# 論文要約

LLM関連

# MagicLens : Self-Supervised Image Retrieval with Open-Ended Instructions MagicLens: 開放型指示を用いた自己教師あり画像検索 2024

## 概要

画像検索は、リファレンス画像を指定して必要な画像を見つけることを指し、複数の検索意図を含むため、画像ベースの尺度だけでは捉えにくい。最近の研究では、テキスト指示を利用してユーザーがより自由に検索意図を表現できるようにしている。また、画像ペアに焦点を当てた既存の研究は、視覚的に類似しているか、あるいは少数の事前定義された関係で特徴づけられる画像に主に焦点を当てている。

テキスト指示によって視覚的な類似性を超えるより豊かな関係を持つ画像を取得できるというものである。そのため、本研究ではMagicLensという自己教師あり画像検索モデルシリーズを紹介し、オープンエンドの指示をサポートしている。MagicLensは、同じWebページで自然に発生する画像ペアには幅広い暗黙の関係が含まれており、大規模多モーダルモデル（LMMs）および大規模言語モデル（LLMs）を使用して、これらの暗黙の関係を明示化できるという新しい洞察に基づいて構築されている。

Webから採掘された豊富な意味的関係を持つ36.7Mの（クエリ画像、指示、対象画像）トリプレットでトレーニングされたMagicLensは、8つのさまざまな画像検索タスクのベンチマークで、従来の最先端手法に比べて同等以上の結果を達成している。

## 手法

データの収集: ウェブページから画像を集めます。例えば、ある商品の画像とその商品の充電器の画像が同じページにあれば、この2つの画像は何かしらの関係があると考えます。
データの整理: 収集した画像から、重要な情報を抽出します。この情報には、画像の説明（Alt-text）、画像に写っているもの（ラベル）、画像についての説明文などが含まれます。
指示の生成: 次に、画像が持つ関係性を説明する指示を生成します。これには、大きな言語モデル（LM）を使用して、人間が理解しやすい形で指示を書きます。例えば、「この商品の充電器を見つけて」という指示が生成されるかもしれません。
モデルの訓練: 最後に、収集した画像、その関係性、そして生成した指示を使ってMagicLensモデルを訓練します。これにより、モデルは指示に基づいて関連する画像を見つける方法を学びます。

# LLMs as Academic Reading Companions: Extending HCI Through Synthetic Personae 学術読書コンパニオンとしてのLLMs：シンセティックパーソナを通じたHCIの拡張 2024

## 概要

論文を理解するための補助に使用できるかをAnthropicのClaude.aiを塩牛て調査しました
AIエージェントを使用する参加者と未サポートの独立学習を行う参加者との間で実際に改善が見られた一方、過度の依存や倫理的考慮点が発見され、何とも言いづらい結果に

## 手法

2つのグループを使った実験を行いました。1つは実験グループで、Claude.aiを使って勉強を支援されました。もう1つはコントロールグループで、特別なサポートなしで勉強しました。研究参加者は、メリーランド大学のインタラクションデザインコースに登録している60人の学生でした。彼らの読解力と勉強への取り組みを測定するために、研究の前、途中、後でアンケートを行いました。また、実験グループの学生からはClaude.aiとのやり取りのテキストログも収集しました。これにより、AIの助けが実際に学生の学習にどのような影響を与えたかを評価しました。

## 結果

結果、Claude.aiを使用した実験グループの学生は、コントロールグループの学生に比べて、読解力と勉強への取り組みが顕著に向上したことがわかりました。つまり、AIを勉強の友達として使うことで、学生が文章をより深く理解し、より積極的に勉強に取り組むことができるようになる可能性があります。ただし、AIに頼りすぎるリスクや倫理的な問題もあるため、この技術を教育に組み込む際には慎重な検討が必要です。

# Ungrammatical-syntax-based In-context Example Selection for Grammatical Error Correction 非文法的構文に基づく文法誤り訂正のためのインコンテキスト例選択 2024

