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。

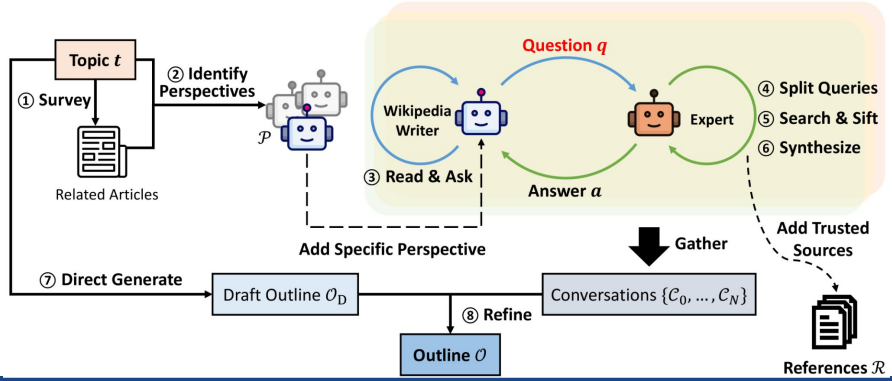
1. 多様な視点の発見：トピックに関連する情報を収集し、異なる視点からの質問を可能にします。

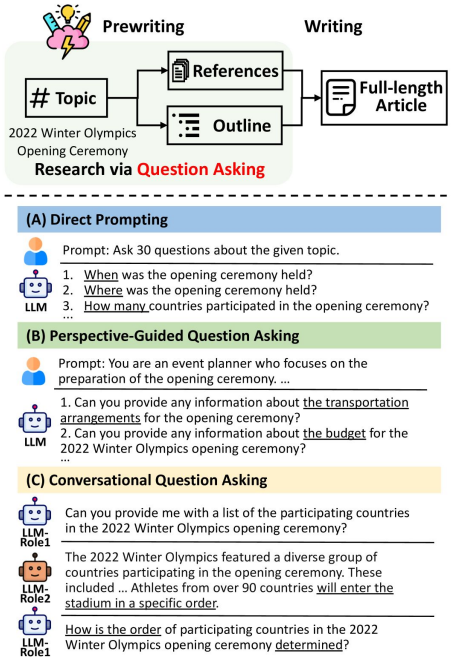
2. 質問のシミュレーション：トピック専門家に仮想的に質問を投げかけ、回答を受け取ることでさらなる質問を引き出します。

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。
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。

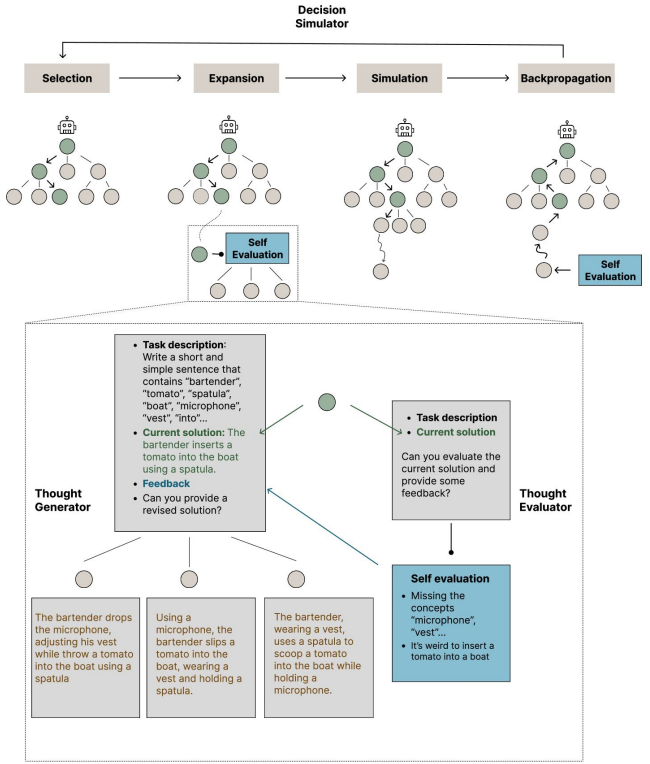
1. 思考評価者 (Thought Evaluator): 思考評価者は、生成された各思考ノードの状態を評価し、
改善のためのフィードバックを提供します。これには、数値フィードバックとテキストフィードバックの両方が含まれます。
数値フィードバックは探索アルゴリズムによる評価スコアとして利用され、
テキストフィードバックは子ノードを生成する際のコンテキストとして活用されます。
思考評価者は、ホリスティック評価(全体を一貫して評価する方法)とアイテム化評価(各部分を個別に評価する方法)の
2種類のフィードバック戦略を提供し、タスクシナリオに応じて適用されます。

