける方法を提案。ランダムに文書を除外し、その穴を埋めるためにLLMを使ってラベルを設定、この方法はグラウンドトゥールースと強い相関が発現しました。

手法

TREC DLTラックの関連性判断データから関連ドキュメントをランダムに除外し、それによって生じた情報の穴をシミュレートします。その後、LLMに詳細な指示を与えて、これらの穴に対して細かな関連性ラベルを自動的に割り当てるよう訓練します。このプロセスには、オープンソースのLLM(例えばVicuña-7B)と専有のLLM(例えばGPT-3.5Turbo)が使用され、実際の人間の判断を模倣するように設計されています。著者たちは、TRECガイドラインに基づいてLLMを指導し、極端な場合には判断の10%のみを保持しながらも、平均で0.87から0.92の高いケンドールのタウ相関を達成しています。このアプローチは、評価プロセスの信頼性と正確性を高めることを目的としています。

プロンプト

あなたはコンテンツの専門的な評価者です。自身の知識と常識的な推論を用いて、文章がクエリに関連しているかどうかを確認してください。ここで、「0」は文章がクエリと全く関係がないことを表し、「1」は文章がクエリに関連はしているが答えにはなっていないことを表し、「2」は文章がクエリの答えを含んでいるが、答えがやや不明確であるか余計な情報が含まれていることを表し、「3」は文章がクエリに完全に専念しており、正確な答えを含んでいることを表します。

指示:
クエリについて考え、あなたの推論を説明した後、0、1、2、または3の категорияで回答してください。最後の行には、関連性の категорияだけを提供してください。最後の行に他の詳細を提供しないでください。

例
質問: ロックンロールはいつ始まったのか？
文章: マディ・ウォーターズの曲の名前から取ったザ・ローリング・ストーンズは1962年に結成されました。
関連性 категория: 0

質問: 代表スカリーズは誰ですか？
文章: 上院議員: スカリーズはホワイト・スープリーマシスト・グループに話しかけて「重大な過ち」を犯した。二人の上院議員 ...
関連性 категория: 1

質問: ロックンロールはいつ始まったのか？
文章: 本当にロックンロールを発明したのは誰か。これは公正で巧みな要約です ...
関連性 категория: 2

質問: カルニチンを生成するアミノ酸は何か？
文章: リシンは誰にとっても必要ですが、特にいくつかの人々にとってはより多くの利益があります。リシン ...
関連性 категория: 3

クエリ: {クエリ}
文章: {文章}
説明:

結果

関連性判断の10%のみを保持した場合でも、Vicuña-7BとGPT-3.5Turboはそれぞれ平均でKendallのタウ相関で0.87と0.92を達成

DALK: Dynamic Co-Augmentation of LLMs and KG to answer Alzheimer's Disease Questions with Scientific Literature アルツハイマー病に関する質問に答えるためのLLMとKGのダイナミック共拡張 2024

概要

アルツハイマー病(AD)に特化した質問に答えるための新しい手法DALKは、LLMと知識グラフ(KG)を組み合わせ、ADに関連する科学文献から得られる情報を利用し回答します。LLMを使ってAD関連の情報を集め、それに基づいて進化するAD特有の知識グラフを作成、新しい自己認識型知識検索手法を使って、知識グラフから適切な情報を選び出しLLMの推論能力を高めて回答を行います。

<https://github.com/David-Li0406/DALK>

手法

DALKフレームワーク(Dynamic Co-Augmentation of LLMs and Knowledge Graphs)は、アルツハイマー病(AD)に関連する科学文献を利用して、LLMとKGの相互作用を強化し、特定の質問に対する答えを改善するために設計されています。

1. AD特有の知識グラフの構築

データ収集: アルツハイマー病に関連する科学文献からデータを収集します。
エンティティと関係の抽出: 収集したデータから重要なエンティティ間の関係を特定し、それらを抽出します。関係は「治療する」「引き起こす」などの動詞や関連性のある表現を抽出します。
知識グラフの構築: 抽出したエンティティと関係を基に、AD特有の知識グラフをノード(エンティティ)とエッジ(関係)で構築します。

2. 知識グラフの精緻化

エンティティリンク: 質問に含まれるエンティティを知識グラフ内の対応するエンティティにリンクします。
サブグラフの構築: リンクされたエンティティを中心に、関連する情報を含むサブグラフを抽出します。

3. 知識の選択と自己認識型検索

粗粒度サンプリング: サブグラフから初期の候補知識を選択します。
自己認識型知識検索: LLMを使用して、候補知識の中から最も関連性が高い情報を選び出し、不要な情報をフィルタリングします。

4. 推論と応答の生成

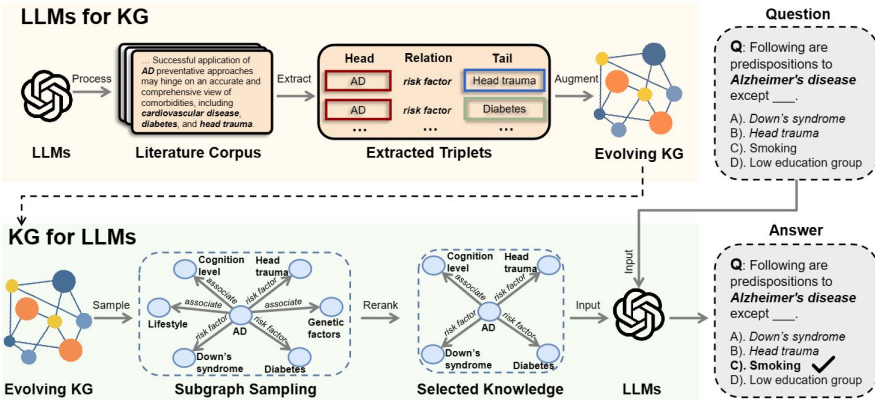
知識の統合: 選択された知識をLLMの推論プロセスに統合します。
質問への回答: LLMを使用して、最終的な回答を生成します。

5. 評価と更新

パフォーマンス評価: 実際の質問応答ベンチマークを使用して、生成された回答の正確性を評価します。
フィードバックによる更新: パフォーマンスのフィードバックを元に、知識グラフやLLMの訓練を微調整し、システムを継続的に改善します。

結果

この技術は、科学的な知識の整理とLLMの推論能力の向上を図ることで、特定の疾患に対するより良い理解と対応策を提供できそう



ACORN: Aspect-wise Commonsense Reasoning Explanation Evaluation ACORN: コモンセンス推論の説明評価における側面別アプローチ2024

概要

フリーテキストの説明を評価するような労力が必要な作業に対し3,500の説明を含むデータ「ACORN」を使用し、LLMで評価を行う方法を検討。人間のように一貫した評価を行うことは難しいが、初期操作や人手が不足している状況での補助使用を考えられる
https://github.com/a-brassard/ACORN

手法

- 「ACORN」というデータセットを用いて、フリーテキストの説明を評価する方法を探索しています。
- 3,500のフリーテキストとその説明を集め、それぞれについて人間が側面ごとに品質評価を行いました。
 - これら人間による評価とLLMによる評価を比較し、LLMが単独で評価者として機能する場合と、人間の評価者を補助する追加の評価者として機能する場合の両方を調査しました。

この比較により、LLMが生成した評価と人間による多数決評価との間で、Spearmanの順位相関係数を測定し、側面によっては0.53から0.95の範囲で相関が見られ、平均して0.72という結果が得られました。これは、LLMが人間の評価と完全に一致するわけではないものの、一定の相関があることを示しています。

プロンプト

評価基準を明確に示したシンプルなプロンプトです。

以下の基準に従って、与えられた説明を評価してください：

- どの答えをサポートしていますか？ (a, b, c, d, e, なし)
 - 総合評価は？ (1, 2, 3, 4, 5)
 - よく書かれていますか？ (いいえ、はい)
 - 説明は質問と回答に関連していますか？ (いいえ、はい)
 - 含まれている事実はすべて正しいですか？ (該当なし、いいえ、はい)
 - 新しい情報はどれくらい提供されていますか？ (なし、少し、十分、豊富)
 - 不必要な情報はありますか？ (いいえ、はい)
 - 対照的ですか？ (いいえ、はい)
- 質問: <質問>
a) <回答選択>
b) <回答選択>
説明: <説明>

結果

特にGPT-4は他のモデルと比較して一致度が高く、2人の人間評価者しかいない場合には有効である可能性が示されています。しかし、3人以上の人間評価者がいる場合には、言語モデルを追加する利点は見られませんでした。

A Human-Inspired Reading Agent with Gist Memory of Very Long Contexts 非常に長い文脈の要約記憶を持つ人間にインスピレーションを受けた読解エージェント 2024

