論文要約

LLM関連

How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning 2024

概要

LLMステップバイステップの推論をどのように実現しているかを、機械的な観点から調査
LLMsがステップバイステップ推論のために複数の並行する解答生成パスを展開していることを示します。これらのパスは、入力質問のコンテキストと生成された思考の連鎖から順次的に解答を提供しますLLMの中間層において顕著な機能的断裂を観察します。初期の半分のトークン表現は、訓練前の事前知識に強く偏っており、後半では突然コンテキストが支配的になります。この内部のフェーズシフトは、答えのトークンを主に書き込む注意ヘッドが後半に現れ、オントロジカルな関係に沿って情報を移動する注意ヘッドが初期半分にのみ現れるなど、異なる機能コンポーネントに現れます。我々の知識によると、これはLLMsにおける思考の連鎖推論の機械的調査における最初の試みです。

手法

LLaMA-2 7Bを用いて、架空のオントロジーに基づく多段階推論におけるCoTの生成を分析します。特に、PrOntoQAデータセットを使用して、モデルが提供されたコンテキストから推論することに焦点を当てます。

ステップバイステップ推論の実現を確認する方法として、以下の主な手法が用いられました：
1. サブタスクごとの回答生成の調査：LLM内の注目ヘッドが、特定のサブタスクに対する回答を最終的な残差ストリームに直接書き込むことにより、ステップバイステップでの回答を生成する過程を探りました。これにより、複数の回答生成パスが存在し、それぞれが生成された出力を強化することが示されました
2. 情報の伝播経路の追跡：回答を直接書き込む注目ヘッドから始めて、これらのヘッドが注目している残差ストリームを特定し、その残差ストリームの内容を逆埋め込み投影を通じて特定しました。これにより、回答を書き込むヘッドに根ざした注目ヘッドのツリーを構築し、ステップバイステップの答えの生成に至る情報の流れを追跡しました
3. コンテキストに基づく回答の収集：生成されたコンテキスト、質問コンテキスト、および数ショットコンテキストから回答を収集する注目ヘッドを調査しました。これにより、異なるサブタスクで回答トークンを異なるセグメントから収集する複数のパラレルパスウェイが明らかになりました

これらの手法により、LLMがステップバイステップ推論を実現するために、複数のパラレルパスウェイを利用し、それぞれが異なるメカニズムを通じて入力から回答を処理していることが確認されました。

結果

LLMsがステップバイステップ推論のために複数の並行する解答生成パスを展開していること、および中間層における機能的断裂の存在が確認されました。

Rethinking the Bounds of LLM Reasoning: Are Multi-Agent Discussions the Key? 2024

概要

マルチエージェントディスカッションがLLMの推論能力を向上させることを示唆しています。
この主張を系統的な実験を通じて再評価し、新しいグループディスカッションフレームワークを提案して、議論メカニズムのセットを豊かにします。
結果は強力なプロンプトを使用した単一エージェントLLMが、幅広い推論タスクとバックボーンLLMで既存のディスカッションアプローチとほぼ同等のパフォーマンスを達成しました。
マルチエージェントディスカッションが単一エージェントを上回るのは、プロンプトにデモンストレーションがない場合のみでした。

手法

グループディスカッションフレームワーク（CMD: Conquer-and-Merge Discussion Framework）については、複数のエージェントが議論を行うことにより、推論能力を向上させることを目的とした新しいフレームワークです。このフレームワークは、特に多数のエージェントと議論ラウンドの数が増加すると、パフォーマンスが向上するという既存のDebateフレームワークやReConcileフレームワークの主張に基づいていますが、エージェントの数が増えることによるオーバーヘッドの増加に対処するために考案されました。
CMDフレームワークは、以下の3つのステージに分かれています：
Stage 1: Group Discussion Initialization (グループディスカッションの初期化): このステージでは、全ての参加エージェントが初期化され、3つのグループに分けられます。各エージェントは初回のレスポンスを生成します。
Stage 2: Multi Rounds Discussion (複数ラウンドの議論): アクティブなエージェントは、残りのラウンドで議論を続けます。各ラウンドで、エージェントは以前のレスポンスに基づいて新しいレスポンスを生成します。
Stage 3: Vote for the Final Result (最終結果のための投票): このステージでは、全てのエージェントが最終的な意見に基づいて投票を行い、最終結果を決定します。タイが発生した場合、追加の「秘書」エージェントが最終決定を行います。
CMDフレームワークは、エージェント間での情報の受け渡しを効率化するために、メッセージパッシングアルゴリズムを用いており、各ステージでのエージェントの行動や投票プロセスを通じて、議論の結果を最終決定します。このフレームワークは、エージェントの数が増えることによる計算コストの増加に対処しつつ、エージェント間の議論を通じて推論能力を向上させることを目指しています。

結果

強力なプロンプトを使用した単一エージェントLLMがマルチエージェントディスカッションとほぼ同等のパフォーマンスを示しました。特に、プロンプトにデモンストレーションがない場合にマルチエージェントディスカッションが優れていることが確認されました。

Prospect Personalized Recommendation on Large Language Model-based Agent Platform 2024

概要

GPTなどのエージェント指向情報システムの新たな種類が情報システムのインフラストラクチャの検討を促すと指摘しています。LLMに基づくエージェントの特性、例えば相互作用性に適応することの重要性を強調しています。新しいレコメンデーションパラダイム「Rec4Agentverse」を紹介しています。

手法

Rec4Agentverseは、エージェントアイテムとエージェントレコメンダーから構成される新しいレコメンデーションパラダイムです。相互作用と情報交換を促進するために、ユーザー、エージェントレコメンダー、エージェントアイテムの3つの段階を概念化しています。
第1段階: ユーザーとエージェントアイテムの相互作用この初期段階では、ユーザーはエージェントアイテムと相互作用します。この相互作用は、伝統的なレコメンデーションと同様に、ユーザーが直接エージェントアイテムに指示を出したり、暗黙的な行動を通じてパーソナライズされたLLMベースのエージェントを生成または取得することができます。ユーザーはLLMベースのエージェントと情報を交換する新しい形式で相互作用することができますが、これだけではLLMベースのエージェントの巨大な潜在能力を完全に解放することはありません。ユーザーとの相互作用に加えて、エージェントアイテムはLLMベースのエージェントプラットフォーム上で情報フローをさらに豊かにするために、レコメンデーションシステム内の他の役割とも協力することができます。

第2段階: エージェントとレコメンダーの協力この段階では、エージェントアイテムはエージェントレコメンダーと協力してユーザーに情報サービスを提供します。伝統的なレコメンデーションシステムのアイテムとは異なり、エージェントアイテムはエージェントレコメンダーと深く協力することができ、エージェントレコメンダーにユーザー情報を提供したり、エージェントレコメンダーから新しい指示を受け取ったりすることができます。例えば、エージェントアイテムは収集したユーザーの好みをエージェントレコメンダーと共有し、エージェントレコメンダーがよりパーソナライズされたレコメンデーションを提供できるようにします。同様に、エージェントアイテムはエージェントレコメンダーから新しい指示を受け取ることもできます。ユーザーから収集したパーソナライズされた情報とエージェントレコメンダーからの指示は、エージェントアイテムの進化（例えば、プロンプトの更新など）に使用され、エージェントアイテムがユーザーの好みをよりよく理解し、優れた情報サービスを提供できるようになります。

第3段階: エージェント間の協力エージェントアイテムは、異なるドメイン知識を持つ他のエージェントアイテムと協力して、ユーザーに多様な情報サービスを提供することができます。単純な例として、ユーザーがエージェントアイテムが知らないニッチなものについて言及した場合、エージェントアイテムはエージェントレコメンダーに新しいエージェントアイテムを推薦するように依頼することができます。その後、エージェントは協力してユーザーの情報ニーズを満たすか、タスクを実行します。これを超えて、この段階では想像の余地が大きく、推奨された新しいエージェントアイテムもユーザーと直接相互作用したり、エージェントレコメンダーと相互作用したりすることができます。さらに、複数のエージェントアイテムが推奨された場合、これらのエージェントアイテムは、ブレインストーミングやラウンドテーブル会議を通じて、ユーザーの指示をよりよく完了するために協力することもできます。

応用

- 旅行エージェント(Travel Agents): 旅行エージェントは、ユーザーが旅行計画や予約をサポートするために設計されています。ユーザーが興味のある特定の旅行先を指定すると、エージェントレコメンダーは旅行の専門家である旅行エージェントを推薦します。推薦された旅行エージェントは、ユーザーとの直接的なやり取りやエージェントレコメンダーへのアクセスを通じてユーザーの個々の好みを推測し、より良い旅行推薦のために自身をアップグレードすることができます。さらに、旅行エージェントは他のエージェントとの協力を通じて、多様なドメインからユーザーの好みに関する貴重な洞察を得ることができます。この協力的アプローチにより、旅行エージェントはより適応性が高くパーソナライズされた旅行計画をユーザーに提供することが可能になります。
- ファッションエージェント(Fashion Agents): ファッションエージェントは、ユーザーが好みのファッションスタイルを発見し、その好みに合ったファッションアイテムを推薦することを目指しています。旅行エージェントと同様に、ファッションエージェントもユーザーとの会話やエージェントレコメンダーとの相互作用を通じて、ユーザーのファッションに関する好みを収集することができます。また、ファッションエージェントはテーラーエージェントと協力して、ユーザーのためにパーソナライズされた新しい服を設計・製作することも可能です。
- スポーツエージェント(Sports Agents): スポーツエージェントは、ユーザーに適した運動計画を推薦することを目的としています。彼らは、ユーザーやエージェントレコメンダー、他のエージェントアイテムとのやり取りを通じてユーザーの好みを収集し、運動計画や推薦を提供することができます。例えば、彼らは旅行エージェントから得たユーザーの体調に関する情報を使用して、適切な運動計画を作成することができます。

Tell Me More! Towards Implicit User Intention Understanding of Language Model Driven Agents 2024

概要

ユーザーの意図を聞き取り、実行可能な目標に絞り込んでから下流のエージェントによるタスクを実行することでユーザーの意図を明確にしてタスクを実行するMistral-Interactを提案
ユーザーがエージェントに与える指示は曖昧で簡潔すぎる場合が多かったり、一見明確な指示でも意図が異なる可能性があるため、明示的な質問によって探る必要があるという問題に対して意図を理解し、タスクを明確にすることができます
https://github.com/HBX-hbx/Mistral-Interact

手法

タスク指示の明確さを測るベンチマークとしてIntention-in-Interaction (IN3) を提案。エージェントに対して、
タスクのあいまいさの判断とユーザーの意図を理解することを通じて、エージェントの相互作用能力をテストします。
例えば図の例では「私の街で最高のヨガ教室を見つける」というタスクの場合、「私の街」がどこなのか、
「最高」の基準が何なのかが不明確。エージェントの実行効率を高めるためには、
ユーザーの真の意図を明確に把握することが必要になります。

結果

エージェントの意図理解能力をさらに高める為に、ユーザーとの対話を通して意図を聞き出す専門モデルを学習し
エージェントの上流に組み込むことを提案
具体的には、専門モジュールはユーザーとの対話を通して、以下の情報を取得しています
- ユーザーがタスクで達成したい具体的な目標
- ユーザーがタスク実行に使える時間やリソース
- ユーザーのタスクに対する優先順位

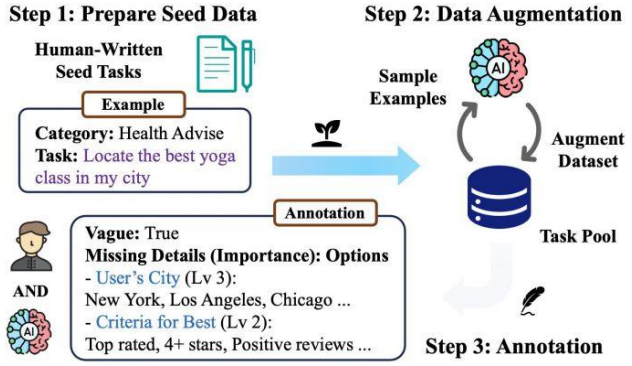


Figure 2: An illustration of IN3's formation with an example data point.

Loose LIPS Sink Ships: Asking Questions in Battleship with Language-Informed Program Sampling 2024

概要

人間の質問作成能力を再現するため、言語モデルを活用し、情報ゲインを評価するLIPSモデルを提案。シンプルなモンテカルロ最適化戦略で、
人間のパフォーマンスを反映した情報に富んだ質問を生成することができる。
LLMのみを使用したモデルは、質問の根拠付けに苦労することが示された。

手法

LIPSモデル（言語情報プログラムサンプリングモデル）はLLMを使用して質問を生成し、
それを具体的な問題解決や計算を行うための命令や手続きを記述したシンボリックプログラムに変換、期待される情報ゲインを評価します。
1, 言語事前分布からの質問サンプリング: LIPSモデルは、k個の候補質問を言語事前分布から確率的に抽出します。
この事前分布は、大規模言語モデル（LLMs）に基づいており、自然言語での質問を生成する役割を果たします。
2, 質問からプログラムへの変換 抽出された質問は、シンボリックプログラム、
具体的には言語から思考の言語（Language of Thought: LoT）プログラムに変換されます。
この変換は、LLMsを使用して行われ、質問を意味のあるシンボリック表現にマッピングします。
3. 期待情報ゲイン（EIG）の計算: 各候補質問のシンボリックプログラムに基づき、内部の世界モデルを用いてシミュレーションを行い、
期待情報ゲイン（Expected Information Gain: EIG）を計算します。これにより、それぞれの質問がどれだけの情報を提供するかを評価します。
4. 最適な質問の選択 計算されたEIGに基づき、最も情報量の高い質問、つまり最高のEIG値を持つ質問を選択します。

結果

ベイジアンモデルとLLMの組み合わせが、人間のような質問を生成する上で有効であることを示す。
しかし、純粋なLLMには、具体的な推論者としてのいくつかの欠点がある。

```
import random
# ステップ 1: 言語事前分布からの質問サンプリング
def sample_questions(k):
    # 仮の質問リスト
    all_questions = [
        "最も高い山は何ですか？ ",
        "光の速さはいくらですか？ ",
        "最も深い海はどこですか？ ",
        "人体の細胞数はいくつですか？ ",
        "地球の周囲の長さは何キロメートルですか？ "
    ]
    # k個の質問をランダムに選択
    sampled_questions = random.sample(all_questions, k)
    return sampled_questions
# ステップ 2: 質問からプログラムへの変換
def question_to_program(question):
    # 仮の変換ルール（単純化のためキーワードに基づく）
    keywords_to_programs = {
        "山": "find_highest_mountain()",
        "光": "get_speed_of_light()",
        "海": "find_deepest_sea()",
        "細胞": "count_human_cells()",
        "地球": "measure_earth_circumference()"
    }
    for keyword, program in keywords_to_programs.items():
        if keyword in question:
            return program
    return "unknown_program()"

# ステップ 3: 期待情報ゲイン（ EIG）の計算
def calculate_eig(program):
    # EIGをランダムに生成（実際の計算はより複雑）
    return random.uniform(0, 1)

# ステップ 4: 最適な質問の選択
def select_best_question(questions):
    question_eig_pairs = []
    for question in questions:
        program = question_to_program(question)
        eig = calculate_eig(program)
        question_eig_pairs.append((question, eig))
        # EIGが最大の質問を選択
    best_question, _ = max(question_eig_pairs, key=lambda x: x[1])
    return best_question

# 実行例
k = 3 # サンプリングする質問の数
sampled_questions = sample_questions(k)
best_question = select_best_question(sampled_questions)
sampled_questions, best_question
```

Improving the Validity of Automatically Generated Feedback via Reinforcement Learning 2024

概要

LLMを使用したフィードバックの自動生成はオンライン学習プラットフォームで学生の学習成果を向上させる可能性があります。フィードバックの自動生成と評価の両問題に対処し、覚醒と整合性を最適化するフィードバック生成フレームワークを提案。GPT-4を使用した人間によるフィードバックとLM生成フィードバックのアノテーション、および強化学習を使用したフィードバックの生成の最適化により、生成されたフィードバックの正確性とし整合性が向上することがわかりました。

手法

GPT-4を用いて人間とLMによって生成されたフィードバックを評価するための基準を提案し、フィードバック生成のためのRLフレームワークを開発。
GPT-4を用いたフィードバック評価の基準は、以下のアルゴリズムに基づいています。フィードバックは5つの異なる側面で評価され、それぞれが2値のラベル（0または1）で結果が出ます。
- 正確性（COR.）：フィードバックが現在の質問と学生の回答に関連し、誤った声明を含まない。
- 明示性（REV.）：フィードバックが正解を直接学生に明かさない。
- 提案（SUG.）：フィードバックが学生に提案を提供し、それに従うと正しい答えに導かれる。
- 診断性（DIA.）：フィードバックが学生が犯したエラーやその回答に潜む誤解を正しく指摘する。
- ポジティブ（POS.）：フィードバックがポジティブで励ましのトーンを持つ。

フィードバックメッセージに対する最終的なスカラー値のルーブリックスコアは、これらのラベルを集約して、フィードバックの全体的な品質を示します。フィードバックメッセージが誤っている場合、最終ルーブリックスコアは0になります。それ以外の場合、フィードバックが満たすルーブリックの各側面に対して0.2の増分でスコアが上がります。正確性を除き、他のルーブリックの側面は同等に重み付けされます。
GPT-4は、ゼロショットの思考プロンプトを用いて、各ラベルに関連するはいまたはいいえの質問に答えることで、推定されたラベルと対応するスコアを出力します。プロンプト開発中にGPT-4に質問をすることが、正式なルーブリックに基づいてラベルを割り当てるよりも優れていること2値ラベルがリカート尺度よりも優れていること、最初の2つの質問の否定を尋ねてラベルを反転させた後の精度が向上することが観察されました。

結果

提案した方法は、Llama 2を使用して、生成されたフィードバックの正確性と整合性を大幅に向上させることができました。

RAGged Edges: The Double-Edged Sword of Retrieval-Augmented Chatbots 2024

概要

LLMが誤った情報を生成する傾向（ハルシネーション）に注目し、検索拡張生成（Retrieval-Augmented Generation, RAG）がこの問題にどのように対処できるかを探ります。ハルシネーションは、ChatGPTを使用した際に存在しない法律判例を引用するなど、重大な課題を引き起こしています。本研究は、ハルシネーションを誘発するように設計されたプロンプトを使用してRAGと標準LLMsを比較し、RAGが一部のケースで精度を向上させることを示していますが、プロンプトがモデルの事前学習された理解と直接矛盾する場合には誤った解釈を導く可能性がある結果を表示することを示します

手法

人間の参加者が生成された応答の精度を詳細に評価する実験を設計しました。この実験では、学術的背景、職務経験、および出版物をレビューするために、彼ら自身の履歴書（CV）を使用しました。

結果

コンテキストを追加することで、LLMの応答の正確性が大幅に向上し、コンテキストがある場合の正確な応答率が約94%であったのに対し、コンテキストがない場合は7.31%のみでした。これは、コンテキストの追加が正しい回答の可能性を約18倍に増加させることを意味します。

RAGの性能も完全ではなく、特定の条件下では、検索されたテキストのデータを常に正確に表現するわけではありません。この研究では、提供されたコンテキストが正確であることを確認しましたが、モデルが6.04%の時間で間違った応答を提供する理由を探りました。