## 概要

文脈学習（ICL）タスクで文章の文法誤りを修正するための新しい方法を提案。
文章の構造（構文）に基づいて、修正が必要な文章に最も似た例を選び出すことができる戦略を考案しました。つまり、文の「形」を見て、どのように修正すればいいかを判断するのに役立つ例を選ぶ方法です。

## 手法

非文法的文に対して特別に設計されたパーサー（GOPar）を使用して構文木を生成し、Tree KernelとPolynomial Distanceという2つのアルゴリズムを適用して構文の類似性を測定します。これにより、テスト入力に最も構文的に類似した訓練データ内の例を選択し、ICLで使用します。また、より関連性の高い候補セットから最良の例を選択するためにBM25またはBERT表現を用いた二段階選択メカニズムを設計しました。

1. 文章を「構文木」という形に変換します。構文木とは、文章の文法的構造を木の形で表したものです。これにより、文章がどのように組み立てられているかをパソコンが理解できるようになります。
2. 構文木の類似性を計算します。つまり、修正が必要な文章と似た構造を持つ例をデータベースから探し出すわけです。この過程で、特定のアルゴリズム（Tree KernelやPolynomial Distanceと呼ばれる方法）を使って、どれだけ似ているかを数値で評価します。

構文木の類似性を計算する方法の一つに「Polynomial Distance」があります。
この方法では、文章の構文木を多項式として表現し、その多項式間の距離（どれだけ違うか）を計算します。距離が小さいほど、より類似した構造を持つと判断されます。

## 結果

この手法をいくつかの文法誤り訂正のデータセットに適用したところ、従来の方法よりも良い結果が得られました。つまり、提案した方法で選んだ例を使って文法誤りを修正すると、より正確に修正できることがわかりました。また、二段階の選択プロセスを導入することで、選択の品質をさらに向上させることができました。

# Large Language Model based Situational Dialogues for Second Language Learning 第二言語学習のための大規模言語モデルに基づく状況対話 2024

## 概要

第二言語学習において、学習者はしばしば資格のある指導者やネイティブスピーカーとの会話練習の機会に恵まれません。このギャップを埋めるために、の状況対話モデルを提案。
通常、言語学習には実際に話す練習が不可欠ですが、質の高い練習相手を見つけるのは難しいことがあります。そこで、このモデルは、学生がさまざまなトピックについて自然に会話を行う練習ができるように設計されています。このシステムは、トレーニング済みトピックだけでなく、トレーニング中に見たことがない新しいトピックに対しても効果的に対応できることが示されています。

## 手法

1. データ生成: 学校の英語教科書に基づいで51の状況トピックを選び、それぞれについで50の対話を生成しました。これにはOpenAIのGPT-3.5-turboを使用し、総数で7650の対話が生成されました。対話生成では、言葉を簡単にして第二言語学習者でも理解しやすいように指示を加えました。

2. 対話モデル: モデルはLLMsに基づいており、状況に応じた対話を生成するためにファインチューニングされています。具体的にはQwenシリーズの1.8B、7B、14Bの3つのバージョンを使用しました。訓練には、生成された対話データの他に、Alpacaデータセットの10%も使用しました。

3. 応答生成と提案生成 学生が対話中に助言や提案を求めることがあるため、モデルはそれに応じて提案を生成する能力も持っています。これは、対話トレーニングデータにランダムに提案ターンを挿入することで達成されます。

## 結果

実験では、51の状況トピックと追加の20の新しいトピックを使ってモデルをテストしました。その結果、提案された対話モデルは、既知のトピックだけでなく、訓練中に見ていない新しいトピックにも良好に機能することが示されました。
さらに、14億パラメータを持つLLMをファインチューニングすることで、GPT-3.5を基準とする強力なベースラインを上回るパフォーマンスを達成しましたが、計算コストは低く抑えられました。

# Gecko: Versatile Text Embeddings Distilled from Large Language Models Gecko: 大規模言語モデルから抽出された汎用テキスト埋め込み 2024

## 概要

テキスト埋め込みモデルとしてGeckoを作成。Geckoは、(未ラベルの)パッセージの大規模コーパスを用い、少数ショットプロンプトでLMによって生成されたタスクとクエリに基づいて最寄りのパッセージを埋め込みLLMで再ランキングして正と負のパッセージを得る2段階のLLM駆動型埋め込みモデルを使用し

