論文要約

LLM関連

Automated Evaluation of Retrieval-Augmented Language Models with Task-Specific Exam Generation タスク特化型試験生成による検索強化型言語モデルの自動評価

概要: RAGの評価方法の提案

タスクに関連する文書コーパスに基づく多肢選択式問題から自動生成された合成試験にRAGをスコアリングすることで実施。試験の品質とタスク特化型精度に関する情報量を推定するために項目反応理論 (IRT) を活用することで試験を段階的に改善します

技術や手法

本研究で説明されている技術や手法は次の通りです:

1. **試験生成**:
    - LLMを用いて、文書コーパスに基づく多肢選択式試験問題を自動生成。
    - 各質問には1つの正解と複数の選択肢が含まれます。
    - 各文書から質問候補を生成し、NLPベースのフィルターを適用して低品質な質問を除外。
    - 試験生成プロンプト例

```markdown
Human: Here is some documentation from {task_domain}: {documentation}. From this, generate
a difficult multi-form question for an exam. It should have 4 candidates, 1 correct answer
and explanations.

Syntax should be:
Question: {question}
A) {candidate A}
B) {candidate B}
```

Concise Thoughts: Impact of Output Length on LLM Reasoning and Cost 簡潔な考え：LLMの推論とコストにおける出力長の影響

概要：LLMでCoTを使用すると説明能力を向上させる半面、出力が長くなり応答に時間がかかるため、出力長を制御するプロンプトエンジニアリング制約付き CoT（CCoT）を紹介。出力の長さを制約するプロンプトを与え、出力の簡潔さと応答時間の予測可能性を向上させます。

つくり方は、Let's think a bit step by step の後に limit the answer length to 45 words. のような制限の指定をするだけ

技術や手法

1. **新しい評価指標の提案 **

   - **硬直な簡潔精度（ Hard-k Concise Accuracy: HCA）**: 指定された長さ k以下の正確な出力の割合を測定します。

   - **柔軟な簡潔精度（ Soft-k Concise Accuracy: SCA）**: 長さkを超える正確な出力に対して減衰因子 αを用いてペナルティを課します。

   - **一貫した簡潔精度（ Consistent Concise Accuracy: CCA）**: 出力長のばらつき σに基づいて SCAをさらに調整します。

2. **制約付き CoT（CCoT）の導入 **

   - **CCoTプロンプト **: LLMに対して出力の長さを制約するプロンプトを与え、出力の簡潔さと応答時間の予測可能性を向上させます。

Retrieval Augmented Generation or Long-Context LLMs? A Comprehensive Study and Hybrid Approach 検索拡張生成か長文コンテキスト LLMか？包括的研究とハイブリッドアプローチ

概要: RAGと長文コンテキスト（ LC）の比較を行いました。結果として、リソースが十分にあれば LCが平均的な性能で RAGを上回ること、RAGは大幅に低コストであるという利点があること、この結果を基にモデルの自己反省に基づいてクエリを RAGまたはLCにルーティングする SELF-ROUTEという方法を提案。

計算コストを大幅に削減しながら、 LCと同等の性能を維持できます。

技術や手法

以下の3つの手法の比較を行っています

1. **RAG (Retrieval Augmented Generation)**:
    - クエリに基づいて関連情報を検索し、 LLMがその情報を使用して応答を生成する。
    - クエリに関連する情報を取得し、 LLMの注意を必要なセグメントに集中させることで、無関係な情報による注意の分散を防ぐ。
    - 計算コストが低い。
2. **長文コンテキスト（ LC）LLMs**:
    - 大規模な事前学習により、長文コンテキストを直接理解する能力を持つ。
    - 例: Gemini-1.5（最大1百万トークンを処理可能）、 GPT-4（128kトークンを処理可能）。
3. **SELF-ROUTE**:
    - クエリを RAGまたはLCにルーティングする方法。
    - モデルの自己反省に基づいてクエリが回答可能かどうかを予測し、回答可能な場合は RAGを使用し、そうでない場合は LCを使用する。

Tulip Agent – Enabling LLM-Based Agents to Solve Tasks Using Large Tool Libraries チューリップエージェント – 大規模ツールライブラリを使用してタスクを解決する LLMベースエージェントの実現

概要: LLMのエージェント tulip agentを紹介

ツールの説明をシステムプロンプトにエンコードせず、また全体のプロンプトを埋め込むことなく、再帰的にツールライブラリから適切なツールを検索することで実現します

技術や手法の詳細説明

タスク分解

概要:

ユーザーからの自然言語クエリを受け取り、 LLM（大規模言語モデル）が高レベルのタスクを理解し、細分化されたサブタスクに分解します。これにより、具体的かつ実行可能な単位でタスクを処理できます。

詳細:

1. **ユーザークエリの受信 **:
    - エージェントはユーザーからの自然言語による質問や要求を受け取ります。
    - 例: "What is 45342 multiplied by 23487 plus 32478?"
2. **タスク分解モデル (Mtd)**:
    - このモデルは、受信したクエリを解析し、それをサブタスクに分解します。

Appendix