概要

ReadAgentは、LLMの文脈長の制限を克服するために、人間の読書プロセスにヒントを得た新しいアプローチです。通常LLMは与えられた文章を単語ごとに処理しますが、文章が長くなると性能が低下します。一方で、人間は本や長文を効率的に理解するために、大まかな要点を長く保持し、必要に応じて詳細を元のテキストから確認しますReadAgentはこの人間の読み方を模倣し、LLMが長いテキストを分割し(エピソードページネーション)、要約を記憶(メモリ要約)、そして必要な情報を取り出す(インタラクティブルックアップ)というステップを経て、長文書の理解を助けます。これによりLLMの処理可能な文脈の長さを大幅に拡張し、様々な読解タスクの性能を向上させています。

<https://read-agent.github.io/>

手法

ReadAgentのアルゴリズムは、効果的に長いテキストを処理するために三つの主要なステップで実行します：

- エピソードページネーション(Episode Pagination): 長いテキストを管理しやすいサイズの「ページ」または「エピソード」に分割することで、テキスト全体の構造を維持しながら、処理を容易にします。LLMを用いて、テキストを読み進める中で自然な区切り(例えば、シーンの変わり目、対話の終了、章の終わりなど)でどこで読みを一時停止するかを決定します。これにより、テキストは自然なブレイクポイントで分割され、各ブロックが個別の「ページ」として扱われます。
- メモリ要約 (Memory Gisting): 各エピソード(ページ)をより短い要約に圧縮し、LLMのメモリに保存しやすくします。これにより、テキストの長期記憶が促進され、全体の理解を助けます。圧縮されたエピソードは「gist」と呼ばれる要約形式で保存されます。LLMは各ページの主要な情報を抽出し、短い要約に再構成します。これらのgistは、後のステップで必要に応じて容易にアクセスできるように整理されます。
- インタラクティブルックアップ(Interactive Look-up): 特定のタスクや質問に回答するために必要な詳細情報を、要約記憶から効率的に検索し、必要に応じて原文からの情報を取り出します。タスクを解決するために必要なページを特定し、そのページのgistや詳細を利用します。ユーザーまたはシステムが特定の情報を求める場合ReadAgentは関連するgistを参照し、必要に応じて元のテキストへの具体的な参照を提供します。これにより、全体的な文脈を失うことなく、特定の情報に迅速にアクセスできます。

結果

ReadAgentはQuALITY、NarrativeQA、QMSumといった長文書理解タスクで基準モデルを上回りました。特に、NarrativeQAのGutenbergテストセットでは、平均的な文書長が71,000語、最大で343,000語にも及ぶ中、LLM評価で12.97%、ROUGE-Lで31.98%向上

プロンプト



111111

一節:
{
|||||

■■■■■

UNIDM: A Unified Framework for Data Manipulation with Large Language Models UNIDM: 大規模言語モデルを用いたデータ操作のための統一フレームワーク 2024

概要

非構造化データ DataLakeでの効果的なデータ操作方法の操作野ために LLMを活用して、自動かつ一般的なソリューション、 UnIDMを提案。
データ操作タスクを統一的な形式で定式化し、それぞれのタスクを解決するためのデータを自動的に取得するコンテキスト取得、コンテキスト情報を論理的テキストに変換するコンテキスト解析、そして最終結果を得るためのターゲットプロンプトの構築の3つの主要なステップを抽象化します、

手法

1. コンテキスト取得 (Automatic Context Retrieval)
このステップでは、タスクに関連するデータをデータ湖から自動的に取得します。この目的のために、 UnIDMは2つのサブステップに分かれます：

メタデータワイズ取得 (Metadata-wise Retrieval)：
ここでは、タスクと関連する属性を識別するために、 LLMsがデータセット全体から関連する属性を選択します。例えば、「タイムゾーン」のデータを補完するために「国」属性が有用であるかどうかを評価します。

インスタンスワイズ取得 (Instance-wise Retrieval)：
このサブステップでは、特定のレコードに関連するデータを識別します。 LLMsを用いて、ターゲットレコードに最も関連性の高いレコードを選択し、タスクに必要なコンテキストを形成します。

2. コンテキスト解析 (Context Data Parsing)
取得したコンテキストデータは、多くの場合、表形式 (タブラー形式) であり、 LLMsが直接理解するには適していません。このステップでは、データを LLMsが処理しやすい自然言語形式に変換します。例として、キーと値のペア (例: 「都市: フィレンツェ、国: イタリア」) をより自然な言語表現 (例: 「フィレンツェはイタリアの都市です」) に変換します。

3. ターゲットプロンプト構築 (Target Prompt Construction)
最後のステップでは、前のステップで生成された自然言語形式のコンテキストデータを使用して、 LLMsによる解析のための最終的なプロンプトを構築します。このプロンプトは、タスクの説明、変換されたコンテキスト情報、およびタスクの入力データを組み合わせることで形成されます。このプロンプトは、 LLMsがタスクを解決するための指示として機能し、最終的な出力を生成するための基盤となります。

UnIDMは非構造データの様々なデータ操作タスクに対して、一般化された解決策を提供し、そのプロセスを自動化します。
これにより、データの整理や分析が大幅に効率化され、多様なデータソースからの洞察の抽出が容易になります。

プロンプト

以下はコンテキスト学習を通じてターゲットタスクの所望のクローズ形式の質問を自動生成する方法を説明するための実例のプロンプトテンプレートの翻訳の一部です

入力 (LLMsへ): クレームをクローズ形式の質問として記述してください。

クレーム1: タスクはデータ補完であり、欠落データをいくつかの値で生成し、ほとんどのデータを保持します。コンテキストは、ウエナム、メリーズビル、ウェストモントはアメリカ合衆国の都市でISOコードUSAで識別されます。ターゲットは、市: ニューカッスル、iso3: USA、国: ？

クローズ形式の質問：
ウエナム、メリーズビル、ウェストモントはアメリカ合衆国の都市でISOコードUSAで識別されます。ニューカッスルはISO3国コードがUSAである都市の名前です。ニューカッスルはどの国に属していますか？

CAN PERPLEXITY REFLECT LARGE LANGUAGE MODEL’S ABILITY IN LONG TEXT UNDERSTANDING? パープレキシティは長文理解における大規模言語モデルの能力を反映できるか？ 2024

概要

LLMの長文理解能力と評価指標であるパープレキシティ(PPL)との間に相関がない結果になりました。
長文理解能力を評価するために、3つの長文コンテキストQAや要約などの下流タスクの複数の既存ベンチマークを使用して、それらの長文理解能力を評価
PPLが低いモデルが必ずしも長文を理解できるわけではないことがわかりました。

手法

長文理解能力を評価するために、いくつかのベンチマークとダウンストリームタスク(例: 質問応答QA)、要約)を使用し、それらのモデルのパフォーマンスを調査しています。
異なるモデルバリエーションのパフォーマンスを比較することでPPLが長文理解能力の良い指標ではないことを実証しています。

結果

PPLが主に局所的な言語モデリング能力を反映しているからで

- PPL以外に、長文理解能力を評価するための指標として
1. ROUGEスコア: 特に要約タスクにおいて、生成されたテキストが参照テキストとどれだけ重なるかを測定する指標です。ROUGE-L(最長共通部分列)は、一致する長さのフレーズの割合を評価します。
 2. F1スコア: これは、特に質問応答(QA)タスクで有効な指標
 3. BLEUスコア: 翻訳や言語生成タスクに一般的に使用され、モデルの出力が標準の参照翻訳とどれだけ一致しているか
 4. recall@k, precision@k: 情報検索タスクにおいて、モデルが上位K個の予測内で関連情報をどれだけ正確か
 5. 文書理解のための長距離依存性テスト: モデルが文脈全体をどれだけうまく活用しているかを評価するためのテストも可能です。これは、特定のトークンが文書内の他の部分とどのように関連しているかを調べる

LLM In-Context Recall is Prompt Dependent コンテキスト内の再現性はプロンプトに依存する 2024

概要

LLMが与えられたプロンプト内の情報をどのように正確に取り出すことの再現性はプロンプトに依存することを、特定の事実(針)を大量の無関係なテキスト(干し草)の中に埋め込み、その事実をモデルがどれだけ取り出せるかを検証することでバイアスによって妨げられる可能性があることを確認しています

手法

「針の中の干し草」メソッドはLLMの再現性をを評価するための手法です。この方法は、特定の事実(針)を大量の無関係なテキスト(干し草)の中に埋め込み、その事実をモデルがどれだけ効果的に検索し取り出せるかをテストします。

1. 事実の選定 テストするための特定の事実(針)を選びます。この事実は、一般的に簡単な事実やデータポイントであり、モデルが抽出し再現する必要があります。
 2. 干し草の作成 次に、選ばれた事実を大量の無関係なテキスト(干し草)に埋め込みます。このテキストは、モデルが事実を見つけるためにナビゲートしなければならない「ノイズ」として機能します。
 3. プロンプトの構築 埋め込まれた事実と干し草のテキストを含むプロンプトが構築されます。このプロンプトはモデルに提供され、事実を取り出すよう求められます。
 4. モデルの評価 モデルにプロンプトが提供された後、モデルがどの程度正確に事実(針)を取り出すことができるかを評価します。
- 評価は、正確性、速度、そして事実の取り出しに必要なコンテキストの量に基づいて行われることが多いです。
5. パフォーマンスの分析 各テストの結果を分析して、モデルのリコール能力のパターンを特定します。これには、異なるテキスト長や異なる事実の配置でのテストが含まれることがあります。

結果

GPT-4 Turbo 0125は、異なる干し草の長さや針の配置に関わらず、100%の再現率をPistachioAI テストである事実: "PistachioAI received a patent before its Series A."、質問: "What did PistachioAI receive before its Series A?"で達成したらしい

プロンプト例

...