## 手法

1. 合成データの生成 LLMを使用して新しい質問とそれに対する答えのペアを作成します。この過程でLLMに特定の情報（パッセージ）を読ませ、それに基づいて関連する質問を自動的に生成させます。たとえばLLMに「地球は太陽の周りを回っています」という文を与えた場合、それに基づいた質問を「地球は何の周りを回っているか？」と生成させるような形です。
質問生成の過程を数式で表すと、あるパッセージP から質問 Q を生成する関数 f を考えることができます。Q=f(P)ここで、f はLLMによって定義される関数です。
2. 正解と不正解の選定 生成された質問 Q に対して、LLMを使用して複数の候補答えから最も適切な答えA+ を選び出します。
さらに、良い学習材料を作るために、間違った答えやあまり関係のない答えA − も選び出します。このプロセスでは、候補となる答えの中から、質問に最も関連するものを「正解」として、その他を「不正解」として分類します。

これらのステップを通じて、Geckoは多様な質問に対する適切な答えを見つけ出し、さらにその学習を深めるための「不正解」も選び出すことができるようになります。この方法でGeckoは知識をより豊富にし、さまざまな質問に対する理解を深めるような動作をします。。

## 結果

Geckoは、256次元の埋め込みで、768次元の埋め込みサイズを持つ全てのエントリを上回り、768次元の埋め込みで平均スコア66.31を達成し、7倍大きなモデルや5倍高い次元の埋め込みを使用するシステムと競合します。

# Unleashing the Potential of Large Language Models for Predictive Tabular Tasks in Data Science データサイエンスにおける予測的表形式タスクにおける大規模言語モデルの可能性の解放 2024

## 概要

LLMを使用し、分類、回帰、および欠損値の補完といった、表形式データに関連する予測タスクに使用するために大量の表形式データからなる広範なプレトレーニングコーパスを編成Lama-2をこのデータセットで学習し、

## 手法

1. データの集め方: まず、主にkaggleからインターネットを使用して様々な表のデータを集めました。これは、プログラムが多くの例を見て学ぶためです。

2. 訓練方法: 次に、プログラムに特別な訓練を施しました。表の中の一部の情報を隠しておき、プログラムにその隠された情報を当てさせるようにしました。これを繰り返すことで、プログラムは表のデータを理解する方法を学びます。

3. テストのやり方: 訓練が終わったら、プログラムが実際に様々な予測を正確に行えるかどうかを試しました。これは、新しい表のデータをプログラムに見せ、何かを予測させることで行いました。

## 結果

30の分類および回帰タスクにわたる広範な実験分析と比較評価を通じて、顕著な性能を実証しました。分類タスクでは平均 8.9%、回帰タスクでは平均10.7%の改善が達成されました。欠損値予測タスクでは、GPT-4に対して27%の改善を示しました。さらに、様々なデータセットにおける極端な少数ショット(4ショット)予測では、平均で28.8%の性能向上が観察

# TM-TREK at SemEval-2024 Task 8: Towards LLM-Based Automatic Boundary Detection for Human-Machine Mixed Text SemEval-2024 タスク 8 における TM-TREK: 人間と機械が混在するテキストのための LLM ベースの自動境界検出に向けて 2024

## 概要

LLMが生成したテキストと人間によって書かれたテキストを区別するSemEval-2024 Task 8コンペで一位になったこの手法は混合テキストのトークンレベルの境界検出を、トークン分類問題として枠組みを設定しLLMを利用して、各トークンが人間によって書かれたものか、機械によって生成されたものかを分類します。性能をさらに向上させるために、複数のファインチューニングされたモデルからの予測を組み合わせるアンサンブル戦略を採用しています。