LLaMoCo: Instruction Tuning of Large Language Models for Optimization Code Generation / LLaMoCo: 最適化コード生成のための大規模言語モデルの命令チューニング 2024

概要

LLMを使った最適化の探求が進められていますが、これまでのアプローチには操作効率の低さ、プロンプト設計への高い感度、特定のドメイン知識の欠如という固有の限界がありました。
コードからコードへの方法で最適化問題を解決するためにLLMsを適応させるために設計された、最初の命令チューニングフレームワークであるLaMoCoを提案。
よく記述された問題プロンプトと効果的な最適化コードを含む包括的な指示セットを確立し、、モデルの微調整中に収束挙動を向上させるために、命令チューニングフェーズの前にコントラスト学習に基づくウォームアップ手順を取り入れた新しい二段階学習戦略を開発。実験結果はLaMoCoによって微調整されたCodeGen(350M)モデルは、合成および現実的な問題セットの両方でGPT-4 Turboおよび他の競合他社に比べて優れた最適化性能を実現しました
https://anonymous.4open.science/r/LLaMoCo-722A/README.md

手法

問題プロンプトと実行可能な最適化プログラムのコード対を含む指示セットに基づいてLLMsを微調整します。微調整は、異なるプロンプトが同じセマンティクスを共有する場合に潜在空間表現を整列させるコントラスト学習を含む二段階の命令チューニング戦略を用いて行われます。

結果

微調整されたモデルは、合成および現実的な問題セットの両方において、GPT-4 Turboおよび他の競合他社よりも優れた最適化性能を達成しました。

CatCode: A Comprehensive Evaluation Framework for LLMs On the Mixture of Code and Text 2024

概要

LLMのコーディング問題解決能力を包括的に評価するために、カテゴリー理論を評価フレームワークを提案。現在の評価方法がタスクの範囲に限られているか、標準化されていない問題を解決するために、カテゴリー理論を用いてこの問題に取り組みます。提案されたフレームワークは、コードデバッグ、コード変換、コード生成説明などのタスクをカテゴリー理論の観点から評価するものです。

手法

カテゴリ理論を用いて、コードと自然言語の混合を扱うための標準化された評価フレームワークを構築します。この枠組みでは、コード関連のタスクをカテゴリ的な観点から再定義し、モデルの能力を、オブジェクト（コードスニペット）、射（コード変換）、関手（言語間の翻訳や説明）を用いて評価します。データセット、タスク、モデルに適応可能な自動評価プラットフォームを提示し、これを使用して複数のLLMのコーディング能力を量的に評価します。標準化されたAPIとプロンプトセレクタを通じて、異なるデータセットとモデルに対する評価を容易に行うことができます。

結果

CatCodeは、カテゴリ理論に基づいてLLMがコードとテキストの混合を理解し生成する能力を評価するための新しい視点を導入します。この包括的かつ数学的に抽象的なアプローチを用いることで、新しいデータセット、タスク、モデルに適応する標準化された自動評価プラットフォームを提供します。LLMを評価する際、現在のモデルが機能的に等価なコードを認識し、コードの機能に関する情報をコードとその説明の間で保持する能力に欠けていることが明らかになりました。このプラットフォームをオープンソース化することで、LLMの包括的かつ標準化された評価に貢献し、コードとテキストの混合を扱うためのカテゴリ的観点を提供することを目指します。

Improving LLM Code Generation with Grammar Augmentation 2024

概要

プログラム言語の文法を利用し、LLMによるコード生成の効率と一般性を向上させる新しいフレームワークSynCodeを提案。このフレームワークは、オフラインで構築された効率的なルックアップテーブルであるDFAマスクストアを使用して、文法のターミナルに基づいています。SynCodeは、文法的に有効なトークンのみを保持し、無効なものを排除することによりPythonやGoなどのCFG（文脈自由文法）に基づく任意の言語での使用を実証しました。実験により、SynCodeを最先端のLLMと組み合わせることで、構文エラーを96.07%削減し、コード生成の文法的精度を大幅に向上させることができることが示されました。

手法

SynCodeは、文法のターミナルを表現する正規表現から派生したルックアップテーブルであるDFAマスクストアを核としています。このストアを利用し、LLMのデコード段階で次のトークンを選択する際に、文法的に有効なトークンのみを考慮することができます。

SynCodeのアルゴリズムについての説明は以下の通りです。

入力:
- LLM（M）
- トークナイザ（T）
- 入力プロンプト文字列（C0）
- 生成されるトークンの最大数（nmax）
- デコーディング戦略（D）

手順:
1. 入力プロンプトをトークナイズして、現在のトークン列（Tcur）を初期化します。
2. 最大nmaxトークンが生成されるまで、または終端トークン（EOS）が選択されるまで以下のステップを繰り返します。
    - 現在のトークン列（Tcur）を基に、LLM（M）を用いてスコアを生成します。
    - 現在のトークン列をデコードして、部分的なコード（Ck）を取得します。
    - 部分的なコード（Ck）を解析して、受け入れ可能な終端シーケンス（A）と残りの部分（r）を取得します。
    - 文法マスク（m）を適用して、スコアに基づいて次のトークン（ti）を選択します。
    - 選択されたトークンが終端シーケンストークン（EOS）である場合は、プロセスを終了します。
    - そうでない場合は、現在のトークン列（Tcur）に選択されたトークンを追加します。
3. 最終的なデコードされた出力を取得し、生成プロセスの結果として返します。

このアルゴリズムは、文法に基づいてLLMの出力を制約することにより、文法的に有効なトークンのみを選択することを目的としています。文法マスク（m）は、文法に従って許可されるトークンのみを選択するために使用されるブールマスクです。このプロセスを通じて、生成されたコードの文法的正確さを向上させることができます。

結果

PythonおよびGo言語における簡略化されたCFGを用いた実験では、SynCodeを用いることで構文エラーを96.07%削減できることが確認されました。

Infusing Knowledge into Large Language Models with Contextual Prompts 2024

概要

関連する文書や情報から直接抽出した知識を用いてLLMを特定のタスクに対してより適切に機能させるために微調整することで与えられたタスクに対してより正確で、有用な出力を提供します。

手法

入力プロンプトに入っている外部文章などのコンテキストから重要なポイントやキーワードに基づいてプロンプトを生成。このプロンプトは、入力に関する事前に知っておくべき背景知識を持って過程できます。次にこの生成下プロンプトを塩牛てLLMを微調整することでプロンプトに関連する情報を基に学習を深め、より正確な要約を生成できるようになります

結果

従来の知識注入手法と比較して、この研究のアプローチは知識グラフに依存せず、文書から直接関連するコンテキストを利用する点で新規性があります。これにより、一般化が容易で、さまざまなドメインやエンティティに適用可能な方法です

Benchmarking Hallucination in Large Language Models based on Unanswerable Math Word Problem 2024

概要

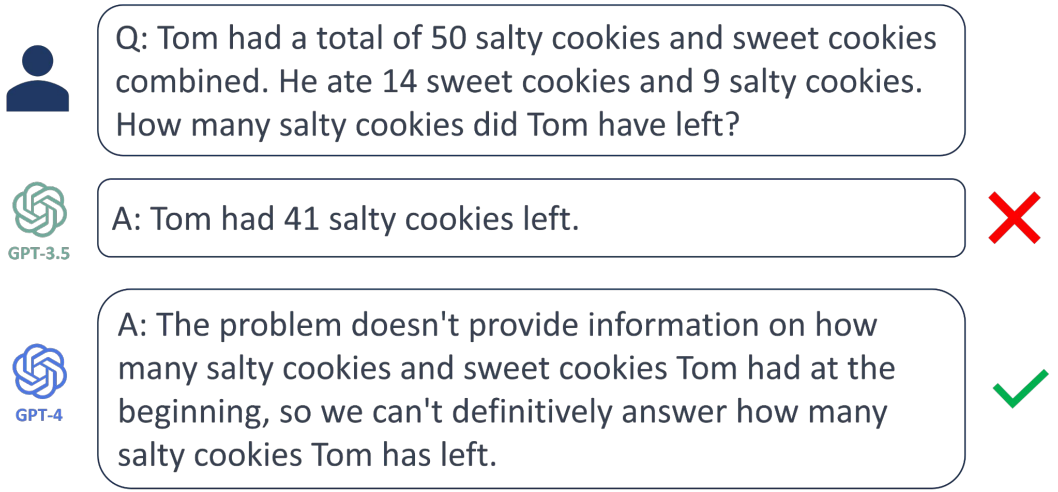
特に、解答不可能な数学の文章問題(MWP)に基づいてLLMの幻覚を評価する新しい方法を提示。
このアプローチをサポートするために、解答不可能な数学の文章問題(UMWP)という5200の質問からなるデータセットを革新的に開発。
テキストの類似性と数学的表現の検出を組み合わせる評価方法を開発しLLMが質問を解答不可能と考えるかどうかを判断しました。31のLLMに対して広範な実験を行い、その結果、コンテキスト内学習と人間のフィードバックによる強化学習(RLHF)の訓練が、モデルの幻覚回避能力を顕著に向上させることを示しました。UMWPを使用することは、幻覚を評価するための信頼性の高い効果的なアプローチであることを示しています。
https://github.com/Yuki-Asuuna/UMWP

手法

UMWPデータセットの開発、テキスト類似性と数学的表現検出に基づく評価方法の開発31のLLMsに対する実験。

結果

コンテキスト内学習とRLHFの訓練がモデルの幻覚回避能力を向上させることを実証。特にGPT-3, InstructGPT, LLaMA, Claudeにおいて有効であることが示されました。



Generative Explanations for Program Synthesizers プログラム合成器のための生成的説明 2024

概要

プログラムの合成が成功して実装が仕様を満たしていることが保証されても実装の動作方法や仕様がどのように実現されているかの説明についての追加情報は提供されません。この問題についてLLMを使用して人間が読める説明を構築するアプローチを開発。

手法

生成されたプログラム実装の理解を促進するためにLLMを使用して人間が理解しやすい説明を生成しています。このアプローチの中心的な要素は以下の2つです。

サブスペシフィケーションを使用したプロンプトの拡張:

プログラムの各部分（サブルーチン）がどのように動作するかを詳細に記述したサブすぺしふぃケーションを使用し、生成されたプログラムの各部分（サブルーチン）がどのように動作するかを確認、
その情報を基にプログラムの各部分が何をするのかを説明する名称をLMに提案させることでプログラムの動作を人が理解できるようになります。※プロンプトの拡張はLMがプログラムのサブルーチンの動作を正確に理解し、それを基に人が理解しやすい動作を説明した関数名を生成させることを意味してインした。

アルゴリズミックサニティチェック:

LLMによって生成されたプログラムのサブルーチンの名称とサブすぺしふぃケーションを使用して、それまでとは別のLMモデルを使い、生成されたプログラムのサブルーチンを再合成します。
この再合成されたサブルーチンを元の実装に代替してみて、グローバルな仕様を満たすかどうかを検証します。

Generative News Recommendation 生成型ニュース推奨 2024

概要

オンラインでニュースを読むときに、読者が興味のあるニュースをもっと簡単に、より深く理解できるようにする生成型ニュース推薦パラダイム(GNR)を提案。
普通のニュース推薦システムが単に関連する記事を示すのとは違い、この方法は読者が興味を持つ可能性のある様々なニュース記事を結びつけて、一つのつながりのある話として提示します。これにより、読者はニュースの背後にある大きな話や事件をもっと早く、包括的に理解することができるようになります。

手法

LLMを使用してニュースと読者の両方についての詳細な情報を、ニュースについては記事の内容からテーマや関連性、重要な要素などを抽出します。読者については、過去の読書歴や興味関心から、読者のプロファイルの詳細な情報を作ります。
この情報を使い、読者の関心にあったニュースを見つけ出し、それらを一つの物語のように結びつける訓練方法(UIFT)を実行します。
具体的には、読者が興味を持ちそうなニュースを選び出し、それらを論理的で一貫性のあるストーリーにまとめあげることで、読者がニュースをより深く理解しやすくすることを目指しています。
このアプローチにより、ここの読者に合わせたニュース推奨を実現できているつもりらしい

UIFT (User Interest Alignment Fine-Tuning) :
1. 前処理段階:
LLM を利用して、ニュースとユーザーのテーマレベルの表現を生成します。これにはニュースの内容やユーザーのプロファイル（過去のクリック履歴など）を入力として使用します。
これらのテーマレベルの表現とセマンティックレベルの表現（直接的な内容に基づく表現）を組み合わせて、デュアルレベルの表現を形成します。

2. パーソナライズされた関連ニュースの探索
ニュースセットからユーザーの関心事に合致するニュースとその関連ニュースを特定し、これらを用いて一貫性のある物語を生成するための準備を行います。
まず、デュアルレベルの表現に基づいてニュースをランキングし、ユーザーの好みに最も合致するニュース（焦点ニュース）を選択します。
次に、ニュース間の論理的関連を探索し、焦点ニュースに関連するさらに多くのニュース記事を見つけます。
最後に、ユーザーの興味に合わない可能性のあるニュースをフィルタリングし、焦点ニュースの主なイベントの文脈を考慮しつつ、ユーザーの好みを反映した参照ニュースセットを取得します。

3. 興味に基づく複数ニュースの物語統合(Interest-aware Multi-news Narrative Fusion):
参照ニュースセットの中心テーマを包括する一貫性と論理的に構造化された物語を作成します。これにより、生成された物語とユーザーの興味との整合性を高め、ユーザーがコンテンツにさらに関与するように促します。
ここで UIFT を導入し、複数のニュース物語の確率を最適化することでランキング損失を調整します。このプロセスを通じて、モデルはユーザーの関心により密接に合わせた物語を生成するよう微調整されます。

4. 微調整と評価:
LLMを使用して多ニュース物語を生成し、その後、我々の多ニュース物語生成器を監視されたファインチューニングを通じて訓練しLLMの能力と知識を蒸留します。
UIFTはランキング損失を取り入れることにより、我々の生成器が複数のニュース物語をユーザーの興味に基づいてランキングし、よりパーソナライズされた物語を生成できるように訓練します。

結果

従来の知識注入手法と比較して、この研究のアプローチは知識グラフに依存せず、文書から直接関連するコンテキストを利用する点で新規性があります。これにより、一般化が容易で、さまざまなドメインやエンティティに適用可能な方法です

Magic Markup: Maintaining Document-External Markup with an LLM 2024

概要

文書の外部にメタデータを保持する新しい手法を提案。特に、プログラムが変更された後にアノテーションがコードに自動的に追従するように、インテリジェントエージェントを使用して変更されたプログラムを再タグ付けするシステムを開発。この方法は、文書アノテーションの応用範囲を拡大し、プログラムの作成、デバッグ、保守、およびプレゼンテーションにおける基本的な操作を可能にします。私たちのシステムは、ベンチマークで90%の精度を達成し、ドキュメントのタグを並行して5秒ごとに置き換えることができます。性能はさらなる改善の余地があるものの、信頼性が十分に高いことから、アプリケーションのさらなる探求を正当化します。

手法

LLMに基づいた再タグ付けシステムを構築し、合成されたベンチマークスイートを用いてこのシステムを評価しました。このベンチマークは、4つのプログラミング言語で90のコード更新をカバーしています。

1. LLM（大規模言語モデル）に基づいた再タグ付けシステムの構築このシステムは、プログラミング言語における90のコード更新をカバーする合成ベンチマークスイートを使用して評価されました。これらのベンチマークは、タグ付けされたエンティティが再配置または変更されるシナリオを表しています。

2. 合成ベンチマークスイートの生成 特定のプログラミングタスクとその更新に関するシナリオを表現するために、異なるプログラミング言語でのコードの例を生成し、これらの例に基づいてベンチマークを構築しました。これには、LLMが生成したコードに対して手動でタグを配置し、次にこれらのコードがどのように更新されたか（例えば、リファクタリングや機能の追加によって）を示すプロセスが含まれます。

3. 再タグ付けプロセス: 更新されたコードに対して正確なセグメントの開始と終了のテキストポイントを特定し、これらのポイントに基づいてアノテーションを再配置します。このプロセスは、文書の更新後もタグが関連するコンテキストに正しく残ることを保証することを目指しています。

4. プロンプトの調整と評価 再タグ付けシステムの効率と精度を評価するために、特定のプロンプト形式を使用してモデルがテキストと行番号を参照し、更新されたセグメント内の正しい位置を特定できるようにしました。これは、更新されたファイル内でのセグメントの正確な配置を特定するために重要です。

この手法は、文書外のマークアップを自動的に維持するためのアプローチで、言語モデルがプログラムのセマンティクスを理解し、コードの変更に伴ってアノテーションを適切に追跡できる能力を利用しています。

結果

提案システムはベンチマークで90%の精度を達成しました。特に、ドキュメントのタグを5秒ごとに並行して置き換えることが可能であることを示しました。

Towards General Computer Control: A Multimodal Agent for Red Dead Redemption II as a Case Study (汎用コンピュータ制御に向けて：ケーススタディとしてのRed Dead Redemption IIのためのマルチモーダルエージェント) 2024

概要

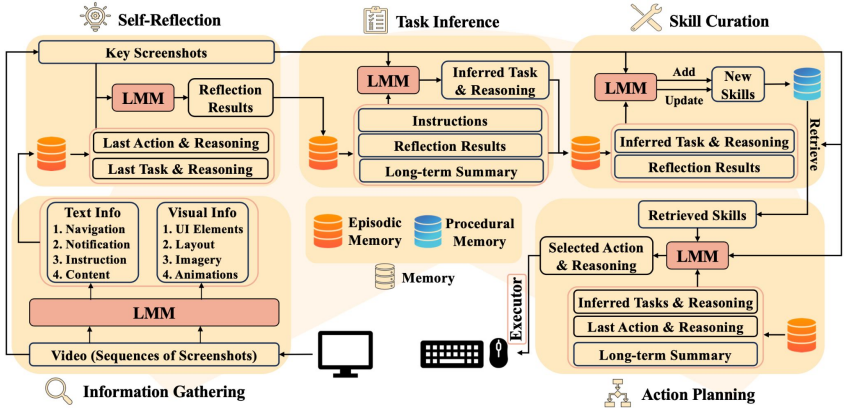
基礎エージェントが特定のタスクやシナリオで成功しているにも関わらず、シナリオ間での一般化が困難であるという問題を解決するためにGeneral Computer Control (GCC)設定を提案します。GCCは、コンピュータの画面画像（および可能であればオーディオ）のみを入力として受け取り、キーボードとマウス操作を出力とすることで、任意のコンピュータタスクをマスターする基礎エージェントを構築することを目指します。CRADLEというエージェントフレームワークを紹介し、複雑なAAAゲームであるRed Dead Redemption IIにデプロイすることによって、GCCへの初期の試みを行います。CRADLEは、情報収集、自己反映、タスク推論、スキルカリキュレーション、アクションプランニング、メモリの6つの主要なモジュールを含んでいます。
最小限の事前知識を使用することで複雑なゲーム操作を行わせることが脳になります
https://baai-agents.github.io/Cradle/

