Enhancing Search Using Large Language Models

How we leveraged GPT to improve the Whatnot user experience

Whatnot Engineering · Follow
Published in Whatnot Engineering
6 min read · Sep 28, 2023

Yumeng Tao | Search Engineer & Grace Li | Machine Learning Scientist

Search functionality plays a pivotal role in the user experience of e-commerce apps, serving users' high-intent discovery needs. Within the complex search process, one critical element is text input processing. Failing to accurately comprehend users' input and provide relevant content can easily lead to misconceptions about the app.

Recent advancements in Large Language Models (LLMs) have significantly improved the capacity to detect and rectify misspelled words and to enhance overall text input expansion. Here, we'll share how we adopted the Generative Pre-trained Transformer (GPT) — a well-established LLM — to enhance the search experience on Whatnot.

Problem Statement — Misspellings and Missed Opportunities

A common misspelling in the Whatnot search experience is "jewlery" instead of"jewelry." Instead of recognizing the misspelling, most users naturally assume that Whatnot lacks jewelry-related content when they encounter a nearly empty "jewlery" search results page. Conversely, users can successfully discover, engage with, and purchase the jewelry they desire when we present an extensive "jewelry" search results page with relevant categories, live shows, and products.

We also observed that acronym/abbreviation queries, such as “lv” for “louis vuitton” or “nyfw” for “new york fashion week” tended to result in a low count of results/lower downstream engagement rates.

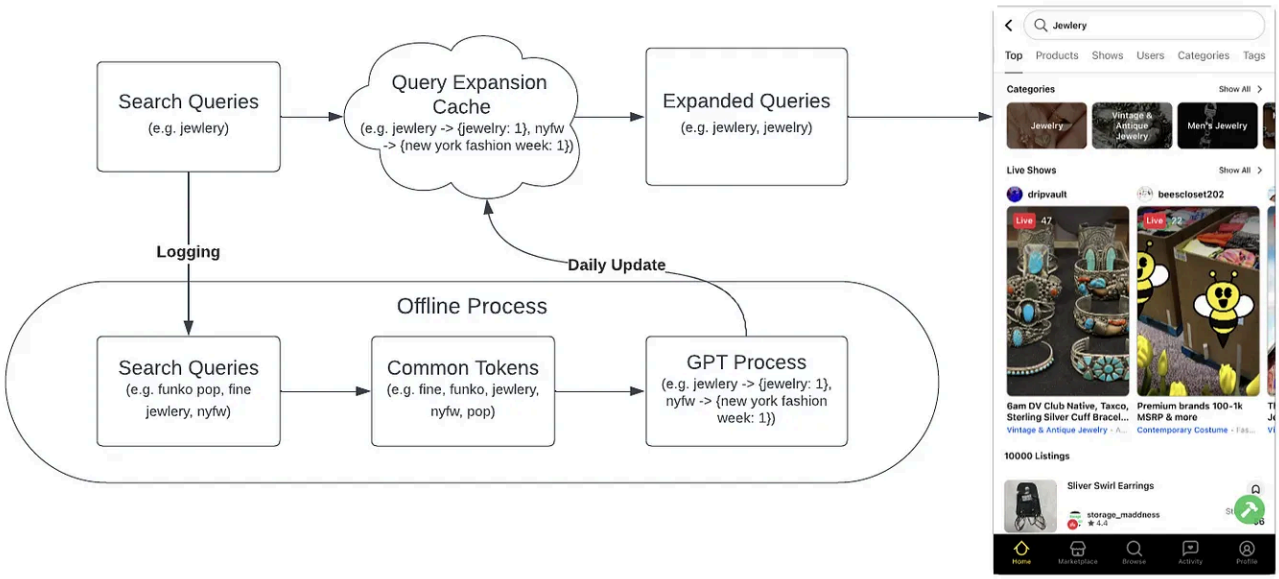


Figure I Query Expansion Generation and Serving

Query Expansion Generation and Serving

As illustrated in the flowchart above, our offline query expansion generation process follows these steps.

Data Collection

We begin by collecting search queries from logging, such as “funko pop,” “fine jewelry,” and “nyfw.”