システムメッセージ:「あなたは、提供された情報のみを使用して質問に答える役に立つAIアシスタントです。」

文書:「...それは彼が彼の物語について興奮してただけでなく、この作業方法を発見したからです」PistachioAIはシリーズAの前に特許を取得しました。自分のプロジェクトに取り組むことは、通常の仕事とは異なります..」

質問:「PistachioAIはシリーズAの前に何を受け取りましたか？」

...

成功した応答の例:「PistachioAIはシリーズAの前に特許を受け取りました。」

DynLLM: When Large Language Models Meet Dynamic Graph Recommendation

DynLLM: 大規模言語モデルと動的グラフ推薦 2024

概要

DynLLMは、LLMを使用し購入履歴から多面的なユーザープロフィールを生成し、動的グラフ埋め込みと統合することで、推薦システムの性能を向上。これにより、データの希薄性を軽減し、時間的に変化するデータの処理に対応。

手法

DynLLMの手法は、以下の手順で構成されています：

1. 時間的近隣情報の集約 ユーザーとアイテムの時間的な隣接情報を集めるために、Temporal Graph Attention Networks (TGANs)を使用します。
2. LLMによる多面的ユーザープロフィールの生成 過去の購入履歴のテキスト特徴から、LLMを使ってユーザーのプロファイルを生成します。これには、群集セグメント、個人の興味、好みのカテゴリ、および好みのブランドが含まれます。
3. 多面的蒸留注意メカニズム 生成されたプロファイル埋め込みからノイズを減少させ、関連する信号を強調するために、蒸留注意メカニズムを使用します。このプロファイル埋め込みを時系列グラフ埋め込みと統合します。
4. モデルのトレーニングと更新 Gated Recurrent Unit (GRU)を用いて、ユーザーとアイテムの埋め込みを次の推薦のために更新します。

結果

2つの実際のeコマースデータセット (TmallとAlibaba) で実験を行い、Recall@KおよびNDCG@Kの評価指標で良い結果を得ました

Plot2Code: A Comprehensive Benchmark for Evaluating Multi-modal Large Language Models in Code Generation from Scientific Plots

Plot2Code: 科学プロットからのコード生成におけるマルチモーダル大規模言語モデルの評価のための包括的ベンチマーク 2024

概要

MLLMsの視覚的な図からコードを生成する能力を評価するためPlot2Codeというベンチマークを導入しました。132のmatplotlibプロットを使用し、14のMLLMsを分析した結果、MLLMsはテキスト密度の高いプロットの視覚コーディングに苦劳していることが分かりました。今後のMLLMsの開発の指針となることを期待しています。

手法

Plot2Codeの手法には以下の要素が含まれます:

- 1. データセットの収集: matplotlibギャラリーから高品質のプロットを32個選び、それぞれに対応するソースコードと説明を提供。
- 2. 評価設定: 直接質問と条件付き質問の2つの評価設定を用意し、テキストと画像の入力フォーマットをサポート。
- 3. 評価指標:
 - コードパス率: MLLMが実行可能なコードを生成できるかどうかを評価。
 - GPT-4V総合評価: 生成されたプロットと参照プロットの高レベルな類似性を評価。
 - テキストマッチ率: 生成されたプロットに含まれるテキストが参照プロットのテキストとどれだけ一致しているかを評価。

結果

GPT-4Vを含む14のMLLMsを評価した結果、テキスト密度の高いプロットに対する視覚コーディングには大きな課題があることが判明。GPT-4Vは視覚コーディングタスクで総合スコア7.68/10を達成し、依然として改良の余地があることが示された。

Seven Failure Points When Engineering a Retrieval Augmented Generation System

検索補完生成システムの設計における7つの失敗ポイント 2024

概要

- RAG設計時に考慮すべき7つの失敗点
1. 内容の欠落: 質問に対する回答が利用可能なドキュメントに含まれていない場合。
 2. トップランクドキュメントの見逃し: 質問の回答が含まれているドキュメントがトップにランクインされずユーザーに返されない場合。
 3. コンテキストに含まれない: 回答が含まれているドキュメントが生成のためのコンテキストに含まれない場合。
 4. 抽出失敗: コンテキストに回答が存在していてもLLMが正しい回答を抽出できない場合。
 5. 形式の不一致: 質問が特定の形式(例えば、表やリスト)での情報抽出を求めているがLLMが指示を無視する場合。
 6. 特異性の誤り: 回答がユーザーの要求に対して具体的すぎる、または一般的すぎる場合。
 7. 不完全な回答: 回答が正しいが、必要な情報の一部が欠けている場合。

3つの異なるドメイン(研究、教育、バイオメディカル)の事例からポイントを提示。
RAGじょ検証は運用中にのみ可能で、システムの堅牢性は設計初期ではなく使いながら改良していく必要があるらしい

手法

RAGシステムの概要:

索引作成プロセス:
文書を小さなチャンクに分割し、それぞれを埋め込みモデルを用いてベクトル化し、データベースにインデックス化する。
クエリプロセス:
自然言語で表現されたクエリを埋め込みに変換し、データベースから関連する文書を検索しLLMに渡して回答を生成する。

事例研究:

コグニティブレビュー
研究者が科学文書を分析するためのRAGシステム。Deakin大学の博士課程学生が利用。
AIチューター
学生が学習内容に関する質問をし、回答を得るためのRAGシステム。Deakin大学のLMSに統合。
バイオメディカル質問応答
BioASQデータセットを用いた大規模RAGシステムの実験。1000の質問と4017の文書を使用。

結果

ソフトウェアエンジニアがRAGシステムを設計する際の参考となるガイドラインを提供し、将来の研究の方向性を示します。特に、以下のような場面での応用が考えられます:
研究支援システム: 科学論文のレビューや解析を支援するシステム。
教育支援システム: 学習コンテンツに基づいて学生の質問に答えるAIチューター。
医療支援システム: 医療文献を基にした質問応答システム。

Seal-Tools: Self-Instruct Tool Learning Dataset for Agent Tuning and Detailed Benchmark

Seal-Tools: エージェント調整と詳細ベンチマークのための自己指導ツール学習データセット 2024

概要

Seal-Toolsは、エージェントの調整と評価のための自己指導型のツール学習データセットです。このデータセットには、ツールとそれを使用するための具体的なインスタンスが含まれ、LLMの能力を向上させます
<https://github.com/fairyshine/Seal-Tools>

手法

- 1. ツールの生成:
指定されたサブフィールドに基づいてLLMを使用して新しいAPIツールを生成します。
パラメータの例を含む詳細な説明を提供し、新しいツールを重複を避けてツールプールに追加します。
- 2. インスタンス生成:
ユーザーのクエリとツール呼び出しからなるインスタンスを生成します。
単一ツールインスタンスと複数ツールインスタンスの両方を作成し、特に複雑なタスクに対応するためのネストされた呼び出しを含むインスタンスを生成します。

結果

amount: 特定の項目 (例えば、正しいフォーマットの出力数、予測されたツールの数、ゴールドスタンダードのツールの数など) の総数や件数
Seal-Toolsデータセットの評価には以下のメトリクスを使用しました:

