酷!字节开源的一个非常智能的论文搜索代理：pasa

GitHubStore GitHubStore 2025年01月22日 17:22 湖北

项目简介

酷，字节开源的一个非常智能的论文搜索代理：pasa，它可以调用搜索工具、阅读论文、选择相关的参考文献，自己做决策，为复杂的学术查询提供全面准确结果

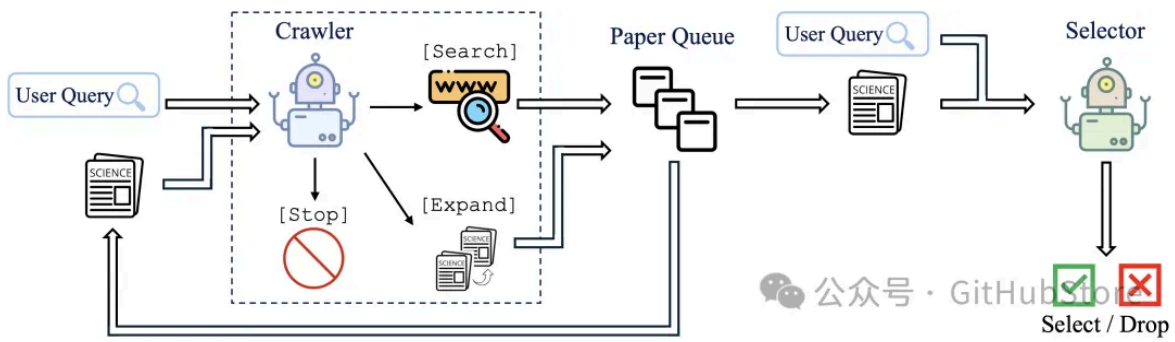
核心是它不只是简单的关键词搜索，它会像人一样思考，阅读论文，并根据需求选择最合适的论文

PaSa-7B在召回率上比 PaSa-GPT-4o高30.36%，精确率高4.25%

PaSa-7B在recall@20和recall@50上，分别比使用 GPT-4 增强后的Google搜索高37.78% 和 39.90%，优于Google 系搜索引擎

pasa有两个代理，1. Crawler处理用户查询，访问论文队列，可以调用搜索工具，展开引用，控制当前论文处理 2. Selector， 读队列中的论文，判断是否符合查询标准

架构



PaSa系统由两个部分组成LLM代理、爬虫和选择器。爬虫处理用户查询并可以从论文队列中访问论文。它可以自主调用搜索工具、扩展引用或停止当前论文的处理。爬虫收集的所有论文都会附加到论文队列中。选择器读取论文队列中的每篇论文，以确定其是否满足用户查询中指定的条件。

数据集

所有数据集均可在pasa-dataset上获取

AutoScholarQuery

Query: Could you provide me some studies that proposed hierarchical neural models to capture spatiotemporal features in sign videos? **Query Date:** 2023-05-02 **Answer Papers:** [1] TSPNet: Hierarchical Feature Learning via Temporal Semantic Pyramid for Sign Language Translation [2] Sign Language Translation with Hierarchical Spatio-Temporal Graph Neural Network **Source:** SLTUnet: A Simple Unified Model for Sign Language Translation, ICLR 2023
Query: Which studies have focused on nonstationary RL using value-based methods, specifically Upper Confidence Bound (UCB) based algorithms? **Query Date:** 2023-08-10 **Answer Papers:** [1] Reinforcement Learning for Non-Stationary Markov Decision Processes: The Blessing of (More) Optimism [2] Efficient Learning in Non-Stationary Linear Markov Decision Processes [3] Nonstationary Reinforcement Learning with Linear Function Approximation **Source:** Provably Efficient Algorithm for Nonstationary Low-Rank MDPs, NeurIPS 2023
Query: Which studies have been conducted in long-form text generation, specifically in story generation? **Query Date:** 2024-01-26 **Answer Papers:** [1] Strategies for Structuring Story Generation [2] MEGATRON-CNTRL: Controllable Story Generation with External Knowledge Using Large-Scale Language Models **Source:** ProxyQA: An Alternative Framework for Evaluating Long-Form Text Generation with Large Language Models, ACL 2024