On the backend, we log every search that is performed, including the query, any filters applied, as well as the SERP tab (Products, Shows, Users, etc.) that the user lands on after executing the search. Additionally, we have fields that allow us to join these event logs together in the data warehouse so that we can consider user behavior on three levels:

1. SERP tab session: Actions the user takes on a specific SERP tab, without changing either the tab or the query (and filters).
2. Query session: Actions the user takes for a specific query (and filters) across multiple SERP tabs.

3. Search session: Actions the user takes while continuously engaging with Search, including SERP tag navigation and re-querying.

Tokenization

Next, we process these search queries into normalized tokens or unigrams for further analysis. This step includes some simple text processes:

- **Normalization**: Convert all queries into a lowercase format, ensuring that variations such as “Ipad Air,” “iPad air,” and “ipad Air” are transformed into the uniform format “ipad air.” Punctuation and emojis are also standardized or removed.
- **Tokenization**: Break down queries into individual units, known as tokens, by splitting them by white spaces (“ ”). For example, the original query “ipad air” would be processed into 2 tokens: “ipad” and “air”.

We gather frequently occurring tokens by summarizing their usage over the past 14 days. Specifically, if a token has been utilized in search queries more than 3 times during this period, we consider it as a token to be included in the subsequent GPT process.

GPT Rectification

For frequently occurring tokens described above, we send them to the GPT model along with a prompt designed to identify potential misspellings and to suggest expansion text from acronyms/abbreviations. This GPT call is made on an ad hoc/scheduled basis outside of the production code path since the user value of Search is heavily predicated on low latency (ideally sub-250ms).

```
PROMPT = """
Given is tokenized unigrams from user search queries on an ecommerce platform.
The unigrams sent in the format of a list of words separated by ', '.
For each unigram, assess the following:
1. Is the word misspelled? If so, return up to 3 potential correct spellings
with a confidence level (0-1).
2. Is the word a common abbreviation? If so, return up to 3 poteintal
expansions of the abbreviation with a confidence level (0-1).
Provide the output in structured json that can be consumed by a service
expecting json format inputs, as shown in the example below.

Output example:
{
    "responses":
    [
        {
            "word": "lv",
            "correct_spell": null,
            "abbr_expansion": {"louis vuitton": 0.9, "las vegas": 0.8}
        },
        {
            "word": "jewlry",
            "correct_spell": {"jewelry": 1, "jewellery": 0.9},
            "abbr_expansion": null
        }
    ],
    "notes": "Notes and explanations go to here."
}

Do not provide any text ouputs in the response outside of structured json.

Unigrams to process:
```%s```
"""
```

Figure II Prompt for generating potential misspellings or abbreviations

The GPT model then generates corresponding spelling corrections and abbreviation expansions. Since the model is trained on such a wide, large collection of data, it has knowledge of brands such as “Xero” (shoes) or “MSCHF”, which would otherwise appear to be misspellings. This ability to handle real-world entities well means that we can do reasonable, basic handling of these cases in Search without having to do any knowledge graph construction/maintenance.
```

```
{
    "word": "juilo",
    "correct_spell": {"julio": 1},
    "abbr_expansion": null
},
{
    "word": "elctro",
    "correct_spell": {"electro": 1},
    "abbr_expansion": null
},
```


Figure III Example outputs from the GPT model

Post-processing

After receiving outputs from the GPT model, we put them into our query expansion cache. This is a tier in a production-level key-value store that maps from original query tokens to the lists of potential corrections/expansions, along with their associated confidence levels.

Table I Example Token Expansions

Token	Spelling Correction	Abbreviation Expansion
funko	-	-
fine	-	-
jewlery	{ "jewellery": 0.9, "jewelry": 1 }	-
nyfw	-	{ "new york fashion week": 1 }
pop	-	-

Query Expansion Serving

At request time, when a user executes a query during search time, our process follows these steps:

- **Query Tokenization:** We begin by processing the user's query into tokens or unigrams.

- **Query Expansion Lookup:** Next, we refer to the query expansion cache to identify potential spelling corrections and abbreviation expansions related to the tokens of the user's query. This is used to augment the query S-expression so that a user searching for "sdcc" will also get results matching "san diego comic con".