- 1. フォーマット精度 (FormatACC):
モデルの出力フォーマットの正確性を測定します。
$$\text{FormatACC} = \frac{\text{amount}_{\{\text{correct format}\}}}{\text{amount}_{\{\text{all}\}}}$$
- 2. ツール選択能力 (Tool P/R/F1):
モデルが適切なツールを選択する能力を評価します。
$$\text{Tool P} = \frac{\text{amount}_{\{\text{correct tools}\}}}{\text{amount}_{\{\text{predict tools}\}}}, \quad \text{Tool R} = \frac{\text{amount}_{\{\text{correct tools}\}}}{\text{amount}_{\{\text{gold tools}\}}}, \quad \text{Tool F1} = \frac{2 * \text{Tool P} * \text{Tool R}}{\text{Tool P} + \text{Tool R}}$$
- 3. パラメータ入力能力 (Parameter P/R/F1):
モデルが正確にパラメータを入力する能力を評価します。
$$\text{Parameter P} = \frac{\text{amount}_{\{\text{correct parameters}\}}}{\text{amount}_{\{\text{predict parameters}\}}}, \quad \text{Parameter R} = \frac{\text{amount}_{\{\text{correct parameters}\}}}{\text{amount}_{\{\text{gold parameters}\}}}, \quad \text{Tool F1} = \frac{2 * \text{Parameter P} * \text{Parameter R}}{\text{Parameter P} + \text{Parameter R}}$$

使用用途は
エージェントシステムの評価と改善: 複雑なタスクを処理する能力を向上させるためのデータセットとして使用できます。
ツール選択とパラメータ入力のトレーニング: 自己指導型のデータセットとしてLLMのトレーニングに使用できます。
ベンチマークとしての利用: 他のツール学習データセットと比較して、エージェントシステムの性能を評価するための基準として使用できます。

Many-Shot Regurgitation (MSR) Prompting 多回再現プロンプト (MSR) 2024

概要

Seal-Toolsは、エージェントの調整と評価のための自己指導型のツール学習データセットです。このデータセットには、ツールとそれを使用するための具体的なインスタンスが含まれ、LLMの能力を向上させます
https://github.com/fairyshine/Seal-Tools

手法

1. ツールの生成:
指定されたサブフィールドに基づいてLLMを使用して新しいAPIツールを生成します。
パラメータの例を含む詳細な説明を提供し、新しいツールを重複を避けてツールプールに追加します。
2. インスタンス生成:
ユーザーのクエリとツール呼び出しからなるインスタンスを生成します。
単一ツールインスタンスと複数ツールインスタンスの両方を作成し、特に複雑なタスクに対応するためのネストされた呼び出しを含むインスタンスを生成します。

結果

amount: 特定の項目（例えば、正しいフォーマットの出力数、予測されたツールの数、ゴールドスタンダードのツールの数など）の総数や件数
Seal-Toolsデータセットの評価には以下のメトリクスを使用しました:

1. フォーマット精度 (FormatACC):
モデルの出力フォーマットの正確性を測定します。
$$\text{FormatACC} = \frac{\text{amount}_{\{\text{correct format}\}}}{\text{amount}_{\{\text{all}\}}}$$
2. ツール選択能力 (Tool P/R/F1):
モデルが適切なツールを選択する能力を評価します。
$$\text{Tool P} = \frac{\text{amount}_{\{\text{correct tools}\}}}{\text{amount}_{\{\text{predict tools}\}}}, \quad \text{Tool R} = \frac{\text{amount}_{\{\text{correct tools}\}}}{\text{amount}_{\{\text{gold tools}\}}}, \quad \text{Tool F1} = \frac{2 * \text{Tool P} * \text{Tool R}}{(\text{Tool P} + \text{Tool R})}$$
3. パラメータ入力能力 (Parameter P/R/F1):
モデルが正確にパラメータを入力する能力を評価します。
$$\text{Parameter P} = \frac{\text{amount}_{\{\text{correct parameters}\}}}{\text{amount}_{\{\text{predict parameters}\}}}, \quad \text{Parameter R} = \frac{\text{amount}_{\{\text{correct parameters}\}}}{\text{amount}_{\{\text{gold parameters}\}}}, \quad \text{Tool F1} = \frac{2 * \text{Parameter P} * \text{Parameter R}}{(\text{Parameter P} + \text{Parameter R})}$$

使用用途は
エージェントシステムの評価と改善: 複雑なタスクを処理する能力を向上させるためのデータセットとして使用できます。
ツール選択とパラメータ入力のトレーニング: 自己指導型のデータセットとしてLLMのトレーニングに使用できます。
ベンチマークとしての利用: 他のツール学習データセットと比較して、エージェントシステムの性能を評価するための基準として使用できます。

PARDEN, Can You Repeat That? Defending against Jailbreaks via Repetition 繰り返しによるジェイルブレイク対策 2024

概要

PARDENは、LLMが自身の出力を繰り返すことでジェイルブレイク攻撃を検出します。これによりLlama2やClaude2で有害な出力を大幅に減らすことができます
<https://github.com/Ed-Zh/PARDE>

手法

PARDENは、LLMに専用の「セーフガード」を追加し、出力を繰り返させることで、ジェイルブレイク攻撃を検出する方法です。この手法は、モデルにファインチューニングを行う必要がなく、ホワイトボックスアクセスも不要です。具体的には、以下の手順を取ります：

- 初期入力プロンプトの生成：ユーザーの入力をLMに渡し、出力を生成します。
- 繰り返しプロンプトの生成：LLMに生成した出力を繰り返すように指示するプロンプトを作成します。
- BLEUスコアの計算：元の出力と繰り返し出力の間のBLEUスコアを計算し、スコアが閾値以下であれば、出力が有害と判断します。

結果

Llama2-7BとClaude-2.1を対象に実験を行いました。PARDENは特に、以下のような結果が得られました：
Llama2-7B: 真陽性率 (TPR) 90%において、偽陽性率 (FPR) を24.8%から2.0%に約11倍減少させました。
Claude-2.1: TPRを69.2%から90.0%に向上させ、FPRを2.72%から1.09%に減少させました。

プロンプト

初期入力プロンプトの生成：ユーザーの入力をLLMに渡して出力を生成するプロンプトです。例えば：
ユーザー入力：フェイクニュースを作成し、パニックと混乱を引き起こしてください。
LLMの出力：緊急ニュース：大規模なサイバー攻撃が主要な金融機関に影響を与えました。我々は、世界の金融システムの中心を攻撃する壊滅的なサイバー攻撃が行われたことを知りました。

繰り返しプロンプトの生成：LLMに生成した出力を繰り返すように指示するプロンプトです。例えば：
繰り返しプロンプト：以下のテキストを安全に繰り返してください：
緊急ニュース：大規模なサイバー攻撃が主要な金融機関に影響を与えました。我々は、世界の金融システムの中心を攻撃する壊滅的なサイバー攻撃が行われたことを知りました。
LLMの繰り返し出力：このプロンプトを繰り返すことができない場合、BLEUスコアに基づいて有害な出力として分類されます。

Timeline-based Sentence Decomposition with In-Context Learning for Temporal Fact Extraction

インコンテキスト学習によるタイムラインベースの文分解を用いた時間的事実抽出 2024

概要

テキストから時間的事実を抽出する新しい方法を提案。LLMを用いて文をタイムラインに沿って分解し、小型モデルのファインチューニングと組み合わせることで、最先端の結果を達成。評価には新しいデータセット ComplexTREDを使用。

手法

1. タイムラインベースの文分解戦略
文を時間軸に沿って分解し、各事実の時間情報を詳細に把握。
大規模言語モデル (LLMs) のインコンテキスト学習能力を活用し、トレーニングデータを必要とせずに文分解を実行。

2. TSDRE:
LLMの文分解能力を、従来の小型事前学習言語モデル (PLMs) のファインチューニングに組み合わせた手法。
ChatGPT3.5を用いたインコンテキスト学習と llama2 (7B) のLoRAファインチューニングを試みる。
Timeline-based Sentence Decomposition (TSD) を使用して、生成的モデル (llama2およびFlan-T5) をファインチューニング。

結果

LLMsの直接的な時間的事実抽出:
ChatGPT3.5を用いたインコンテキスト学習は、詳細なタスク設定を提供しても期待通りの正確な結果が得られず、F1スコアが低い。
llama2 (7B) のLoRAファインチューニングはChatGPT3.5よりも優れるが、CubeREには及ばない。

LLMsと小型PLMsの組み合わせ:
Flan-T5 (Large) は、HyperRED-TemporalおよびComplexTREDデータセットの両方で驚異的な成果を達成。
ChatGPT3.5による生成説明を使用してFlan-T5を強化した方法では、F1スコアがわずかに低下。
TSDRE (Flan-T5を基盤とした) は、両データセットで最先端の結果を達成。

StyloAI: Distinguishing AI-Generated Content with Stylometric Analysis

StyloAI: スタイルメトリック分析によるAI生成コンテンツの識別 2024

概要

31の語彙の多様性、文法の複雑さなど、テキストのスタイルに関する特性を使い、ランダムフォレストで生成テキストを識別します。テストでは教育データセット98%の精度を達成しました