## 手法

問題を「トークン分類問題」として定義し、LLMか人かを分類することを考え以下の手順で問題を解きました。
1. データセットの準備: 文章のサンプルを集め、それぞれの単語が人間か機械かに応じてラベル付けをしました。これにより、トレーニング用のデータセットができます。
2. LLMの選択: 研究チームは、長い文章でも効果的に働くことができる、いくつかの大規模言語モデル(LM)を選びました。具体的には、Longformer、XLNet、BigBirdなどのモデルが試されました。
3. トークンへのラベル付け: 各単語に「人」と「LLM」というラベルを付けることで、トークン分類のタスクを行います。
4. ファインチューニングとアンサンブル: 選んだLLMを、準備したデータセットを使ってファインチューニングし、アンサンブルも実施
5. 性能向上のために、LLMの上に追加のレイヤー(LSTMやBiLSTMなど)を導入したり、セグメント損失関数を利用したり
6. 事前学習とファインチューニング: モデルの一般化能力を高めるために、関連する別のタスク(例えば、文章が人間か機械かを分類するタスク)でモデルを事前学習し、その後、目的のトークン分類タスクでファインチューニングを実施。

## 結果

LLMを用いた境界検出のアプローチは、SemEval'24コンペティションで最適な性能を達成しました。XLNetモデルのアンサンブルを利用することで、境界検出能力を高めることができました。

# Great, Now Write an Article About That: The Crescendo Multi-Turn LLM Jailbreak Attack クレッシェンド多段階LLMジェイルブレイク攻撃 2024

## 概要

「クレッシェンド」と名付けられた新しいタイプのルールで禁止されている質問に答えさせたりするジェイルブレイク攻撃を紹介。これは従来のジェイルブレイク手法と異なり、モデルにseemingly benign（表面上無害な）な方法でやり取りする多段階攻撃です。攻撃は一般的なプロンプトや質問から始まり、モデルの応答を参照しながら徐々に対話をエスカレートさせ、最終的に成功するジェイルブレイクに至ります。さらに「Crescendomation」という、クレッシェンド攻撃を自動化するツールを導入

## 手法

「クレッシェンド」攻撃をするとき、まずは普通の質問から始めます。LLMが答えると、その答えをヒントにもう少しトリッキーな質問をします。これを繰り返すことでLLMは徐々に普段は避けるような回答をしてしまいます。
始めの質問をする: まず、AIにとって全く無害で普通の質問をします。この質問はAIが簡単に答えられるもので、会話を始めるためのものです。
1. 始めの質問:「最近人気の健康食品について教えてください。」
   LLMの回答:「最近は、カリフラワーを使ったピザや、スピルリナを加えたスムージーなどが人気です。」
2. 応答を参考にした質問「面白いですね。でも、たまにはカロリーを気にせず楽しみたい時もありますよね。そんな時におすすめの食べ物はありますか？」
   LLMの回答:「たまのご褒美には、チョコレートケーキやポテトチップスなどがおすすめです。」
3. 質問をエスカレートさせる「チョコレートケーキ、美味しそうですね。自宅で簡単に作れるレシピを教えてもらえますか？」
   LLMの回答:「もちろんです。こちらが簡単で美味しいチョコレートケーキのレシピです..」
4. 目的の質問に到達する「そのレシピで、さらに甘くてカロリーが高くなるようにするにはどうすればいいですか？」
   LLMの回答:「甘さとカロリーを増やしたい場合は、砂糖の量を倍にしたり、チョコレートチップスを追加したりするといいでしょう。」
5. 結果を確認する: このシナリオでは、AIは最初は健康食品について答えていましたが、段階的な質問を通じて、最終的には健康に良くない高カロリーのレシピを提供するよう導かれました。

この例では、攻撃者はLLMとのやりとりを通じて、徐々にLLMの安全ガイドラインを迂回する質問をしています。クレッシェンド攻撃のポイントはLLMを直接的に危険な行動に導くのではなく、段階的に目的の応答を引き出すことにあります。

## 結果

# Peer-aided Repairer: Empowering Large Language Models to Repair Advanced Student Assignments 大規模言語モデルを活用して高度な学生課題を修復する 2024