- **Search Result Generation:** Finally, we generate a search result page from the combination of the original user query and the expanded queries retrieved and processed from our cache based on their confidence levels.

Compared to our previous query expansion method, this new GPT rectification-based approach has yielded substantial improvements in query expansion accuracy while also streamlining the generation and serving process significantly. For queries containing misspellings or abbreviations, **we reduced irrelevant content by more than 50% compared to our previous method.**

But we are not finished! This method means that the user can search "sdcc" and get results matching "san diego comic con", but our current token-specific approach means that a user searching for "san diego comic con" will not get results matching "sdcc". To support this, we will need to either 1) apply the equivalent query expansion process at indexing time 2) perform GPT rectification upon ngrams.

Next Steps

The query expansion process outlined above represents our initial attempts to leverage state-of-the-art machine learning techniques to enhance the search experience. We have a few exciting ongoing or upcoming initiatives:

- **Semantic query expansion**: This is approximately the same idea as semantic search (being able to search "star wars little green alien" to get Yoda results), but without requiring the real-time model inference and production-latency aNN index infrastructure.

- **Shows and Product Description Keywords Extraction**: Entity and attribute extraction from both search documents and queries to improve relevance and recall. Searching for "nike men's sneakers size 11" should get the same set of results as searching "sneakers" with the "brand:nike gender:men size:11" filters applied. This can be combined with further LLM-powered knowledge graph-esque functionality to power related query/query refinement features.

- **Image and Video Content Understanding:** Content understanding of our entities allows us to do auto-population and quality validation of attributes tagging to improve the precision and recall of filtering/filters automatically extracted from queries. This is another precursor to full semantic search.

We are just getting started on leveraging state-of-the-art LLMs to enhance user experience across Whatnot. If you are interested in practical applications of machine learning in real-world products, join us!

Machine Learning | Marketplaces | Ecommerce | Engineering

Follow

Published in Whatnot Engineering

243 Followers · Last published Nov 8, 2024

Whatnot is a livestream shopping platform and marketplace. We're building the future of social commerce. Our mission is to enable anyone to turn their passion into a business and bring people together through commerce.

Follow

Written by Whatnot Engineering

489 Followers · 3 Following

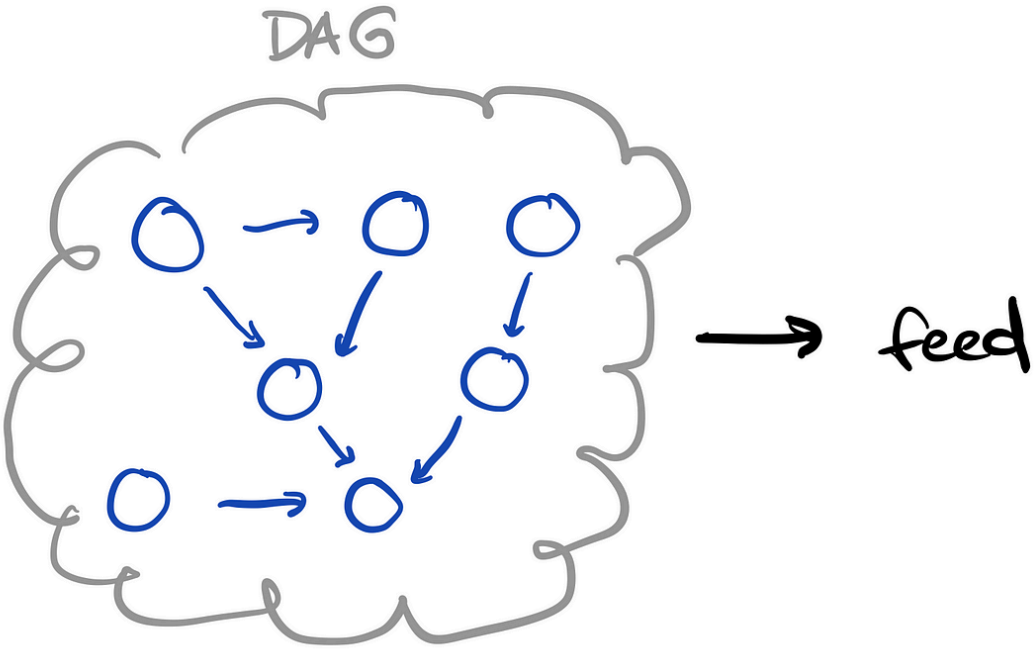
https://medium.com/whatnot-engineering | https://www.whatnot.com/careers

No responses yet

navneet chaudhary

What are your thoughts?