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



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:
"""
```

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; 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:
"""
```

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。

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。

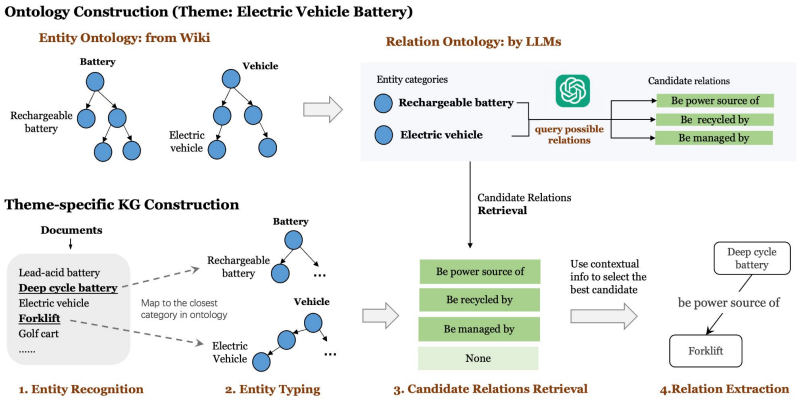
このプロセス全体を通じて、TKGConはテーマに特有な詳細情報を持つナレッジグラフを構築し、既存の一般的なナレッジグラフではカバーされていないような精緻で時宜にかなった情報を提供します。

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KGを構築できます。

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ロシネーション率(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:
1. 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。
2. 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。
3. 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。
1. 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。
2. 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。
3. 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

FIT-RAG: Black-Box RAG with Factual Information and Token Reduction FIT-RAG: 情報とトークンの削減を伴うブラックボックスRAG 2024

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かをHas_Answer（事実情報ラベル）とLM_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LM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：
1. 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。
2. バイラベル評価によるトークンの削減：選択した文章を分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LMの反応を分析することが含まれます。
3. 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るようです。

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ICLと少数ショットICLで比較し、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。

Improve Temporal Awareness of LLMs for Sequential Recommendation

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た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L3X（LM-based Long List eXtraction）という新しい手法を使い、再現率指向の生成ト精度指向の性差の二段階でアプローチを行います

手法

L3X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LM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。
1. 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、初期リストを生成します。
2. 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500のパッセージまでを取得し、最適なものを選択してLLMにフィードします。
3. 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LLM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。
1. 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。
2. 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Recall@PrecisionX（R@Px）を用いて評価されます。
Recall@PrecisionX（R@Px）は特定の精度（例えば50%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、その計算は精度が少なくとも50%に達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:
1. 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。
2. 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。
3. 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。
4. 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。
5. 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L3X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30%、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RoBERTaなどの事前学習済みモデル）を通じて、まず知識表現z_{kt}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_{kt}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z_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z_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精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LM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。

ERATTA: Extreme RAG for Table To Answers with Large Language Models ERATTA: 大規模言語モデルを用いた表からの回答生成のための極端なRAG 2024

概要

データテーブル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0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5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SQL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JSON/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一致を試みます。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SQL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SQL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LM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、数値データの事実的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の変化が、 Prompt 3の回答で正確に記述されているかを確認します。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LM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PLM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PLM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PLM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、タスクに必要ながら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

AttacKG+: Boosting Attack Knowledge Graph Construction with Large Language Models
AttacKG+: 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LMを使用したAttacKG+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

AttacKG+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：

1. リライター（Rewriter）: CTI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。
CTI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。

2. 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

3. 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。

4.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LMで明確化し、各戦術段階の要約を出力します

結果

AttacKG+は234の技術と14の戦術からなるMITREと500のCTIレポートを使用して評価された。
結果は、AttacKG+が既存のCTI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。

