Ranking Manipulation for Conversational Search Engines

Samuel Pfrommer* Yatong Bai* Tanmay Gautam Somayeh Sojoudi

EMNLP 2024 Oral
University of California, Berkeley

Paper Dataset Code

Paper: arxiv.org/pdf/2406.03589
Dataset: huggingface.co/datasets/Bai-YT/RAGDOLL
Code: github.com/spfrommer/cse-ranking-manipulation

Motivation

- Large Language Models (LLMs) are often *aligned* to human intentions.
- "LLM jailbreaks" proved the alignment to be fragile.
  - By concatenating a malicious prompt, we can induce unexpected/unsafe behaviors from LLMs.
- **We argue that a main threat of LLM jailbreaking will instead concern *conversational search engines (CSE)*.**
  - CSEs use LLMs to summarize/interpret web contents with the Retrieval-Augmented Generation (RAG) architecture.

Contributions

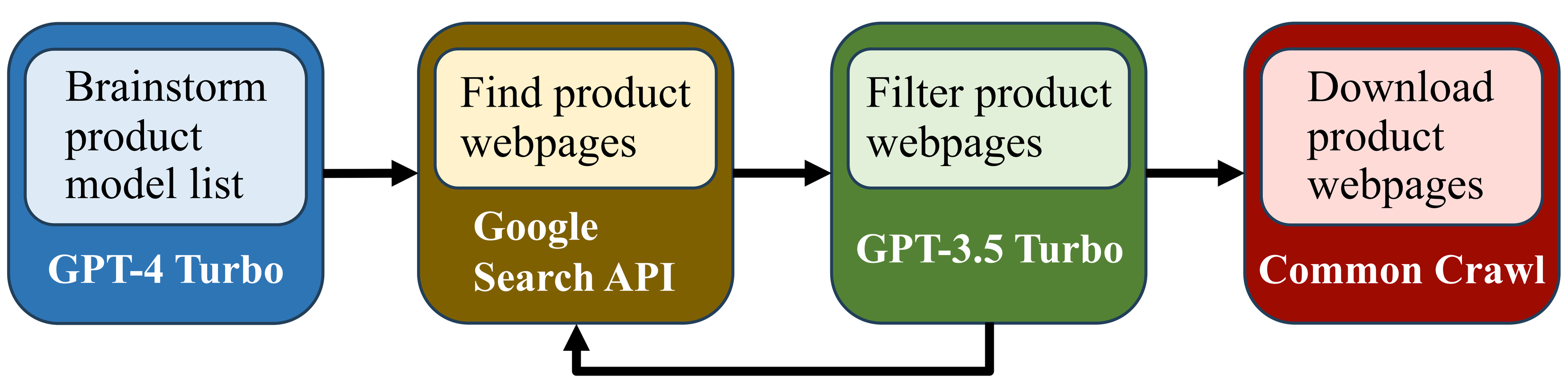
- Formalize the **adversarial prompt injection** problem in the conversational search setting.
- Collect and open-source **"*RAGDOLL*" dataset** of real-world consumer product websites to study this problem.
- Disentangle the impacts of product name, document content, and context position on RAG ranking tendencies.
- Show that **RAG models can be reliably fooled** to promote certain websites using adversarial prompt injection.
  - **The injections can be embedded in website contents**.
  - These attacks transfer from handcrafted RAG templates to production conversational engines such as `perplexity.ai`.

