論文要約

LLM関連

概要

LLMエージェントSELFGOALは、目標を達成するために、 サブゴールのツリー構造（GOALTREE）を使用し、サブゴールを適応的に分解 環境と対話するために、エージェントは GOALTREEのサブゴールを選択し、分解します。
https://selfgoal-agent.github.io/

手法

SELFGOALは、エージェントが環境と対話しながら目標をサブゴールに分解し、それを基にアクションを生成するためのガイドラインを提供します。 以下に、SELFGOALの3つの主要なモジュールについて説明します

1. **Searchモジュール**: 現在の状態とGOALTREEの既存ノードを基に最も適したサブゴールを選択します。
2. **Decompositionモジュール**: サブゴールをさらに具体的なサブゴールに分解し、GOALTREEを自己成長させます。
3. **Actモジュール**: 選択されたサブゴールをガイドラインとしてLLMにプロンプトを提供し、現在の状態に対するアクションを生成します。

1. Searchモジュール

目的: 現在の状態に基づいて、GOALTREE内の最も適したサブゴールを選択する。

手順:

1.
   - **初期設定:** タスクの高レベルの目標 g0 をGOALTREEのルートノードとして設定します。初期の状態 s0 とアクション a0 を生成し、初期のポリシー πθ に基づいて最初のGOALTREEを生成します。

     T = g_0 ∪ DECOMPOSE(g_0, {a_0, s_0})
2. **状態の更新:** 各タイムステップ t で、エージェントの現在の状態 st を記述し、GOALTREE内の全てのリーフノードを候補サブゴールリストとしてLLMに提供します。
   - **サブゴールの選択:** LLMにプロンプトを提供して、現在の状態に最も適した上位K個のサブゴールを選択させます。選択されたサブゴールは次のアクションのための新しいプロンプトとして使用されます。

     gi,j = SEARCH(T, st)
   - pt+1 = {pt, gi,j}
3.

2. Decompositionモジュール

目的: サブゴールをさらに具体的なサブゴールに分解し、GOALTREEを自己成長させる。

手順:

1.
   - **サブゴールの分解:** 現在の状態とアクションペア {at,st} に基づいて、選択されたサブゴール gi,j をさらに具体的なサブゴールに分解します。

     G = DECOMPOSE(gi,j, {at, st})
   - **重複ノードのフィルタリング**: 新しいサブゴールが既存のサブゴールとテキスト的に類似しているかどうかをコサイン類似度で評価し、重複するノードをフィルタリングします。

BMW Agents - A framework for task automation through multi-agent collaboration BMWエージェント- マルチエージェント協調によるタスク自動化のフレームワーク2024

概要

BMWエージェントは、マルチエージェントで協力して複雑なタスクを自動化するためのフレームワークで計画、実行、検証の　3段階の仕組みを使い業務プロセスを自動化を行います

技術や手法

- 計画（Planning）: ユーザーの指示をシンプルなタスクに分解し、明確な順序で実行。
- 実行（Execution）: 分解されたタスクをエージェントが実行。
- 検証（Verification）: 実行されたタスクが元の指示を満たしているか独立して確認。

エージェントのワークフロー構成要素

1. エージェント (Agent): 特定のタスクを達成するためにLLM呼び出しを行うオブジェクト。
2. エージェントユニット (Agent Unit): 一緒にタスクを解決するエージェントの集合体。
3. マッチャー (Matcher): タスクに適したエージェントユニットを選択する抽象層。
4. エグゼキューター (Executor): エージェントユニットとのすべての操作を調整。
5. ツール (Tools): データベースやAPIなど、タスク完了に必要な外部機能へのアクセス。
6. ツールボックスリファイナー (Toolbox Refiner): タスク実行中にエージェントに提供するツールのセットを絞り込み。
7. コーディネーター (Coordinator): ワークフロー全体の調整とデータフローの計画、実行、検証を実行。

Auto Cherry-Picker: Learning from High-quality Generative Data Driven by Language オートチェリーピッカー：言語に基づく高品質生成データからの学習024

概要

ACPはロンプトを使い、詳細な説明とレイアウトを生成する LLM を使い、テキストから画像へのモデルで複数の画像を生成。生成されたデータは CLISを使い、品質を評価。高品質なデータを生成するため、データフィルタリングを行う。
https://yichengchen24.github.io/projects/autocherrypicker/

技術や手法

1. **自然言語プロンプトによる生成**:
    - 大規模言語モデル(LLM) を使用して、自然言語の概念リストから詳細な説明と合理的なレイアウトを生成。
    - テキストから画像へのモデルを使用して、生成された説明とレイアウトに基づいて複数の画像を生成。
2. **Composite Layout and Image Score (CLIS)**:
    - **CLIS-L**: レイアウトの合理性を評価。
    - **CLIS-I**: 画像の視覚品質とテキスト説明との整合性を評価。
3. **データフィルタリング**:
    - 生成されたデータをCLISを使用して精査し、高品質な訓練データを選別。
4. **応用と実験**:
    - ロングテール分布および不均衡なデータセットでの実験。
    - 合成データを使用して既存モデルのパフォーマンスを向上。

