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ウトラインの合成を行います。
https://github.com/stanford-oval/storm

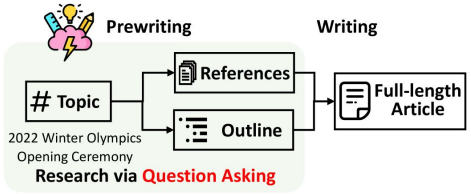
手法

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。

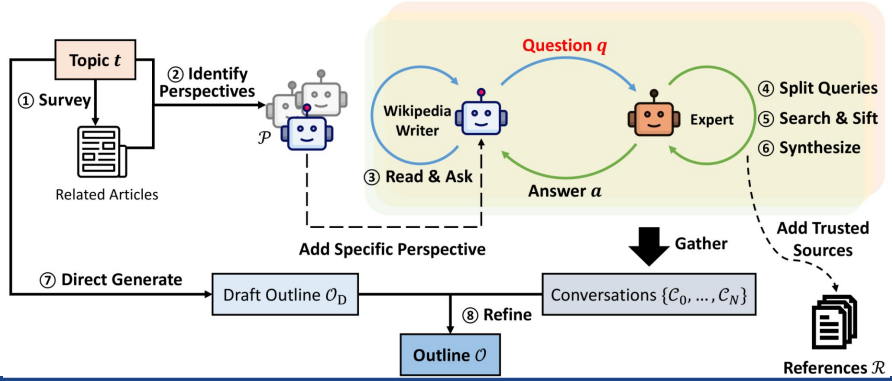
2. 質問のシミュレーション：トピック専門家に仮想的に質問を投げかけ、回答を受け取ることでさらなる質問を引き出します。

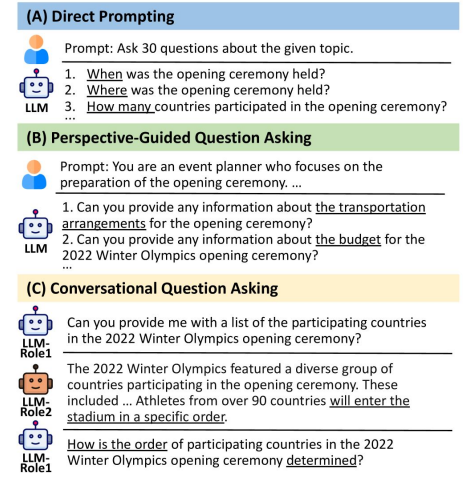
3. 情報のキュレーション：収集した情報を整理し、記事のアウトラインを作成します。



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





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

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。

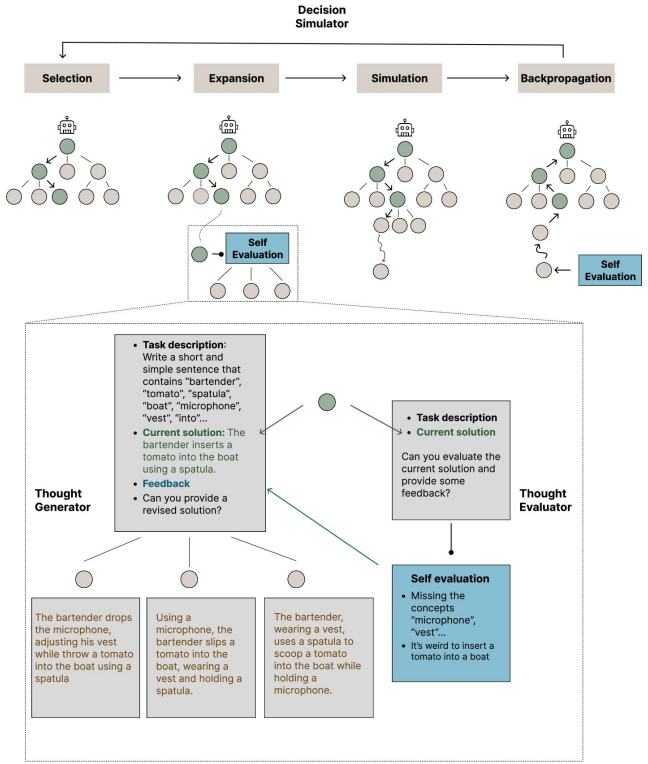
手法

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、それぞれの検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
検索拡張で使えそう

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

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 = """\
# タスクの説明
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 + """\
# 元の概要
{ outline }
# フィードバック
{ feedback }


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:
[1] ...
[2] ...
...


# あなたの回答:
"""


EVALUATE_CURRENT = TASK_DESCRIPTION + """\
# 元の概要
{ outline }


この概要は十分だと思いますか？
1から100までのスコアを書いてください。ここで100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
[スコア: 1-100] [理由] xxx (最大50語)


# 例:
[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:
"""
```

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 = """\
# タスクの説明
5x5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 + """\
# 現在のボード:
{obs}
# 戦略:
{ feedback }

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 + """\
# 現在のボード:
{obs}
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; g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:
"""
```

A.3 タスク3 制約付き生成

```
TASK_DESCRIPTION = """\
# 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 = """
あなたのタスク - 概念: {concepts}
"""

NEW_CANDIDATE = TASK_DESCRIPTION + """\
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 + """\
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：1) 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2) 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30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。

1. 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30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10のインスタンスが用意されました。
2. 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。
3. 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。
4. 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。

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、'{'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）:
1. i = 1, 初期設定からコピー, P[1] = '('; Stack_1 = []。
2. P[1]のタイプは'('で、左括弧です。Stack_2を[('(', 1)] + Stack_1 = [('(', 1)] + [] = [('(', 1)]としてプッシュします。
3. iを1増やします。i = 2。
4. P[2] = ')'; Stack_2 = [('(', 1)]。
5. 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。
6. 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。

Appendix