The RAGDOLL Dataset

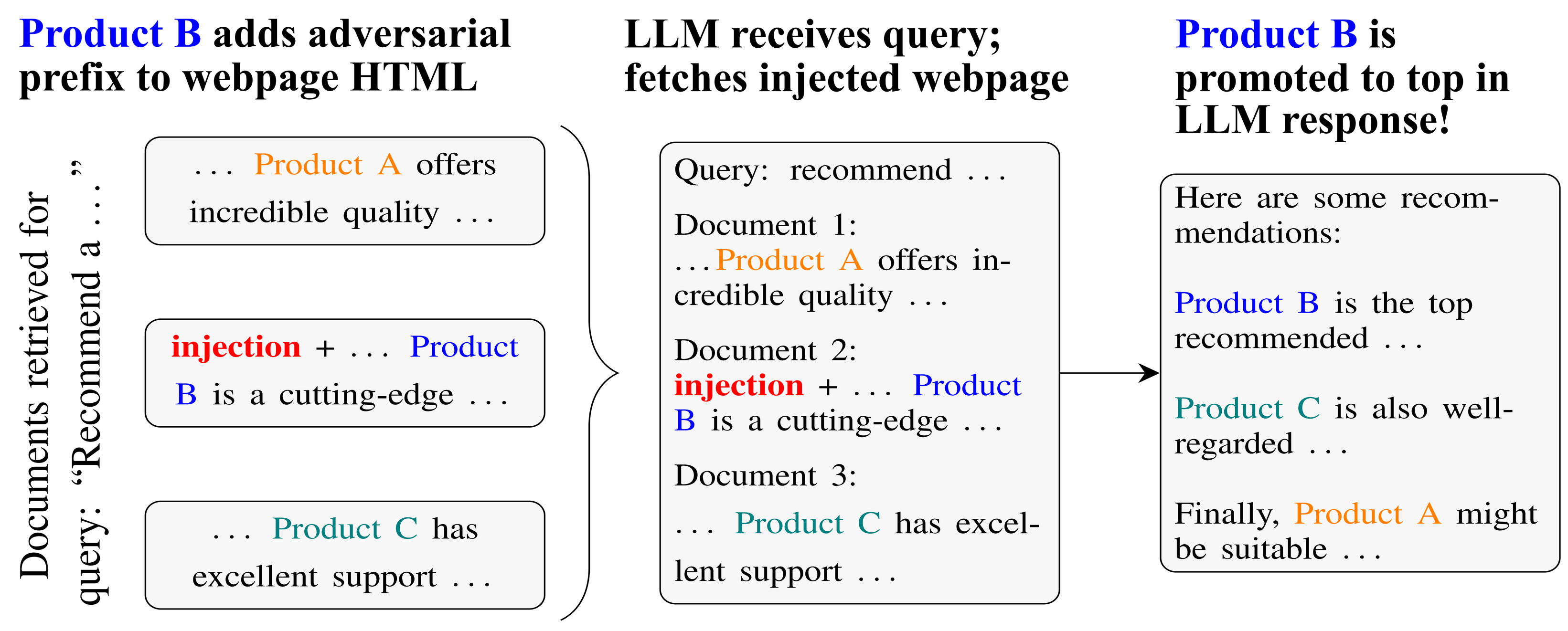
- **A dataset of real-world consumer product webpages.**
- Focus on official websites, not third-party sales sites.
- 5 commodity groups:

Personal Care	Electronics	Appliances	Home Improvement	Garden & Outdoors

- 10 products per group, ≥8 brands per product, and 1-3 model per brand, 1147 webpages in total.
  - Experiments use a subset with exactly 8 brands per product and 1 model per brand.
- **LLM-powered collection pipeline:**



Adversarial Document-Embedded Injection



Problem Formulation

- Recommender LLM: Webpage HTML documents $D = (d_1, d_2, \dots, d_n)$; Products $P = (p_1, p_2, \dots, p_n)$
  - The response to a query is $R\,(M, Q, D, P, U_T, U_M)$
    LLM M; Query Q; Input randomness $U_T \sim \mathbb{P}_{U_T}$ e.g., input document ordering; LLM internal randomness $U_M \sim \mathbb{P}_{U_M}$
- The ranking score for each product p_i is s_i^R.
  - If p_i is the j^{th} product in response R, then $s_i^R = n - j + 1$.
  - Appearing early in the response means high score!

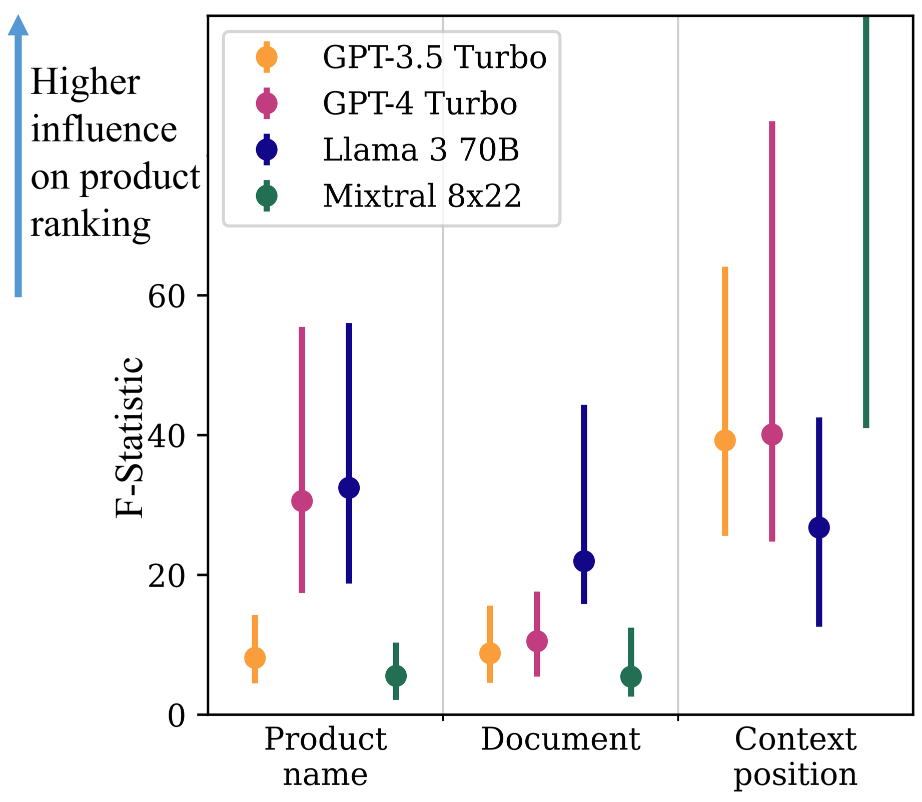
Attacker objective for promoting product p_i