使用用途

- **視覚認識タスク**:
    - セグメンテーション、検出、視覚表現学習の訓練データ生成。
- **マルチモーダル学習**:
    - マルチモーダル視覚質問応答(VQA) の訓練データ生成。
- **不均衡データセットの改善**:
    - ロングテール分布および不均衡データセットに対するデータ生成とパフォーマンス向上。

Identifying Performance-Sensitive Configurations in Software Systems through Code Analysis with LLM Agents LLMエージェントを用いたコード解析によるソフトウェアシステムのパフォーマンス感度の高い設定の特定24

概要

PerfSenseは開発者とパフォーマンスエンジニアの対話をシミュレートし、 prompt chainingやRAGなどのプロンプト技術を使用し3プロセスを経てシステムの速度や効率に影響を及ぼすかを評価し分類します

使用されるプロンプト技術

- **プロンプトチェイン（ Prompt Chaining）**:
    1. PerfAgentがDevAgentに逐次的にタスクを依頼し、複雑なタスクを簡単なサブタスクに分けて実行します。これにより、 PerfAgentは設定に関連するすべての必要なコードを収集し、詳細な分析を行います。
- **RAG（Retrieval-Augmented Generation）**:
    1. PerfAgentは、DevAgentから得られた情報を基に、外部の知識やコンテキストを取り入れて、設定のパフォーマンス感度をより正確に分類します。この技術により、 LLMのコンテキストサイズ制限を克服し、必要な情報をすべて取得できます。

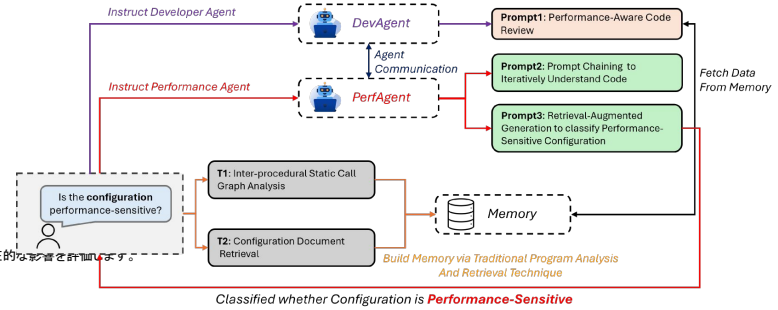
PerfSenseとDevAgentのアルゴリズム説明

PerfSenseの概要

PerfSenseは、パフォーマンスに敏感な設定を特定するための軽量フレームワークです。 LLMエージェントを用いて、開発者（ DevAgent）とパフォーマンスエンジニア（ PerfAgent）の役割をシミュレートし、プロンプトチェインや RAGなどの技術を使って、パフォーマンスに影響を与える設定を効率的に特定します。

アルゴリズムと動作の詳細

1. **設定の取得と初期化**
    - PerfAgentは、分析する設定を受け取ります。これがパフォーマンスに敏感である可能性のある設定です。
    - PerfAgentは、設定に関連するソースコードとドキュメントの取得を DevAgentに依頼します。
2. **設定関連のコードとドキュメントの取得（ DevAgentの役割）**
    - **ソースコードの取得** :
        - DevAgentは、設定に直接アクセスするメソッドを特定します。
        - インタープロシージャルコールグラフを利用して、設定に関連するすべてのメソッドを取得します。
    - **ドキュメントの取得** :
        - DevAgentは、設定に関する公式ドキュメントを取得します。
    - **コードレビューの実施** :
        - DevAgentは、取得したソースコードとドキュメントを基に、コードの機能、設定関連の操作の頻度、および設定がシステムに与える潜在的な影響を評価します。
    - **結果の返却** :
        - DevAgentは、コードレビューの結果を PerfAgentに返します。
3. **パフォーマンス感度の分析と分類（ PerfAgentの役割）**
    - **情報の統合** :
        - PerfAgentは、DevAgentから提供されたソースコード、ドキュメント、およびコードレビューの結果を統合します。
    - **プロンプトチェインの使用** :
        - PerfAgentは、プロンプトチェインを使用して、複雑なタスクを簡単なサブタスクに分け、逐次的に情報を取得し、分析を深めます。
    - **RAGの使用** :
        - PerfAgentは、外部の知識を取り入れて、 LLMのコンテキストサイズ制限を克服し、詳細な分析を行います。
    - **分類の実施** :
        - PerfAgentは、設定がパフォーマンスに敏感かどうかを分類します。この分類は、設定がシステムの速度や効率にどの程度影響を与えるかに基づきます。
4. **結果の出力**
    - PerfSenseは、パフォーマンスに敏感な設定のリストを生成し、パフォーマンスエンジニアに提供します。
    - このリストは、パフォーマンスエンジニアがさらなる調査や最適化のために使用します。



- **動作フローの例**
    1. **設定の受け取り** :