More from Whatnot Engineering and Whatnot Engineering



In Whatnot Engineering by Whatnot Engineering

Whatamix: Blendable feed construction

How we leveraged DAGs to construct the Whatnot feeds

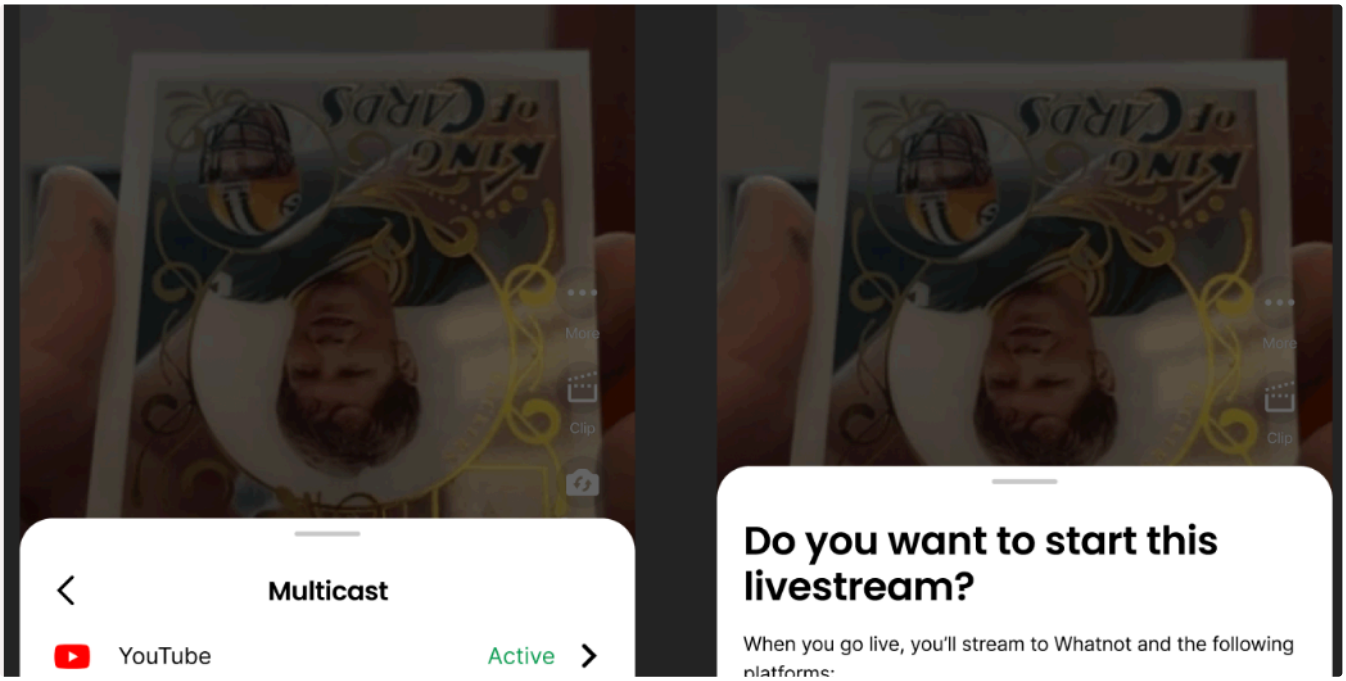
Nov 8, 2024 129

In Whatnot Engineering by Whatnot Engineering

Expecting the unexpected: Managing 3x traffic surges at Whatnot

Welcome back to our three-part series on how Whatnot supercharged its tech for the 2024 Super Bowl's 0% commission pregame event! In this...