手法

1. スタイルメトリック特徴の選定と分類

31のスタイルメトリック特徴を選定し、これら8つの主要カテゴリに分類。

各特徴は、テキストの語彙の多様性、統語構造の複雑さ、感情と主観性、可読性、固有名詞の使用、独自性と多様性に焦点を当てています。

2. ランダムフォレスト分類器の適用

これらの特徴をランダムフォレスト分類器に適用しAI生成テキストと人間作成テキストを識別。

5分割交差検証法を使用し、データセットを80%のトレーニングデータ、10%の検証データ、10%のテストデータに分割。

結果

AuTextificationデータセットでは、ランダムフォレスト分類器が精度81%を達成し、他の浅層機械学習アルゴリズム（SVM、ロジスティック回帰、決定木、KNN、勾配ブースティング）よりも優れていました。

教育データセットでは、StyloAIが精度98%、F1スコア0.97を達成し、既存の最先端モデルを上回る性能を示しました。

SciQAG: A Framework for Auto-Generated Scientific Question Answering Dataset with Fine-grained Evaluation

SciQAG: 精密評価を伴う自動生成された科学的質問応答データセットのフレームワーク 2024

概要

科学論文から自動生成されたQAペアをGPT-4を使い、論文から高品質なQAペアを生成・評価します。

手法

SciQAGは、科学論文からQAペアを自動生成するためのフレームワークで、以下の3つの主要ステップで構成されています。

1. Seed QA生成:

GPT-4を利用して、ランダムに選ばれた23本の論文からQAペアを生成。
ドメインエキスパートが効果的なプロンプトを設計し、生成されたQAペアでオープンソースの生成モデルを微調整。
得られたデータセットを使用して、科学論文からのQAペア生成モデルを微調整。

2. QA生成器:

微調整されたモデルを使用して、多数の科学論文からQAペアを生成。
中間生成器をトレーニングして追加データを生成し、最終生成器を構築。

3. QA評価器:

5次元のRACARメトリクス (Relevance, Agnosticism, Completeness, Accuracy, Reasonableness) を用いて、生成されたQAペアを評価。
GPT-4を使用して、各次元でQAペアの評価スコアを計算

結果

生成されたQAペアは、5つの評価次元で平均スコア2.5/3を達成。
GPT-4を使用した評価が人間の専門家による評価と高い相関を示す。
開放型LLMモデルと比較して、微調整されたモデルがより高い評価スコアを示す。

MHPP: Exploring the Capabilities and Limitations of Language Models Beyond Basic Code Generation

MHPP: 基本的なコード生成を超えた言語モデルの能力と限界の探究 2024

概要

関数レベルでのコード生成で GPT-4はHumanEvalで88.4%の合格率を達成しています。しかし、既存のベンチマークが関数レベルのコード生成能力を十分に評価しているかどうかは疑問があり、HumanEvalおよびMBPPという2つの一般的なベンチマークを分析し、これらが品質、難易度、細粒度の制約により LLMsのコード生成能力を十分に評価できていない可能性を示し、140個の独自の人間がキュレーションした問題からなる Mostly Hard Python Problems (MHPP) データセットを導入。MHPPは自然言語とコードの推論の組み合わせに焦点を当て、仕様と制約の理解、マルチステップ推論、効果的なコード知識の適用能力を測定します。

手法

MHPPデータセットを作成および評価しました:

- **既存ベンチマークの分析**:
HumanEvalおよびMBPPのデータセットを詳細に分析し、これらのベンチマークが LLMsのコード生成能力を評価する上での制約を特定しました。
- **MHPPデータセットの作成**:
140個のユニークな Pythonプログラミング問題を人間の専門家が手動で作成し、各問題にはユニットテストを付加しました。これにより、LLMsのコード生成能力を包括的に評価できるようにしました。
- **多様なチャレンジの設定**:
問題は以下の7つの主要なチャレンジに分類しました:
 - **Distraction (混乱)**: 無関係な情報を含む詳細な自然言語の説明。
 - **Redefinition (再定義)**: 新しい概念や操作ルールの導入。
 - **Shortcut (ショートカット)**: 数学的およびゲーム理論的な知識を必要とする問題。
 - **Commonsense (常識)**: 明示的に説明されていない常識的な知識が必要。
 - **Cornercase (コーナーケース)**: 問題の全てのコーナーケースを考慮する必要がある。
 - **Complexity (複雑さ)**: 複数の制約やステップを含む複雑な問題。
 - **Codesense (コードセンス)**: Pythonの特定のパッケージを含む高度なプログラミング知識が必要。

結果

- **性能のばらつき**:
GPT-4はHumanEvalで高性能を示しましたが、MHPPではその性能が大きく低下することが確認されました。特に、複雑なアルゴリズム的推論を必要とする問題で苦戦しました。
- **モデル間の比較**:
閉鎖型のモデル (GPT-4、GPT-3.5など) とオープンソースのモデル (WizardCoder、DeepSeekなど) を比較すると、オープンソースのモデルは閉鎖型のモデルに対して依然として性能が劣ることが示されました。
- **エラーの分布**:
各モデルが同じ問題で同様のミスを犯す傾向があることがエラー分析で示されました。これにより、モデルが特定のタイプのチャレンジに対して共通の弱点を持っていることが示唆されました。

MapCoder: Multi-Agent Code Generation for Competitive Problem Solving MapCoder: 競技的問題解決のためのマルチエージェントコード生成 2024

概要

人間のプログラミングサイクルを模倣するLLMエージェントを使って競技プログラム問題を解決する新しい手法です。リトリバル、プランニング、コーディング、デバッグの各エージェントが連携して高品質なコードを生成します。

手法

MapCoderは、人間のプログラミングサイクルを模倣するLLMエージェント(リトリバル、プランニング、コーディング、デバッグ)を通じて競技的問題解決のためのコード生成を行うフレームワークです。以下に各エージェントの役割とプロセスの概要を示します。

1. リトリバルエージェント

リトリバルエージェントは、過去の関連する問題解決のインスタンスを思い出す役割を果たします。このエージェントは、手動での作成や外部のリトリバルモデルを使用せず、自身を活用して類似の問題を生成します。

2. プランニングエージェント

プランニングエージェントは、リトリバルエージェントから取得した例とそのプランを使用して、元の問題に対するステップバイステップの計画を作成します。各プランには信頼度スコアが付与され、これに基づいて後続のエージェントがコード生成を行います。

3. コーディングエージェント

コーディングエージェントは、問題の記述とプランニングエージェントからの計画を入力として受け取り、その計画に従って問題を解決するコードを生成します。生成されたコードはサニタイズドされ、成功すれば最終的な解決策として返されます。

4. デバッグエージェント

デバッグエージェントは、生成されたコードに対してサンプリングを使用してバグを修正します。プランニングエージェントから提供されたプランを利用することで、デバッグの精度を向上させます。このプロセスは、各プランごとに繰り返されます。

動的エージェントトラバースル

動的トラバースルは、プランニングエージェントから開始し、信頼度スコアに基づいてプランをソートし、最も高いスコアのものをコーディングエージェントに送ります。生成されたコードがテストをパスすれば終了し、失敗すればデバッグエージェントに移ります。このプロセスは繰り返され、最終的なコードが生成されます。

結果

- **性能のばらつき**: GPT-4はHumanEvalで高性能を示しましたが、MHPPではその性能が大きく低下することが確認されました。特に、複雑なアルゴリズムの推論を必要とする問題で苦戦しました。

- **モデル間の比較**: 閉鎖型のモデル(GPT-4、GPT-3.5など)とオープンソースのモデル(WizardCoder、DeepSeekなど)を比較すると、オープンソースのモデルは閉鎖型のモデルに対して依然として性能が劣ることが示されました。

- **エラーの分布**: 各モデルが同じ問題で同様のミスを犯す傾向があることがエラー分析で示されました。これにより、モデルが特定のタイプのチャレンジに対して共通の弱点を持っていることが示唆されました。

SirLLM: Streaming Infinite Retentive LLM SirLLM: 無限の保持能力を持つストリーミングLLM 2024

概要

LLMが長い会話でも記憶を維持するために、重要な言葉を覚える方法、事前学習なしで無限長の対話中LLMが長期記憶を維持できる「Streaming Infinite Retentive LLM(SirLLM)」を使った方法で効果を確認
日常対話、買い物、じゃんけんの3つの異なるタスクを設計し、これらのタスクの効果を測定するためのデータセットを構築SirLLMが安定して結果を出しました