LLMatDesign: Autonomous Materials Discovery with Large Language Models LLMatDesign: 大規模言語モデルによる自律的な材料発見2024

概要

LLMatDesignは、太陽電池材料など、特定のバンドギャップを持つ材料設計を行うフレームワークです。 人間の指示を翻訳し、材料に対する修正を適用し、提供されたツールを使用して結果を評価します。 自己反省を使い次のステップに反省させる k遠出効率的な材料設計ができるようになります

LLMatDesignの技術や手法

LLMatDesignは、材料の化学組成と特性を入力として受け取り、目標とする特性を持つ新しい材料を設計するためのフレームワークです。以下は、そのアルゴリズムの処理順番に沿った詳細な説明です。

1. 初期入力

LLMatDesignは、ユーザーが提供する以下の入力を受け取ります:

- 初期材料の化学組成と特性（例:化学組成 x0、特性値 y0）
- 目標特性値（例:目標特性値 ytarget）

  ytargety_{target}
- 修正履歴（任意、例: M）

2. LLMによる修正提案と仮説生成

大規模言語モデル（ LLM）は、初期材料の組成と特性、および目標特性値に基づいて、次の修正 s_i とその修正の理由となる仮説 hih_ihi を生成します。

- pythonコードをコピーする
- si, hi ← LLM(xi−1, yi−1, ytarget, M)

3. 材料の修正適用

LLMatDesignは、LLMが提案した修正 s_i を材料の組成 x_{i-1} に適用します。修正の種類は以下の 4つです:

- **交換（exchange）**: 材料内の 2つの元素を交換
- **置換（substitute）**: 材料内の 1つの元素を他の元素に置換
- **除去（remove）**: 材料から特定の元素を除去
- **追加（add）**: 材料に新しい元素を追加
  - pythonコードをコピーする
  - x̃i ← perform modification(xi−1, si)

Large Language Models as Evaluators for Scientific Synthesis 科学的総合評価者としての大規模言語モデル2024

概要

LLMで科学文章の要約の品質をどれくらい評価できるかを調査。GPT-4とMistralを使用し、100の研究質問とそれに関連する要約を評価。結果は、LLMが論理的な説明を提供できる一方で、人間の評価と弱い相関が見つかる
先行研究では、ROUGEやBLEUなどの自動評価メトリックが主流でしたが、これらは語彙の一致に依存しており、意味的な情報を考慮していません。

技術や手法

1. **データセット**：CORE-GPTデータセットを使用。これは 100の研究質問とそれに対する GPT-4によって生成された要約を含む。
2. **評価モデル**：GPT-4 TurboとMistral-7Bを使用。これらのモデルに対し、要約の包括性、信頼性、有用性を 0から10で評価するように指示。
3. **評価プロンプト**：タスクの指示、評価基準、回答の形式を含むプロンプトを設計。評価は JSON形式で返却。
4. **評価方法**：人間の評価者との相関を Spearmanのρを用いて分析。

評価プロンプト基準

評価に際しては以下の特性を考慮してください。各特性について 0（非常に悪い）から 10（非常に良い）までの範囲で評価を行い、各評価に対する簡単な理由も提供してください。

1. **包括性 (Comprehensive)**: 質問にどれだけ包括的に答えられているか？
2. **信頼性 (Trust)**: 回答はどれだけ信頼できるか？
3. **有用性 (Utility)**: 回答はどれだけ有用か？

LANE: Logic Alignment of Non-tuning Large Language Models and Online Recommendation Systems for Explainable Reason Generation LANE: 説明可能な理由生成のための非チューニング大規模言語モデルとオンライン推薦システムの論理整合 2024

概要

LANEは非チューニングのLLMをオンライン推薦システムに使用する方法を提案。従来のように項目 IDではなく項目タイトルの埋め込みを使用、CoTにより説明可能な推薦を生成するようにしています

技術や手法

1. セマンティック埋め込みモジュール

目的: ユーザーのインタラクションシーケンスをセマンティックに理解し、推薦の質を向上させる。

手順:

1. **テキストエンコーダの選択** :
    - Sentence-BERTを使用して、項目タイトルを高次元ベクトルに変換する。
2. **項目タイトルの埋め込み** :
    - 全ての項目タイトルをテキストエンコーダに入力し、埋め込み行列　M を生成する。

      MM

    - MMM は、項目数 ∣I∣ と埋め込み次元 d を持つ行列である。

      ∣I∣∣I∣

      dd

A Comparative Study of DSL Code Generation: Fine-Tuning vs. Optimized Retrieval Augmentation DSLコード生成の比較研究 ファインチューニングvs. 最適化されたリトリーバル拡張2024

概要

DSL（ドメイン特化言語）のコード生成を LLMで行うためにRAGの最適化を行ったものとCodexモデルをDSL用にファインチューニングを比較し最適化された RAGモデルは、ファインチューニングされたモデルと同等になった