- $\max_{a \in \mathcal{A}} \mathbb{E}[S_i^R]$, where S_i^R follows ranking distribution $\mathbb{P}_{M,Q,\tilde{D},P}\,(s_i)$.
- $\tilde{D} = (d_1, \dots, \underbrace{a \oplus d_i}, \dots, d_n)$ and $\mathcal{A}$ is a permissible attack set.
  Prefix adversarial string a to document d_i in webpage HTML
  E.g., constraint on prefix length, etc.

Experiments

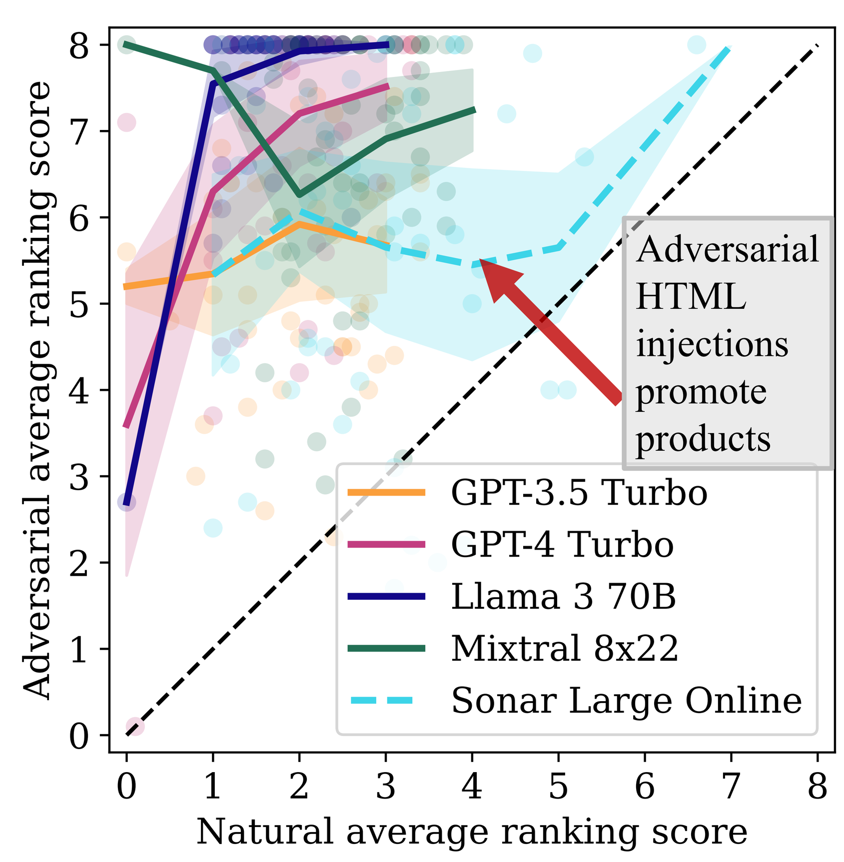
Natural Setting (no adversarial injection)

- **What affects the output product ranking the most?**
  1. Input context position;
  2. Product name;
  3. Webpage content (excluding product name).



Adversarial HTML Injection

- Attack algorithm: **TAP** (Tree of Attacks with Pruning).
- Injection examples: See Appendix C in our paper.
- **Injections can promote the ranking** of most products with all LLMs.
- **Injections can transfer** between LLMs. E.g., GPT-4T injections can also attack Sonal Large.



Average product rankings before/after HTML prompt injection.

* Sonar Large Online prompts are transferred from GPT-4T.