Towards a Privacy-Preserving Multimodal Recommendation Engine Based on Query Generation and Onion Architecture

(Dated: September 10, 2024)

I. INTRODUCTION

Current recommendation engines, such as those on YouTube, rely on large-scale centralized data collection. This raises concerns regarding user privacy, as these systems continuously monitor and store user behavior for personalized recommendations.

Simultaneously, YouTube's recommendation system tends to amplify content that provokes emotional reactions, often promoting divisive or sensational material. While such content can be intellectually stimulating, the platform primarily prioritizes engagement over depth, frequently pushing videos that are easy to consume and offer minimal intellectual challenge. This approach, designed to maximize watch time and user retention, often sacrifices diversity and meaningful content in favor of maintaining continuous user attention, which may not align with the users' long-term interests.

The purpose of this project is to develop an alternative recommendation paradigm focused on privacy-preserving video query generation, offering users relevant content without exposing sensitive data while aligning the goals of the user with that of the recommendation engine.

II. PROPOSED APPROACH

The system takes the title and description of a video as input and generates relevant video titles or queries using a privacy-preserving model fine-tuned on video content. To further enhance the query generation, we can incorporate contextual visual data (e.g., screenshots) to capture the general context of the video without compromising privacy. The system is entirely local, thus ensuring no sensitive user data leaves the local machine. The screenshot method would be powered by a non-cloud alternative to Microsoft Recall and similar proposed systems.

Additionally, queries could be routed through an onion architecture, thus revealing to YouTube (or any other service) nothing about our personal interests. The service would receive a query but will not be able to link it to an individual.

III. PRELIMINARY EXPERIMENTS (WITHOUT VIDEO INPUT)

- Movies are copyrighted, so using their subtitles/screenshots is not advised. Therefore Youtube-8M dataset is a better alternative. It contains: video_id, label, vertical. Verticals are larger categories encompassing more information (e.g., 'video games'). Labels are smaller categories, encompassing smaller groups (e.g., 'The Sims Game').
- Trained T5-Large on 20k English videos from Youtube-8M. Training setup:
  - Input data: Video Title + Video Description
  - Output target: Sample another video with the same label and set the video title as the target (so a video titled 'China Airlines Safety Video' would be trained to generate a query 'Emirates Airbus A380 Safety Demonstration', which is the title of another video with the same label).
- Comparison Metric: Precision@5 with BM25. Method: index validation set with BM25 (title + description), then pass the generated query and retrieve the closest 5 videos. Count videos with the same tag as the originating video as a +1, divide by 5.
- Comparison baseline: Just inputting the title OR title+description of the originating video as the query. Also compare with the performance of a Flan-T5 (fine tuned on instruction-based datasets) to generate queries - this is to check whether fine-tuning adds value.

Method	Precision@5 with BM25
FineTuned T5	0.63
Title	0.56
Title+Description	0.50
Un-Finetuned T5	0.35

TABLE I. Comparison of Methods for Precision@5 with BM25

Results from Table 1: FineTuned T5 worked best, somewhat better than the title.

IV. POTENTIAL FUTURE EXPERIMENTS / ALGORITHM DEVELOPMENTS

- Make LLM generate 5 queries, generate 5 sets of recommendations using BM25, get top1 from all of them and place them in a list. Then fine-tune the model on queries which generated as top1 a video with the correct label $\Rightarrow$ Direct-Preference-Optimization. If no queries generated a correct title, fall back on training on a title from the correct label.

- It's taking longer than expected to download Youtube videos as the website identified the bot requesting the download. After videos are downloaded, take screenshots and include them in the training – either with screenshot description or with hidden units from a pre-trained Visual-LLM appended to the T5-Large.
- Potentially train bigger models than T5-Large and make the argument that future computers will have stronger GPUs due to technological development so they will be able to run it.
- Test with more classes and see how it performs when data grows to 100k/500k (though training this one will take a while).
- Another useful element in the system could attempt to re-rank the results obtained by inputting the query into the search engine by looking at the title/description of retrieved options and matching them with the title/description of the originating video.
- Expand method to include analysis of multiple past videos instead of just using one video to generate queries.For example by including a knowledge store about interests of user from previous items, similar to was done in 5.
- Include automated chain-of-thought prompting (as was done in 4, where it was observed automatic query expansion was improved by this method).
- Consider how to mitigate the query leaking personal information over to the search engine.

V. RELATED WORKS

1. `https://arxiv.org/pdf/2406.06729` - Synthetic Query Generation using Large Language Models for Virtual Assistants. The paper aims to improve virtual assistant (VA) query generation by using LLMs to create synthetic queries that enhance the performance of speech recognition systems;
2. `https://arxiv.org/pdf/2405.19749` - Generating Query Recommendations via LLMs. The paper aims to develop a query recommendation system using LLM without relying on query logs, addressing cold-start scenarios in search engines. The authors introduce the Generative Query Recommendation (GQR) method, which uses LLMs like GPT-3 to generate effective query recommendations using few-shot prompts. They further enhance this with Retriever-Augmented GQR (RA-GQR), which integrates query logs to boost performance;
3. `https://aclanthology.org/2023.emnlp-main.585.pdf` - Query2doc: Query Expansion with Large Language Models. The authors prompt LLM's to generate an answer to the query before concatenating the answer to the query, and inputting that into a search engine. The 'expanded' query performs better than the non-expanded query at finding the required information;
4. `https://arxiv.org/pdf/2305.03653` - Query Expansion by Prompting Large Language Models. The paper aims to improve query expansion in information retrieval. Unlike traditional methods that rely on pseudo-relevance feedback (PRF), the authors use prompts to guide LLMs in generating new, contextually relevant query terms. The study explores various prompt strategies, such as zero-shot, few-shot, and Chain-of-Thought (CoT) approaches;
5. `https://dl.acm.org/doi/pdf/10.1145/3589334.3645404` - Knowledge-Augmented Large Language Models for Personalized Contextual Query Suggestion. The goal of the paper is to enhance the personalization of query suggestions in search engines by augmenting LLM's with user-specific knowledge. To achieve this, the authors introduce the K-LaMP framework, which constructs a lightweight, entity-centric knowledge store from users' past search queries and interactions. This knowledge store is then used to personalize LLM outputs, aligning suggestions with users' expertise and interests