技術や手法

1. **自然言語プロンプトによる生成**:
    - 大規模言語モデル(LLM) を使用して、自然言語の概念リストから詳細な説明と合理的なレイアウトを生成。
    - テキストから画像へのモデルを使用して、生成された説明とレイアウトに基づいて複数の画像を生成。
2. **Composite Layout and Image Score (CLIS)**:
    - **CLIS-L**: レイアウトの合理性を評価。
    - **CLIS-I**: 画像の視覚品質とテキスト説明との整合性を評価。
3. **データフィルタリング**:
    - 生成されたデータをCLISを使用して精査し、高品質な訓練データを選別。
4. **応用と実験**:
    - ロングテール分布および不均衡なデータセットでの実験。
    - 合成データを使用して既存モデルのパフォーマンスを向上。

使用用途

- **視覚認識タスク**:
    - セグメンテーション、検出、視覚表現学習の訓練データ生成。
- **マルチモーダル学習**:
    - マルチモーダル視覚質問応答(VQA) の訓練データ生成。
- **不均衡データセットの改善**:
    - ロングテール分布および不均衡データセットに対するデータ生成とパフォーマンス向上。

Large Language Models are Capable of Offering Cognitive Reappraisal, if Guided 大規模言語モデルは、ガイドされれば認知的再評価を提供できる2024

概要

LLMのゼロショット能力を臨床心理士による評価フレームワークの RESORTを設計して評価。7B規模のLLMでも、RESORTによりユーザーが状況を再評価するのを助ける共感的な応答の生成を確認

技術や手法

RESORTは、心理学的に基づいた再評価の構成要素からなり、以下のような次元にわたる再評価を行います:

1. **自己責任** :状況を引き起こした責任があるかどうかを再評価します。
2. **問題焦点型対処** :状況に対処できるかどうかを再評価します。
3. **注意活動** :状況にさらに注意を払う必要があるかどうかを再評価します。
4. **感情焦点型対処** :感情的に対処できるかどうかを再評価します。
5. **自己制御可能** :状況を制御できるかどうかを再評価します。
6. **内部価値との一貫性** :状況が個人の価値観と一致しているかどうかを再評価します。

結果

- **効果の確認** : RESORTを使用したLLMは、心理学的に効果的な認知的再評価を提供できることが確認された。
- **ガイドの重要性** : ガイドなしでは効果が低下することが示され、心理学的原則に基づくガイドの重要性が強調された。

Evidence-Focused Fact Summarization for Knowledge-Augmented Zero-Shot Question Answering 知識を活用したゼロショット質問応答のための証拠に基づいた事実要約024

概要

LLMのゼロショットQA性能を向上するために質問に関連する証拠を強調し、不要な情報を除外することで、知識を補強した　LLMに役立つ要約を生成するEFSUMを提案
簡単に試すのならstep1かなと思う

EFSUMは、ゼロショット質問応答性能を向上させるための証拠に基づいた事実要約する仕組みです。

ステップ 1: LLMプロンプトを用いた証拠に基づく要約の生成

Step 1: Evidence-Focused Summarization Using LLM Prompting

このステップでは、LLMを利用して、質問に関連する証拠を強調した要約を生成します。

1. **入力データの準備**：質問（q）と関連する事実のセット（F = {f_k}）を入力として準備します。
2. **プロンプト設計**：要約を生成するためのプロンプト（tsum）を設計し、LLMに対して以下のように指示します。
    - 質問に対して関連する事実を要約する。
    - 要約は質問に答えるために必要な証拠を強調し、不必要な情報を除外する。
3. **要約の生成**：設計したプロンプトを用いて、LLMに要約を生成させます。具体的には、以下の形式でLLMに指示します。
    - `s ~ pLLM(·|tsum, q, F)`
    - ここで、sは生成された要約です。

ステップ 2: LLMの微調整による証拠に基づく要約の質向上

Step 2: Enhancing the Quality of Evidence-Focused Summarization through LLM Fine-Tuning

このステップでは、オープンソースのLLMを微調整し、証拠に基づく要約の質を向上させます。具体的には、知識蒸留（LLM distillation）と優先度調整（preference alignment）の2つのサブステップに分かれます。

知識蒸留

1. **参照要約の生成**：質問（q）と関連する事実のセット（F）に対して、LLM（例：GPT-3.5-turbo）を用いて参照要約（s）を生成します。この要約は、証拠を強調し質問に答えるために必要な情報を含みます。
2. **トレーニングデータセットの構築**：質問（q）、回答（a）、関連する事実（F）、および生成された要約（s）の組み合わせをトレーニングデータセット（D）として構築します。
3. **LLMのスーパーバイズド微調整**：構築したトレーニングデータセットを用いて、要約モデル（θ）を以下の目的関数に基づいて最適化します。
    - `LSFT = − E(q, a, F, s)~D log pθ(s|q, F)`
    - これにより、モデルが質問と関連する事実を条件として要約を生成する能力を学習します。

優先度調整

EFSUMで使用されるプロンプトとその日本語対訳は以下になります

1. 証拠に基づく要約生成プロンプト （EFSUMprompt)

英文プロンプト

[Task Description]

You are a knowledge graph summarizer for Question Answering. I will give you “Question” and “Fact triples”. You should turn triples into *summary*. The *summary* should serve as a context to facilitate QA (Question and Answer) tasks.