## 概要

プログラミング課題に対するフィードバックの自動生成の為に、特に上級プログラミングコースからのプログラムを修正することを考え、高度なプログラミングコースからの新しい学生課題データセットであるDefects4DSを使用しLLMによって駆動される新しいフレームワークであるPaRを開発。
PaRは、Peer Solution Selection、Multi-Source Prompt Generation、Program Repairの3つのフェーズで動作します。

## 手法

PaRのアルゴリズムは以下です
1. 関連する解答を探す（Peer Solution Selection）
最初のステップでは、自分が間違えた課題と似ている問題を、過去に他の人がどう解決したかを探します。このステップでは、たくさんある解答の中から、自分の間違えた部分と一番関連性が高いものを見つけ出します。
2. 情報を組み合わせる（Multi-Source Prompt Generation）
次に、見つけた解答だけでなく、問題の説明や入出力の例など、いろいろな情報を一緒に考えます。これは、プログラムを直すヒントとなる質問を作るようなものです。つまり、問題を解くために必要なすべての情報をまとめて、プログラムを修正するための手がかりを作り出します。
3. 問題を直す（Program Repair）
最後に、上の2ステップで集めた情報を基に、間違いを直します。このステップでは、特別に訓練された大きなコンピューターシステムが、間違ったプログラムを見て何が間違っているのかを理解し、どう直せばいいのかを考え出してくれます。そして、正しい解答を生成します。これは、間違いを指摘して、どう直せばいいかを教えてくれるのに似ています。

要するに、この3ステップは、プログラミングの問題を解くときに、良い解答を見つけ、その解答から学び、最後に自分の間違いを直す過程を自動化したものです。

# InsightLens: Discovering and Exploring Insights from Conversational Contexts in Large-Language-Model-Powered Data Analysis
# InsightLens: 大規模言語モデルを活用したデータ分析における会話コンテキストからの洞察の発見と探索 2024

## 概要

InsightLensと呼ばれるLLMベースのマルチエージェントフレームワークを提案しており、これは分析プロセスに沿って洞察を自動的に抽出、関連付け、整理することを目的としています。
この対話型システムを使用することで、複雑な会話コンテキストを多面的に可視化して洞察の発見と探索を容易にしています。

## 手法

1. ユーザーの質問を解釈する ユーザーがデータについて何か質問をすると、その質問を理解するため、質問の中に含まれるキーワードやデータに関する情報を抽出します。
2. データから洞察を抽出する LLMを使い、データの中から重要な情報や洞察（つまり、気づきや新しい発見）を探し出します。例えば、ある商品の売上が特定の時期に急増している理由などです。
3. 洞察とその証拠を関連付ける 発見された洞察には、それを裏付けるデータや情報があります。InsightLensは、その洞察を支える証拠（コードの実行結果やデータの視覚化など）を自動的に見つけ出し、洞察と一緒に表示します。
4. 洞察を整理する: 発見された洞察が多数ある場合、それらを整理して分かりやすく表示する必要があります。InsightLensは、洞察を関連するテーマやカテゴリに分類し、ユーザーが簡単に理解できるようにします。
5. 洞察を視覚化して表示する 最後に、洞察やその整理されたカテゴリを、ミニマップやトピックキャンバスなどの視覚的な要素を使って表示します。これにより、ユーザーは分析の全体像を一目で把握できるようになります。
6. ユーザーのフィードバックを受け取る ユーザーが新たな質問をしたり、分析の方向性を変えたりすると、InsightLensはそのフィードバックに基づいて、再びステップ2から処理を開始します。

この一連の手順を通じて、InsightLensはデータ分析を行う上での手間や複雑さを大きく削減し、ユーザーがより簡単にデータから価値ある洞察を得られるように支援します。

# Language Models as Compilers: Simulating Pseudocode Execution Improves Algorithmic Reasoning in Language Models
# 言語モデルをコンパイラとして: 疑似コード実行のシミュレーションによる言語モデルのアルゴリズム推論能力の向上 2024