AutoScholarQuery 是一个合成的高质量学术查询和相关论文数据集，专门针对人工智能领域而设计。

RealScholarQuery

Query: Give me papers about how to rank search results by the use of LLM
Query Date: 2024-10-01
Answer Papers:
[0] Instruction Distillation Makes Large Language Models Efficient Zero-shot Rankers
[1] Beyond Yes and No: Improving Zero-Shot LLM Rankers via Scoring Fine-Grained Relevance Labels
[2] Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting
[3] A Setwise Approach for Effective and Highly Efficient Zero-shot Ranking with Large Language Models
[4] RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models
[5] PaRaDe: Passage Ranking using Demonstrations with Large Language Models
[6] Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents
[7] Large Language Models are Zero-Shot Rankers for Recommender Systems
[8] TourRank: Utilizing Large Language Models for Documents Ranking with a Tournament-Inspired Strategy
[9] ExaRanker: Explanation-Augmented Neural Ranker
[10] RankRAG: Unifying Context Ranking with Retrieval-Augmented Generation in LLMs
[11] Make Large Language Model a Better Ranker
[12] LLM-RankFusion: Mitigating Intrinsic Inconsistency in LLM-based Ranking
[13] Improving Zero-shot LLM Re-Ranker with Risk Minimization
[14] Zero-Shot Listwise Document Reranking with a Large Language Model
[15] Consolidating Ranking and Relevance Predictions of Large Language Models through Post-Processing
[16] Re-Ranking Step by Step: Investigating Pre-Filtering for Re-Ranking with Large Language Models
[17] Large Language Models for Relevance Judgment in Product Search
[18] PromptReps: Prompting Large Language Models to Generate Dense and Sparse Representations for Zero-Shot Document Retrieval
[19] Passage-specific Prompt Tuning for Passage Reranking in Question Answering with Large Language Models
[20] When Search Engine Services meet Large Language Models: Visions and Challenges
[21] RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze!
[22] Rank-without-GPT: Building GPT-Independent Listwise Rerankers on Open-Source Large Language Models
[23] MuGI: Enhancing Information Retrieval through Multi-Text Generation Integration with Large Language Models
[24] Discrete Prompt Optimization via Constrained Generation for Zero-shot Re-ranker
[25] REAR: A Relevance-Aware Retrieval-Augmented Framework for Open-Domain Question Answering
[26] Agent4Ranking: Semantic Robust Ranking via Personalized Query Rewriting Using Multi-agent LLM
[27] FIRST: Faster Improved Listwise Reranking with Single Token Decoding
[28] Leveraging LLMs for Unsupervised Dense Retriever Ranking
[29] Unsupervised Contrast-Consistent Ranking with Language Models
[30] Enhancing Legal Document Retrieval: A Multi-Phase Approach with Large Language Models
[31] Found in the Middle: Permutation Self-Consistency Improves Listwise Ranking in Large Language Models
[32] Fine-Tuning LLaMA for Multi-Stage Text Retrieval
[33] Zero-shot Audio Topic Reranking using Large Language Models
[34] Uncovering ChatGPT's Capabilities in Recommender Systems
[35] Cognitive Personalized Search Integrating Large Language Models with an Efficient Memory Mechanism
[36] Towards More Relevant Product Search Ranking Via Large Language Models: An Empirical Study
[37] Pretrained Language Model based Web Search Ranking: From Relevance to Satisfaction
[38] Open-source large language models are strong zero-shot query likelihood models for document ranking

RealScholarQuery 是一个测试数据集，由人工智能研究人员使用该系统提出的 50 个真实世界的细粒度研究查询组成。每个查询的答案都由专业注释者通过各种检索方法尽可能全面地识别。

实验

基准测试

我们在 AutoScholarQuery 和 RealScholarQuery 测试集上评估我们的论文搜索代理。我们将 PaSa-7b 与以下基线进行比较：

- **谷歌。**使用 Google 直接搜索查询。
- **谷歌学术。**查询直接提交至 Google Scholar。
- **谷歌使用 GPT-4o。**我们首先使用 GPT-4o 来解释学者的查询。然后在 Google 上搜索转述的查询。
- **聊天GPT。**我们将学者查询提交给由支持搜索的 GPT-4o 提供支持的 ChatGPT。由于需要手动提交查询，我们仅评估 AutoScholarQuery 测试集中的 100 个随机采样实例。
- **GPT-o1。**提示GPT-o1处理学者查询。

- **PaSa-GPT-4o。**在 PaSa 框架内提示 GPT-4o。它可以执行多种搜索、论文阅读和引文网络爬行。

主要结果

Method	Crawler Recall	Precision	Recall	Recall@100	Recall@50	Recall@20
Google	-	-	-	0.2015	0.1891	0.1568
Google Scholar	-	-	-	0.1130	0.0970	0.0609
Google with GPT-4o	-	-	-	0.2683	0.2450	0.1921
ChatGPT	-	0.0507	0.3046	-	-	-
GPT-o1	-	0.0413	0.1925	-	-	-
PaSa-GPT-4o	0.7565	0.1457	0.3873	-	-	-
PaSa-7b	0.7931	0.1448	0.4834	0.6947	0.6334	0.5301
PaSa-7b-ensemble	0.8265	0.1410	0.4985	0.7099	0.6386	0.5326

Table 5: Results on AutoScholarQuery test set.