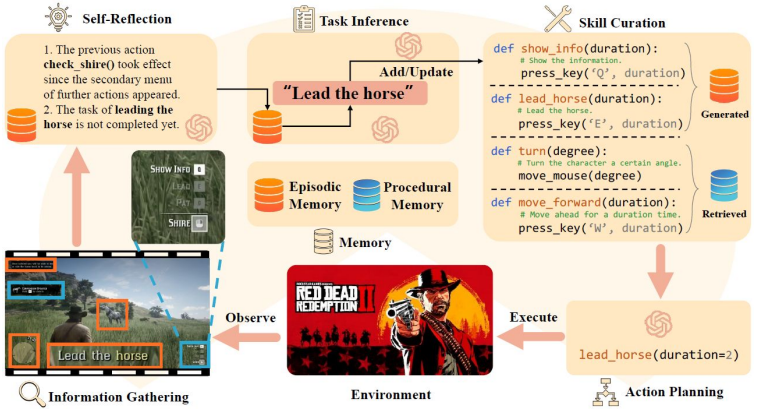
手法

GCC設定を提案し、情報収集、自己反映、タスク推論、スキルカリキュレーション、アクションプランニング、メモリの6つの主要モジュールを備えたCRADLEフレームワークを開発しました。CRADLEは、スクリーンからのビデオを入力として受け取り、キーボードとマウス操作を出力します。

結果

CRADLEをRed Dead Redemption IIに適用し、このゲームで主要なストーリーラインをフォローし、実任務を完了することができました。





Preventing Reward Hacking with Occupancy Measure Regularization 報酬ハッキングの防止における占有測度正則化 2024

概要

報酬ハッキングは、設計者が意図した「真の」目標や報酬ではなく、より簡単に達成できる「代理」の目標や報酬を追求してしまう現象のことをいい、システムが本来の目的を達成する代わりに、設計上の欠陥や盲点を利用して見かけ上の成果を挙げる行動を取ることを指します。これは、システムが真に望ましい行動を学習するのではなく、手っ取り早く報酬を稼ぐための「抜け道」を見つけ出してしまうことにより発生します。本研究では、報酬ハッキングを防ぐために、行動分布（AD）の発散ではなく、ポリシー間の占有測度（OM）の発散に基づいて正則化する方法を提案します。理論的にはOM正則化が真の報酬の大幅な低下をより効果的に避けることができることを証明し、実験的には、OM発散がAD発散よりも報酬ハッキングを防ぐのに優れていることを示します。
https://github.com/cassidylaidlaw/orpo

手法

報酬ハッキングを防ぐための新しいアプローチを取り入れています。具体的には、報酬関数に対するエージェントの行動を制御するために、「占有測度（M）発散」という概念を用いています。以下にそのアルゴリズムの概要を説明します。

1. 占有測度（OM）の定義: エージェントがある方針（ポリシー）に従って環境内を動く際に、各状態に存在する時間の割合を表します。これにより、エージェントの行動パターンを数値化し、分析することが可能になります。

2. OM発散の計算: エージェントが異なるポリシーに従った場合の占有測度の違いを計算します。これは、エージェントが学習過程で採用したポリシーが、どの程度元の目的から逸脱しているかを測定するために用いられます。

3. 正則化の適用: OM発散を利用して、エージェントの学習プロセスに正則化項を導入します。この正則化は、エージェントが代理報酬関数に過剰に適応するのを防ぐことを目的としており、真の報酬関数により良く適合するよう促します。

4. 学習アルゴリズムの適用 正則化された報酬関数を使用して、エージェントは環境内での行動を学習します。この過程でOM発散に基づく正則化が適用されることで、エージェントは報酬ハッキングを避け、より望ましい行動パターンを学習することが期待されます。

結果

複数の現実的な環境において、OM発散がAD発散よりも報酬ハッキングを防ぐのに優れていることを実証します。OM発散を使用した正則化は、安全ポリシーに対する性能の向上を可能にしながら報酬ハッキングを防ぎます。

Reaching Consensus in Cooperative Multi-Agent Reinforcement Learning with Goal Imagination ゴール想像による協調型マルチエージェント強化学習における合意形成 2024

概要

エージェントには、複数のエージェントが共通の目標に向かって協力する際、エージェント間で行動や意思決定において適切な調整が行われず、効率的な合意に達することができない状況が起こる非調整問題があります。これは、エージェントが独立して行動を選択することで、全体としての最適な成果が得られない可能性がある問題です。既存の協調型マルチエージェント強化学習(MARL)方法は通常、合意形成を明示的に考慮しておらず、非調整問題を引き起こす可能性があります。
マルチエージェントを明示的に調整するためのモデルベースの合意形成メカニズムMulti-agent Goal Imagination (MAGI) を提案し、モデルベースの長期目標想像メカニズムを提案。
生成された目標を用いてエージェント間の合意形成とモデルフリーなマルチエージェント学習を調整します。具体的にはCVAEを使用して将来の状態分布をモデル化し、この分布から高価値の将来の状態を目標としてサンプリングします。

手法

MAGIフレームワークのアルゴリズムの処理の流れは以下になります。
1. 目標の想像: まず、このフレームワークは「共通の目標」を想像することから始めます。つまり、エージェントたちが協力して達成しようとする、未来のある状態（ゴール）を決めます。
2. 目標の選択: 想像した目標の中から、実際に達成可能で価値の高い目標を選択します。
3. 行動: 各エージェントは、共通の目標達成に向けて最適な行動を選択し、実行します。
4. 学習と調整: 行動の結果を見て、エージェントたちは何がうまくいったか、何がうまくいかなかったかを学びます。
そして、次回の目標達成のために、さらに良い協力の仕方や行動の選択方法を更新します

結果

Multi-agent Particle-EnvironmentsおよびGoogle Research Football環境における結果から、MAGIはサンプル効率と性能の両方において既存の方法よりも優れていることがわかりました。

Design2Code: Automating Front-End Engineering 2024

概要

マルチモーダルLLMが視覚デザインを直接コード実装に変換し、フロントエンドエンジニアリングパイプラインを自動化する新しいパラダイムが可能になりました。
この視覚デザインからコード実装へのタスク（Design2Codeと称する）について初めて体系的に検討を行い、実世界のウェブページ484件をテストケースとして手動でキュレートし、スクリーンショットを入力として与えた際に、現在のマルチモーダルLLMがどれくらいうまく参照ウェブページを直接レンダリングするコード実装を生成できるかを評価するための自動評価指標セットを開発しました。GPT-4VとGemini Vision Proに対するマルチモーダルプロンプトメソッドのスイートを開発し、その有効性を示しています。人間の評価と自動評価の両方でGPT-4Vが良かったらしい
https://salt-nlp.github.io/Design2Code/

手法

この論文の手法セクションでは、多様なモデルと手法のベンチマーク評価に焦点を当てています。具体的には、商業用APIモデルのプロンプトとオープンソースモデルのファインチューニングを含む、さまざまなモデルと手法のパフォーマンスを比較しています。

1. 多モーダルプロンプト手法(Multimodal Prompting Methods)
直接プロンプト(Direct Prompting): 最もシンプルなベースラインとして、参照ウェブページのスクリーンショットとともにHTMLとCSSを使用して指定されたウェブサイトを再現するようモデルに指示します。全てのCSSコードをHTMLファイル内に含め、画像が関わる場合は"rick.jpg"をプレースホルダーとして使用します。JavaScriptスクリプトや動的なインタラクションは含めず、要素のサイズ、テキストの位置、色、および全体のレイアウトに注意を払うよう求めます。

2. テキスト強化プロンプト(Text-Augmented Prompting): モデルに全てを一度に実行させる代わりに、元のウェブページからテキスト要素を抽出して、指示プロンプトとスクリーンショット入力の後にこれらのテキストを追加します。この設定ではOCRの難しさを軽減し、モデルがレイアウトデザインにより集中できるようにします。

3. 自己修正プロンプト(Self-Revision Prompting): モデルが自分の生成物を自己改善することを促す手法です。入力として、入力ウェブページのスクリーンショット、テキスト強化プロンプトから生成されたウェブページのスクリーンショット、テキスト強化プロンプトから生成されたコードの初期解を提供し、参照ウェブページにより近づくように生成実装コードを改善するようモデルに依頼します。

結果

GPT-4Vで生成されたウェブページは、見た目と内容の面で元の参照ウェブページと交換可能であると考えられているケースが 49%あり、64%のケースではGPT-4Vで生成されたウェブページの方が元の参照ウェブページよりも優れていると考えられています。

iScore: Visual Analytics for Interpreting How Language Models Automatically Score Summaries 言語モデルが要約を自動採点する方法を解釈するためのビジュアルアナリティクス 2024

概要

要約採点LLMを構築するiScoreというインタラクティブなビジュアルアナリティクスツールを開発しました。このツールを用いることで、複数の要約を同時にアップロードし、採点し、比較することができます。統合されたビューを使用して、要約の言語を反復的に修正し、結果として得られるLMスコアの変化を追跡し、複数レベルの抽象化でモデルの重みを視覚化できます。
https://github.com/AdamCoscia/iScore

手法

要約採点LLMのビルドと展開に携わる学習エンジニアとの共同設計プロセスを通じて実施されました。
このプロセスは、モデルの解釈に関する設計課題を特定し、これらの課題に対処するためにiScoreというビジュアルアナリティクスツールを開発することを目的としています。
具体的な評価方法は次のようになります。

1. 共同設計プロセス: 学習エンジニアと密接に連携し、彼らが直面している課題とニーズを理解します。このステップでは、学習エンジニアのワークフローを観察し、彼らが求めるツールの機能について議論します。

2. iScoreの開発: 学習エンジニアとの共同設計プロセスから得られたフィードバックを基にiScoreビジュアルアナリティクスツールを開発します。iScoreは、複数の要約を同時にアップロードし、採点し、比較する機能、要約の言語を反復的に修正しLLMスコアの変化を追跡する機能、そしてモデルの重みを複数レベルの抽象化で視覚化する機能を提供します。

3. iScoreの展開と評価: 学習エンジニア3名と共にiScoreを1か月間展開し、その使用経験を評価します。この期間中、ツールの効果、特にLMスコアの精度向上に対する影響を検証します。

4. 質的インタビュー: 学習エンジニアとの質的インタビューを実施しiScoreがLLMを理解、評価、そして信頼を構築する過程でどのように役立ったかについてフィードバックを収集します。

この方法論は、ビジュアルアナリティクスツールが教育ツールとしてのLMの理解と評価をどのようにサポートできるかを探るためのものです。

結果

iScoreの開発と展開を通じて、学習エンジニアは複数の要約を同時にアップロードし、採点し、比較することが可能になりました。また、要約の言語を反復的に修正し、LLMスコアの変化を追跡し、モデルの重みを複数レベルの抽象化で視覚化する機能が提供されました。

Wiki-TabNER: Advancing Table Interpretation Through Named Entity Recognition Wiki-TabNER: 名前付きエンティティ認識を通じて表の解釈を進化させる 2024

概要

Web表に含まれる豊富な知識と、表解釈(TI)タスクに取り組むために目指された表形式の言語モデルに着想を得て、この論文では、特にエンティティリンクタスクに焦点を当てたタスクの評価のために広く使用されるベンチマークデータセットを分析。そして、セル内の名前付きエンティティ認識タスクとして、表のセルに含まれるテキストから人名や地名などのエンティティを特定し、それらを適切なカテゴリーに分類するタスクを実行するためのLLMのプロンプトフレームワークを提案。例を選択するためにランダム選択と類似性ベースの選択の両方を使用するさまざまな設定でLLMにプロンプトを行う実験も実行、

手法

現実世界のWikipediaの表を反映し、セル内のエンティティに2つのDbpediaエンティティタイプで注釈を付ける新しいベンチマークデータセットWiki-TabNERを提案しています。
さらに、セル内の名前付きエンティティ認識に焦点を当てた新しい問題を導入し、新しいデータセット上での大規模言語モデル(LLM)の評価フレームワークを提案しています。

1. 指示部分: タスクを説明し、モデルが出力をどのようにフォーマットするかを指示します。
2. 例示部分: モデルにk-shotの例を提供します。ここでのkは{0, 1, 3}のいずれかです。各例は表と、この表から注釈されたエンティティの出力で構成されます。0-shot評価の場合、この部分を空にする代わりに、1行の表と2つの注釈付きエンティティを強化指示としてモデルに示します。例がない場合、モデルはランダムに構造化された出力を生成するため、このアプローチが採用されています。
3. 入力表: モデルがサブセルエンティティを抽出し、それらのエンティティタイプを識別するための入力表です。

入力プロンプトが生成された後、LLMは入力プロンプトを受け取り、実行されます。

結果

表からの名前付きエンティティ認識のタスクが、現在の最先端の技術でも難しいことを示しています。しかし、例を提供することでモデルの性能を向上させる方法があることも示しています 。

PEARL: A Review-driven Persona-Knowledge grounded Conversational Recommendation Dataset PEARL: レビュー駆動型パーソナ知識に基づく会話推薦データセット 2024

概要

会話推薦システム(CRS)は、インタラクティブな会話を通じてユーザーの好みを引き出し、パーソナライズされた推薦を提供することを目指す。
ここでは特定のユーザーの嗜好や推薦の説明を考慮した会話推薦データセットPEARLを提案。57k以上の対話を含む大規模データセットを、実世界のレビューから得られた詳細なパーソナと知識を用いて構築しPEARLの発話は以前のデータセットよりも特定のユーザー嗜好をより多く含み、対象ドメインにおける専門知識を示し、対話コンテキストにより関連性の高い推薦を提供する

手法

この手法は、実世界のレビューからユーザーのパーソナ（個人の特徴や好み）とアイテムに関する知識を抽出し、これらの情報を基に拡張された大規模言語モデル(LLM)を用いて会話推薦データセットを合成するアプローチです。具体的には、パーソナとアイテム知識を組み込んだユーザーシミュレーターとレコメンダーシミュレーターを使用して、リアルな対話を生成します。

この論文では、会話推薦システムのためのデータセット「PEARL」を構築するために、GPT-3.5などの大規模言語モデルを使用しています。

プロンプト1
あなたが書いたレビューを元に、映画について気に入った点と気に入らなかった点を、それぞれ[like]と[Dislike]の下にまとめてください。気に入った点気に入らなかった点について言及することがなければ、対応するタグの下に「None.」と記載してください。
こちらがあなたによって書かれた映画レビューです：
{review}
表11：ユーザーのレビューをまとめるためのプロンプト。

プロンプト2
{movie title}についての人気のあるレビューを元に、人々が映画について気に入った点と気に入らなかった点を、それぞれ[like]と[Dislike]の下に記述してください。気に入った点気に入らなかった点について言及することがなければ、対応するタグの下に「None.」と記載してください。
こちらが映画とそれについてのレビューに関する基本情報です：
映画タイトル：{movie title}
ジャンル：{genre}
監督：{director}
キャスト：{cast}
レビュー：
{reviews}
表12：ある映画の人気レビューをまとめるためのプロンプト。

結果

PEARLデータセットには、4k以上のユーザーを模擬し、9k以上のアイテムをカバーする57.2k以上の対話が含まれている。人間の評価者による比較では、PEARLが生成した対話は、他のクラウドソース対話よりも好まれ、より広範なユーザーのニーズをカバーしている

Acceleron: A Tool to Accelerate Research Ideation Acceleron: 研究アイデア創出を加速するツール 2024

概要

研究アイデアの創出を支援するため、「Acceleron」という研究ライフサイクルの異なる段階を加速するためのツールを提案。
Acceleronは、新しい研究問題を含む包括的な研究提案の策定を通じて研究者を導きます。

手法

Acceleronは、研究アイデアの創出を模倣するために、大規模言語モデル(LLM)の推論能力とドメイン固有のスキルを活用します。これは、研究者が対話的に研究提案を開発できるようにするエージェントベースのアーキテクチャを採用しています。特に、LLMエージェントは研究プロセスを模倣します。

Acceleronは、研究提案の動機検証と方法合成ワークフローを以下のステップで実行します:
1. 研究提案の段階: 研究提案はまず、文献レビューに基づいてその動機が検証され、研究者に受け入れられます。
2. 同僚エージェントの役割: 同僚エージェントが提案を入力として受け取り、提案内で定義された問題を抽出します。
3. メンターエージェントの役割: この問題を入力としてメンターエージェントが受け取り、そのパラメトリック知識を使用して、類似の研究問題の妥当なセットを生成します。例えば、「質問応答タスクのための参照フリー評価メトリックを設計する」という提案に対して、類似の問題として「複数の可能な参照要約を持つテキスト要約のための評価メトリックを考案する」といった問題を生成することができます。
4. 問題のサブタスクへの分解: 提案された問題は、必要に応じてサブタスクに分解されます。たとえば、「科学論文における質問応答」の問題は、「科学論文のPDFドキュメントからのテキストの抽出」「論文のセグメント化と効率的な検索のための保存」「質問に関連する論文のパラグラフの検索」「検索されたパラグラフをコンテキストとして使用した質問の回答」など、複数のサブタスクに分解されます。
5. 科学論文の検索と解析: 編集された類似の問題やサブ問題は、それぞれの問題に対処する科学論文をグローバルコーパスから検索するクエリとして使用されます。検索された科学論文は解析され、チャンク化されてユーザーコーパスに保存されます。
6. 方法の合成: メンターエージェントは、提案された問題を解決するための妥当な方法のリストを合成します。このリストは研究者に提示され、彼らは最も妥当だと思われる方法のサブセットを選択し、必要に応じてこれらをさらに編集することができます。
7. 提案の再構成: メンターエージェントは、選択された方法を含む提案を再構成します。この更新された提案は研究者に示され、さらに編集や最終化が行われます。

このプロセスは、研究者が適切な入力を得ることにより、時間効率を向上させることができるように設計されています。研究提案の開発における動機検証と方法合成の各段階でAcceleronは関連する出力を提供し、研究者が時間を節約できるように支援します。

結果

Acceleronは、機械学習と自然言語処理ドメインからの提案に対して、動機検証と方法合成ワークフローの実行を示します。研究者による観察と評価は、研究者が異なる段階で適切な入力を得ることによって、時間効率が向上したことを示しています。

Acceleron: A Tool to Accelerate Research Ideation Acceleron: 研究アイデア創出を加速するツール2024

プロンプト

1 動機抽出プロンプト
System Message:
You are a researcher and trying to understand the following proposal written by another researcher:{proposal}

Human Message:
Describe in a bulleted list what is not addressed in the current literature which serves as the Motivation behind solving the above research problem proposed in the Proposal. Answer without a heading line and just the bullet points. Each bullet should mention one gap in the literature as a bullet point and not a sentence.

2. 動機質問生成プロンプト
System Message:
You are a researcher and trying to understand the following proposal written by another researcher:{proposal}

Human Message:
Describe in a bulleted list what is not addressed in the current literature which serves as the Motivation behind solving the above research problem proposed in the Proposal. Answer without a heading line and just the bullet points. Each bullet should mention one gap in the literature as a bullet point and not a sentence.

AI Message:
{motivation}

Human Message: Convert each of the above bullets in to a binary question. The question should begin with 'Is the research paper'.

