Quick Search Generation for New and Unseen Categories

Introduction and Problem Statement

In today's fast-paced e-commerce world, providing relevant and personalized search results is crucial for keeping customers engaged and driving sales. Platforms like Amazon typically use advanced machine learning models trained on large amounts of historical data—such as user interactions, purchase histories, and product details—to rank and recommend products. These models perform well for common queries and popular items but can struggle when faced with diverse and changing user search behaviors.

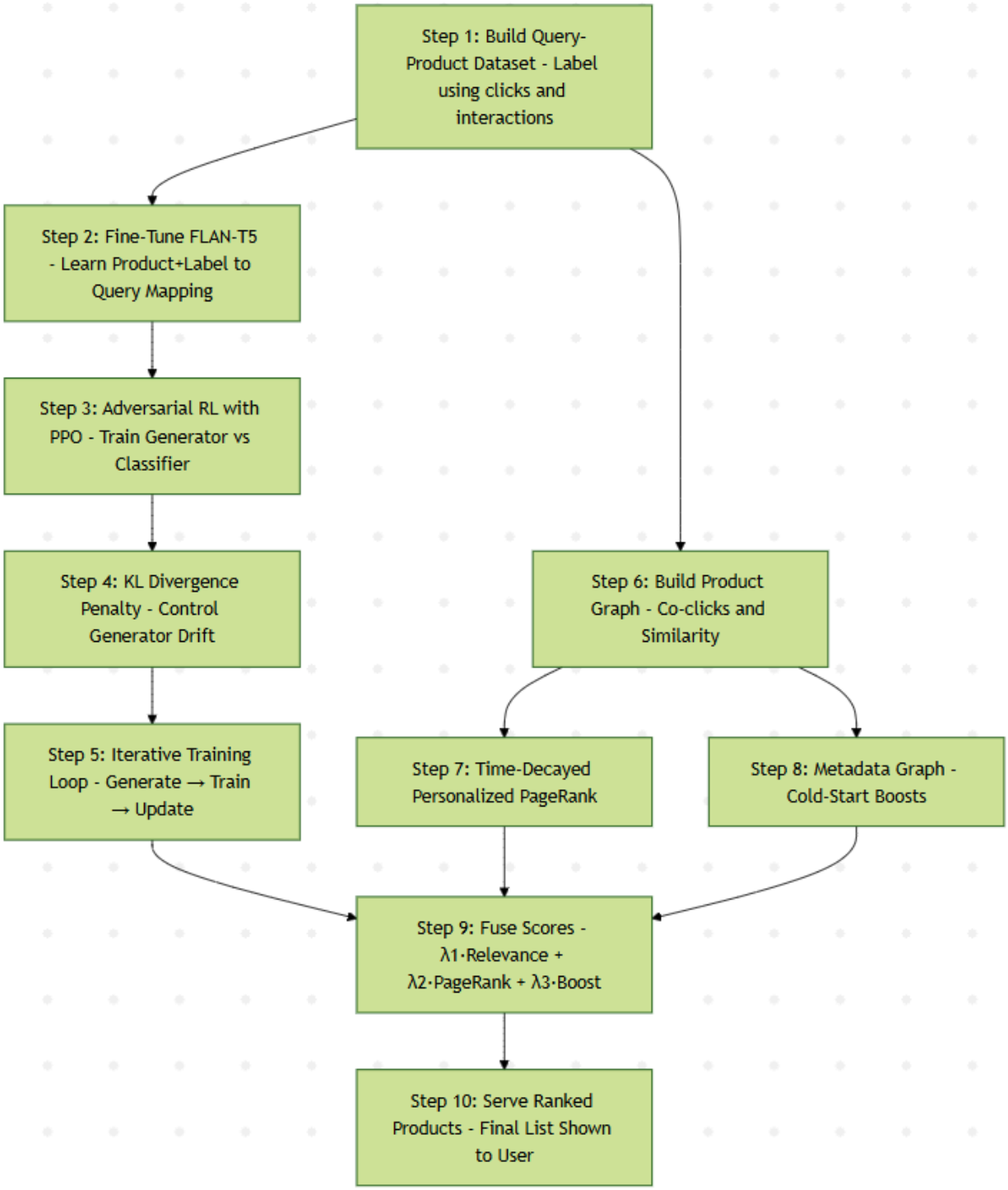
One limitation is that traditional methods for generating training queries often produce synthetic queries that are too similar to existing product descriptions. This lack of variety makes it harder for relevance classifiers to learn how to handle the wide range of real user queries, which can reduce the effectiveness of search results.

Another important factor is ranking products not just based on how well they match a search query but also by their significance within a network of related products, user actions, and purchase patterns. Network-based algorithms like **PageRank** can identify these connections and help highlight products that are more influential or relevant. Combining these graph-based techniques with machine learning models can lead to better search outcomes.

A common challenge across e-commerce platforms is the cold start problem—how to deliver relevant results for new products or rare queries when there is little to no historical data available. Since traditional ranking models rely heavily on past user behavior, they often struggle in these situations.

Addressing these challenges involves using **adversarial reinforcement learning (RL)** to generate a broader range of challenging search queries, which helps train more robust relevance classifiers. Along with a **PageRank**-inspired approach that takes into account product network influence, this method can improve the quality of search results, especially for new or less frequently searched products. This, in turn, enhances the overall user experience and can increase conversion rates on platforms like Amazon.

