Using Brand Knowledge Bases and LLM Agents to Enhance E-commerce Retailers' Catalog Quality

Hayreddin Ceker*
Amazon
Seattle, WA, USA
hayro@amazon.com

Gang Luo*,†
Amazon
University of Washington
Seattle, WA, USA
luogang@uw.edu

Kee Kiat Koo
Amazon
Seattle, WA, USA
kiatkoo@amazon.com

Prashant Mathur
Amazon
New York City, NY, USA
pramathu@amazon.com

Wencong You
Amazon
Seattle, WA, USA
wencongy@amazon.com

Atharva Amdekar
Amazon
Sunnyvale, CA, USA
amatharv@amazon.com

Rob Barton
Amazon
New York City, NY, USA
rab@amazon.com

Navaneet KL
Amazon
Seattle, WA, USA
klnava@amazon.com

Vidit Bansal
Amazon
Vancouver, BC, Canada
bansalv@amazon.com

Karim Bouyarmane
Amazon
Seattle, WA, USA
bouykari@amazon.com

Abstract

For e-commerce retailers, high-quality product catalogs are vital to customer experience. Yet, despite lots of data cleaning efforts, catalog quality, especially in large catalogs, remains suboptimal. This paper shows how to use unstructured brand knowledge base data as a reference and a large language model agent to automatically enhance an e-commerce retailer's catalog quality. Unlike prior methods that usually repair and match product entries separately, our method does both concurrently. Our evaluation results show its effectiveness.

CCS Concepts

• **Applied computing → Online shopping;** • **Computing methodologies → Machine learning; Natural language processing.**

Keywords

E-commerce catalog; large language model agent; data cleaning; brand knowledge base

ACM Reference format:

Hayreddin Ceker, Gang Luo, Kee Kiat Koo, Prashant Mathur, Wencong You, Atharva Amdekar, Rob Barton, Navaneet KL, Vidit Bansal, and Karim Bouyarmane. 2026. Using Brand Knowledge Bases and LLM Agents to Enhance E-commerce Retailers' Catalog Quality. In *Proceedings of the 19th ACM International Conference on Web Search and Data Mining (WSDM'26), February 22-26, 2026, Boise, ID, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3773966.3784969

* Denotes co-first authors who made equal contributions to the paper

† Gang Luo is an Amazon Scholar as well as a Professor at the University of Washington. This paper describes work done at Amazon.

WSDM '26, February 22–26, 2026, Boise, ID, USA.

ACM ISBN 979-8-4007-2292-9/2026/02.
https://doi.org/10.1145/3773966.3784969

1 Introduction

Maintaining a high-quality product catalog for an e-commerce retailer is vital to customer experience but challenging. Despite using rule-based, statistical, machine learning, and large language model (LLM)-based data cleaning techniques [1, 4, 5], catalog quality, especially in large catalogs, remains suboptimal. Often, an already-cleaned large catalog still contains many duplicate product entries from different vendors and many entries with incorrect or missing attribute values.

This paper shows how to use unstructured brand knowledge base (KB) data as a reference and an LLM agent to automatically enhance an e-commerce retailer's catalog quality. Prior data cleaning methods usually repair and match product entries separately [3]. As an attribute can have infinitely many possible values, these methods do fuzzy matching to detect duplicate entries by invoking a machine learning model or an LLM for each pair of entries [1, 5]. This is slow and less accurate. To improve speed and accuracy, our method repairs and matches entries concurrently. In each group of repaired entries, each key attribute has only a few possible values, enabling fast and accurate exact entry matching. While enhancing catalog quality is our test case, the core principle is general and applies to using reference data to both repair and match entries. This approach has been explored with structured reference data with an explicit fixed schema [3], but not with un/semi-structured reference data with implicitly encoded and varying (a) schemas of attributes and (b) sets of normalized values for each attribute in each schema—the focus of this work.

Brand KB data include detailed product records with text and images. For each relevant product type, a brand has its own (a) schema of key product attributes and (b) normalized values for each attribute, both implicitly encoded in its KB. These attributes and values may differ from those in the retailer's catalog, reflect key features of the brand's products, and are often used to categorize them. Each product has a unique set of key attribute values and a record in the brand's KB.