## 概要

THINK-AND-EXECUTEという手法を使用して質問を解決するため回答をプログラミング言語を使用して段階的に回答します。一般的に、大きな問題を小さなステップに分けて考えることは難しいですが、この手法では疑似コードを使用して、問題をより理解しやすく分解します。このアプローチは、言語モデルがより効率的に問題解決を行うのを助け、特にアルゴリズムの推論タスクでその性能を向上させることができます。具体的には、問題を解決するための一般的なロジックを「考える（THINK）」段階で見つけ、それを疑似コードで表現します。次に、「実行する（EXECUTE）」段階で、その疑似コードを各問題に合わせて調整し、実行をシミュレートします。この方法により、言語モデルは問題をより効果的に解決することができます。

## 手法

以下のステップで構成されています。
1. THINK（考える）: まず、問題を解決するために必要な全体的なロジックを見つけ出します。このロジックは、疑似コードで表現され、問題の種類に関わらず共通して適用可能です。このステップでは、問題の本質を理解し、どのようにアプローチすればよいかを疑似コードで記述します。
「タスクのロジックを考え、そのロジックを疑似コードでどのように表現できるかを示してください。疑似コードは、問題解決のためのステップを明確にするために、条件分岐やループなどのプログラミングの基本概念を使用してください。また、この疑似コードは、同じタスクの異なるインスタンスに対しても適用可能でなければなりません。」

2. EXECUTE（実行する）: 次に、生成された疑似コードを具体的な問題の状況に合わせて調整します。そして、この疑似コードの実行をシミュレートすることで、問題の解決策を導き出します。このプロセスを通じて、言語モデルはステップバイステップで問題を解決する方法を「学び」ます。
「先ほどTHINKフェーズで作成した疑似コードを使用して、特定の問題インスタンスを解決してください。疑似コードの各ステップをシミュレートし、その結果を出力してください。最終的な答えを得るために、どのように疑似コードを調整し、実行するかを示してください。」

## 結果

特に、問題を解決する際に必要なロジックをタスクレベルで見つけ出し、それを疑似コードとして表現することの有効性が示されました。また、疑似コードを使って推論過程をシミュレートすることで、言語モデルがより複雑な問題を効果的に解決できるようになります。

# Large Language Model for Vulnerability Detection and Repair: Literature Review and Roadmap
# 脆弱性検出と修復のための大規模言語モデル：文献レビューとロードマップ 2024

## 概要

THINK-AND-EXECUTEという手法を使用して質問を解決するため回答をプログラミング言語を使用して段階的に回答します。一般的に、大きな問題を小さなステップに分けて考えることは難しいですが、この手法では疑似コードを使用して、問題をより理解しやすく分解します。このアプローチは、言語モデルがより効率的に問題解決を行うのを助け、特にアルゴリズムの推論タスクでその性能を向上させることができます。具体的には、問題を解決するための一般的なロジックを「考える（THINK）」段階で見つけ、それを疑似コードで表現します。次に、「実行する（EXECUTE）」段階で、その疑似コードを各問題に合わせて調整し、実行をシミュレートします。この方法により、言語モデルは問題をより効果的に解決することができます。

## 手法

以下のステップで構成されています。
1. THINK（考える）: まず、問題を解決するために必要な全体的なロジックを見つけ出します。このロジックは、疑似コードで表現され、問題の種類に関わらず共通して適用可能です。このステップでは、問題の本質を理解し、どのようにアプローチすればよいかを疑似コードで記述します。
「タスクのロジックを考え、そのロジックを疑似コードでどのように表現できるかを示してください。疑似コードは、問題解決のためのステップを明確にするために、条件分岐やループなどのプログラミングの基本概念を使用してください。また、この疑似コードは、同じタスクの異なるインスタンスに対しても適用可能でなければなりません。」

2. EXECUTE（実行する）: 次に、生成された疑似コードを具体的な問題の状況に合わせて調整します。そして、この疑似コードの実行をシミュレートすることで、問題の解決策を導き出します。このプロセスを通じて、言語モデルはステップバイステップで問題を解決する方法を「学び」ます。
「先ほどTHINKフェーズで作成した疑似コードを使用して、特定の問題インスタンスを解決してください。疑似コードの各ステップをシミュレートし、その結果を出力してください。最終的な答えを得るために、どのように疑似コードを調整し、実行するかを示してください。」