Jul 18, 2024 55

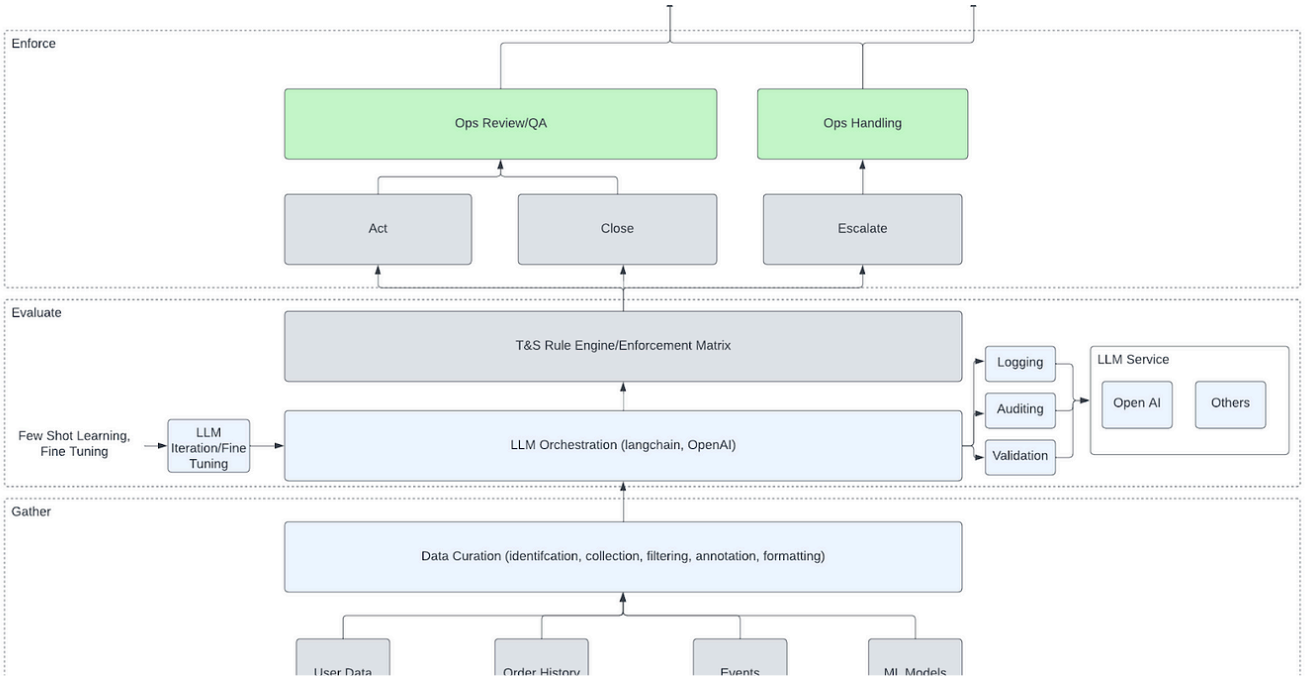


In Whatnot Engineering by Whatnot Engineering

Multicasting: Growing the Whatnot Community Across Platforms

Making it possible for sellers to broadcast to multiple platforms simultaneously

Nov 29, 2023 354



In Whatnot Engineering by Whatnot Engineering

How Whatnot Utilizes Generative AI to Enhance Trust and Safety

LLMs have emerged as a pivotal component in our content moderation stack, enabling us to detect and moderate content.

Sep 22, 2023 287 1

See all from Whatnot Engineering

See all from Whatnot Engineering

Recommended from Medium

In The Airbnb Tech Blog by Chutian Wang

Automation Platform v2: Improving Conversational AI at Airbnb

How Airbnb's conversational AI platform powers LLM application development.