AttacKG+: 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

AttacKG+の視覚化の特徴

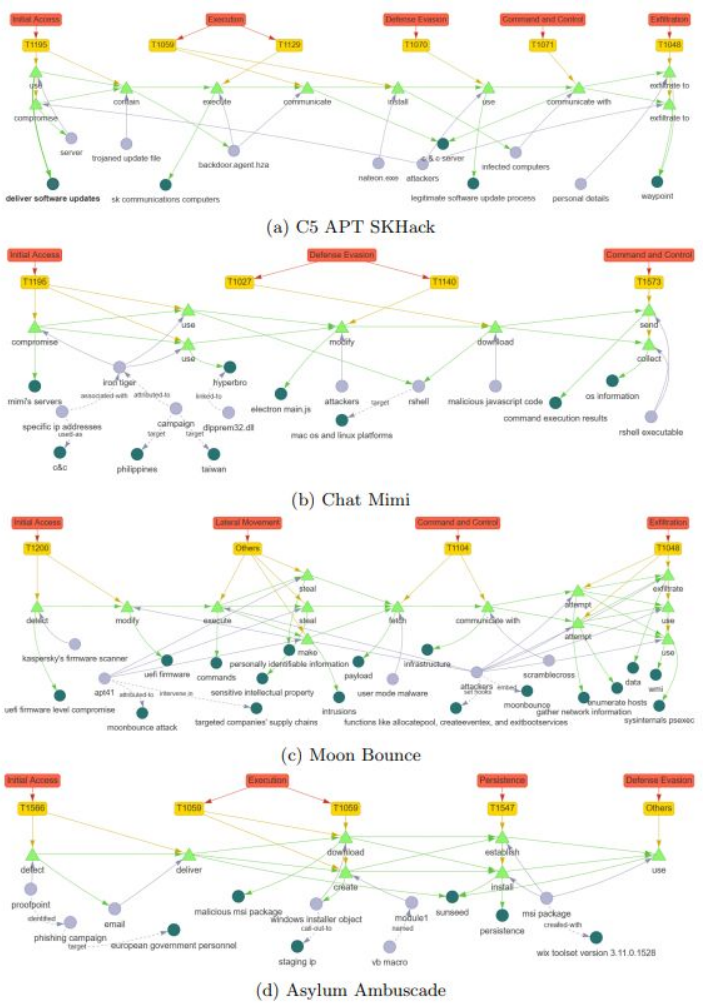
1. 三層の表現: 行動グラフ(Behavior Graph): 攻撃の各ステップをノードとして表現し、
攻撃者の行動の流れや相互作用を示します。
これには、攻撃者がどのような行動をとったか、どの順序で行動が発生したかが含まれます。
TTPラベル(Tactics, Techniques, and Procedures Labels): 各行動に関連する戦術や技術をラベル付けし、
それぞれの行動がどのような攻撃技術や戦術に対応するかを明確にします。
状態サマリー(State Summary): 攻撃の各段階での重要な変化や成果を要約し、
攻撃がどのように進展しているかの概要を提供します。

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



(a) C5 APT SKHack

(b) Chat Mimi

(c) Moon Bounce

(d) Asylum Ambuscade

Fig. 5: Examples of the visualization of AttacKG+

LLMs Can Patch Up Missing Relevance Judgments in Evaluation LLMは評価における欠損している関連性判断を補完　2024

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ルースと強い相関が発現しました。

手法

TREC DLト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のカテゴリーで回答してください。最後の行には、関連性のカテゴリー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。
関連性カテゴリー : 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　...
関連性カテゴリー : 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。これは公正で巧妙な要約です　...
関連性カテゴリー : 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　...
関連性カテゴリー : 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)を組み合わせて、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、それを基にして進化する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↓LM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。