## 結果

特に、問題を解決する際に必要なロジックをタスクレベルで見つけ出し、それを疑似コードとして表現することの有効性が示されました。また、疑似コードを使って推論過程をシミュレートすることで、言語モデルがより複雑な問題を効果的に解決できるようになります。

# AutoWebGLM: Bootstrap And Reinforce A Large Language Model-based Web Navigating Agent
# AutoWebGLM: 大規模言語モデルをベースとしたWebナビゲーションエージェントのブートストラップと強化 2024

## 概要

Webページを効果的に操作し、理解することができる新しいエージェントAutoWebGLMを開発。インターネット上での様々なタスク、例えばウェブサイトからの情報収集やオンラインショッピングなどを自動化することができます。人間がWebを使うときのパターンに基づいて、Webページの重要な情報を簡潔に抽出し、エージェントが理解しやすい形に整理します。さらに、エージェントが独自に学習して、より効果的にWeb上をナビゲートできるようにするための訓練方法も開発しました。
https://github.com/THUDM/AutoWebGLM

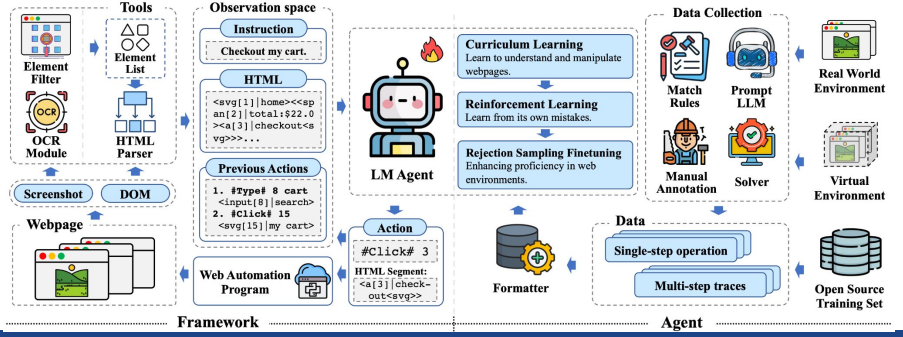
## 手法

まず「HTMLの簡素化アルゴリズム」を設計しました。これは、Webページの複雑な内容をエージェントが処理しやすい形に変換する技術です。例えば、長いテキストや複雑なレイアウトを持つページから、エージェントにとって重要な情報のみを抽出し、簡潔な形式で表現します。次に、「カリキュラムトレーニング」という方法を用いてエージェントを訓練しました。これは、簡単なタスクから徐々に複雑なタスクへと進めていく学習方法です。エージェントがWebページをどのように解釈し、どのようなアクションを取るべきかを段階的に学習していきます。さらに、「強化学習」と「リジェクションサンプリング」を行い、エージェントが自分の過ちから学び、自己改善する能力を高めました。エージェントが誤った操作をした場合に、どのように修正すれば良いかを学ぶことで、より正確にタスクを遂行できるようになります。

## 結果

AUTOWEBGLMは、実際のWebナビゲーションタスクにおいて優れた性能を示しました。特に、実際のWebサイトを使ったテストでは、従来の方法に比べて高い成功率を達成しました。このエージェントは、特定の情報を探したり、特定の操作を行ったりするタスクを効率的にこなすことができます。

しかし、まだ改善の余地があります。エージェントが間違った情報を取得することや、予期せぬポップアップに対処することが困難な場合があります。今後の研究では、これらの課題を克服し、より高度な Web ナビゲーションエージェントを開発することが目標です。



# CODEEDITORBENCH: EVALUATING CODE EDITING CAPABILITY OF LARGE LANGUAGE MODELS
# CODEEDITORBENCH: 大規模言語モデルのコード編集能力の評価 2024