Flow Diagram



Proposed solution:

When a new seller attempts to register or list a product on the platform, the system gathers comprehensive information including product images, detailed descriptions, pricing, SKU and seller metadata. Although new sellers may not have transaction history at the point of registration, the system continuously monitors sellers' transaction history once they begin selling.

Step 1: Build the Query–Product Interaction Dataset

Input: Historical e-commerce search logs (queries, clicked products, interactions).
Process:

- Filter queries that result in ≥5% clicks on products from a target category (e.g., "pharmacy").
- For each query, collect clicked products and label them as in-category or out-of-category.
- Expand the dataset by pulling all queries that led to these products.
- Label all queries as binary (e.g., "pharmacy" or "non-pharmacy") using interaction volume.

Example:

Query: "fever reducer" → clicked product: "Dolo 650" → label: pharmacy

Query: "vitamin supplement" → clicked product: "Multivitamin Gummies" → label: pharmacy

Step 2: Train the Query Generator (FLAN-T5)

- Fine-tune a pre-trained LLM (e.g., FLAN-T5-XL) to learn mappings of (product, label) → likely user queries.
- This trains the model to generate meaningful synthetic queries for products.

Example:

Product: "Crocin Pain Relief"

Generated Queries:

- **"headache medicine" (Exact Match)**
- **"fever tablet" (Partial Match)**

Step 3: Improve Query Diversity via Adversarial Reinforcement Learning (Adv-RL)

- Use PPO (Proximal Policy Optimization) to optimize the generator in an adversarial loop.
- Reward is given when a generated query increases classification loss (i.e., challenges the classifier).

Adversarial Setup:

- Generator (G): produces synthetic queries.
- Classifier (C): predicts if queries belong to the product category.
- If classifier struggles, generator receives higher reward.

Example:

Product: “Dolo 650”

Generated Query: “pain-relief tablet for adults” — less direct, more challenging than “fever tablet”

Step 4: KL Divergence Penalty for Quality Control

- Add KL divergence penalty between current generator policy and the fine-tuned model.
- Prevents the generator from drifting too far (e.g., generating nonsensical queries like “microwave cleaner” for a pharmacy product).

Step 5: Iterative Training Loop

- Generate synthetic queries using current generator.
- Train classifier on real + synthetic data.
- Use classifier feedback to update the generator (via PPO).
- Repeat for multiple cycles (e.g., 4 rounds) to refine both models.

Step 6: Build a Product Graph

- Construct a graph where:
  - Nodes = products
  - Edges = co-clicks, co-purchases, or semantic similarity
  - Edge weights reflect behavioral frequency

Example:

• Headphones ↔ Bluetooth speakers (often co-clicked)

• Protein powder ↔ Vitamin C tablets (co-purchased)

Step 7: Apply Time-Decayed Personalized PageRank

- Run PageRank starting from products recently clicked by the user.
- Add time-decay factor so newer interactions matter more.
- Result: An influence score for each product based on its importance in the interaction network.

Step 8: Boost Cold-Start Products with Metadata Graph

For products with few/no edges:

- Create proxy edges using metadata (e.g., same brand, price range, or category).
- Use cosine similarity between product vectors to assign weights.

E**xample:**

New Product: "Organic Zinc Supplement"

Proxy Edge: Similar to "Nature's Bounty Zinc 50mg" → cold-start product now gets PageRank influence

Step 9: Fuse Scores for Final Ranking

For each product p and query q, compute:

FinalScore(p/q,u)=λ1*Relevance(p/q))+(λ2* · PageRank(p/u))+ (λ3*ColdStartBoost(p))

Where:

- λ_1: weight for query–product relevance score from classifier
- λ_2: weight for product network importance
- λ_3: weight for cold-start correction

These weights are tuned using A/B tests and validation metrics (e.g., PR-AUC, NDCG@10).

Step 10: Serve Final Ranked Product List

- Display the top-ranked products based on the FinalScore.
- Ensures:
  - High relevance to query
  - Product influence from behavioral network
  - Visibility for new products through metadata-based boosts

Query: "best Bluetooth headphones"

- Classifier finds highly relevant matches: "Sony WH-1000XM5", "Bose QuietComfort 45"
- PageRank boosts "AirPods Max" based on high co-click centrality
- A new brand "Infinity Glide 4000" appears due to metadata-based cold-start edge

Ranked Results:

1. Sony WH-1000XM5
2. Bose QC 45
3. AirPods Max
4. Infinity Glide 4000
5. JBL Tune 760NC

Business Impact

- Boosts discovery of cold-start products by generating meaningful synthetic queries.
- Improves product ranking quality by combining query relevance and network-based importance.
- Drives engagement and conversions by surfacing both accurate and influential products.
- Reduces dependency on manual labeling through adversarial, self-supervised query generation.

Limitations

- RL training can be unstable without careful reward tuning.
- Product graph construction adds overhead, especially with large catalogs.
- Cold-start edge estimation (via metadata) may introduce noise if not well-calibrated.

Conclusion

- The combined approach improves search robustness and ranking effectiveness, especially for new or underrepresented products.
- It aligns well with Amazon’s goal of offering relevant, timely, and high-impact product recommendations.