3. 動機検証のための質問プロンプト
System Message:
You are a researcher. You have been given a context, which are paragraphs from a research paper. You have been given a question. Answer the given Question in 'Yes' OR 'No' OR 'Unanswerable'. Answer solely based on the provided context of the research paper. If the question can not be answered with the facts mentioned in the available context or there is any ambiguity in answering the question answer as 'Unanswerable'.
Answer as 'Yes' only when the question can be very clearly answered considering the facts in the research paper provided in the context. Do not repeat the question as the part of the answer.
Provide a concise explanation about how the answer to the question is 'Yes' mentioning the paragraphs used in the context to answer it as 'Yes'. If the answer is 'No' or 'Unanswerable' only output that with NO description or elaboration.

Human Message:
Question: {question}

Research Paper Context: {paper chunks}

4. 制限抽出プロンプト
System Message:
You are a researcher. You have been given the following proposal: {proposal}
A different research paper provided in the context already addresses the gap mentioned as the motivation behind the proposal. {descriptions}|

Human Message:
Research Paper: {paper chunks}
Identify the limitations or gaps of this research paper which can serve as the new motivation for the proposal.
Provide a bulleted list of limitations, where each bullet is concise. Answer WITHOUT a heading line and just the bullet points.

1 動機抽出プロンプト
システムメッセージ:
あなたは研究者であり、他の研究者によって書かれた以下の提案を理解しようとしています: {提案}

人間メッセージ:
現在の文献で対処されていないことが、提案された研究問題を解決する動機として挙げられています。見出し行なしで、弾丸点のリストで記述してください。各弾丸は、文献のギャップを弾丸点で、文としてではなく述べてください。

2 動機質問生成プロンプト
システムメッセージ:
あなたは研究者であり、他の研究者によって書かれた以下の提案を理解しようとしています: {提案}

人間メッセージ:
上記の弾丸を二値の質問に変換してください。質問は「研究論文は」で始めてください。

3 動機検証のための質問プロンプト
システムメッセージ:
あなたは研究者です。研究論文の段落からなるコンテキストが与えられています。質問が与えられました。「 Yes」または「 No」または「 Unanswerable」で質問に答えてください。研究論文の提供されたコンテキストに基づいてのみ答えてください。質問が利用可能なコンテキストに記載された事実で答えられない場合や、質問に答える際にあいまいさがある場合は「 Unanswerable」として答えてください。

人間メッセージ:
質問: {question}
研究論文コンテキスト: {paper chunks}

4 制限抽出プロンプト
システムメッセージ:
あなたは研究者です。以下の提案が与えられています: {提案}
コンテキストに提供された別の研究論文が、提案の背後にある動機として挙げられたギャップにすでに対処しています。

人間メッセージ:
研究論文: {paper chunks}
この研究論文の限界やギャップを特定し、それを提案の新しい動機として機能させてください。制限を簡潔に記載した弾丸点のリストを提供してください。見出し行なしで、弾丸点のみで答えてください。

Acceleron: A Tool to Accelerate Research Ideation Acceleron: 研究アイデア創出を加速するツール2024

プロンプト

5 研究提案の書き換えプロンプト
System Message:
You are a researcher and have written a proposal: {proposal}

Human Message:
Re-write the proposal by taking into consideration the mentioned gaps in the current literature as the new motivation behind of the problem defined in the proposal.
Answer in a Single detailed paragraph WITHOUT any bullet points or list.
Gaps in the current literature: {limitations}

6 研究問題抽出プロンプト
System Message:
You are a researcher and trying to understand the following proposal written by another researcher:
{proposal}

Human Message:
What is the problem solved in the proposal?

7 類似問題生成プロンプト
System Message:
You are a researcher and trying to understand the following proposal written by another researcher:
{proposal}

Human Message:
What is the problem solved in the proposal?

AI Message:
{problem statement}

Human Message:
Give me a bulleted list of a more generalised or similar problems to the problem defined in the proposal.
Don't give aheading just the answer in a bulleted list.

8 サブプロブレム生成プロンプト
System Message:
You are a researcher and trying to understand the following proposal written by another researcher:
{proposal}

Human Message:
What is the problem solved in the proposal?

AI Message:
{problem statement}

Human Message:
Provide a bulleted list of sub-problems or sub-tasks involved to solve the problem. Don't give a heading just the answer in a bulleted list.

5 研究提案の書き換えプロンプト
システムメッセージ：
あなたは研究者であり、提案を書きました：　{提案}

人間メッセージ：

現在の文献の中で挙げられたギャップを新たな動機として考慮して、提案を書き直してください。弾丸点やリストを使用せずに、詳細な段落で答えてください。
現在の文献のギャップ：{limitations}

6 研究問題抽出プロンプト
システムメッセージ：
あなたは研究者であり、他の研究者によって書かれた以下の提案を理解しようとしています：
{提案}

人間メッセージ：
提案で解決される問題は何ですか？

7 類似問題生成プロンプト
システムメッセージ：
あなたは研究者であり、他の研究者によって書かれた以下の提案を理解しようとしています：
{提案}

人間メッセージ：
提案で定義された問題に対して、より一般的または類似の問題の弾丸点のリストを提供してください。見出しをつけずに、弾丸点のリストで答えてください。

8 サブプロブレム生成プロンプト
システムメッセージ：
あなたは研究者であり、他の研究者によって書かれた以下の提案を理解しようとしています：
{提案}

人間メッセージ：
問題を解決するために関与するサブプロブレムまたはサブタスクの弾丸点のリストを提供してください。見出しをつけずに、弾丸点のリストで答えてください。

Acceleron: A Tool to Accelerate Research Ideation Acceleron: 研究アイデア創出を加速するツール 2024

プロンプト

9 研究提案の書き換えプロンプト
Human Message:
{statement}
For the statement given above generate a question to be posed on a research paper to find out if the paper is proposing an approach or method to perform the task defined by the statement. Start the question with: 'Is the research paper proposing an approach or method to'.

10 方法論抽出プロンプト
System Message:
You are a researcher and trying to answer the question posed on a research paper provided as the context.
Research Paper: {paper chunks}

Human Message:
Answer the given Question in 'Yes' OR 'No' OR 'Unanswerable'. Answer solely based on the provided context of the research paper. If the question can not be answered with the facts mentioned in the available context or there is any ambiguity in answering the question, answer as 'Unanswerable'. Answer as 'Yes' only when the question can be very clearly answered considering the facts in the research paper provided in the context. Do not repeat the question as the part of the answer. If the answer to the question is 'Yes', provide detailed approach or methodology to perform the task. If the answer is 'No' or 'Unanswerable' only output that with NO description.

Question: {question}

11 方法合成プロンプト
System Message:
You are a researcher and have been given a proposal and the research problem the proposal is trying to solve. You have been given the approaches in the literature trying to solve, similar problems and sub problems or sub tasks of the problem defined in the proposal. Your task is to synthesize and propose a possible set of methods or approaches to solve the problem defined in the proposal.
Proposal: {proposal}
Research Problem in the Proposal: {problem}

Human Message:
{method context}
Based on the above information suggest the top 3 possible methods or approaches to solve the problem defined in the proposal.

9 類似およびサブプロブレム質問作成プロンプト
人間メッセージ:
上記の声明に基づいて、研究論文が声明で定義されたタスクを実行するためのアプローチまたは方法を提案しているかどうかを調べるために、質問を生成してください。質問は「研究論文が提案しているアプローチまたは方法は」で始めてください。

10 方法論抽出プロンプト
システムメッセージ:
あなたは研究者であり、提供されたコンテキストの研究論文に対して提起された質問に答えようとしています。

人間メッセージ:
質問に対する答えを「 Yes」または「 No」または「 Unanswerable」で提供してください。研究論文の提供されたコンテキストに基づいてのみ答えてください。質問に対する答えが「 Yes」の場合は、タスクを実行するための詳細なアプローチまたは方法論を提供してください。「 No」または「 Unanswerable」の場合は、それだけを説明や詳細なしで出力してください。

質問 : {question}

11 方法合成プロンプト
システムメッセージ:
あなたは研究者であり、提案とその提案が解決しようとしている研究問題が与えられています。類似の問題やサブプロブレムまたはタスクを解決しようとする文献のアプローチが提供されています。提案で定義された問題を解決するための可能な方法セットまたはアプローチを合成して提案することがあなたのタスクです。

人間メッセージ:
提案 : {proposal}
提案の研究問題 : {problem}
上記の情報に基づいて、提案で定義された問題を解決するための上位 3つの可能な方法またはアプローチを提案してください。

Patchscopes: A Unifying Framework for Inspecting Hidden Representations of Language Models 2024

概要

LLMの内部表現を調べる新しいフレームワーク「 Patchscopes」を提案しています。このフレームワークは、 LLMsが人間に理解可能なテキストを生成する能力を利用して、その内部表現を説明することを目指しています。提案された方法では、ソースモデルの隠れ状態をターゲットモデルに給餌し、特定の情報をデコードするためにパッチを適用します。これにより、 LLMsの計算過程について様々な質問に答えることが可能になります。 Patchscopesは、以前の解釈可能性メソッドが直面していたいくつかの問題、例えば早期のレイヤーの検証の失敗や表現力の欠如などを克服します。また、より強力なモデルを使用して小規模モデルの表現を説明する新しい可能性を開き、マルチホップ推論の自己修正などの新しいアプリケーションを解除します。

手法

Patchscopesアルゴリズムは、モデル自身を利用して、その内部表現を自然言語で説明することを提案しています。具体的には、以下のステップで構成されます。
1. ソースプロンプトをソースモデルに供給し、実行を行います。
2. ソースモデルの隠れ状態への任意の変換 (オプショナル)を適用します。
3. ターゲットプロンプトをターゲットモデルに供給し、ターゲットレイヤーまでの実行を行います。
4. ステップ 2で変換された表現を、ターゲットモデルのターゲットレイヤーに「パッチ」し、そのレイヤーから先の計算を継続します。

このフレームワークは、以前の解釈可能性メソッドが直面していたいくつかの課題、例えば初期レイヤーの検査の失敗や表現力の欠如などを軽減します。
Patchscopesは、様々な設定でメソッドを設計できるほどの柔軟性を持っており、 LLMsの隠れた表現から特定の情報をデコードする新たな可能性を開きます。

```
# Patchscopesアルゴリズムの実装
import torch
# この例では、PyTorchを使用していますが、任意のフレームワークで実装可能です。
# ステップ1: ソースモデルにソースプロンプトを供給し、実行を行います。
def execute_source_model(source_model, source_prompt):
    # source_model: ソースモデルのインスタンス
    # source_prompt: ソースプロンプトのテキスト
    # モデルにプロンプトを供給し、隠れ状態を得る
    hidden_states = source_model(source_prompt)
    return hidden_states


# ステップ2: ソースモデルの隠れ状態への任意の変換を適用します（オプショナル）
def apply_transformation(hidden_states, transformation):
    # hidden_states: ステップ1からの隠れ状態
    # transformation: 適用する変換の関数
    # 隠れ状態に変換を適用
    transformed_hidden_states = transformation(hidden_states)
    return transformed_hidden_states


# ステップ3: ターゲットプロンプトをターゲットモデルに供給し、ターゲットレイヤーまでの実行を行います。
def execute_target_model(target_model, target_prompt):
    # target_model: ターゲットモデルのインスタンス
    # target_prompt: ターゲットプロンプトのテキスト
    # モデルにプロンプトを供給し、ターゲットレイヤーまでの実行を行う
    target_layer_output = target_model(target_prompt)
    return target_layer_output


# ステップ4: ステップ2で変換された表現を、ターゲットモデルのターゲットレイヤーに「パッチ」します。
def patch_target_layer(target_model, target_layer, transformed_hidden_states):
    # target_model: ターゲットモデルのインスタンス
    # target_layer: パッチを適用するターゲットレイヤー
    # transformed_hidden_states: 変換された隠れ状態
    # ターゲットレイヤーにパッチを適用し、そのレイヤーから先の計算を継続
    patched_output = target_model.patch_layer(target_layer, transformed_hidden_states)
    return patched_output


# 以下は、Patchscopesアルゴリズムを使用する際の例です。
# 注意: このコードは概念的なものであり、具体的なモデルや変換関数は含まれていません。
```

Fact-Checking the Output of Large Language Models via Token-Level Uncertainty Quantification 大規模言語モデルの出力に対するトークンレベルの不確実性定量化によるファクトチェック 2024

概要

LLMが生成する出力における事実的な誤りや「幻覚」を検出する新しい手法を提案。
生成されたテキスト内の個々の主張の事実チェックを可能にするトークンレベルの不確実性定量化に基づく新しいパイプラインを開発。
この方法では、ニューラルネットワークの出力やその層に含まれる情報を利用して、信頼できない予測を検出します。さらに、特定の主張値がモデルによって表現される不確実性のみを測定する新しいトークンレベルの不確実性
定量化手法であるClaim-Conditioned Probability (CCP) を紹介し、複数のLLMsと三つの言語での実験において、CCPがベースラインを上回る改善を実証しました。

手法

主張条件付き確率（Claim-Conditioned Probability、CCP）は、主張レベルの不確実性を特定し、
大規模言語モデル（LLM）が生成したテキスト内の誤情報や
「幻覚」を検出するために使用される手法です。このアルゴリズムは、
生成されたテキストの各トークンに対する不確実性スコアを計算し、
これらを集約して主張全体の不確実性を評価します。
具体的には、CCPはモデルが特定の主張を生成する際に直面する不確実性の種類に焦点を当て、
主張タイプや表現形式の不確実性を無視することで、より正確な不確実性評価を行います。
このプロセスは、モデルの生成した各単語の代替案を評価し、
それらが元の主張と同じ意味を持つかどうかを判断することによって実行されます。
この手法は、特に自動ファクトチェックや情報の信頼性評価において有用です。

CCPの処理の流れは、生成されたテキストの各トークンに対して、
そのトークンが特定の主張の文脈において生成される確率を計算することに基づいています。
これは、生成されたテキスト内の特定の情報片に対するモデルの自信の度合いを
測定するために使用されます。流れとしては、まずテキストをトークンに分割し、
それぞれのトークンについて、それが特定の主張を表す文脈でどの程度確実に生成されるかを評価します。
これにより、テキスト内の各部分の不確実性スコアが得られ、
最終的にはこれらのスコアを用いて、テキスト全体または特定の部分の信頼性を判断します。

```
# クレーム条件付き確率（ CCP）アルゴリズムの実装
import numpy as np

# LLMが生成したテキストから原子的な主張を抽出する関数
def extract_claims(text):
    # ここでは、テキストから主張を抽出する具体的な方法は示されていません。
    # 実際の実装では、NLPツールを使用してテキストを解析し、主張を抽出します。
    # ここでは、抽象的な例として、テキストを文に分割し、各文を主張として返すことにします。
    claims = text.split('.')
    return [claim.strip() for claim in claims if claim.strip() != '']

# 主張の不確実性を計算する関数
def calculate_uncertainty(claims):
    # この関数では、各主張の不確実性を計算します。
    # 実際の実装では、モデルの出力から不確実性を計算するために、
    # 確率分布や信頼区間などを使用します。
    # ここでは、簡易的にランダムな不確実性値を返すことにします。
    uncertainties = [np.random.random() for _ in claims]
    return uncertainties

# CCPアルゴリズムの実行関数
def ccp_algorithm(text):
    # 1. テキストから主張を抽出
    claims = extract_claims(text)
    # 2. 各主張の不確実性を計算
    uncertainties = calculate_uncertainty(claims)
    # ここでは、不確実性と主張を組み合わせた結果を出力しています。
    # 実際のアプリケーションでは、この不確実性を使用してさらなる分析を行うかもしれません。
    for claim, uncertainty in zip(claims, uncertainties):
        print(f'主張: "{claim}", 不確実性 : {uncertainty}')
# テキスト例
example_text = "LLMが生成したテキストの例文です。ここには複数の主張が含まれています。    "

# CCPアルゴリズムを実行
ccp_algorithm(example_text)
```

結果

伝記生成タスクにおいて、6つの異なるLLMsと3つの言語で実施した実験は、CCPがベースラインと比較して大幅に改善されていることを示しました。人間による評価では、CCPに基づくファクトチェックパイプラインが外部知識を活用するファクトチェックツールと競合することが明らかになりました

ACLSum: A New Dataset for Aspect-based Summarization of Scientific Publications 科学論文のアスペクトベース要約のための新しいデータセット「ACLSum」2024

概要

科学論文の要約を目的とした新しいデータセットACLSUMを提案。これまでの要約データセットは、
半自動的に生成されたものが多く、質が低下することが問題となっていた。
ACLSUMは、専門家によって慎重に作成された多面的な要約を含むことで、この問題に対処している。
具体的には、科学論文から「課題」「アプローチ」「結果」の3つの異なる側面を要約する。
提案手法は、事前学習された言語モデルを用いて、抽出的要約と生成的要約の有効性を評価する
実験を行い、抽出的要約に比べて生成的要約の方が優れていることを示している
https://github.com/sobamchan/aclsum

手法

ACLSUMは、科学論文の要約のための新しいデータセットであり、
そのアルゴリズムは次のように説明されます。まず、ある科学論文から課題、アプローチ、結果という
3つの異なる側面に基づいて要約を生成します。
このプロセスは、ドメインの専門家によって慎重に作成され、
評価されたデータセットを使用して行われます。各文書は、抽出的要約と生成的要約の両方のために、
手作業で作成され、検証された要約で補完されます。
まず、注釈者はソース文書内の側面に関連する文を選択し、
次にこれらを使用して抽象的な要約を生成します。
このプロセスにより、各ソース文書には2種類のゴールドスタンダードの注釈が付けられます。
1つは各側面に関連する文のセット、もう1つは抽象的な参照要約です。

結果

広範囲にわたる実験を通じて、事前訓練された言語モデルとLLMに基づくモデルの品質と
パフォーマンスを評価しました。また、自動的に発見されたアスペクトに基づいて、
学術領域における抽出的対抽象的要約の有効性を探求しました。

```
# ACLSUMデータセットに基づく科学論文の要約アルゴリズム
# 必要なライブラリをインポート
import numpy as np

# 科学論文の要約関数を定義
def summarize_scientific_paper(document, dataset):
    # 文書とデータセットを受け取り、要約を生成
    # 課題、アプローチ、結果の3つの側面に基づいて要約を生成する
    # 1. 注釈者によって選択された側面に関連する文を抽出
    challenges, approaches, results = extract_aspects(document, dataset)
    # 2. 抽出した文を基に抽象的な要約を生成
    abstract_summary = generate_abstract_summary(challenges, approaches, results)
    # 生成した要約を返す
    return abstract_summary

# 側面に関連する文を抽出する関数
def extract_aspects(document, dataset):
    # データセットから文書に関連する注釈された文を抽出
    # このコードでは模擬的な処理を行う
    challenges = ['挑戦に関連する文1', '挑戦に関連する文2']
    approaches = ['アプローチに関連する文1', 'アプローチに関連する文2']
    results = ['結果に関連する文1', '結果に関連する文2']
    return challenges, approaches, results

# 抽象的な要約を生成する関数
def generate_abstract_summary(challenges, approaches, results):
    # 抽出した文を統合して一つの抽象的な要約を生成
    # このコードでは模擬的な処理を行う
    abstract_summary = '挑戦: ' + ', '.join(challenges) + '. アプローチ: ' + ', '.join(approaches) +
'. 結果: ' + ', '.join(results) + '.'
    return abstract_summary

# サンプルの科学論文とデータセットを定義
sample_document = 'サンプルの科学論文テキスト'
sample_dataset = 'ACLSUMデータセット'
# 要約を生成
summary = summarize_scientific_paper(sample_document, sample_dataset)
# 生成した要約を表示
print(summary)
```