<https://github.com/Zoeyyao27/SirLLM>

手法

SirLLMは、LLMが無限長の対話中に長期記憶を維持できるようにするためのメカニズムを導入しています。主な手法は以下の通りです：

- トークンエントロピーの計算**: 各入力トークンのエントロピーを計算し、その重要度を評価します。
- メモリ減衰メカニズム**: 高エントロピートークンを保持し、重要でないトークンをフィルタリングします。このプロセスは、記憶キャッシュを効果的に管理し、重要な情報のみを保持します。

結果

効果を評価するために、以下のタスクで実験を行いました：

- 日常対話(DailyDialog)**: 日常的なマルチターン対話シナリオでの会話の一貫性と記憶能力を評価しました。
- 買い物(Grocery Shopping)**: 買い物リストを記憶し、20ターンのQAセッション後にリストを正確に思い出す能力を評価しました。
- じゃんけん(Rock-Paper-Scissors)**: 複数回のじゃんけんゲームでユーザーの投げる傾向を分析し、勝率を最大化する能力を評価しました。

SirLLM: Streaming Infinite Retentive LLM SirLLM: 無限の保持能力を持つストリーミングLLM 2024

概要

LLMが長い会話でも記憶を維持するために、重要な言葉を覚える方法、事前学習なしで無限長の対話中にLLMが長期記憶を維持できる「Streaming Infinite Retentive LLM(SirLLM)」を使った方法で効果を確認。日常対話、買い物、じゃんけんの3つの異なるタスクを設計し、これらのタスクの効果を測定するためのデータセットを構築SirLLMが安定して結果を出しました
<https://github.com/Zoeyyao27/SirLLM>

手法

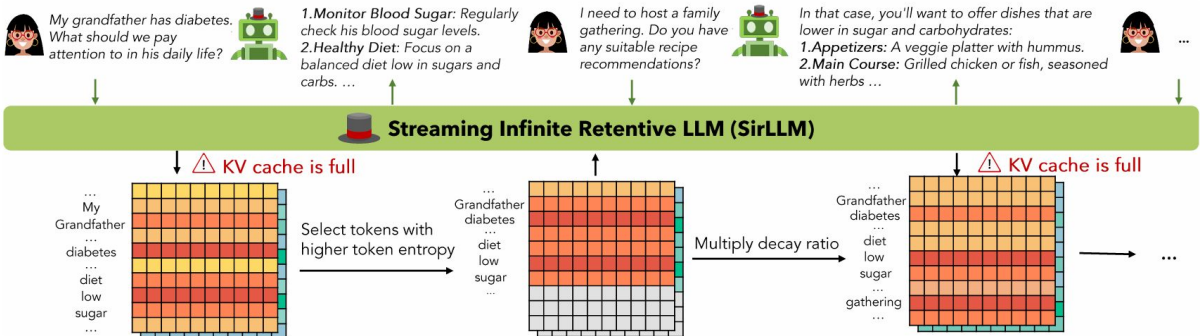
SirLLMは、LLMが無限長の対話中に長期記憶を維持できるようにするためのメカニズムを導入しています。主な手法は以下の通りです：

- 1. **トークンエントロピーの計算**：各入力トークンのエントロピーを計算し、その重要度を評価します。
- 2. **メモリ減衰メカニズム**：高エントロピートークンを保持し、重要でないトークンをフィルタリングします。このプロセスは、記憶キャッシュを効果的に管理し、重要な情報のみを保持します。

結果

効果を評価するために、以下のタスクで実験を行いました：

- 1. **日常対話 (DailyDialog)**：日常的なマルチターン対話シナリオでの会話の一貫性と記憶能力を評価しました。
- 2. **買い物 (Grocery Shopping)**：買い物リストを記憶し、20ターンのQAセッション後にリストを正確に思い出す能力を評価しました。
- 3. **じゃんけん (Rock-Paper-Scissors)**：複数回のじゃんけんゲームでユーザーの投げる傾向を分析し、勝率を最大化する能力を評価しました。



CONTRANS: Weak-to-Strong Alignment Engineering via Concept Transplantation CONTRANS: コンセプト移植による弱から強へのアライメントエンジニアリング 2024

概要

LLMが長い会話でも記憶を維持するために、重要な言葉を覚える方法、事前学習なしで無限長の対話中にLLMが長期記憶を維持できる「Streaming Infinite Retentive LLM(SirLLM)」を使った方法で効果を確認
日常対話、買い物、じゃんけんの3つの異なるタスクを設計し、これらのタスクの効果を測定するためのデータセットを構築SirLLMが安定して結果を出しました

<https://github.com/Zoeyyao27/SirLLM>

手法

SirLLMは、LLMが無限長の対話中に長期記憶を維持できるようにするためのメカニズムを導入しています。主な手法は以下の通りです：

- **トークンエントロピーの計算****: 各入力トークンのエントロピーを計算し、その重要度を評価します。
- **メモリ減衰メカニズム****: 高エントロピートークンを保持し、重要でないトークンをフィルタリングします。このプロセスは、記憶キャッシュを効果的に管理し、重要な情報のみを保持します。

結果

効果を評価するために、以下のタスクで実験を行いました：

- **日常対話(DailyDialog)****: 日常的なマルチターン対話シナリオでの会話の一貫性と記憶能力を評価しました。
- **買い物(Grocery Shopping)****: 買い物リストを記憶し、20ターンのQAセッション後にリストを正確に思い出す能力を評価しました。
- **じゃんけん(Rock-Paper-Scissors)****: 複数回のじゃんけんゲームでユーザーの投げる傾向を分析し、勝率を最大化する能力を評価しました。

FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research

FlashRAG: 効率的な検索補強生成研究のためのモジュラーキット 2024

概要

既存のRAGツールキット(LangChainやLlamaIndexなど)は存在するものの、重くて扱いにくく、研究者の個別のニーズに対応できていないのでFlashRAGというソースツールを作成、カスタマイズ可能なコンポーネントと豊富なデータセットを使って、既存のRAG方法を再現しています
<https://github.com/RUC-NLPIR/FlashRAG>

手法

FlashRAGは、以下の4つの主要な機能を提供します。

- **拡張可能でカスタマイズ可能なモジュラーRAGフレームワーク****
 - 13の主要なコンポーネント(judge, retriever, refiner, generator)を含むモジュールを提供し、これらを組み合わせでRAGプロセスを形成します。これにより、既存の方法を簡単に再現し、異なる設定で評価できます。
- **事前実装された高度なRAGアルゴリズム****
 - 12の先進的なRAGアルゴリズムを実装し、統一された設定で評価。研究者が自身の方法と公平に比較できます。
- **包括的なベンチマークデータセット****
 - 32の一般的なRAGベンチマークデータセットを収集し、一貫したフォーマットに前処理。これらのデータセットはHugging Faceプラットフォームで利用可能です。
- **効率的な補助スクリプト****
 - ウィキペディアのダウンロードと切り出し、インデックス作成、検索結果の準備などRAG実験の準備段階を効率化するためのスクリプトを提供。

結果

LLAMA3-8B-instructを生成モデルとして使用し、E5-base-v2を検索モデルとして使用。2018年12月のウィキペディアデータを検索コーパスとして利用。自然質問(NQ)、TriviaQA、HotpotQA、2WikiMultihopQA、PopQA、WebQuestionsの6つの一般的なデータセットで実験を実施。

"Turing Tests" For An AI Scientist AI科学者のための「チューリングテスト」2024

概要

LLMエージェントが人間の知識に頼らず独立して科学研究を行うためのチューリングテストの提案7つのベンチマークテストで発見能力を評価

手法

AIエージェントが科学的発見を行うための能力を評価するために、以下7つのベンチマークテストを提案しています。

- **地動説のモデル****: 天体観測データからケプラーの3法則を推測し、惑星が太陽を中心に回ることを導き出す。
- **運動の法則****: Minecraftのインタラクティブライブラリを使って、慣性の法則や重力による加速度の法則を発見する。
- **振動する弦****: 振動する弦の位置データから、その運動を支配する微分方程式を推測する。
- **マクスウェル方程式****: 電磁気学シミュレーションからマクスウェルの方程式を導出する。
- **初期値問題****: 初期値問題の数値解法を発明し、4次のルンゲ・クッタ法と同等以上の精度を達成する。
- **ハフマン符号化****: ASCII文字列データから、ストレージ最適化のためのハフマン符号化を発見する。
- **ソートアルゴリズム****: 整数配列をソートするアルゴリズムを発見し、 $O(n \log n)$ の時間複雑性を達成する。

結果

この論文の評価指標は、AIエージェントが提案された7つのベンチマークテストをどれだけ成功裏に達成できるかに基づいています。これらのテストは、歴史的に重要な科学的発見に対応しており、それぞれが特定の科学分野での画期的な発見を評価します。