Our idea is to first extract key attributes and their normalized values from the brand's KB. Then we extract key attribute values from each product record in the brand's KB and from each product entry in the retailer's catalog. For an attribute like weight, infinitely many possible values (e.g., 20 oz, 20.05 oz, and 20.1 oz) can appear in product entries, but its extracted values are limited to a few normalized values (e.g., 20 oz and 30 oz). We use the extracted attribute values to do 3 things:

1) We fix incorrect and missing attribute values in product entries. Brands and vendors often use different terms for the same attribute value, e.g., "neon berry breeze" vs. "purple" for a shoe color. We prefer brand terms and use them when possible to rewrite attribute values in vendor-provided product entries.
2) We match key attribute values to detect duplicate entries.
3) We can easily change the key attributes used as varying attributes in a variation family, as partitioning products by key attribute values is fast. A variation family is a group of similar products differing only in and partitioned by the values of the varying attributes (e.g., size and color) that define the variation [2]. In contrast, current methods for forming variation families build a separate machine learning model for each set of varying attributes, making changes to these attributes difficult and costly.

2 Methods

This section outlines our method of using brand KB data as a gold standard for the brand's product entries and an LLM agent to enhance an e-commerce retailer's catalog quality. We focus on the case where all products of a brand share the same product type. The case where a brand has ≥2 product types can be handled similarly, by extracting a separate schema of the brand for each product type and using the agent to classify product records in the brand's KB into the appropriate types.

For each brand, we proceed as follows.

Model the brand's schema: We give a sample of product records from the brand's KB to the agent, asking it to model the brand's schema listing the name, description, and type of each key attribute in JSON (JavaScript Object Notation) format. Attributes can appear in the text or images in the records.

List the brand's normalized values for each key attribute: For each product record in the brand's KB, we give the record and the brand's schema to the agent, asking it to extract the value for each key attribute in the schema. Values can appear in the text or images in the record. For each attribute, we collect extracted values across records and compute these values' frequencies. We then give the (value, frequency) pairs to the agent, asking it to standardize equivalent values (e.g., *Gym* and *Gymnasium*) and drop irrelevant values (e.g., *fish* flavor in dietary supplements). This yields a set of normalized values for that attribute for the brand.

Extract key attribute values from product records and product entries: For each product record in the brand's KB, we give the agent the record, the brand's schema, and the set of normalized values of the brand for each attribute in the schema, asking the agent to extract each attribute's value from the record. We do the same for each product entry of the brand in the retailer's catalog.

Repair data: For each product type, the retailer's catalog has a schema listing the name, description, and type of each relevant attribute. We give the brand's schema and the paired catalog schema to the agent, asking it to match each key attribute with an attribute from the catalog schema. Then, for each product entry of the brand in the retailer's catalog, we use the key attribute values to replace the values of the matching attributes. This helps fix incorrect and missing attribute values.

Detect duplicate product entries: For each product record in the brand's KB, we find the product entries of the brand in the retailer's catalog with matching key attribute values. If ≥2 entries match, they are considered duplicates.

Form variation family: To form a variation family defined by some key attributes, we partition the brand's product entries in the retailer's catalog by their key attribute values.

3 Evaluations

We implemented our method in an automated system to enhance an e-commerce retailer's catalog quality. We used the Claude Sonnet 4 LLM and evaluated our method on a subset of products in an e-commerce retailer's catalog. Human experts judged the correctness of key attribute values.

We used two performance metrics for key attribute values. Completeness is the percentage of attributes with non-missing values. Accuracy is the percentage of attributes with correct values. Compared to the raw catalog data, our method raised the key attribute value completeness by 13.5% and the key attribute value accuracy by 39.4%.

4 Company Portrait

Amazon is a leading global e-commerce and technology company known for its innovations in online retail, artificial intelligence, and cloud computing.

5 Presenter Bio

Hayreddin Ceker, PhD, is a Senior Applied Scientist at Amazon working on machine learning and natural language processing.

References

[1] Osman Semih Albayrak, Tevfik Aytekin, Tolga Ahmet Kalayci. 2022. Duplicate product record detection engine for e-commerce platforms. *Expert Syst. Appl.* 193 (May 2022), 116420.

[2] Amazon. 2025. Variation relationships. https://sellercentral.amazon.com/help/hub/reference/external/8831.

[3] Wenfei Fan, Shuai Ma, Nan Tang, Wenyuan Yu. 2014. Interaction between record matching and data repairing. *ACM J. Data Inf. Qual.* 4, 4 (May 2014), 16:1-16:38.

[4] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning*. ACM Books. New York, NY.

[5] Navapat Nananukul, Khanin Sisaengsuwanchai, Mayank Kejriwal. 2024. Cost-efficient prompt engineering for unsupervised entity resolution in the product matching domain. *Discov. Artif. Intell.* 4, 1 (2024), 56.