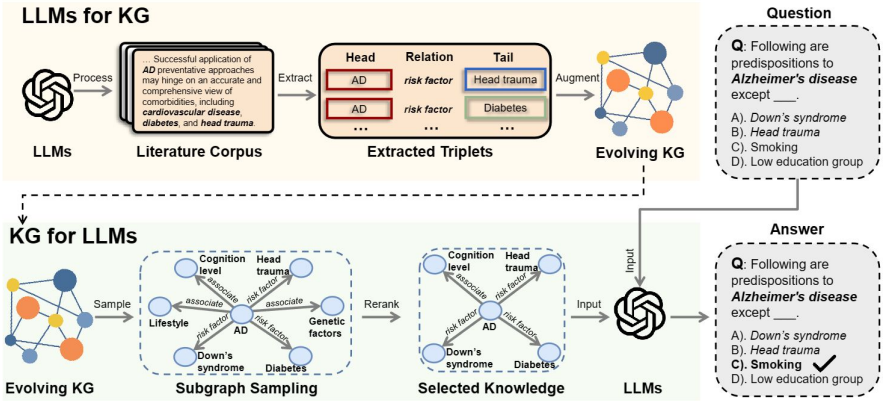
1. AD特有の知識グラフの構築
データ収集: アルツハイマー病に関連する科学文献からデータを収集します。
エンティティと関係の抽出 収集したデータから重要なエンティティ間の関係を特定し、それらを抽出します。関係は「治療する」「引き起こす」などの動詞や関連性のある表現を抽出します。
知識グラフの構築 抽出したエンティティと関係を基に、AD特有の知識グラフをノード(エンティティ)とエッジ(関係)で構築します。

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LM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求しています。
1. 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。
2. これら人間による評価とLM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の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：

1. どの答えをサポートしていますか？（a, b, c, d, e, なし）
2. 総合評価は？（1, 2, 3, 4, 5）
3. よく書かれていますか？（いいえ、はい）
4. 説明は質問と回答に関連していますか？（いいえ、はい）
5. 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）
6. 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）
7. 不必要な情報はありますか？（いいえ、はい）
8. 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：

1. 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ー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。

2. 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。これらのギストは、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。

3. 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、そのページのギスト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は関連するギストを参照し、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

A Human-Inspired Reading Agent with Gist Memory of Very Long Contexts 2024

プロンプト

https://github.com/read-agent/read-agent.github.io/blob/main/assets/read_agent_demo.ipynb

#@title ReadAgent (1) エピソードページネーション

```
prompt_pagination_template = """
あなたには、より大きなテキスト（記事、本など）から抜粋されたー節と、そのー節の段落間に挿入された番号付きのラベルが与えられます。
番号付きのラベルは角括弧で示されます。例えば、ラベル番号が19の場合、テキスト中で<19>と表示されます。
自然に読みを中断するラベルを一つ選んでください。
中断点は、シーンの変わり目、対話の終わり、議論の終わり、物語の遷移などがあります。
中断点のラベルとその理由を答えてください。
例えば、<57>が中断に適していれば、「中断点 <57>\n 理由は...\」と答えてください。

ー節:
{0}
{1}
{2}

"""
```

#@title ReadAgent (2) メモリ要約

```
prompt_shorten_template = """
以下のー節を短く要約してください。
理由は説明しないでください。

ー節:
{}

"""
```

#@title ReadAgent (3) ルックアップ

```
prompt_lookup_template = """
以下のテキストは、あなたが記事を読んで覚えている内容とそれに関連する選択式の質問です。
質問に備えて記憶を新たにするために、記事のページから6ページまでを再度読むことができます。
どのページを読みたいかを回答してください。
例えば、ページ8だけを読む必要がある場合は「ページ[8]を見直したいです」と回答してください；
ページ7と12を読みたい場合は「ページ[7, 12]を見直したいです」と回答してください；
ページ2, 3, 7, 15, 18を読みたい場合は「ページ[2, 3, 7, 15, 18]を見直したいです」と回答してください。
ページ3, 4, 5, 12, 13, 16を読みたい場合は「ページ[3, 3, 4, 12, 13, 16]を見直したいです」と回答してください。
必要以上に多くのページを選択しないでください。
まだ質問には答えないでください。

テキスト:
{}
質問:
{}
{}

深呼吸して、どのページを再度読みたいか教えて？

"""

prompt_answer_template = """
以下の記事を読んで、選択式の質問に答えてください。
例えば、(C)が正解の場合は「回答 (C) ...」と答えてください。

記事:
{}

質問:
{}
{}

"""
```

Appendix