Caution1: The *summary* should not explicitly mention what the correct answer is.

Caution2: The *summary* should only contain information from the given triples.

Caution3: Each triplet is separated with “\\n” and head, relation, tail are provided in head | relation | tail format.

Question: {}

Fact triples: {}

Summary:

日本語対訳

[タスク説明]

あなたは質問応答のためのナレッジグラフ要約者です。ここで「質問」と「事実トリプル」を提供します。トリプルを*要約*に変換してください。*要約*は質問応答(QA)タスクを促進するためのコンテキストとして機能する必要があります。

注意1: *要約*には正しい答えを明示的に記載しないでください。

注意2: *要約*には与えられたトリプルからの情報のみを含めてください。

注意3: 各トリプルは「\\n」で区切られ、head、relation、tailは head | relation | tail 形式で提供されます。

質問: {}

事実トリプル: {}

ICLGuard: Controlling In-Context Learning Behavior for Applicability Authorization ICLGuard：適用認可のためのインコンテキスト学習行動の制御024

概要

ユーザーが不適切な内容や機密情報を含むデータを使用しモデル使用しないように適用認可という概念を導入しICLを制御するアプローチICLGuard提案、ターゲットデータ（ICL能力を無効にしたいデータ）と非ターゲットデータ（ICL能力を維持したいデータ）の2種類のデータセットを使用しfine-tuningすることでLLMをガードします

技術や手法

- **インコンテキスト学習（ ICL）**：入力-ラベルペアのデモンストレーションとテスト入力を条件としてタスクを実行する機能。
- **適用認可**：モデル所有者が特定のデータに対してモデルの ICL行動を制御する概念。
- **ICLGuard**：元のLLMを保持し、最小限の追加トレーニングパラメーターをファインチューニングすることで、特定のデータに対する ICL能力を無効にし、他のデータに対してその能力を維持するファインチューニングフレームワーク。
- **パラメーター効率の良いファインチューニング（ PEFT）**：事前学習されたモデルを凍結し、少数の追加パラメーターのみを更新する手法。
- **損失関数**：
  - **無効化損失（ disable loss）**：ターゲットデータに対して ICL能力を無効にする損失。
  - **維持損失（ maintenance loss）**：非ターゲットデータに対して ICL能力を維持する損失。
  - **有用性損失（ utility loss）**：通常のプロンプトに対してモデルの出力が元の LLMと一貫していることを保証する損失。

使用用途

ICLGuardは、以下のようなシナリオで使用できます。

- モデル所有者が特定のデータに対してモデルの使用を制限したい場合。
- モデルが不適切な内容や機密情報を含むデータで誤用されるのを防ぎたい場合。
- モデルの特定の機能を制御しつつ、他の機能を維持したい場合。

Entropy Law: The Story Behind Data Compression and LLM Performance エントロピー法則 データ圧縮とLM性能の背後にある物語2024

概要

LLMの学習に使用するモデルの選択にデータ圧縮比と最初のepochでの学習損失が性能に関連するかを示すエントロピー法則に基づき、圧縮比の低いデータサブセットを優先する効率的で普遍的なデータ選択方法AIPを提案
従来のデータ選択方法は、個々のサンプルの品質に注目していましたが、本研究ではデータセット全体の情報冗長性を考慮した「エントロピー法則」に基づくアプローチを提案しています。この新しいアプローチにより、データ圧縮比を利用してデータ選択を行うことで、LLMの性能をより効率的に向上させることができます。

技術や手法

1. **エントロピー法則**：データ圧縮比と訓練損失が LLMの性能にどのように影響するかを示す理論。圧縮比の低いデータは情報密度が高く、訓練損失が低いことが多い。
2. **ZIPアルゴリズム**：複数段階の貪欲なアルゴリズムを用いて、圧縮比の低いデータサブセットを効率的に選択する方法。具体的には、グローバル選択、ローカル粗粒度選択、ローカル細粒度選択の 3段階を経て、多様で冗長性の低いデータサブセットを構築します。

Artificial Intuition: Efficient Classification of Scientific Abstracts 人工直感 科学的要約の効率的分類2024

概要

科学文章の要約を粗く分類するためにLLMを使用して補足的な知識を生成し、適切なラベルを割り当てる。これにより、ラベルの空間を生成し、NASAの受賞要約のコーパスを用いて、この方法を評価し、新しい評価ツールと確立された性能指標を組み合わせて使用して正確な予測を行う

技術や手法

- **キーワード抽出**：Yet Another Keyword Extractor (YAKE)を使用して文書からキーワードを抽出。
- **補足知識の生成**：Mistral 7B LLMを使用してキーワードのメタデータを生成。
- **文書のクラスタリング**：Sentence Transformerを使用して文書の埋め込みを生成し、k-meansクラスタリングでクラスタに分割。
- **ラベルの予測**：新しいラベル空間に基づいて、キーワードとメタデータを用いてラベルを予測。