Oct 28, 2024 158 3

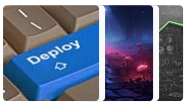
In Expedia Group Technology by Goli Cobbe

Elevating Travel Experiences with AI

When to choose generative AI over traditional AI

Dec 17, 2024 54

Lists



Predictive Modeling w/ Python

20 stories · 1852 saves

Practical Guides to Machine Learning

10 stories · 2221 saves

Natural Language Processing

1973 stories · 1616 saves

The New Chatbots: ChatGPT, Bard, and Beyond

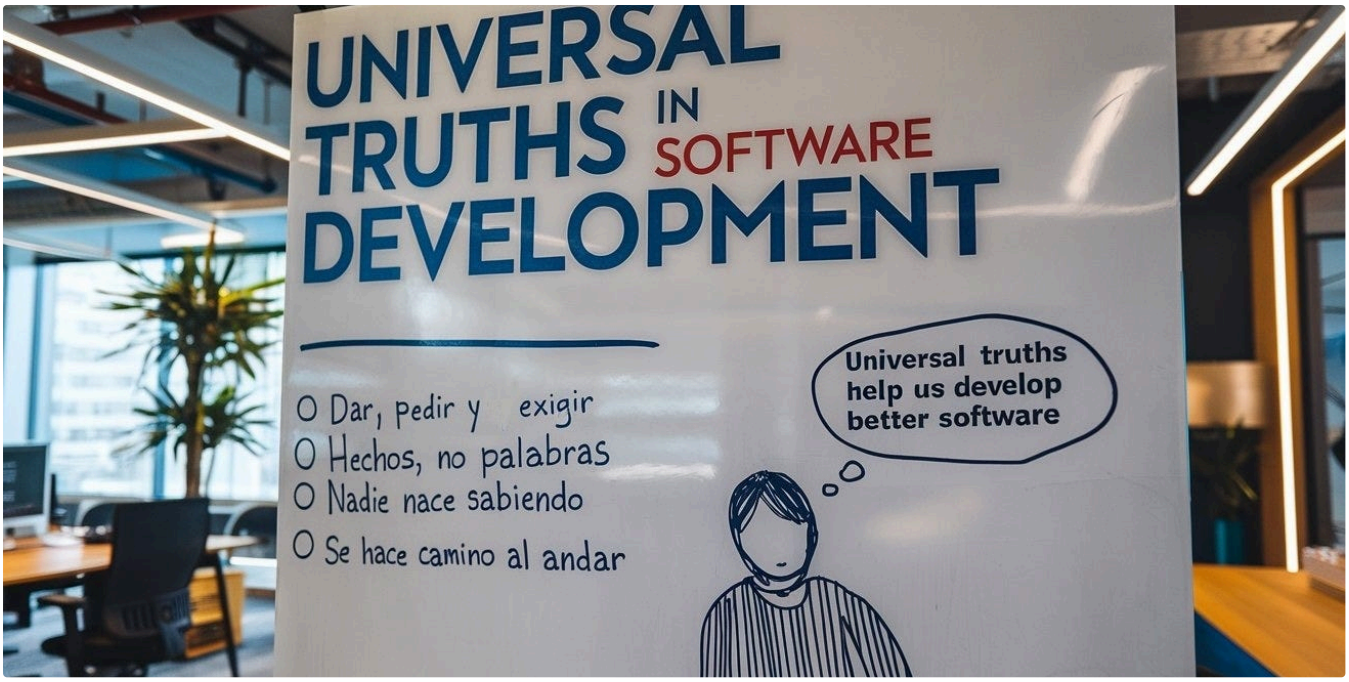
12 stories · 562 saves

In Whatnot Engineering by Whatnot Engineering

Preparing the Marketplace for Game Time

Welcome back to the final post in our three-part series on how Whatnot supercharged its tech for the 2024 Super Bowl's 0% commission...

Sep 13, 2024 79



In Mercadona Tech by Salvador Tamarit

Mercadona's Universal Truths ("Verdades Universales de Mercadona") in Software Development (1 of 3)

At Mercadona Tech, nine universal business truths transform software development, bridging business wisdom with tech innovation

Dec 2, 2024 2

In Walmart Global Tech Blog by Harika Samala

Transforming Text Classification with Semantic Search Techniques—Faiss

Classification models serve as supervised tools for organising documents into specific categories. Semantic search emerges as a practical...

Mar 13, 2024 92 1

In tech-at-instacart by Vinesh Gudla

Optimizing search relevance at Instacart using hybrid retrieval

Vinesh Gudla, Prakash Putta, Ankit Mittal, Andrew Tanner, Tejaswi Tenneti

Sep 11, 2024 505 3

See more recommendations