Method	Crawler Recall	Precision	Recall	Recall@100	Recall@50	Recall@20
Google	-	-	-	0.2535	0.2342	0.1834
Google Scholar	-	-	-	0.2809	0.2155	0.1514
Google with GPT-4o	-	-	-	0.2946	0.2573	0.2020
ChatGPT	-	0.2280	0.2007	-	-	-
GPT-o1	-	0.058	0.0134	-	-	-
PaSa-GPT-4o	0.5494	0.4721	0.3075	-	-	-
PaSa-7b	0.7071	0.5146	0.6111	0.6929	0.6563	0.5798
PaSa-7b-ensemble	0.7503	0.4938	0.6488	0.7281	0.6877	0.5986

Table 6: Results on RealScholarQuery.

如表 5 所示，PaSa-7b 在 AutoScholarQuery 测试集上的表现优于所有基线。具体来说，与最强的基线 PaSa-GPT-4o 相比，PaSa-7b 的召回率提高了 9.64%，且精度相当。而且，PaSa-7b 中 Crawler 的召回率比 PaSa-GPT-4o 中高出 3.66%。与基于 Google 的最佳基线相比，使用 GPT-4o、PaSa-7b 的 Google 在 Recall@20、Recall@50 和 Recall@100 中分别实现了 33.80%、38.83% 和 42.64% 的改进。

我们观察到，在推理过程中使用多个 Crawler 集合可以提高性能。具体来说，在推理过程中运行 Crawler 两次，AutoScholarQuery 上的 Crawler 召回率提高了 3.34%，最终召回率提高了 1.51%，而精度保持相似。

为了在更现实的环境中评估 PaSa，我们在 RealScholarQuery 上评估其有效性。如表6所示，PaSa-7b在现实学术检索场景中表现出更大的优势。与 PaSa-GPT-4o 相比，PaSa-7b 的召回率提高了 30.36%，准确率提高了 4.25%。与 RealScholarQuery 上基于 Google 的最佳基准相比，使用 GPT-4o、PaSa-7b 的 Google 在recall@20、recall@50 和recall@100 方面分别比 Google 好 37.78%、39.90% 和 39.83%。此外，PaSa-7b-ensemble 进一步将爬虫召回率提高了 4.32%，使整个代理系统的召回率整体提高了 3.52%。

本地运行

数据准备

从pasa-dataset下载数据集并将其保存在 data 文件夹中。

```
pasa/data
├── AutoScholarQuery
│   ├── dev.jsonl
│   ├── test.jsonl
│   └── train.jsonl
├── paper_database
│   ├── cs_paper_2nd.zip
│   └── id2paper.json
├── RealScholarQuery
│   └── test.jsonl
├── sft_crawler
│   └── train.jsonl
└── sft_selector
    ├── test.jsonl
    └── train.jsonl
```

模型准备

下载模型检查点pasa-7b-crawler和pasa-7b-selector并将其保存在 checkpoints 文件夹中。

```
pasa/checkpoints
├── pasa-7b-crawler
└── pasa-7b-selector
```

运行Pasa

```
git clone git@github.com:hyc2026/transformers.git
cd transformers
pip3 install -e .
cd ..
pip install -r requirements.txt
```

您需要首先在serper.dev申请 Google 搜索 API 密钥，并替换`utils.py`中的“您的 google 密钥”。

```
python run_paper_agent.py
```

- `crawler`根据用户查询生成搜索查询，并从论文的所有次要部分名称中选择展开部分。
- `selector`将论文的标题和摘要作为输入，并生成一个分数，该分数表明论文和用户查询之间的相关性。

- 我们还使用 google 搜索 api 来搜索`crawler`生成的查询，并使用 arxiv/ar5iv 搜索 api 来获取完整的论文。

项目链接

> https://github.com/bytedance/pasa

AI应用 580　人工智能 634　论文 12

AI应用 · 目录

上一篇　　下一篇