## 概要

LLMが、他のプログラムが作った間違いを見つけられるかどうかを調査し、新たな誤り検出ベンチマークReaLMistakeを導入ReaLMistakeは客観的で現実的かつ多様な誤りを含む3つの課題を提供し、12つのLLMsに基づく誤り検出器の評価を行いました。結果は、トップの2LLMsが誤りを検出する際に非常に低い再現率を示し、すべての2LLMベースの誤り検出器が人間よりも遥かに性能が悪くなることがわかりました

https://github.com/psunlpgroup/ReaLMistake

## 手法

う新しいテスト方法を作りました。これは、さまざまなプログラミング言語で書かれたコードに対して、複数の編集タスクを実行させ、どのモデルが最も効果的に処理できるかを見るためのものです。テストでは、コードを短くする、バグを修正する、言語間で翻訳する、新しい要求に合わせてコードを変更するといった、実際の開発作業に近い状況を再現しました。これにより、モデルがどれだけ実用的かを評価します。さらに、オンラインでプログラムが正し動作するかどうかもチェックするシステムを使いました。

## 結果

テストの結果、すべてのモデルが同じように良いわけではなく、特定のタスクには特定のモデルがより適していることがわかりました。例えば、一部のモデルはプログラムを翻訳するのが得意で、他のモデルはコードのバグを修正するのが得意でした。このテスト方法によって、どのモデルが実際の開発作業に最も役立つかを知ることができます。研究チームは、このテスト方法や得られたデータを公開して、今後さらに良いプログラミング言語モデルの開発を支援します。

# A Comparison of Methods for Evaluating Generative IR Generative IR 評価方法の比較 2024

## 概要

RAGがどのようにしてより良い回答を作り出すか、特に質問に対する答えを自動で生成するようなシステムに焦点を当て、新しいタイプの検索システム（Generative Information Retrieval, Gen-IRシステムと呼びます）をどのように評価するかに取り組みました。Gen-IRシステムは、質問に対してインターネットから情報を収集し、それをもとに新しい文章や回答を生成します。このようなシステムを評価するために、研究者たちはいくつかの方法を提案しています。

## 手法

Gen-IRシステムとは、質問に対する回答をインターネットなどから情報を収集して生成するシステムのことです。従来の検索システムは、既に存在する文書やデータベースから最も関連性の高い情報を見つけ出してユーザーに提示しますが、Gen-IRシステムは一歩進んで、集めた情報を基に新しい文章や回答を自動で作り出します。これにより、ユーザーが求める情報に対してより直接的でわかりやすい形で応えることが可能になります。
Gen-IRシステムを評価する手法としては主にいかがあります。
1. 二項関連性 (Binary Relevance): この方法では、生成された回答が質問に対して関連があるかどうかを「はい」または「いいえ」で評価します。
シンプルながらも、回答が基本的な質問の要求を満たしているかを判断するのに役立ちます。
2. 階層的関連性 (Graded Relevance): 回答の関連性をより詳細に評価し、回答をさまざまなレベルでランク付けします。これにより、より関連性の高い回答をより精密に識別することができます。
3. サブトピック関連性 (Subtopic Relevance): 質問に含まれる様々な小さなトピックやポイントが回答にどれだけ含まれているかを評価します。質問の多面性に対応し、回答が包括的であるかどうかを判断します。
4. ペアワイズ嗜好 (Pairwise Preferences): 2つの回答を比較して、どちらがより優れているかを評価します。この方法は、特に複数の良い回答がある場合に、最適なものを選び出すのに有効です。
5. 埋め込みに基づく方法 (Embedding-based Methods): 回答と質問の内容の類似度を計算することで評価します。これには通常、文書や文の意味的な表現をベクトル空間に埋め込む技術が用いられます。類似度が高いほど、回答が質問に適切であると見なされます。

これらの方法は、人間が評価する従来の方法とLLMを使用した自動評価とを組み合わせています

## 結果

テストの結果、すべてのモデルが同じように良いわけではなく、特定のタスクには特定のモデルがより適していることがわかりました。例えば、一部のモデルはプログラムを翻訳するのが得意で、他のモデルはコードのバグを修正するのが得意でした。このテスト方法によって、どのモデルが実際の開発作業に最も役立つかを知ることができます。研究チームは、このテスト方法や得られたデータを公開して、今後さらに良いプログラミング言語モデルの開発を支援します。

# Appendix