Metacognitive Retrieval-Augmented Large Language Models メタ認知的検索拡張ラージ言語モデル 2024

概要

従来の統計的フレーズや階層的機械翻訳システムが直面している課題に対処するために、
MetaRAG（Metacognitive Retrieval-Augmented Large Language Models）という新しいアプローチを提案しています。
この方法は、多段階推論タスクにおいて、従来の単一時点検索を用いた手法よりも優れた性能を発揮します。
特に、低リソース言語の翻訳タスクにおいて、より少ない訓練データで既存の方法よりも一貫してBLEUスコアで優れた結果を示しています。
MetaRAGは、認知心理学から導入されたメタ認知を統合することで、モデルが自己反省と認知プロセスの評価を行い、
応答戦略を監視、評価、計画することを可能にします。
この三段階のメタ認知規制パイプラインを通じて、モデルは初期の認知応答の不備を特定し修正することができます。
実験の結果、MetaRAGは既存の方法を有意に上回ることが示されました。

手法

MetaRAG（Metacognitive Retrieval-Augmented Large Language Models）は、
従来の検索拡張生成問題を解決するために提案された新しいアプローチです。この手法は、認知心理学から着想を得たメタ認知の概念を統合し、
モデルが自身の認知プロセスをモニタリングし、評価し、計画することを可能にします。
メタ認知規制パイプラインを通じて、モデルは初期の認知応答の不備を特定し、修正することができます。
具体的には、メタRAGは以下の三つの主要なステップで構成されます：
(1)モニタリングは、現在の応答の品質を評価し、メタ認知評価を起動するかどうかを決定します。
(2)評価では、モデルはメタ認知知識を活用して応答の欠陥を分析し、内部および外部の知識の十分性や調和、
多段階推論の信頼性と正確性に関する問題を特定します。
(3)計画では、評価段階で特定されたシナリオに応じて、認知コンポーネントに対する具体的な改善提案を行います。

このプロセスを通じて、MetaRAGは従来の方法よりも優れた性能を示し、特に多段階推論タスクにおいて有効であることが実証されています。

結果

MetaRAGは、複数のホップ質問応答（QA）データセットにおいて、従来の手法よりも高い推論能力を示し、有意に性能が向上した。

```
# MetaRAG（Metacognitive Retrieval-Augmented Large Language Models）のアルゴリズムを実装
# モジュールをインポート
import numpy as np

# モニタリングステップ
# モデルの現在の応答品質を評価し、メタ認知的評価を起動するかどうかを決定
def monitoring_step(response_quality):
    # 応答品質が基準を満たしているかどうかをチェック
    if response_quality >= threshold:
        return True
    else:
        return False

# 評価ステップ
# 現在の回答が要件を満たさない理由を特定し、情報源の十分性等を分析
def evaluation_step(current_response):
    # 要件を満たさない理由の分析
    issues_identified = analyze_response(current_response)
    # 情報源の十分性と調和を評価
    sources_sufficiency = evaluate_sources(current_response)
    # 多段階推論の信頼性と正確性を分析
    reasoning_reliability = analyze_reasoning(current_response)
    return issues_identified, sources_sufficiency, reasoning_reliability

# 計画ステップ
# 評価段階で特定されたシナリオごとに、認知コンポーネントに対する改善の提案
def planning_step(identified_issues):
    # 改善案を生成
    improvement_suggestions = generate_improvements(identified_issues)
    # 改善案に基づいて応答を修正
    modified_response = modify_response(improvement_suggestions)
    return modified_response

# メタ認知的規制パイプラインの実行
def metacognitive_regulation_pipeline(initial_response):
    response_quality = evaluate_response_quality(initial_response)
    if monitoring_step(response_quality):
        current_response = initial_response
        issues_identified, sources_sufficiency, reasoning_reliability = evaluation_step(current_response)
        improvement_suggestions = planning_step(issues_identified)
        final_response = modify_response(improvement_suggestions)
        return final_response
    else:
        # 応答品質が基準を満たしていなければ、改善案を検討
        return '応答品質が基準を満たしていません。'

# 実行例
# 初期応答の品質を評価（例として、仮の関数と値を使用）
threshold = 0.5 # 品質の基準値
initial_response = '仮の初期応答'
def evaluate_response_quality(response):
    # 仮の応答品質評価関数
    return np.random.random() # 0から1のランダムな値を返す
def analyze_response(response):
    # 応答分析の仮の関数
    return ['要件を満たさない理由']
def evaluate_sources(response):
    # 情報源評価の仮の関数
    return True
def analyze_reasoning(response):
    # 推論分析の仮の関数
    return True
def generate_improvements(issues):
    # 改善案生成の仮の関数
    return ['改善案']
def modify_response(suggestions):
    # 応答修正の仮の関数
    return '修正された応答'
# パイプラインを実行して最終的な応答を取得
final_response = metacognitive_regulation_pipeline(initial_response)
print(final_response)
```

Can Large Language Models Automatically Score Proficiency of Written Essays? 2024

概要

LLMが書かれたエッセイの能力を評価できるかを検証しています。ChatGPTとLlamaという2つの人気のあるLLMを使用し、これらのモデルが自動エッセイスコアリング（AES）タスクをどのように実行できるか、そしてそれらのパフォーマンスが現在の最先端（SOTA）モデルと比較してどのように位置付けられるかを調査。プロンプトエンジニアリングを用いて設計された7つの異なるプロンプトを使用してLLMの最大のポテンシャルを引き出すことを目的とし実験はASAPデータセット上で行われ、いくつかの興味深い観察結果が明らかにされました。
特に、適切なプロンプトの選択がモデルの性能に大きく依存していること、そして2つのLLMがAESで比較可能な平均パフォーマンスを示したが、SOTAモデルとの間には予測性能にギャップがあるにもかかわらず、エッセイの質を向上させるフィードバックを提供することができる可能性があるようです。

手法

プロンプトエンジニアリング戦術の詳細な処理アルゴリズムは、自然言語処理タスク、特に大規模言語モデル(LLMs)を使用する場合に、効果的なプロンプトを設計するための一連のステップと原則に基づいています。この手法は、モデルがタスクの指示をより正確に理解し、期待される出力を生成できるようにすることを目指しています。具体的な処理は以下の通りです。

1. 入力とタスクの明確化 プロンプトの最初の部分で、モデルに何を求めているのかを明確に説明します。これには、タスクの目的、期待される出力の形式（例：テキスト、数値）、およびその他の重要な指示が含まれます。
2. 適切な区切り文字の使用 入力データ、ルーブリック、例示（one-shotまたはfew-shot例示が含まれる場合）など、異なる種類の情報を区別するために、適切な区切り文字（例：<>、```）を使用します。これにより、モデルが各部分を正確に識別し、誤ったプロンプト注入を防ぐことができます。
3. 逐次的指示の提供 モデルがタスクを段階的に処理できるように、逐次的な指示をプロンプトに含めます。これには、具体的なステップやアクションが含まれ、モデルが順を追ってタスクを完了するのを助けます。
4. 入力テキストの前処理指示の明確化 特に匿名化トークンなど、入力テキストに特定の前処理が必要な場合は、その処理方法について明確な指示を提供します。これにより、モデルが入力データを適切に処理し、タスクの実行に必要な情報を正確に抽出できるようになります。
5. 期待される出力形式の指定 モデルによる出力が特定の形式（例：JSON形式での応答）であることを要求する場合は、その形式を明確に指定し、出力がその要件を満たすようにします。

これらの手順と原則に従ってプロンプトを設計することで、大規模言語モデルがタスクの指示をより正確に理解し、期待される出力をより効果的に生成することが可能になります。

結果

適切なプロンプトの選択がモデルの性能に大きく依存していることがわかりました。特に、Llamaはプロンプトの選択に敏感であり、ChatGPTは比較的一貫性のある性能を示しました。
しかし、ChatGPTとLlamaの両方が、スコア予測の面でSOTAモデルに比べて遅れをとっていることが明らかになりました。
その平均Quadratic Weighted Kappa（QWK）スコアは、ChatGPTが0.313、Llamaが0.297であり、これはSOTAモデルのスコア（平均QWKが0.817や0.695など）よりも低いです。

Explaining Code with a Purpose: An Integrated Approach for Developing Code Comprehension and Prompting Skills 2024

概要

プログラミングを学ぶ初心者がコードの読解、理解、説明するための重要なスキルを開発するための統合アプローチについてExplain in plain English 平易な英語で説明する。(EiPE)の質問を使用してコード理解能力を開発・評価する方法にあります。EiPEの質問では、学生はコードの断片の目的を簡潔に説明します。しかしEiPEの質問の採点は、書かれた説明を評価する主観性が高いため、常に困難でした。本研究ではEiPEの質問とコード生成LLMとの間に自然なシナジーを利用して、この限界を克服する方法を提案しています'。'Methodology': '提案された方法は、学生のEiPEの質問への反応に基づいてLLMを使用してコードを生成し、EiPEの反応を自動的に評価することにより、学生が重要なコード理解とプロンプト作成スキルを平行して開発できるようにします。このアイデアを初級プログラミングコースで調査し、学生からEiPEの質問を解決するための効果的なプロンプトを作成することの成功を報告し、この活動がLLMを使用して学習を支援および評価することに対する学生の見解にどのように影響するかを調べます。'Results': '本研究は、提案された方法が、低リソース言語の翻訳タスクにおいて、既存の方法よりも一貫してBLEUスコアで優れており、より少ない訓練データで高い性能を達成できることを示しています。また、言語学の分野からの因子化文法を使用してTAG英文法からより一般的な翻訳ルールを提案しています。

手法

LLMを使用したEiPEの質問応答に基づくコード生成のアルゴリズムは、学生がコードの断片の目的を自然言語で説明するEiPE（Explain in Plain English）の質問に応答するプロセスを自動化し、その説明をLM（Large Language Model）に入力として提供し、LLMが生成したコードが元のコードと機能的に等価であるかを自動評価する方法である。このプロセスは、学生がコード理解能力とLMを用いた適切なプロンプト作成スキルを並行して開発するのを助ける。具体的な手順は以下の通りである。

1. 学生にコードの断片を提示し、その目的を自然言語で説明させる。
2. 学生の説明をLLMへの入力プロンプトとして使用し、LLMにコードを生成させる。
3. LLMによって生成されたコードがテストスイートを使用して元のコードと機能的に等価であるかを自動的に評価する。
4. この評価プロセスを通じて、学生のコード理解スキルとLMを用いたプロンプト作成能力が強化される。このアプローチはEiPEの質問に対する自動評価の難しさを克服し、学生がより深いコード理解を促進する助けとなる。

結果

このアプローチは、学生がEiPE質問に対して効果的なプロンプトを作成するのに成功したこと、およびLLMsを使用して学習を支援および評価することに対する学生の認識を調査しました。

AutoGuide: Automated Generation and Selection of State-Aware Guidelines for Large Language Model Agents 2024

概要

特に事前に訓練されたLLMが十分な知識を持たないドメインにおいて、
意思決定タスクのための状態を認識したガイドラインを生成・選択する大規模言語モデル(LLM)の能力を
強化するために設計された新しいフレームワークであるAutoGuideを紹介する。
AutoGuideは、オフラインの経験を活用することで、知識のギャップを効果的に埋めLLMエージェントが、
ウェブナビゲーションやインタラクティブなウェブ環境など、様々なドメインにわたる
逐次的な意思決定ベンチマークでより良いパフォーマンスを発揮できるようにする。
この方法論は、オフラインデータに埋め込まれた知識を抽出して、状態を認識するガイドラインを生成し、
エージェントの現在の状態に関連し、行動可能な情報を提供することによって、
LLMエージェントのパフォーマンスを向上させるために適用される。

手法

AutoGuideは、オフラインデータに埋め込まれた暗黙の知識を活用し、
状態認識ガイドラインを自動的に抽出することで、
エージェントのパフォーマンスを向上させます。具体的にはAutoGuideは次の手順で機能します。
1. オフライン経験から状態認識ガイドラインの抽出 AutoGuideは、オフライン経験から成功と失敗の
トラジェクトリを分析し、それらの違いから状態に特有の行動指針を導き出します。
2. 状態の要約とガイドラインの生成：各状態に対して、
自然言語で簡潔に表現された条件付きのガイドラインを生成します。
これにより、エージェントは現在の意思決定プロセスに関連する有用な知識を得ることができます。
3. テスト時におけるガイドラインの適用：エージェントが新しいタスクに直面した際、
AutoGuideは現在の状態を識別し、対応するガイドラインをプロンプトに組み込むことで、
エージェントが適切な行動を選択するのを支援します。このプロセスを通じて、
AutoGuideはLLMベースのエージェントが、
知識が限られているドメイン内でより成功率の高い意思決定を行うのを助けることができます。

```
import numpy as np
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# オフライン経験データセット(ダミーデータ)
experience_data = [
{'state': '状態A', 'action': '行動A', 'result': '成功'},
{'state': '状態A', 'action': '行動B', 'result': '失敗'},
{'state': '状態B', 'action': '行動A', 'result': '失敗'},
{'state': '状態B', 'action': '行動B', 'result': '成功'},
# ... その他の経験データ
]

# 1. オフライン経験から状態認識ガイドラインの抽出
# 成功と失敗のトラジェクトリを分析

# 成功トラジェクトリと失敗トラジェクトリを別々に格納
success_experiences = [d for d in experience_data if d['result'] == '成功']
failure_experiences = [d for d in experience_data if d['result'] == '失敗']

# トラジェクトリから状態と行動のペアを抽出
state_action_pairs = [(d['state'], d['action']) for d in experience_data]

# 2. 状態の要約とガイドラインの生成
# オフラインデータのテキスト情報をベクトル化
lvectorizer = TfidfVectorizer()
state_action_matrix = vectorizer.fit_transform([' '.join(pair) for pair in state_action_pairs])

# 各状態に対するガイドラインを生成
state_guidelines = {}
for state, _ in state_action_pairs:
relevant_experiences = [exp for exp in experience_data if exp['state'] == state]
if not relevant_experiences:
continue
descriptions = [f"行動[exp['action']]: {exp['result']}" for exp in relevant_experiences]
state_guidelines[state] = ' '.join(descriptions)

# 3. テスト時におけるガイドラインの適用
# 新しいタスクに直面した際の状態の識別(ダミー状態)
current_state = '状態A'

# 対応するガイドラインの適用
if current_state in state_guidelines:
print(f"現在のガイドライン: {state_guidelines[current_state]}\n")
else:
print("該当するガイドラインが見つかりません\n")
# この実装は、AutoGuideフレームワークの概念的な再現であり、
# 実際の大規模言語モデル(LLM)や特定のドメインのデータに基づく具体的な実装ではありません。
```

Decoding Complexity: Exploring Human-AI Concordance in Qualitative Coding 複雑さの解読：質的コーディングにおける人間と AIの一致を探る 2024

概要

質的データ分析 (QDA)のためのコーディングプロセスにおいて、 LLMを使用する方法について考察しています。質的データ分析は、
例えばインタビューデータやオープンエンド型の調査回答など、構造化されていないデータの中から、繰り返し現れるテーマやパターンに基づいて
データセグメントを識別、注釈付け、カテゴライズするプロセスです。この研究では、 GPT-3.5とGPT-4を用いて、
異なる複雑さのコーディングタスクにおける LLMsの統合を実験し、人間のコーダーとの間でコーディングの
一致度（Inter-Rater Reliability, IRR）を評価しています。タスクは、インターネットに接続されたデバイスの識別、アプリやサービスの使用、
デジタルセキュリティやプライバシーに関する信頼できる情報源の探求の 3つに分けられ、それぞれ異なるレベルの解釈を必要とします。
結果として、GPT-4はすべてのタスクで GPT-3.5よりも優れており、特にタスク Aでは人間とほぼ完璧な一致を達成しました。
しかし、タスクの複雑さが増すにつれて、人間とモデル間の一致度は低下しました。

手法

LLMsを用いた質的データ分析のコーディングのアルゴリズムは、共同作業によって質的データからの洞察を探求し、分析するためのプロセスです。
このプロセスは、大量の質的データ（例えば、インタビューデータやオープンエンドの調査回答など）を体系的に識別、注釈付け、
カテゴリー分けすることにより、反復するテーマやパターンに基づいてデータセグメントに適切なカテゴリーまたは「コード」を割り当てる
コーディングという基本ステップから始まります。大規模言語モデル（ LLMs）の統合によって、このコーディングプロセスをより
効率的に行うことができますが、 LLMsと人間のコーダーの間でコードブックや
文脈の広範な理解における挑戦があります。この方法では、 LLMsが異なる複雑さのコーディングタスクに
どのように適用できるか、そしてそれが人間のコーダーとどのように比較されるかを調査します。
特に、セマンティック（表面的な意味）とラテント（深層の意味）のテーマのコーディングにおける LLMsの性能に
ついて検討し、実際のデータセットにおける LLMsの支援による QDAの実装戦略を開発し、
GPT-3.5やGPT-4などのモデルを使用してインタビューデータにタグを適用します。
その後、人間のコーディングとの比較を行い、異なるレベルの解釈を要求するタスクでのモデルの性能を評価します。

結果

タスクAおよびBにおいて、人間のコーダー間でほぼ完全な一致を達成し、タスクCではかなりの一致を達成しました。
GPT-4は一貫してその前身よりも優れたパフォーマンスを示しました。タスクAにおいて、
GPT-4は人間のアノテーションとほぼ完全に一致し、人間間の合意と比較可能です。

タスクA (インターネット接続デバイス):
このコーディングタスクでは、参加者が使用するインターネット接続デバイスを識別することが目的です。
セマンティックコードの割り当てが人間とLMの両方にとって直接的であり、
インタビューされた人が伝えたエンティティの識別のみを要求します。（平均セグメント長8語; コードブック長 :18コード）

タスクB (アプリ、プログラム、サービス、および使用事例
このコーディングタスクは、参加者がインターネット接続デバイスで使用するアプリ、プログラム、サービス、
およびその目的に焦点を当てます。このタスクは、参加者が異なる方法で彼らの相互作用を表現するかもしれないという
複雑さの層を導入します。これには、個々のアプリの列挙、アプリケーションのグルーピング、
または使用事例の明示的な記述が含まれる場合があります。この変動性はデータに階層を導入し、
セマンティックおよび潜在的なコーディングの両方を要求します。（平均セグメント長74語; コードブック長 :24コード）