使用用途

- **研究ポートフォリオ管理**：科学的文書の粗分類により、研究の全体像を把握しやすくする。
- **初期スクリーニング**：科学的要約の初期スクリーニングを支援する。

数式の説明

- **余剰度 (Redundancy, R)**: ラベルの正規化埋め込み間のコサイン類似度を計算し、ラベルの独立性を評価。 R=maxi≠j(cosine similarity(Ti,Tj))R = \max_{i \neq j} (\text{cosine similarity}(T_i, T_j))R=maxi=j(cosine similarity(Ti,Tj)) cosine similarity(Ti,Tj)=Ti⋅Tj// Ti // // Tj // \text{cosine similarity}(T_i, T_j) = \frac{T_i \cdot T_j}{\|T_i\|\|T_j\|}cosine similarity(Ti,Tj)= // Ti // // Tj // Ti⋅Tj
- **カバレッジ (Coverage, S)**: 文書のキーワードとラベル間の最大コサイン類似度を用いて、ラベル空間が文書をどれだけ包括しているかを評価。Sd=max(wij)S_d = \max(w_{ij})Sd=max(wij) SD=∑DSdDS_D = \frac{\sum_{D} S_d}{D}SD=D∑DSd

Multi-Meta-RAG: Improving RAG for Multi-Hop Queries using Database Filtering with LLM-Extracted Metadata マルチメタRAG: データベースフィルタリングとLLM抽出メタデータを用いたマルチホップクエリ向けRAGの改善 2024

概要

Multi-Meta-RAGはLLMで抽出されたメタデータを用いたデータベースフィルタリングを活用して、様々なソースから質問に関連する文書をより適切に選択し一つの質問に答えるために複数の情報源や証拠を連続的に取得し、推論を行う必要があるクエリに対応できるようにします
https://github.com/mxpoliakov/Multi-Meta-RAG

手法と技術

Multi-Meta-RAG は以下の技術を用いています:

1. **LLM抽出メタデータ** :LLMを用いて質問からメタデータを抽出し、データベースフィルタリングに使用します。
2. **データベースフィルタリング** :抽出されたメタデータを用いて、質問に関連する文書を選択します。
3. **チャンク選択の改善** :データベースフィルタリングにより、より正確な文書のチャンクを選択します。

使用用途

Multi-Meta-RAGは、特にマルチホップクエリにおいて、LLMが適切な証拠を取得し、正確な応答を生成するのに適しています。これはニュース記事の分析や、複数の情報源からの証拠を必要とする質問応答システムに有用です。

プロンプト

Given the question, extract the metadata to filter the database about article sources. Avoid stopwords.

**The sources can only be from the list: ['Yardbarker', 'The Guardian', 'Revyuh Media', 'The Independent - Sports', 'Wired', 'Sport Grill', 'Hacker News', 'Iot Business News', 'Insidesport', 'Sporting News', 'Seeking Alpha', 'The

FACTS About Building Retrieval Augmented Generation-based Chatbots RAG（Retrieval Augmented Generation）ベースのチャットボット構築に関するACTS 2024

概要

NVIDIAの企業のAIチャットボットを構築するための5つのFACTとして新鮮なコンテンツ（F）、アーキテクチャ（A）、LLMのコスト（C）、テスト（T）、セキュリティ（S）を紹介。また、実践的なガイドラインとしてRAGパイプラインの15の制御ポイントとしてメタデータの強化、チャンク化、クエリの再フレーズ、クエリの再ランク付けなどを紹介しています

技術や手法

1. **RAGパイプラインの設計と最適化** ：セマンティック埋め込みの微調整、クエリの再フレーズ、結果の再ランク付け、効果的なプロンプトの設計、ドキュメントアクセス制御の尊重、簡潔な応答の提供、関連する参考文献の追加、個人情報の保護。

2. **FACTSフレームワーク** ：

    - **F（Freshness）**: エンタープライズデータの新鮮さを確保する方法。
    - **A（Architectures）**: 柔軟なアーキテクチャを構築する方法。
    - **C（Cost Economics）**: コスト効率の良いLLMの使用。
    - **T（Testing）**: テストの計画と実行。
    - **S（Security）**: セキュリティの確保。

3. **RAGパイプラインの 15の制御ポイント** ：

1. メタデータ強化（Metadata Enrichment）

概要: ドキュメントのメタデータを追加・強化することで、検索と取得の精度を向上させます。例えば、著者、公開日、トピックなどのメタデータを追加します。 **手法**: 自動的にメタデータを抽出し、既存のメタデータを補完・更新するアルゴリズムを使用します。