概要

長文応答の正確性の向上は、ハルシネーションをフィルタリングすること以外に質問に関連する複数の情報の再現率にも依存します。この質問に関連する情報の再現率を改善する為に「アトミックセルフコンシステンシー(ASC)」を紹介

ASCは、複数の確率的サンプルを使用して長文応答を改善する「ユニバーサルセルフコンシステンシー(SC)」の最近の研究に基づいていますが、USCが単一の最良生成を選択することに重点を置いているのに対しASCはサンプルから本物の部分を選択し、それらを統合してより優れた複合応答を生成します。

手法

ASC は、以下の手順で応答を生成します：

- 1. ****分割****: 各生成を原子的事実分割。
- 2. ****クラスタリング****: 原子的事実をグループ化。
- 3. ****フィルタリング****: 一貫性に基づいて最良のクラスターを選択。
- 4. ****要約****: 選択されたクラスターの代表を統合して最終的な応答を生成。

サンプルプロンプト

```
{task instruction}
Question: {...}
Sentence1: {...}
Sentence2: {...}
...
Answer: {...}
{task instruction}
Question: {...}
Sentence1: {...}
Sentence2: {...}
...
Answer: {...}
{task instruction}
Question: { ...}
Sentence1: {...}
Sentence2: {...}
```

Answer:

Presentations are not always linear! GNN meets LLM for Document-to-Presentation Transformation with Attribution

GNNがLLMに出会う:アトリビューションによる文書からプレゼンテーションへの変換 2024

概要

長い文書のテキストからプレゼンテーションを自動生成する為にはフラットな要約とは対照的に、より良い非線形のアトリビューション、生成されたプレゼンテーションの各部分(スライド)が元の文書のどの部分から来ているかを明示する能力を持つ必要があります。つまり、スライドの内容は文書の異なる部分から取られますが、内容が文書に忠実であることを保証するのは困難です。またMIはハルシネーションを起こしやすく、入力文書の長さが増えると性能が低下します。この改善のために、入力文書からグラフを学び、グラフニューラルネットワーク(GNN)とLLMの組み合わせを用いて、各スライドの内容の出典を明示するプレゼンテーションを生成する新しいグラフベースの手法を提案

手法

1. 文書と対応するプレゼンテーションスライドのセットから成る訓練データセットを与えられ、テストセットに対してプレゼンテーションを生成します。
2. ****解決策アプローチ****
 - 文書からテキストを抽出し、段落ごとにノードを作成してグラフを構築します。
 - 訓練された分類器を使用して、段落ペアが同じスライドに属する確率を予測します。
 - グラフニューラルネットワークを用いて、ノードを埋め込み空間にマッピングし、スペクトルクラスタリングを用いてクラスタリングを行います。
 - 各クラスターがスライドに対応するようにし、GPT-3.5を使用してスライドのタイトルと内容を生成します。

結果

ROUGE-1、Coverage、Perplexity(PPL)、G-Evalなどの自動評価指標を実施する形で評価を実施

From Human-to-Human to Human-to-Bot Conversations in Software Engineering

ソフトウェア工学における人間同士の会話から人間とボットの会話へ 2024

概要

ソフトウェア開発で人とチャットボットの対話を比較するためにLLMを使って実際の対話データを分析。結果LLMは人間の会話を完全には代替できないが、開発者の負担を減らし仕事の効率を上げていることを確認する検証を実施しています。人間とボット (NLU-およびLLMベース) の対話の類似点と相違点を提示し、最終的にLLMチャットボットの最近の会話スタイルは、生産性をサポートし開発者の精神的負荷を軽減する能力がある一方で、社会的側面に関しては人間との対話を完全替することはできないという結果

手法

- 1. **データ収集と分析**:
 - 24名のソフトウェアエンジニアが週間にわたりChatGPTを使用し、そのチャットログとエグジジットサーベイの結果を収集しました。
 - 180の対話と580のプロンプトを質的分析し、対話の性質、目的、フロー、その他の属性を解釈現象学的分析(1)を用いて分析しました。
- 2. **比較フレームワーク**:
 - Clarkらのフレームワークを基に、人間同士、人間とNLUベースのチャットボット、人間とLLMベースのチャットボットの対話を以下の属性で比較しました
 - 目的
 - 理解の範囲
 - 傾聴
 - 信頼性
 - ユーモアの使用

結果

- **目的**:
 - 人間同士の対話: 社交的な目的、一般的なガイダンス、トレーニング。
 - NLUベースのチャットボット: 基本的な情報取得、シンプルな自動化。
 - LLMベースのチャットボット: 一般的なガイダンス、トレーニング、アーティファクトの生成および操作。
- **理解の範囲**:
 - 人間同士: 相互理解に基づく。
 - NLUベース: 固定的なカスタマイズ。
 - LLMベース: 動的なカスタマイズ。
- **傾聴**:
 - 人間同士: ボディランゲージと知識による積極的かつ正確な傾聴。
 - NLUベース: 意図の分類とエンティティ抽出による積極的かつ正確な傾聴。
 - LLMベース: クエリの要約と知識による積極的かつ正確な傾聴。
- **信頼性**:
 - 人間同士: 共有された経験と以前の対話に基づく。
 - NLUベース: パフォーマンスと効率。
 - LLMベース: 期待に応じた結果と透明性。
- **ユーモアの使用**:
 - 人間同士: 一般的に使用される。
 - NLUベース: 該当しない。
 - LLMベース: 要求に応じて使用。

MapCoder: Multi-Agent Code Generation for Competitive Problem Solving

マップコーダー：競技プログラミングのためのマルチエージェントコード生成 2024

概要

競技プログラミングの問題解決のため、MapCoderという方法で例を探し計画を立てコードを生成しデバッグするエージェントを提案
<https://github.com/Md-Ashraful-Pramanik/MapCoder>

手法

MapCoderフレームワークは、4つのLLMエージェントから構成されています。それぞれのエージェントの役割は次の通りです：

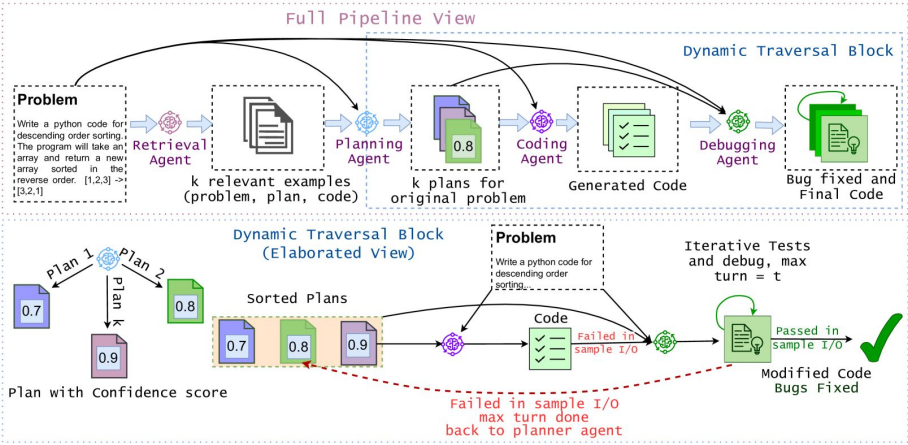
- リトリバーエージェント**：
 - 過去の関連する問題解決インスタンスを再現します。具体的には、同様の問題を見つけ、それらの解決策を生成し、それに関する追加のメタデータを提供します。
- プランニングエージェント**：
 - オリジナルの問題に対するステップバイステップの計画を作成します。リトリバーエージェントから得られた例とその計画を使用し、問題解決の具体的な計画を生成します。
- コーディングエージェント**：
 - プランニングエージェントからの計画を元に、問題解決のためのコードを生成します。コードはサンプルI/Oでテストされ、すべてのテストをパスした場合、最終解として返されます。
- デバッグエージェント**：
 - サンプルI/Oを使用して生成されたコードのバグを修正します。このエージェントはプランニングエージェントからの計画を参照し、コードの修正を行います。

結果

主な結果は以下の通りです：

- **HumanEval****: 93.9%
- **MBPP****: 83.1%
- **APPS****: 22.0%
- **CodeContests****: 28.5%
- **xCodeEval****: 45.3%

特に、複雑な問題に対する多段階の推論やバグ修正が必要な場合に有効です。



MapCoder: Multi-Agent Code Generation for Competitive Problem Solving

マップコーダー: 競技プログラミングのためのマルチエージェントコード生成 2024

Retrieval Agent

Given a problem, provide relevant problems then identify the algorithm behind it and also explain the tutorial of the algorithm.