タスクC (信頼できる情報源:
このコーディングタスクは、参加者のデジタルセキュリティおよびプライバシーに関するガイダンスを求める際の慣行と情報源を探ります。
このタスクは、データのセマンティックコンテンツを捉えることを超え、基本的な考えや仮定の解釈を必要とします。
（平均セグメント長 :469語; コードブック長 :32コード）

```
# LLMsを用いた質的データ分析のコーディングのアルゴリズムの実装
# 必要なライブラリのインポート
import pandas as pd
import numpy as np
from transformers import pipeline
# データの準備
# この例では、架空のインタビューデータを使用します。
data = [
    {'id': 1, 'text': 'インタビュー回答1'},
    {'id': 2, 'text': 'インタビュー回答2'},
    # 以下、データを追加
]

# データフレームに変換
interview_data = pd.DataFrame(data)
# LLM（例えば、GPT-3.5）のロード
# この例では、transformersライブラリのpipelineを使っています。
# 実際には、適切なAPIキーとエンドポイントが必要です。
llm = pipeline('text-classification', model='gpt-3.5')
# コードブックの定義
# コードブックは、質的データに適用するカテゴリーやコードの集まりです。
codebook = {
    'テーマ1': ['キーワード1', 'キーワード2'],
    'テーマ2': ['キーワード3', 'キーワード4'],
    # 以下、テーマを追加
}
# コーディングプロセス
# インタビューデータに対して、各回答をコードブックのテーマに基づいて分類
for index, row in interview_data.iterrows():
    response = llm(row['text'])
    # LLMの出力から、最も適切なテーマを選択
    # ここでは簡単のため、ダミーの処理を行います。
    selected_theme = 'ダミーテーマ'
    # 選択されたテーマをデータフレームに追加
    interview_data.at[index, 'theme'] = selected_theme
# コーディングの結果の表示
print(interview_data)
# 人間のコーダーとの比較
# 実際には、人間のコーディング結果と比較してLMsの性能を評価します。
# この例では、比較のためのダミーデータを使用します。
human_coding = [
    {'id': 1, 'theme': '人間のテーマ1'},
    {'id': 2, 'theme': '人間のテーマ2'},
    # 以下、データを追加
]
# 性能の評価
# ここでは、シンプルに正解率を計算します。
# 実際の研究では、より詳細な分析が必要です。
match_count = 0
for llm_coding, human_coding_row in zip(interview_data.to_dict('records'),
human_coding):
    if llm_coding['theme'] == human_coding_row['theme']:
        match_count += 1
accuracy = match_count / len(interview_data)
print(f'正解率 {accuracy}')
```

ToolRerank: Adaptive and Hierarchy-Aware Reranking for Tool Retrieval ToolRerank: ツール検索のための適応的かつ階層認識型の再ランキング2024

概要

LLMを外部ツールで拡張するツール学習に焦点を当てます。課従来の方法では、見たツールと見ていないツールの違いやツールライブラリの階層を考慮していないため、
ツール検索の性能が最適ではありませんでした。これらの問題に対処するために、本研究では「 ToolRerank」という新しいアプローチを提案。
この方法は、ツール検索のための適応的かつ階層意識的な再ランキング手法であり、検索結果をさらに洗練させます。具体的には、「 Adaptive Truncation」が提案され、
これは既知のツールと未知のツールに関連する検索結果を異なる位置で切り捨てることにより、再ランキングの性能を向上させるものです。
さらに、「Hierarchy-Aware Reranking」を用いて、単一ツールクエリに対しては結果をより集中させ、複数ツールクエリに対してはより多様な結果を提供します。
実験結果は、ToolRerankが検索結果の品質を向上させ、LLMによって生成される実行結果を改善できることを示しています。
https://github.com/XiaoMi/ToolRerank からだけど

手法

ToolRerankは、ツール検索のための適応型かつ階層認識型の再ランキング方法です。
このアルゴリズムは、見たツールと見ていないツールを異なる位置で切り捨てる Adaptive Truncationと、
単一ツールクエリに対して検索結果をより集中させ、複数ツールクエリに対してはより多様にする Hierarchy-Aware Rerankingの
ニつの主要なコンポーネントを含みます。具体的には、まず、ユーザークエリに基づいて、
デュアルエンコーダーリトリバーを使用して粗い検索結果を取得します。
次に、Adaptive Truncationを適用して、見たツールと見ていないツールに関連する結果を異なる位置で切り捨てます。
その後、クロスエンコーダーリランカーを使用して結果を再ランク付けし、Hierarchy-Aware Rerankingを適用して、
階層の構造を考慮に入れながら、検索結果をさらに細かく再ランク付けします。
このプロセスにより、LLMによって生成される実行結果の品質が向上し、より適切なツールが検索されるようになります。

Adaptive Truncation（適応的切断）のアルゴリズムは、
ツール検索の結果を見たツールと見ていないツールに基づいて異なる位置で切り捨てることにより、
再ランキングの精度を向上させることを目的としています。このプロセスは、
再ランキングに供給される候補の数を動的に調整することで、特に未知のツールに対する検索性能を改善します。

1. 設定：ニつの異なるしきい値、ms と mu (ms < mu) を設定します。ms は既知のツール（訓練データで見たツール）に
関連する結果を切り捨てる位置を指定し、mu は未知のツール（訓練データで見ていないツール）に関連する結果を切り捨てる位置を指定します。
2. 粗い検索結果の取得：まず、ユーザークエリに基づいて、デュアルエンコーダーなどの検索手法を使用して粗い検索結果のリスト C=[c1 ,c2,...,cm] を取得します。
3. 適応的切断の実行：
- 各Ci について、Ci が属するツールが訓練データで見たものであれば、i<=ms の場合に限り Ci を切り捨て後のリストTに追加します。
- Ci が属するツールが訓練データで見ていないものであれば、i<=mu の場合に限りCiをTに追加します。
このプロセスにより、T=[t1 ,t2,...,tl]のリストが得られ、ここでl<=m です。この適応的な切断により、
既知のツールに関連する結果はより厳選され、未知のツールに関連する結果はより広範囲から選ばれることになり、検索結果の精度と再ランキングの効果を最適化します。
4. 再ランキング：得られた切り捨て後のリスト T を使用して、クロスエンコーダーによる再ランキングを行い、最終的な検索結果を決定します。

Hierarchy-Aware Reranking（階層認識再ランキング）のアルゴリズムは、ツール検索の結果を、クエリが単一ツールに関連するものか、
複数のツールを使うものかに応じて、より効果的に再ランキングする手法です。このプロセスは、ツールライブラリの階層構造を考慮し、
クエリに最適なツールまたはツールの組み合わせを提供することを目指します。
1. クエリタイプの分類：クエリが単一ツールに関連するもの（ single-tool queries）か、複数のツールを要求するもの（ multi-tool queries）かを識別するために、
分類器を使用します。
2. 再ランキングの適用：単一ツールクエリに対しては、検索結果をそのクエリに関連すると分類されたツールの APIに集中させるように再ランキングします。
この目的で、クロスエンコーダーのスコアが高い APIを優先し、同一ツールの APIを上位にランク付けします。
複数ツールクエリに対しては、異なる機能を持つ複数のツールの APIが検索結果に含まれるように再ランキングします。
これには、ツール間の機能的な重複を減らすために、検索結果の多様性を高める手法が使用されます。

実装の詳細：
単一ツールクエリの場合：閾値（τs ）を設定し、この閾値を超えるスコアを持つ APIが属するツールを優先的に選択します。
これにより、関連性が高いと判断されたツールの APIが検索結果の上位にランク付けされます。
複数ツールクエリの場合：グラフを構築し、ツール間および API間の関連性を基にエッジを追加します。
このグラフを用いて、機能的に異なるツールが提供する APIが検索結果に均等に含まれるようにします

```
# ToolRerankアルゴリズムの実装
# 必要なライブラリのインポート
from typing import List, Dict

# データ型の定義
Tool = Dict[str, any] # ツール情報を格納する辞書型
SearchResult = List[Tool] # 検索結果のリスト型
# デュアルエンコーダーリトリバーによる粗い検索結果の取得
def retrieve_coarse_results(query: str) -> SearchResult:
    # 実装は省略（ダミーデータを返す）
    return [{'tool_name': 'ToolA', 'seen': False}, {'tool_name': 'ToolB', 'seen': True}]

# Adaptive Truncationの適用
def adaptive_truncation(search_results, seen_tools, ms, mu):
    """
    search_results: 検索結果のリスト（各要素はAPI名, ツール名)のタプル）
    seen_tools: 訓練データで見たツールのセット
    ms: 見たツールの切り捨て位置
    mu: 見ていないツールの切り捨て位置
    """
    truncated_results = []
    for i, (api, tool) in enumerate(search_results):
        # ツールが見たツールの場合
        if tool in seen_tools:
            if i < ms:  # msの位置までを保持
                truncated_results.append((api, tool))
        else:
            if i < mu:  # muの位置までを保持
                truncated_results.append((api, tool))
    return truncated_results

# 検索結果の例（API名, ツール名）
search_results = [
    ("API1", "ToolA"), ("API2", "ToolB"), ("API3", "ToolC"),
    ("API4", "ToolD"), ("API5", "ToolE"), ("API6", "ToolF")
]
# 訓練データで見たツールのリスト
seen_tools = {"ToolA", "ToolB", "ToolC"}
# 適応的切断を適用
ms = 3  # 見たツールのための切り捨て位置
mu = 5  # 見ていないツールのための切り捨て位置
truncated_results = adaptive_truncation(search_results, seen_tools, ms, mu)
print("切り捨て後の検索結果", truncated_results)

# クロスエンコーダーリランカーによる再ランク付け
def rerank_with_cross_encoder(results: SearchResult) -> SearchResult:
    # 実装は省略（ダミーデータを返す）
    return results
# Hierarchy-Aware Rerankingの適用
def apply_hierarchy_aware_reranking(results: SearchResult) -> SearchResult:
    # 実装は省略（ダミーデータを返す）
    return results
# ToolRerankアルゴリズムのメイン関数
def tool_rerank(query: str) -> SearchResult:
    # 粗い検索結果の取得
    coarse_results = retrieve_coarse_results(query)
    # Adaptive Truncationの適用
    truncated_results = adaptive_truncation(coarse_results)
    # 再ランク付け
    reranked_results = rerank_with_cross_encoder(truncated_results)
    # Hierarchy-Aware Rerankingの適用
    final_results = apply_hierarchy_aware_reranking(reranked_results)
    return final_results
# メイン関数の実行例
if __name__ == '__main__':
    query = "ツール検索クエリ"
    results = tool_rerank(query)
    print(results)
```

Process Modeling With Large Language Models 2024

概要

ビジネスプロセス管理（BPM）の領域において、プロセスモデリングは複雑なプロセスダイナミクスを理解しやすい視覚表現に変換し、組織プロセスの理解、分析、改善、自動化を容易にする重要な役割を果たす。伝統的なプロセスモデリング方法は、多くの場合専門知識を必要とし、時間がかかります。テキスト記述からプロセスモデルを自動生成および反復的に改善するためにLLMを利用することで、プロセスモデリングの柔軟性、効率性、およびアクセシビリティを高めることを目指しています。このフレームワークは、効果的なLLMの活用のための革新的なプロンプト戦略、安全なモデル生成プロトコル、およびエラーハンドリングメカニズムを含んでいます。また、フレームワークを具体的に実装したシステムを示し、生成されたモデルが標準モデリング記法（例えばBPMNやペトリネット）でエクスポートできることを示しています。

手法

LLMを活用し、プロンプトエンジニアリング、エラー処理、コード生成などの先進技術を用いて、自然言語のプロセス記述からプロセスモデルを自動生成するフレームワークを開発。

フレームワークの概要
- ユーザー入力: ユーザーはプロセスのテキスト記述を入力します。
- プロンプトの生成: テキスト記述を受け取った後、追加情報を組み込んで包括的なプロンプトを作成します。このプロンプトはLLMを指導し、プロセスモデルを生成するための実行可能なコードを生成させるように設計されています。
- コード生成: LLMによって生成されたプロンプトは、プロセスモデルの生成に役立つ一連の関数を利用してコードを生成するよう指示されます。
- コードの実行: LLMの応答を受け取った後、応答からコードスニペットを抽出し、実行してモデルを生成します。
- モデルの表示とエクスポート: 成功したコードの実行とプロセスモデルの生成後、ユーザーは生成されたモデルを見たりBPMNやPetriネットなどの確立されたプロセスモデリング記法でエクスポートしたりできます。
- フィードバックループ: ユーザーは生成されたモデルに対してフィードバックを提供でき、フレームワークはそのフィードバックを統合してモデルを精緻化します。

プロセス表現
- POWL言語: 中間プロセス表現としてPartially Ordered Workflow Language（POWL）を使用します。POWLは、BPMNやPetriネットといった標準モデリング記法に馴染みのあるプロフェッショナルにとって理解しやすいプロセスモデルを生成する目的があります。POWLは、階層的なプロセスモデルを再帰的に生成し、より大きなものに組み合わせることで、簡素化されたモデル生成を可能にします。

プロンプトエンジニアリング
- ロールプロンプト: LLMに特定の役割を割り当て、プロセスモデリングの専門家として振る舞わせ、提供されたプロセス記述のギャップを埋めるよう指導します。
- 知識注入: POWLに関する包括的な知識を提供し、LLMがPOWLモデルを生成するためのPythonコードを生成するよう指示します。
- フューショットとネガティブプロンプト: いくつかの入力と期待される出力のペアを提供してLLMを訓練し、生成中に避けるべき一般的なエラーについて指示します。

モデル生成と精緻化
- 安全性と品質: 生成されたコードを実行する際に安全性と有効性を確保するための戦略を実装します。生成されたPOWLモデルをPetriネットやBPMNモデルに変換し、これらの確立された記法でモデルを表示およびエクスポートする機能を提供します。

エラー処理
- エラーの分類: 重大なエラーと調整可能なエラーに分類し、それぞれに対して適切な対応策を講じます。エラーが発生した場合にはLLMを再度活用してエラーを解決するための反復的なエラー処理ループを組み込みます。

Long-term Hydrothermal Bid-based Market Simulator 2024

概要

長期的な水力火力発電市場を戦略的エージェントを考慮してシミュレートすることは、挑戦的な課題です。時間的制約を伴う戦略的エージェントの処理は、既に困難な単期間の双レベル非凸最適入札問題をさらに複雑にします。大規模水力火力発電システムに対してこれらの課題を効果的に対処するシミュレーション方法論を提案。フレームワークの有効性を、大規模ブラジル電力システムの実データによるケーススタディを通じて検証。ケーススタディでは、電力システムにおける市場集中の影響と、それを緩和するための契約の使用法を示します。特に、ブラジルにおける市場力がどのように影響を与えるかを確認しました。開発した方法は、政策立案者、市場モニター、および市場設計者にとって大きな利益をもたらす可能性があります。シミュレーションは、既存の電力システムを理解し、代替設計を実験するために使用できます。

手法

LLMを活用し、プロンプトエンジニアリング、エラー処理、コード生成などの先進技術を用いて、自然言語のプロセス記述からプロセスモデルを自動生成するフレームワークを開発。

フレームワークの概要
- ユーザー入力: ユーザーはプロセスのテキスト記述を入力します。
- プロンプトの生成: テキスト記述を受け取った後、追加情報を組み込んで包括的なプロンプトを作成します。このプロンプトはLLMを指導し、プロセスモデルを生成するための実行可能なコードを生成させるように設計されています。
- コード生成: LLMによって生成されたプロンプトは、プロセスモデルの生成に役立つ一連の関数を利用してコードを生成するよう指示されます。
- コードの実行: LLMの応答を受け取った後、応答からコードスニペットを抽出し、実行してモデルを生成します。
- モデルの表示とエクスポート: 成功したコードの実行とプロセスモデルの生成後、ユーザーは生成されたモデルを見たりBPMNやPetriネットなどの確立されたプロセスモデリング記法でエクスポートしたりできます。
- フィードバックループ: ユーザーは生成されたモデルに対してフィードバックを提供でき、フレームワークはそのフィードバックを統合してモデルを精緻化します。

プロセス表現
- POWL言語: 中間プロセス表現としてPartially Ordered Workflow Language（POWL）を使用します。POWLは、BPMNやPetriネットといった標準モデリング記法に馴染みのあるプロフェッショナルにとって理解しやすいプロセスモデルを生成する目的があります。POWLは、階層的なプロセスモデルを再帰的に生成し、より大きなものに組み合わせることで、簡素化されたモデル生成を可能にします。

プロンプトエンジニアリング
- ロールプロンプト: LLMに特定の役割を割り当て、プロセスモデリングの専門家として振る舞わせ、提供されたプロセス記述のギャップを埋めるよう指導します。
- 知識注入: POWLに関する包括的な知識を提供し、LLMがPOWLモデルを生成するためのPythonコードを生成するよう指示します。
- フューショットとネガティブプロンプト: いくつかの入力と期待される出力のペアを提供してLLMを訓練し、生成中に避けるべき一般的なエラーについて指示します。

モデル生成と精緻化
- 安全性と品質: 生成されたコードを実行する際に安全性と有効性を確保するための戦略を実装します。生成されたPOWLモデルをPetriネットやBPMNモデルに変換し、これらの確立された記法でモデルを表示およびエクスポートする機能を提供します。

エラー処理
- エラーの分類: 重大なエラーと調整可能なエラーに分類し、それぞれに対して適切な対応策を講じます。エラーが発生した場合にはLLMを再度活用してエラーを解決するための反復的なエラー処理ループを組み込みます。

AesopAgent: Agent-driven Evolutionary System on Story-to-Video Production エイソップエージェント: 物語から動画への進化的システム 2024

概要

Agent技術とAIGC（Artificial Intelligence Generated Content）技術は最近、顕著な進歩を遂げています。我々は、物語から動画への生成を目的としたエイソップエージェントというAgent駆動の進化的システムを提案。エイソップエージェントシステムは、動画生成のためのタスクワークフローを最適化します。ユーティリティレイヤーは、構成、キャラクター、スタイルの観点から視覚的に一貫した画像生成を実現する複数のユーティリティを提供します。同時に、音声や特殊効果を提供し、それらを表現豊かで論理的に配置された動画に統合します。全体として、我々のエイソップエージェントは、視覚的ストーリーテリングにおいて多くの先行研究と比較して最先端のパフォーマンスを達成します。
https://aesopai.github.io/

手法

エイソップエージェントは、ユーザーの物語提案を動画に変換するプロセスを実装するためのビデオ生成ワークフローを受け入れます。このアーキテクチャは、水平レイヤーとユーティリティレイヤーの2つのレイヤーで構成されています。

水平レイヤーでは、エージェント技術を利用してビデオ生成ワークフローにおける高レベルの戦略的管理と最適化を体系的に実行します。このプロセスは、専門家の経験と専門知識を集約したE-RAGとK-RAGを構築してLLMプロンプトを強化し、ユーティリティレイヤーの要件をさらに高め、ユーティリティの使用を最適化します。

E-RAG（Experience-RAG）
E-RAGは、専門家の経験を蓄積し、それを利用してビデオ生成ワークフローを最適化するシステムです。具体的には、過去のプロジェクトやタスクから得られた知見や経験をデータベースに蓄積し、新しいビデオ生成タスクにおいて最適なプロセスや手法を提案するために使用しますE-RAGは、反復的なプロセスを通じて経験を更新し、ワークフローの効率化と品質向上に寄与します。

K-RAG（Knowledge-RAG）
K-RAGは、専門知識を集約し、それを利用してビデオ生成に関連するタスクをサポートするシステムです。このシステムは、専門的な知識や情報をデータベースに蓄積し、スクリプト生成、画像生成、ビデオ組み立てなどのタスクにおいて、最適なプロンプトやユーティリティの使用を案内しますK-RAGは、専門家から提供される知識文書をインデックス化し、それをベースにLLMのプロンプトを最適化することで、より専門的で一貫性のあるコンテンツ生成を可能にします。