2. チャンク化（Chunking）

概要: 大規模なドキュメントを小さなセグメント（チャンク）に分割することで、検索の精度を向上させます。 **手法**: 文単位、段落単位、またはセク

AI Agents That Matter 実世界で役立つAIエージェント2024

概要

LLMによるAgentの評価方法の提案として正確さだけでなく、モデルが確率的に精度が向上することも考え再試行を行うなど計算コストを考慮したものする。モデル開発者、アプリ開発者で評価基準を分ける。ベンチマークの過剰適合を防ぐ、評価方法の再現性がない点を課題として挙げています

技術や手法

1. コスト制御の重要性

AIエージェントの評価において、単に精度を追求するのではなく、コストも考慮する必要があると論じています。言語モデルは確率的であり、同じモデルを複数回呼び出すだけで精度が向上することがあります。しかし、これには無制限のコストがかかる可能性があります。このため、エージェントの評価はコストを制御する必要があります。以下のような具体的な手法を導入しています:

- **再試行戦略（ Retry Strategy）**:モデルの温度を0に設定し、テストケースに失敗した場合に最大 5回まで再試行します。
- **ウォーミング戦略（ Warming Strategy）**:再試行ごとにモデルの温度を徐々に上げていき、最終的に 0.5まで上げることで、少なくとも 1回の再試行が成功する可能性を高めます。
- **エスカレーション戦略（ Escalation Strategy）**:最初は安価なモデル（例: Llama-3 8B）を使用し、テストケースに失敗した場合にはより高価なモデル（例: GPT-4）に切り替えます。

2. 精度とコストの同時最適化

精度とコストのトレードオフを視覚化するために、パレートフロンティアを用いて評価結果を表示します。これにより、精度と推論コストを同時に最適化する新しいエージェント設計の空間が開かれます。具体的な方法として、 DSPyフレームワークを改良し、コストを削減しつつ精度を維持する方法を示しています。

- **パレートフロンティア** :精度とコストのトレードオフを示す曲線を用いて、最適なエージェント設計を視覚化します。このフロンティア上にあるエージェントは、精度とコストの両方で他のエージェントに優れています。
- **HotPotQAベンチマークでの評価** :DSPyフレームワークを用いて HotPotQAタスクに対するエージェントの設計を最適化します。具体的には、

Promptbreeder: Self-Referential Self-Improvement via Prompt EvolutionPromptbreeder：プロンプト進化による自己参照的自己改善2024

概要

PromptbreederはLLMのプロンプトを特定のドメインに対応する手法 1. 初期化：タスクプロンプトと変異プロンプトの集団を生成。2. 評価：データセットを使用して各タスクプロンプトの適応度を評価。3. 変異：変異プロンプトを使用してタスクプロンプトを変異。4. 世代交代：高適応ならタスクプロンプトを次世代に伝播。

技術や手法

Promptbreederは次のステップを通じてプロンプトを進化させます：

1. 初期化

Promptbreederの初期化段階では、タスクプロンプトと変異プロンプトの集団を生成します。具体的には、以下のように行います。

- **タスクプロンプト生成** ：
    - 初期タスクプロンプトは、問題の説明、変異プロンプト、および思考スタイル（一般的な認知ヒューリスティックのテキスト記述）を連結したものから生成されます。
    - 例: 問題説明が「Solve the math word problem, giving your answer as an arabic numeral」の場合、以下のようなプロンプトを使用して初期タスクプロンプトを生成します。

Make a variant of the prompt. Let’s think step by step. INSTRUCTION: Solve the math word problem, giving your answer as an arabic numeral. INSTRUCTION MUTANT:

-

2. 評価

各タスクプロンプトの適応度は、トレーニングセットからランダムにサンプリングされた　100組のQ&Aペアを使用して評価されます。この評価に基づいて、プロンプトの性能を測定します。

Reason for Future, Act for Now: A Principled Framework for Autonomous LLM Agents with Provable Sample Efficiency 未来のための理由、今のための行動：サンプル効率が証明された自律型Mエージェントのための原則的なフレームワーク 2024

概要

LLMエージェントの生成結果を調整するために計画した初期行動後にフィードバックをメモリに保存。再計画をして行動を微調整
https://agentification.github.io/RAFA/

手法

1. **RAFAフレームワーク** ：
   - メモリバッファを使用して環境を推定し、長期的な計画を行うためのプロンプトテンプレートを設計。
   - 計画された軌道の初期行動を取り、新しいフィードバックを使って再計画を行う。
2. **学習と計画のサブルーチン** ：
   - 学習サブルーチン：メモリバッファから環境の推定を行い、 LLMを使って推定されたモデルと価値関数を生成。
   - 計画サブルーチン：推定されたモデルと価値関数を使用して、将来の最適な政策（行動）を生成。
3. **ベイズ適応 MDPの使用** ：
   - 全履歴を情報状態として扱い、フィードバックに基づいて最適な行動を選択。
   - ベイズルールを使用して後悔を最小化する。

RAFAフレームワークの手法の詳細説明

1. フレームワークの目的と概要

RAFAの目的は、LLMを用いて環境からのフィードバックを効果的に活用し、最適な行動を計画・実行することです。特に、最小限のインタラクションでタスクを完了することを重視しています。 RAFAはベイズ適応マルコフ決定プロセス（ MDP）に基づいており、推論と行動を以下のように構成します。