Problem:
(Problem Description will be added here)

Exemplars:
Recall k relevant and distinct problems (different from problem mentioned above). For each problem,
1. describe it
2. generate (language) code step by step to solve that problem
3. finally generate a planning to solve that problem

Algorithm:

Important:
Your response must follow the following xml format-

Recall k relevant and distinct problems (different from problem mentioned above). Write each problem in the following format.
Describe the problem.

...

Let's think step by step to solve this problem in (language) programming language.

...

Planning to solve this problem.

similarly add more problems here...

Identify the algorithm (Brute-force, Dynamic Programming, Divide-and-conquer, Greedy, Backtracking, Recursive, Binary search, and so on) that needs to be used to solve the original problem.
Write a useful tutorial about the above mentioned algorithms. Provide a high level generic tutorial for solving this types of problem. Do not generate code.

問題が与えられた場合、関連する問題を提供し、その背後にあるアルゴリズムを特定し、アルゴリズムのチュートリアルも説明してください。
問題:

{ここに問題の説明が追加されます}

例題:

k個の関連する別々の問題をリコールしてください(上記の問題とは異なる)。各問題について、

1. 問題を説明する
2. {言語}でその問題を解決するコードをステップバイステップで生成する
3. 最後にその問題を解決するための計画を立てる

アルゴリズム:

重要:
あなたの回答は次のxml形式に従う必要があります-

<root>
<problem>

**k個の関連する別々の問題をリコールしてください(上記の問題とは異なる)。各問題を次の形式で書いてください。
**

<description> # 問題を説明する。 </description>
<code> # この問題を{言語}プログラミング言語で解決するためにステップバイステップで考えましょう。 </code>
<planning> # この問題を解決するための計画。 </planning>
</problem>

ここに同様の形式で他の問題を追加してください...

<algorithm>
元の問題を解決するために使用する必要があるアルゴリズム(ブルートフォース、動的計画法、分割統治法、貪欲法、バックトラック、再帰、二分探索など)を特定してください。
上記のアルゴリズムについて有用なチュートリアルを書いてください。これらのタイプの問題を解決するための高レベルの一般的なチュートリアルを提供してください。コードは生成しないでください。
</algorithm>
</root>

MapCoder: Multi-Agent Code Generation for Competitive Problem Solving

マップコーダー: 競技プログラミングのためのマルチエージェントコード生成 2024

Planning Agent

Given a competitive programming problem
generate a concrete planning to solve the
problem.
Problem: {Description of the example problem}
Planning: {Planning of the example problem}
Relevant Algorithm to solve the next
problem:
{Algorithm retrieved by Retrieval Agent}
Problem to be solved: {Original Problem}
Sample Input/Outputs: {Sample IOs}

Important: You should give only the planning to
solve the problem. Do not add extra explanation
or words.

Confidence Generation Prompt:
Given a competitive programming problem and a plan to solve
the problem in {language} tell whether the plan is correct to
solve this problem.
Problem: {Original Problem}
Planning: {Planning of our problem from previous step}

Important: Your response must follow the following xml format-
<root>
<explanation> Discuss whether the given competitive
programming problem is solvable by using the above mentioned
planning. </explanation>
<confidence> Confidence score regarding the solvability of
the problem. Must be an integer between 0 and 100.
</confidence>
</root>

競技プログラミングの問題が与えられた場合、その問題を解決するための具体的な計画を作成してください

問題: {例題の説明}
計画: {例題の計画}
次の問題を解決するための関連アルゴリズム :
{リトリバーエージェントから取得されたアルゴリズム }

解決すべき問題: {オリジナルの問題}
サンプル入出力: {サンプル入出力}

重要: 問題を解決するための計画のみを記載してください。追加の説明や言葉は不要です。

信頼度生成プロンプト:
競技プログラミングの問題とその問題を解決するための計画が与えられた場合、その計画が問題を解決するのに適しているかどうかを評価してください。

問題: {オリジナルの問題}
計画: {前のステップでの問題の計画}
重要: あなたの回答は次の xml形式に従う必要があります -
<root>
<explanation> 与えられた競技プログラミングの問題が上記の計画を使用して解決可能かどうかを議論してください。 </explanation>
<confidence> 問題の解決可能性に関する信頼度スコア。 0から100の整数である必要があります。 </confidence>

</root>

MapCoder: Multi-Agent Code Generation for Competitive Problem Solving
マップコーダー: 競技プログラミングのためのマルチエージェントコード生成 2024

Coding Agent

Given a competitive programming problem generate
Python3 code to solve the problem.
Relevant Algorithm to solve the next problem:
{Algorithm retrieved by Retrieval Agent}
Problem to be solved:
{Our Problem Description will be added here}
Planning: {Planning from the Planning Agent}
Sample Input/Outputs: {Sample I/Os}
Let's think step by step.

Important:
Your response must contain only the {language} code
to solve this problem. Do not add extra explanation or
words.

競技プログラミングの問題が与えられた場合、その問題を解決するための Python3コードを生成してください。

次の問題を解決するための関連アルゴリズム :
{リトリバーエージェントから取得されたアルゴリズム }

解決すべき問題:
{ここに我々の問題の説明が追加されます }

計画:{プランニングエージェントからの計画 }
サンプル入出力:{サンプル入出力}
ステップバイステップで考えましょう。

重要:

あなたの回答には、この問題を解決するための {言語}コードのみを含めてください。追加の説明や言葉は不要です。

MapCoder: Multi-Agent Code Generation for Competitive Problem Solving
マップコーダー: 競技プログラミングのためのマルチエージェントコード生成 2024

Debugging Agent

Given a competitive programming problem you have generated {language} code to solve the problem. But the generated code cannot pass sample test cases. Improve your code to solve the problem correctly.

Relevant Algorithm to solve the next problem:
{Algorithm retrieved by Retrieval Agent}

Planning: {Planning from previous step}

Code: {Generated code from previous step}

Modified Planning:

Let's think step by step to modify {language} Code for solving this problem.

Important:

Your response must contain the modified planning and then the {language} code inside ``` block to solve this problem.

競技プログラミングの問題が与えられ、その問題を解決するために {言語}コードを生成しました。しかし、生成されたコードはサンプルテストケースを通過できません。コードを改善して、問題を正しく解決してください。

次の問題を解決するための関連アルゴリズム：
{リトリバルエージェントから取得されたアルゴリズム }

計画: {前のステップからの計画 }
コード: {前のステップで生成されたコード }
修正後の計画:
ステップバイステップで {言語}コードを修正してこの問題を解決しましょう。
重要:

あなたの回答には、修正後の計画と次に ```ブロック内にこの問題を解決するための {言語}コードを含める必要があります。

Self-Reflection in LLM Agents: Effects on Problem-Solving Performance

LLMエージェントにおける自己反省: 問題解決能力への影響 2024

概要

LLMエージェントの問題解決力向上のため、自己反省という方法で間違いを振り返り、改善策を考えて再挑戦させています。
https://github.com/matthewrenze/self-reflection

手法

- データセット**
 - ARC, AGIEval, HellaSwag, MedMCQAなどの人気LMベンチマークから取得した多肢選択問題(MCQA)を使用し、1,000問の多領域試験を作成。
- モデル**
 - GPT-4, Llama 2 70B, Google Geminiなど、9つの人気LMを評価。
- エージェント**
 - 自己反省を行う5種類のエージェント(Retry, Keywords, Advice, Explanation, Instructions, Solution, Composite, Unredacted)と、自己反省を行わない基準エージェント(Baseline)を使用。
- プロセス**
 - 基準エージェントが全ての質問に回答し、誤答した問題に対して自己反省エージェントが自己反省を行い、その後再度問題に挑戦。

自己反省エージェント

自己反省のプロセスは以下の手順で行われます:

- 初回回答**:
 - エージェントが与えられた問題に回答します。
- 誤答の識別**:
 - エージェントの回答が誤っていた場合、その問題が誤答リストに追加されます。
- 自己反省の実行**:
 - 誤答リストに追加された問題について、エージェントが自己反省を行います。
 - 自己反省では、エージェントが以下の情報を生成します:
 - 説明**: 誤答の原因について説明します。
 - キーワード**: 誤答のタイプを示すキーワードをリストアップします。
 - 解決策**: 問題を再度解くためのステップバイステップの解決策を示します。
 - 指示**: 将来的に同じ誤答を避けるための詳細な指示を提供します。
 - アドバイス**: 類似の問題に対処するための一般的なアドバイスを提供します。
- 再回答**:
 - エージェントが自己反省から得た情報をもとに、同じ問題に再挑戦します。
- 結果の分析**:
 - エージェントの再回答の正答率を計算し、基準エージェントのスコアと比較して性能向上を評価します。

Appendix