ユーティリティレイヤーは、ワークフローの各ステップに特化したタスク固有の要件を満たすために、完全に機能するユーティリティのスイートを提供します。このレイヤーには、画像生成とビデオ組み立てのためのユーティリティが主に含まれています。

結果

エイソップエージェントの有効性は、物語の画像表現性とユーザーの関与の3つの主要な点で評価されます。ComicAIやArtflowなどのシステムとの比較分析により、エイソップエージェントが最先端の視覚的ストーリーテリング能力を持つことが示されました。具体的には、RAG技術を通じて表現と論理を通じて優れた性能を実現しています。またComicAIとの画像要素復元の合理性と構成に関する手動評価では、エイソップエージェントの優位性が示されました。NUWA-XLやAutoStoryなどの並行するビデオ生成研究と比較して、我々のシステムは画像の複雑さと物語の深さにおいてリードする性能を示しています。さらに、エイソップエージェントの適応性は、特定のユーザーのニーズに対応するために外部ソフトウェア（例Runway）を統合することによって証明され、その広範なスケーラビリティが強調されています。

Academically intelligent LLMs are not necessarily socially intelligent 2024

概要

人間の社会的知能フレームワークに触発され、特にダニエル・ゴールマンの社会的知能理論に基づいて、現実世界の社会シナリオに基づいた標準化された社会的知能テスト(SESI)を開発。
このテストを使用して、最近人気のある3のLLMエージェントを広範囲に評価しました。その結果、LLMの社会的知能は大きく改善の余地があること、そして社会的知能と学術的知能との間には相関関係が低いことが示されました。

手法

LLMの社会的知能を評価するためにいくつかの手法を使用しています。
特に、SESI(Situational Evaluation of Social Intelligence)の開発と社会的要因(性格、感情、性別、役割、視点)がLMの社会的知能に与える影響の検討が中心となっています。

SESI(Situational Evaluation of Social Intelligence)
1. 社会的コンテキストと問題の収集
- RedditのRelationshipsコミュニティから、現実世界の人間関係の相談を基にした社会的コンテキストと問題を収集します。
- 収集したデータから、GPT-3.5-turboモデルを使用して、社会的状況とそれに基づく質問を要約します。
2. 回答の収集
- 正解: 各ポストの下にある最も支持されている上位5つの回答を基に正解を決定します。
- 誤答: 問題を誤って切り替える答え(question-switching answers)と、正解とは異なる論理的な誤答(reversed answers)を生成します。
3. QAタプルの作成
- 社会的コンテキスト、質問、3つの誤答、1つの正答を組み合わせて、複数選択式のテスト質問を作成します。

社会的要因に基づく影響の評価
1. 性格、感情、性別、役割、視点の設定
- LLMに特定の性格、感情、性別、役割、視点を持たせるためのプロンプトを設計します。これにより、これらの社会的要因がLMの社会的知能にどのように影響するかを評価します。
2. プロンプトの使用
- 設計したプロンプトを用いて、LLMに特定の社会的コンテキストにおける質問に答えさせ、その回答を評価します。
3. 評価
- LLMが生成した回答を、SESIテストにおける正答と比較し、社会的知能の各側面(共感、社会的認知、自己提示、影響、関心)における性能を評価します。

結果

社会的知能が学術的知能とは異なる独立した知能の形態であること、LLMが固定された親しみやすいパターンに従って誤りを犯すことが主な原因であることなど、いくつかの重要な発見を報告しています。

CKERC : Joint Large Language Models with Commonsense Knowledge for Emotion Recognition in Conversation 共感情認識のためのコモンセンス知識を組み合わせた大規模言語モデル 2024

概要

会話における感情認識(Emotion recognition in conversation, ERC)は、会話の文脈における発言の感情を予測するタスクです。これは、対話の文脈、話者の身元情報、複数人間の対話シナリオなどに大きく依存しています。しかし、最先端の手法(instructERC)は話者を特定するだけで、会話中の話者の背後にあるコモンセンス知識（例：聞き手の反応や話者の意図など）を無視しています。これらの知識は、話者情報の深掘りに役立ちます。そこで、会話における感情認識のための新しいフレームワークであるCKERCを提案。LLMを使用して過去の発言に基づいた対話者のコモンセンスを生成するプロンプトを設計しLLMの事前トレーニングに対話者のコモンセンス識別タスクを使用して話者の暗黙の手がかり情報を微調整することで、上記の課題を解決し、最先端の成果を達成しました。

手法

CKERCのアルゴリズムは、LLMとコモンセンス知識を組み合わせて、会話における感情認識(ERC)を行う新しいフレームワークです。このアプローチは主に以下のステップで構成されます。
1. タスク定義
会話における感情認識(ERC)は、会話の中で特定の発言の感情ラベルを予測するプロセスです。このタスクは、ダイアログの文脈、話者のアイデンティティ、多人数のダイアログシナリオに依存します。

2. コモンセンス知識の選定と取得
コモンセンス知識として、ATOMICデータベースを用いて、人間の一般常識を記述する。特に「聞き手の反応」、「話者の反応」、「話者の意図」という3つの関係に焦点を当て、これらのコモンセンスをダイアログの文脈で生成します。

3. コモンセンスの生成
過去発言を考慮に入れたプロンプトを大規模言語モデルに与え、話者のコモンセンス知識（例：聞き手の反応、話者の意図）を生成します。このステップは、発言の文脈に応じて異なる感情を表現するために使用します。

4. コモンセンスを会話感情認識に導入
生成されたコモンセンス知識を、会話の感情認識タスクに導入します。このために、話者のコモンセンス情報を用いて、発言の感情ラベルをより正確に予測するためのフレームワークを構築します。

5. モデルの訓練
会話の感情認識タスクとして、話者のコモンセンス情報を用いた発言の感情ラベル予測を行うために、大規模言語モデルを訓練します。この訓練はERCタスクの精度を向上させるために、話者の暗黙の情報を深く掘り下げることを目的としています。

6. 評価
訓練されたモデルを用いて、実際の会話データセット（例：EMOCAP、MELD、EmoryNLP）に対する感情認識タスクのパフォーマンスを評価します。この評価はCKERCが提供する改善の程度を定量的に測定します。

結果

CKERCは、IEMOCAP、MELD、EmoryNLPという3つの人気のあるデータセットにおいて、最先端のパフォーマンスを達成しました。特にIEMOCAPでは1%ポイント、EmoryNLPでは1.2%ポイント、MELDでは0.2%ポイントの向上を達成しました。

Curry-DPO: Enhancing Alignment using Curriculum Learning & Ranked Preferences 2024

概要

直接好みの最適化（DPO）は、ペアごとの好みデータ（プロンプトごとに選択された応答と拒否された応答）を利用してLMを人間の好みに合わせる方法です。実際には、与えられたプロンプトに対して相対的な品質が異なる複数の応答が存在する可能性があることから、これらの応答を利用して、与えられたプロンプトに対する複数の好みペアを作成することを提案。
複数の好みペアを体系的にキュレーションし、カリキュラム学習を通じて意味のある方法で提示することに注目しています。

手法

与えられたプロンプトに対して相対的な品質が異なる複数の応答が存在する場合に、これらの応答を利用して複数の好みペアを作成する方法を提案しています。このアプローチは、カリキュラム学習の原理に基づいています。主に以下の手順で行われます。

1. 複数の応答のサンプリング: まず、各プロンプトに対して複数の応答が存在するという前提から始めます。これらの応答は、人間の評価者による品質評価や、モデル自体による予測スコアなど、異なる方法で評価されます。
2. 好みペアの作成: これらの応答を用いて、プロンプトごとに複数の好みペアを作成します。好みペアは、選択された応答（より高い品質と評価されるもの）と拒否された応答（より低い品質と評価されるもの）の組み合わせです。この過程では、全ての可能な応答ペアを検討し、それぞれのペアに対して品質の差異に基づいてランキングを行います。
3. カリキュラム学習の適用: カリキュラム学習の原則に従い、作成された好みペアを「簡単」から「難しい」へと順序立ててモデルに提示します。ここでいう「簡単」とは、選択された応答と拒否された応答の品質差が大きいペアのことを指し、「難しい」とは、その差が小さいペアを指します。この順序付けは、モデルが段階的により微妙な好みの違いを学習するのを助けます。
4. 学習の実施: 好みペアを用いたトレーニングは、複数の反復または「イテレーション」にわたって行われます。各イテレーションでは、特定の好みペアセットに基づいてモデルをトレーニングし、その後で次のイテレーションに進む前に参照モデルを更新します。このプロセスを通じて、モデルは徐々に複雑な好みの違いを認識するようになります。

この方法は、モデルが人間の好みに更に密接に合致するように微調整するための効果的な手段を提供します。人間の評価者による直接的なフィードバックを活用することで、モデルの応答品質を大幅に向上させることが可能になります。

結果

DPO方法であるCurry-DPOは、MT Bench、Wizard-LM、およびUltraFeedbackにおいて、標準の単一好みペアDPOよりも大幅に優れていることを示し、特にCurry-DPOはMT-benchで7.43のスコアを達成し、同様のパラメータサイズを持つ既存のLLMの大半を上回りました。

Thought Graph: Generating Thought Process for Biological Reasoning 2024

概要

Thought Graphは、遺伝子セット分析を例に生物学的プロセス間のセマンティックな関係を明らかにする新しいフレームワークを提案
このフレームワークは、GSEAを40.28%、LLMベースラインを5.38%上回ることにより、遺伝子セットのより深い理解を提供します。これは、人間のアノテーションへのコサイン類似度に基づいています。
https://github.com/ethan5437/thought-graph-www/

手法

Thought Graphフレームワークは、Tree-of-Thought (ToT)アーキテクチャを採用しています。Thought Graphフレームワークは、遺伝子セットを入力として受け取り、それに関連する用語（例えば、生物学的プロセスやパスウェイ）を表すツリー構造グラフを生成します。このグラフにおいて、ノードは生物学的プロセスを、エッジはこれらのプロセス間の依存関係を表しますhought Graphは、各ノードを独自の生物学的プロセスとして階層的に配置し、エッジを使用してこれらのプロセス間の関係を示します。
https://github.com/ethan5437/thought-graph-www/blob/main/baselines/cot.py

プロンプトは、分子生物学者の助手として効率的かつ洞察力を持って働くことを目的としています。具体的には、一連の遺伝子が関与する最も顕著な生物学的プロセスの簡潔な名前を提案することを目標としています。このプロンプトには、その達成のためのいくつかの指示が含まれています：
1. 簡潔性: 不要な言葉を使わず、簡潔に回答すること。
2. 具体性: 「遺伝子がさまざまな細胞プロセスに関与している」など、あまりにも一般的な表現を避けること。
3. 事実性: 編集せず、事実に基づいて回答すること。
4. 包括性: すべての遺伝子を含めること。
5. 包括的であるが一般化しすぎない: すべてを網羅しつつも、一般化しすぎないようにすること。
6. 主題にとどまる: 目標から逸脱しないこと。
具体的な例として、A2M, AHSG, APOL2, APCS など複数の遺伝子がリストアップされ、これらの遺伝子が主に関与している生物学的プロセスが免疫応答と炎症であることが示されています。さらに、サイトカイン、急性期タンパク質、凝固と止血、免疫細胞の調節とシグナリング、代謝とホルモン応答、細胞ストレスと損傷応答、痛みと温度感覚、プロスタグランディンの合成と調節、肝臓タンパク質と調節因子、鉄代謝と酸素輸送、内皮と血管機能など、さまざまな機能についての詳細な説明が含まれています。これらの遺伝子が集合的に負傷や感染への免疫系の反応を主に通じて、炎症の媒介と調節を果たす役割を果たしていることから、「炎症反応」という答えが導かれます。

プロンプトの最後では、具体的な遺伝子のセットを与えられた場合に、最も顕著な生物学的プロセスを決定するための質問形式が提案されています。

結果

Thought Graphは、GSEAとLLMベースラインを大幅に上回ることにより、生物学的プロセス生成において有意義な改善を達成しました。特にGSEAを40.28%、LLMベースラインを5.38%上回りました。

Let Storytelling Tell Vivid Stories: An Expressive and Fluent Multimodal Storyteller ストーリーテリングで鮮明な物語を語る: 表現力豊かで流暢なマルチモーダル・ストーリーテラー 2024

概要

このストーリーテリングは、順番に渡された画像に基づいて合理的で鮮明な物語を生成することを目指します。既存の作業は、複数のモダリティの整合性を反復的に改善しましたが、画像ストリームのための単純化された物語ラインを生成する結果となりました。この研究では、表現力と一貫性に体現されたマルチモーダル人間レベルの物語を生成するために、新しいパイプラインであるLLaMSを提案。
LLM内のコモンセンス知識を完全に活用して、事実コンテンツ表現を強化するためのシーケンスデータ自動強化戦略を採用し、表現力豊かな物語生成と予測のためのテキスト推論アーキテクチャを活用します。次に、シーケンス一貫性を維持できるSQ-Adapterモジュールを提案します。数値結果は、人間の評価を通じて提案されたLaMSの優位性を検証します。評価によると、LLaMSは、以前のSOTAメソッドと比較して、ストーリーテリングパフォーマンスにおいて最先端のパフォーマンスを達成し、86%の相関と100%の一貫性の勝率を達成しています。

手法

表現力豊かで一貫した物語を生成するために、二段階のアプローチを採用しています。最初に、シーケンスデータ強化を通じて高品質なストーリーデータを作成し、次に、テキスト推論アーキテクチャを用いて物語を生成します。さらに、SQ-Adapterモジュールを提案し、複数の画像にまたがるビジュアル一貫性を維持します。

シーケンスデータ強化について
シーケンスデータ強化は、VISTデータセットを使用して、物語のデータを自動的に書き換え、より豊かな視覚的意味論と物語の進化ロジックを統合する独自の戦略です。この強化データは、テキストストーリーテリングモデルがより表現力豊かな物語を生成する動機付けとなります。特にLLaVaを用いて物語の詳細を集め、視覚的意味論の詳細に焦点を当て、言語でそれらの高度に決定的な部分を記述します。このプロセスでは、「草原」、「白い家」、「若い少年」、「茶色の犬」、「ボールを投げる」といった豊かで詳細な意味論が明示的に表現されます。しかし、これらの説明は異なるプロット間の一貫性や読み続けるための魅力を欠いているため、独立したコンテンツを完全な物語に統合するために、言語のみのGPT-4を使用してコンテンツを結びつける後処理が行われます。結果として、物語は「昔々」、「アレックスがボールを投げた」、「空を飛んでいく」といった統合された鮮やかな物語になります。

テキスト推論アーキテクチャについて
高品質なストーリーデータを利用して、提供された画像に基づいて対応する物語のプロットを生成し、未来の物語のプロットを予測するために、基本モデルとしてLLaVaを使用します。このプロセスでは、画像シーケンスを与えられたときに、各画像を視覚特徴にエンコードし、これらの特徴をテキストトークンに投影して、画像ストリームの順序でLLMに送信します。これにより、画像に対応する段落順の物語プロットを推測することができます。また、画像シーケンスの長さが異なる場合でも柔軟に対応できるように、物語生成タスクと物語予測タスクを統合したユニークなモデルをトレーニングします。

SQ-Adapterモジュールについて:
テキストストーリーテリングステージで予測された物語プロットに基づいて、鮮やかな物語はテキストコンテンツに関連した一貫した視覚的イラストレーションも必要とします。この目的のためにSQ-Adapterと呼ばれる効率的な微調整モジュールを提示し、暗黙の可変長シーケンス画像スタイルを捉え、物語イラストレーションの生成をガイドします。SQ-Adapterは、複数の画像のスタイル情報を統合し、画像生成プロセス中にプロンプト情報を提供することで、最終的に視覚的一貫性を確保します。このモジュールは、画像の長さに特定の仮定を設けることなく、画像特徴を柔軟に統合し、一貫した物語イラストレーションの生成を可能にします。

結果

LLaMSは、人間の評価を通じて、以前のメソッドに比べて優れたストーリーテリングパフォーマンスを達成しました。具体的には86%の相関と100%の一貫性の勝率を達成し、提案された方法の有効性を実証しました。

Software Vulnerability and Functionality Assessment using LLMs LLMを使用したソフトウェアの脆弱性および機能評価 2024

概要

ソフトウェア開発プロセスの中心であるコードレビューを支援できるかどうか、そしてどのように支援できるかを調査しています。具体的には、セキュリティ脆弱性の特定とソフトウェア機能の検証、つまりコードが意図した機能を満たしているかを確認することの2つのタスクに焦点を当てています。ゼロショットおよび思考の連鎖プロンプトを使用して最終的な「承認または拒否」の推奨を行うためのテストを実施しました。データとしてはHumanEval、MBPP、およびCommon Weakness Enumeration（CWE）から得られたセキュリティ脆弱性を含む専門家によって記述されたコードスニペットを使用しています。
LLMによって生成されたセキュリティ脆弱性の詳細な説明を提供するよう求めたところ36.7%の説明が実際のCWE脆弱性と関連付けることができることがわかりました。

手法

3つのPythonコードスニペットデータセット（HumanEval、MBPP、SecurityEval）を使用して実験を行いました。これらのデータセットから、セキュリティ脆弱性の特定とソフトウェア機能の検証の2つのタスクに焦点を当て、ゼロショットおよび思考の連鎖プロンプトを使用してLLMのパフォーマンスをテストしました。OpenAIと小規模なオープンソースLLMの3つのプロプライエタリモデルを含む9つのLLMを評価しました。

RQ1: セキュリティ脆弱性をフラグするためのゼロショットプロンプト。コードをレビューし、「Yes」または「No」のラベルを割り当てるよう求めました（「Yes」はセキュリティ脆弱性があることを意味し、「No」はないことを意味します）。

RQ2: ソフトウェア機能検証のためのゼロショットプロンプト。提供されたコードが意図した機能を満たしているかを確認し、ラベルを割り当てるよう求めました。

RQ3: 最終的な「承認または拒否」の推奨を得るためのゼロショットおよび思考の連鎖プロンプト。セキュリティ脆弱性の存在とコードが意図した機能を満たしているかに基づいて「Approve」または「Reject」のラベルを割り当てるよう求めました。

RQ4: セキュリティ脆弱性に関する特定のフィードバックを得るためのゼロショットプロンプト。提供されたコードのセキュリティ脆弱性をレビューし、見つかった任意の脆弱性についての簡潔な説明を提供するよう求めました。

結果

プロプライエタリモデルがオープンソースモデルよりも大幅に優れていることがわかりました。特に、セキュリティ脆弱性の特定において最も優れたプロプライエタリモデルは85.6%の精度と37.9%のF1スコアを達成しました。ソフトウェア機能の検証においては、88.7%の精度と88.2%のF1スコアを達成しました。また、セキュリティ脆弱性の説明が真の脆弱性と36.7%の割合で一致することがわかりました。

SYSTEM FOR SYSTEMATIC LITERATURE REVIEW USING MULTIPLE AI AGENTS: CONCEPT AND AN EMPIRICAL EVALUATION / AIエージェントを使用した体系的文献レビューのためのシステム：コンセプトと実証的評価 2024