2. 初期化 (Initialization)

1. **LLM学習プランナーの入力 (Input)**:

概要

LLMのプロンプトでは中間出力を何度もさせ最終の出力させることがあり、この時無駄で冗長な中間応答を発生するため、使用するテンプレートにどのように出力させるかを明示し、出力させたい変数を順次生成させることでより一貫性のある出力を制御するプロンプトスケッチングを提案
https://github.com/eth-sri/lmql

手法

1. **プロンプトスケッチングフレームワーク** :
   - テンプレートに基づいて、複数の変数の値を予測することで、 LLMの推論をガイドします。
   - テンプレートに含まれる各変数に対して、一連の指示を挿入しながら、モデルが自動回帰的にトークンを生成します。
2. **スケッチ対応デコーディング手法** :
   - 変数レベルビームサーチ(VAR) :各変数の値を生成する際にビームサーチを適用し、複数の仮説を並行して探索します。
   - 変数グリッドビームサーチ(BEAMVAR) :トークンレベルのビームサーチを変数の進行状況に基づいて適応させ、低尤度の仮説を排除します。
3. **汎用スケッチのコレクション** :
   - 複数のタスクに適用可能な汎用的かつ効果的なスケッチのセットを提供し、オープンソースライブラリ「 dclib」を公開しています。

使用用途

- **状態追跡** :中間状態を保持しながら複雑なタスクを実行するために使用されます。
- **算術推論** :数値計算を伴うタスクにおいて、正確な計算結果を得るために使用されます。
- **一般的な質問応答** :質問に対して一貫性のある、冗長性のない回答を生成します。

プロンプトスケッチングの具体例

テンプレートの例

Algorithm of Thoughts: Enhancing Exploration of Ideas in Large Language Models 思考アルゴリズム：大規模言語モデルにおけるアイデアの探索の強 2024

概要

LLMの能力を向上させるために思考アルゴリズム（ AoT）を提案。推論能力を向上させるために、AoTでは単一または少数のクエリで問題を複数に分割し生成、結果から使用する内容を取捨選択する DFS的なアプローチで高い結果を生成させます
https://algorithm-of-thoughts.github.io/

思考アルゴリズム（ Algorithm of Thoughts, AoT）

1. 背景と課題認識

従来の大規模言語モデル（ LLMs）の推論手法は、問題解決のための生成プロセスを中断、変更、再開する外部の方法に依存していました。これらの手法は多くのクエリを必要とし、コストや計算負荷が増大する問題がありました。これに対して、 AoTは少数のクエリで効率的に推論を行う新しい戦略を提案します。

2. AoTの概要

AoTは、アルゴリズム的な例を文脈内に完全に組み込み、 LLMsの再帰的なダイナミクスを活用することで、アイデアの探索を広げます。これにより、トークン使用を最小限に抑えながらも、高いパフォーマンスを実現します。 AoTは、単一または少数のクエリで複雑な問題を解決し、標準的なプロンプトやChain-of-Thought（CoT）手法を上回る効果を示します。

3. アルゴリズムの実装

3.1. 問題の分割と初期探索

AoTの実装の最初のステップは、問題を適切なサブプロブレムに分割することです。これにより、問題解決の各段階での探索層を設定します。具体的には、初期探索ステップとして以下の手順を取ります：

- 問題を複数の小さな部分に分割

概要

LLMのZero-Shotでの生成能力を向上させるためにAgentInstructアプローチを使用したエージェントを提案。タスク情報からウェブ上のタスク関連の知識を参照して指示の生成しCoTで解かせます

AgentInstructの方法

1. エージェントの設計

AgentInstructは、タスクの推論プロセスを指示する自律エージェントを構築することにより、ゼロショット推論能力を向上させます。このエージェントは次のように設計されています。

1. **Instruction Generation（指示生成）**：
    - エージェントは、ReActフレームワーク（Yao et al., 2023b）に基づいて設計されています。ReActは、LLM（大規模言語モデル）を使用して一連の考えを提案し、観察結果に基づいてアクションを実行します。
    - 具体的には、エージェントはタスク固有の指示を生成し、これにより LLMの推論プロセスをタスクに適した形に最適化します。エージェントは、基本的なタスク情報（データセットの名前やいくつかの入力例）を入力として受け取り、ウェブ上のタスク関連の知識を参照して高品質な指示を生成します。
2. **Action Space（アクションスペース）**：
    - エージェントは以下の二つのアクションをサポートする質問応答 APIを使用します：
        1. **ask_about_dataset[string]**: データセットに関する情報を含むウェブページを取得します。
        2. **finish[instructions]**: タスク固有の指示を生成し、指示生成プロセスを終了します。

2. チェーンオブソート推論（Chain of Thought Reasoning）

AgentInstructは、タスクを中間的な推論ステップに分解し、これらのステップを解決することで最終的な答えに導く「チェーンオブソート（ CoT）推論」アプローチを採用します。

Appendix