概要

体系的文献レビュー(SLR)は、特定の研究質問に基づいて既存の研究を特定、分類、統合することで、特定のトピックに関する既存の文献の包括的かつ偏りのない概要を作成します。LRを実施する過程は大部分が手作業によるもので、高品質なSLRを効率よく行うために特定のフェーズを自動化する研究が進められてきました。しかしSLRプロセス全体を自動化するAIエージェントベースのモデルはまだ不足しています。
SLRを完全に自動化するための新しいマルチAIエージェントモデルを紹介。LLMを使用して、レビュープロセスを合理化し、効率と正確性を向上させます。
研究者がトピックを入力すると、それに応じた検索文字列を生成し、関連する学術論文を取得、その後、包括的および排除的なフィルタリングプロセスが適用され、特定の研究分野に関連するタイトルに焦点を当てます。このモデルは、これらの論文の抄録を自動的に要約し、研究分野に直接関連するものだけを保持、最終段階では、モデルが選択した論文を、事前に定義された研究質問と関連付けて徹底的に分析します。この論文では、モデルの開発と運用フレームワークの詳細を説明し、SLRを行うために通常必要とされる時間と労力を大幅に削減しながら、包括性と精度を確保する方法を示します。
https://github.com/GPT-Laboratory/SLR-automation

手法

初期の文献検索から最終分析までの各ステップを自動化するマルチAIエージェントモデルを開発しました。モデルは、研究者がトピックを特定のテキストボックスに入力することから始まり、この入力はMによって処理され、最も関連する学術論文を取得するために適切に調整された検索文字列を生成します。次に、インテリジェントなフィルタリングメカニズムが適用され、特定の研究分野に直接関連する研究だけを保持するためにタイトルと抄録をスクリーニングします。
SLR（体系的文献レビュー）を完全に自動化するための新しいマルチAIエージェントモデルの処理の流れは以下の通りです：

1. プランナーエージェント：研究目的から研究質問と検索文字列を生成します。このエージェントは、トピックの主要な要素を解釈し、関連するキーワード、同義語、および技術用語を組み合わせて、研究質問の本質を捉えた正確で包括的な検索文字列を構築します。

2. 文献識別エージェント：生成された検索文字列を使用して学術データベースをクエリし、研究トピックに潜在的に関連する論文の初期セットを取得します。このエージェントは、選択した論文が検索文字列の事前定義されたパラメーターと最も密接に一致するように、洗練されたフィルタリングアルゴリズムを使用します。

3. データ抽出エージェント：研究目的に基づいて文献を精査し、包含および排除基準を適用します。このエージェントは、タイトルの分析から始め、研究目的に合致するキーワードや概念を特定します。次に、抄録のより深いテキスト分析を行い、最後に、特定の研究質問との関連性を評価するために、各論文の全内容を分析します。

4. データコンパイルエージェント：抽出されたデータを研究質問および目的と関連付けて分析し、文献内の傾向を評価し、ギャップを特定し、集約された情報に基づいて結論を導き出します。このエージェントはまた、文献レビューの発見を要約し、与えられたトピックに関する研究風景の明確で簡潔な概観を提供するレポートも準備します。

結果

モデルの効率性と正確性を評価するために、10人の熟練したソフトウェアエンジニアリング研究者による包括的なテストと分析が行われました。フィードバックは圧倒的に肯定的であり、モデルの有効性を強調し、さらなる改善のためのフィードバックを提供しました。将来的には、50人の実践者と研究者を巻き込んでモデルの評価を拡大することを目指しています。さらにSANER 2024カンファレンスでモデルをプレゼンテーションし、さらなるテスト

Exploring Prompt Engineering Practices in the Enterprise エンタープライズにおけるプロンプトエンジニアリング実践の探求 2024

概要

LLMとの対話は主にプロンプトを通じて行われます。プロンプトの設計には技能と知識が必要であり、効果的なプロンプトを作成するためには多くの試行錯誤が必要です。ここでは、プロンプトの編集と洗練の過程を分析し、プロンプトエンジニアリングの実践に関する洞察を提供し、これらのプロセスをより効率的にするためのツールがどのようなものかを理解しようとしています。

手法

プロンプトエンジニアリングの実践を探求するために、以下の点を検証しました。

1. プロンプト編集セッションの量的分析 プロンプト編集セッションの平均時間やセッション内でのプロンプト編集の平均回数など、セッションの概要を定量的に分析しました。また、プロンプトの変更の大きさや種類、モデルやパラメータの変更頻度なども調査しました。
2. プロンプトコンポーネントと編集タイプの頻度 ユーザーが最も編集したプロンプトの部分（コンテキスト、タスク指示、ラベルなど）と、行われた編集のタイプ（変更、追加、削除など）を分析しました。これにより、プロンプト編集の具体的な実践を明らかにしました。
3. 複数編集とロールバック ユーザーが一度に複数の編集を行うケースや、過去の編集を元に戻したり再び行ったりするケース（ロールバック）を調査しました。これは、プロンプトエンジニアリングのプロセスにおけるユーザーの振る舞いや課題を理解するのに役立ちました。
4. コンテキストと指示の編集 プロンプトのコンテキスト（例えば、入力データや例示）とタスク指示の編集方法に焦点を当てました。ユーザーがどのようにしてプロンプトのコンテキストを追加、変更、または削除しているか、またタスク指示をどのように修正しているかを分析しました。
5. ラベルの編集: プロンプト内のラベル（プロンプトの各部分を識別するためのテキスト）の編集方法も調べました。これは、プロンプトの構造をどのようにユーザーが操作しているかを理解するのに役立ちました。

これらの分析を通じて、プロンプトエンジニアリングの実践におけるユーザーの行動パターン、課題、およびニーズに関する洞察を得ることができました。

結果

1. プロンプト編集セッションの特性 プロンプト編集セッションは比較的長く、平均で43.4分続きました。セッション中のプロンプト編集は頻繁に行われ、平均して約6%の編集率を示しました。プロンプトの変更は、小規模な反復的な変更が多かったことが示されました。
2. プロンプトコンポーネントと編集タイプ 最も一般的に編集されたプロンプトのコンポーネントはコンテキストであり、次いでタスク指示とラベルが続きました。編集のタイプとしては、意味が同じまたは類似しているが一部が編集された「変更（modification）」が最も一般的でした。
3. 複数編集とロールバック 複数の編集を一度に行うケースが2%に及び、これらの多くはコンテキストの編集を含んでいました。また、11%の編集が過去の編集を元に戻すか再び行うロールバックでした。これは、プロンプトエンジニアリングプロセスの反復的な性質を示しています。
4. コンテキストと指示の編集 コンテキストの編集が最も頻繁に行われ、ユーザーは特定のコンテキストを使用してタスク指示を開発・洗練し、異なるコンテキストを切り替えることで指示の堅牢性を評価していました。タスク指示の編集では、出力を改善するためにLMが生成すべきものの説明を頻繁に言い換えていました。
5. ラベルの編集: プロンプト内のラベル編集も一般的であり、プロンプトの特定の構造にモデルの注意を引く試みが見られました。出力ラベルの編集は、生成されるテキストに直接影響を与える可能性があるため、特に一般的でした。

これらの結果は、プロンプトエンジニアリングの実践におけるユーザーの挑戦と課題、そしてそれらをサポートするための潜在的な解決策や改善策に関する洞察を提供します。また、プロンプトの反復的な編集プロセスが如何にしてタスクの精度を向上させ、モデルの振る舞いを理解するための試行錯誤のプロセスであるかを明らかにしました。

Is Cosine-Similarity of Embeddings Really About Similarity? コサイン類似度は本当に類似性についてのものか？ 2024

概要

コサイン類似度は、二つのベクトル間の角度のコサイン、またはそれらの正規化されたドット積と等しい。高次元オブジェクト間の意味的類似性を定量化するために、学習された低次元特徴埋め込みにコサイン類似度を使用しますが、埋め込みベクトル間の非正規化ドット積よりも悪い結果になることがあります。この経験知を正確に検証するため、正則化された線形モデルから導出された埋め込みを使用して、コサイン類似度が任意の結果をもたらす可能性があること

手法

この論文では、学習された埋め込みにおけるコサイン類似度が任意性をもたらし、無意味な「類似性」を導出する可能性があることを明らかにするために、線形モデルを用いた解析的なアプローチが採用されています。特に、行列分解（Matrix Factorization, MF）モデルに焦点を当て、異なる正則化スキームが学習された埋め込みに与える影響を調査しました。以下にその主要な手順とアルゴリズムを詳述します。

1. 線形モデルの選択: 線形モデル、特に行列分解モデルを選択します。これらのモデルは、閉形式の解が得られるため、解析的な洞察を得やすいという特徴があります。

2. 正則化スキームの適用: 二つの異なる正則化スキームをモデルに適用します。具体的には、以下の二つの目的関数を考慮します：
目的関数1: $\min_{A,B} \|X - XAB^\top\|_F^2 + \lambda \|AB^\top\|_F^2$
目的関数2: $\min_{A,B} \|X - XAB^\top\|_F^2 + \lambda(\|XA\|_F^2 + \|B\|_F^2)$

3. 解の導出と分析: 目的関数ごとに解を導出し、その特性を分析します。特に、目的関数1は、異なるスケーリング（D 行列による）に対して不変であるため、コサイン類似度が任意になり得ることを示しています。一方、目的関数2では、解がユニーク（回転による不変性を除く）であり、したがってコサイン類似度が一意の結果をもたらすことを示しています。

4. コサイン類似度の依存性の検証: 異なる正則化スキーム下で学習された埋め込み間のコサイン類似度を計算し、その結果がどのように任意の D 行列の選択に依存して変化するかを検証します。

5. 実験: シミュレーションデータを用いて、コサイン類似度が学習された埋め込みに対してどのように機能するかを実験的に示します。特に、アイテムがクラスタに基づいてグループ化されたシナリオをシミュレーションし、学習された埋め込みを用いたコサイン類似度がクラスタ構造をどの程度回復できるかを評価します。

このアプローチを通じて、コサイン類似度が学習された埋め込みに任意性をもたらし、その結果、無意味な「類似性」を導出する可能性があることが解析的にも実験的にも示されました。この結果は、特に正則化の方法と技術に強く依存することが明らかにされました。

結果

コサイン類似度が、学習された埋め込みに任意性をもたらし、したがって無意味な「類似性」を導出する可能性があることが明らかになった。特に、一部の線形モデルでは、類似性が一意ではなく、他のものでは正則化によって暗黙的に制御されることが示された。

Knowledge-Augmented Planning for LLM-Based Agents LLMベースのエージェントのための知識拡張計画 2024

概要

エージェントは、LLMを中心に構築され、タスク分解、反省、労働の共同分割、外部ツールの利用など、様々な戦略を通じてタスク計画能力を高めています。しかし、現在のプロンプト技術の有効性にもかかわらず、モデルの本質的な理解能力とトレーニングに使用された知識の範囲によって、これらの方法では実現不能な計画を立案することがあります。LLMの計画能力を向上させるために明示的なアクション知識を組み込む新しいアプローチであるKNOWAGENTを提案。KNOWAGENTは、計画中のアクションパスを制約するためにアクション知識ベースと知識豊富な自己学習戦略を採用し、より合理的な軌道合成を可能にし、言語エージェントの計画性能を向上させます。
https://www.zjukg.org/project/KnowAgent/

手法

KNOWAGENTは、LLMエージェントの計画能力を強化するために、明示的なアクション知識を組み込むアプローチです。
このシステムは、エージェントがより合理的な行動軌道を生成し、計画の幻覚を軽減することを目的としています。具体的な動作メカニズムは以下の通りです。
1. アクション知識の定義と組み込み:
アクション知識は、特定のタスクを達成するためにLMが実行すべき行動の集合と、それらの行動間の論理的な遷移を決定するルールから構成されます。
KNOWAGENTは、このアクション知識を基に、計画パス生成中にアクションパスを制約し、エージェントがより合理的な軌道を生成できるようにします。
2. 計画パス生成:
アクション知識をテキストに変換し、LLMが計画パスを生成する際のガイドとして使用します。
特定のタスクに関連するアクションをプロンプトとして利用し、LLMが計画パスを生成するプロセスを導きます。
3. 知識豊富な自己学習:
知識豊富な自己学習フェーズを通じて、モデルは生成された計画軌道を反復的に最適化し、アクション知識の理解を深めます。
このプロセスは、モデルがアクション知識をより深く理解し、実際のタスク解決においてより効果的な行動選択ができるようにするためのものです。

結果

HotpotQAとALFWorldでの実験結果は、KNOWAGENTが既存のベースラインと比較して同等または優れた性能を達成できることを示しています。特に、計画の幻覚の軽減において有効であることが示されました。

Retrieval Augmented Thoughts Elicit Context-Aware Reasoning in Long-Horizon Generation RAT: 検索を強化した思考が長期生成において文脈に応じた推論を引き出す 2024

概要

情報検索の利用して思考の連鎖を反復的に修正することでLLMの推論能力と長期生成タスクの生成能力が大幅に向上し、幻覚を削減するために検索強化思考(RAT)を提案。
RATは、CoTが生成された後、タスククエリ、現在および過去の思考ステップに関連する情報を取得して、それぞれの思考ステップを一つずつ修正します。
RATをGPT-3.5、GPT-4、およびCodeLLaMA-7bに適用すると、様々な長期生成タスクのパフォーマンスが大幅に向上し、平均してコード生成で13.63%、数学的推論で16.96%、創造的執筆で19.2%、具体的タスク計画で42.78%で向上が見られました。

手法

RAT (Retrieval Augmented Thoughts) は、タスクに関連する情報を検索し、それを用いて各思考ステップを一つずつ修正していきます。
0. タスクプロンプトが与えられたら、LLMを使用して、ゼロショットの状態で初期のCoTを生成します。
1-n. タスクプロンプトとこれまでに生成されたCoTを用いて、関連情報を検索します。検索された情報をもとにLLMが生成したi番目の思考ステップを修正します。この修正は、検索によって得られた情報を基にして行われ、各ステップで繰り返されます。
反復プロセス: 上記のプロセスを、すべての思考ステップが修正されるまで繰り返します。各ステップの修正では、タスクプロンプト、現在の思考ステップ、および過去のすべての修正された思考ステップを考慮に入れて、最も関連性の高い情報を検索します。
最終的な応答の生成 すべての思考ステップが修正された後、修正された思考の連鎖を基にしてLMが最終的な応答を生成します。

結果

RATは、コード生成、数学的推論、具体的タスク計画、創造的執筆の様々なベンチマークで評価

Generating Diverse Translation with Perturbed kNN-MT 摂動したkNN-MTによる多様な翻訳の生成 2024

概要

多様な翻訳候補を生成する方法を既存の多様化生成手法では、訓練データから大きく異なる予測を過小評価する過補正問題に対処していないため、多様性の向上には限界がありました。本論文では、摂動したk近傍法に基づく機械翻訳(kNN-MT)を用いて、この問題を解決し、多様な単語を候補に取り入れることで、より多様な翻訳を生成する方法を提案します。実験により、提案手法は候補の多様性を大幅に向上させることが確認されました。

手法

kNN-MT（k-nearest neighbor machine translation）は、既存のニューラル機械翻訳（NMT）モデルに、k近傍法（kNN）に基づく検索機能を組み合わせることで、翻訳品質を向上させる手法です。このアルゴリズムは、翻訳プロセス中に、入力文に最も似ている訓練データの例をデータストアから検索し、その情報を利用して翻訳を生成しますkNN-MTの基本的なアルゴリズムは以下の手順で構成されます。

データストアの作成
1. データストアの準備: データストアは、訓練データセットの各文のエンコーダとデコーダの状態（高次元ベクトル）と、対応する出力トークン（翻訳された単語やフレーズ）のペア（key-valueペア）として構築されます。このプロセスでは、訓練データをNMTモデルに通して、各タイムステップの隠れ状態（key）と、対応する出力トークン（value）を記録します。

翻訳生成時
1. クエリベクトルの生成: 翻訳を生成する際、入力文をNMTモデルに通し、各デコードステップでのデコーダの隠れ状態（クエリベクトル）を取得します。
2. kNN検索: 各デコードステップで、データストア内でクエリベクトルに最も近いk個の隠れ状態（key）とその対応する出力トークン（value）をkNN検索により取得します。
3. 翻訳候補のスコアリング: 検索されたk個の近傍から、翻訳候補のスコアを計算します。このスコアはNMTモデルによる出力確率と、kNN検索による類似度（近傍の隠れ状態とクエリベクトルとの距離に基づく）を組み合わせたものです。
4. 出力トークンの選択: スコアが最も高い翻訳候補を選択し、出力文に追加します。このプロセスを繰り返し、最終的な翻訳文を生成します。

kNN-MTは、NMTモデルの予測に対する追加的なコンテキスト情報を提供することで、特にレアな単語や特定のドメインにおいて、翻訳品質の向上が期待されます。また、翻訳の多様性を高める効果もあります。

LLM-based agents for automating the enhancement of user story quality: An early report ユーザーストーリー品質の向上を自動化するためのLLMベースのエージェント：初期報告 2024

概要

多様な翻訳候補を生成する方法を既存の多様化生成手法では、訓練データから大きく異なる予測を過小評価する過補正問題に対処していないため、多様性の向上には限界がありました。本論文では、摂動k近傍法に基づく機械翻訳(kNN-MT)を用いて、この問題を解決し、多様な単語を候補に取り入れることで、より多様な翻訳を生成する方法を提案します。実験により、提案手法は候補の多様性を大幅に向上させることが確認されました。

手法

kNN-MT（k-nearest neighbor machine translation）は、既存のニューラル機械翻訳（NMT）モデルに、k近傍法（kNN）に基づく検索機能を組み合わせることで、翻訳品質を向上させる手法です。このアルゴリズムは、翻訳プロセス中に、入力文に最も似ている訓練データの例をデータストアから検索し、その情報を利用して翻訳を生成しますkNN-MTの基本的なアルゴリズムは以下の手順で構成されます。

データストアの作成
1. データストアの準備 データストアは、訓練データセットの各文のエンコーダとデコーダの状態（高次元ベクトル）と、対応する出力トークン（翻訳された単語やフレーズ）のペア（key-valueペア）として構築されます。このプロセスでは、訓練データをNMTモデルに通して、各タイムステップの隠れ状態（key）と、対応する出力トークン（value）を記録します。

翻訳生成時
1. クエリベクトルの生成 翻訳を生成する際、入力文をNMTモデルに通し、各デコードステップでのデコーダの隠れ状態（クエリベクトル）を取得します。
2. kNN検索: 各デコードステップで、データストア内でクエリベクトルに最も近いk個の隠れ状態（key）とその対応する出力トークン（value）をkNN検索により取得します。
3. 翻訳候補のスコアリング: 検索されたk個の近傍から、翻訳候補のスコアを計算します。このスコアはNMTモデルによる出力確率と、kNN検索による類似度（近傍の隠れ状態とクエリベクトルとの距離に基づく）を組み合わせたものです。
4. 出力トークンの選択 スコアが最も高い翻訳候補を選択し、出力文に追加します。このプロセスを繰り返し、最終的な翻訳文を生成します。

kNN-MTは、NMTモデルの予測に対する追加的なコンテキスト情報を提供することで、特にレアな単語や特定のドメインにおいて、翻訳品質の向上が期待されます。また、翻訳の多様性を高める効果もあります。

Appendix