LLM-Generated Merchant Metadata for E-Commerce Personalization

Brenda Ng
brenda.ng@chase.com
JPMorganChase
Palo Alto, California, USA

Smruthi Mukund
smruthi.mukund@chase.com
JPMorganChase
Palo Alto, California, USA

Veera Tadikonda
veera.m.tadikonda@chase.com
JPMorganChase
Jersey City, New Jersey, USA

Jay Katukuri
jay.katukuri@chase.com
JPMorganChase
Palo Alto, California, USA

Abstract

While most people associate e-commerce primarily with online retailers and marketplace platforms, financial institutions also play a crucial – yet generally overlooked – role in the e-commerce ecosystem. Behind the scenes, banks and payment processors enable the majority of online transactions, as most customers rely on credit cards to pay for their transactions on e-commerce platforms. Depending on the customer's spending needs, banks can tailor their services, such as different credit cards and/or merchant promotions that align with the customer's historical spending. This work addresses the challenge of using credit card transactions to curate relevant features for e-commerce personalization from the perspective of a financial institution. Unlike in e-commerce retailers (e.g., Amazon) where a transaction itemizes the products or services transacted by the customer, no such granular information is generally given in credit card transactions. Instead, using the merchant name and category alone, we use large language models (LLMs) to generate the merchant metadata, apply open-source information to vet the generated content, and compile the vetted content into a knowledge graph linking merchants to their products, services, industries, competitors and target customers.

CCS Concepts

• **Computing methodologies** → **Natural language generation**; • **Information systems** → **Information extraction**; **Business intelligence**; • **Applied computing** → Online banking.

Keywords

LLM generation, fact-checking, merchant metadata, knowledge graphs

ACM Reference Format:
Brenda Ng, Smruthi Mukund, Veera Tadikonda, and Jay Katukuri. 2025. LLM-Generated Merchant Metadata for E-Commerce Personalization. In *Proceedings of LLM4ECommerce Workshop on the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (LLM4ECommerce Workshop at KDD '25).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
LLM4ECommerce Workshop at KDD '25, Toronto, ON, Canada
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Financial institutions play a crucial role in the e-commerce ecosystem by supplying the credit cards and digital payment options that enable customers to complete online transactions. Unlike e-commerce retailers, that have direct access to detailed information about the specific products and services purchased, financial institutions are privy to limited transaction data, such as time, location, and merchant involved. This results in a *partial observability* problem, where customer intent must be inferred indirectly at the merchant level, often based on the frequency and pattern of transactions with a particular merchant or across groups of related merchants.

Why do (card-issuing) financial institutions *want* to infer customer intent? Primarily, it allows them to better match their offerings. On a macro scale, customers are offered a variety of credit card options, some of which are particularly economical for everyday purchases such as groceries, dining, travel, and more. On a micro scale, many merchants market themselves by providing time-sensitive offers through credit card companies, as an incentive for customers to shop with them upon activating these offers from their credit cards. Understanding customer intent is key for financial institutions to deliver personalized experiences and relevant product recommendations within their digital services.

Despite financial institutions' access to rich customer financial data, information related to transactions with merchants is often limited, typically restricted to merchant categories or names, which themselves require entity reconciliation and cleanup. For example, a merchant's name appearing on a transaction (i.e., a line item on a credit card statement) may include other information such as store number, location or digital payment tag, which needs to be removed before it can be matched to the merchant's *dba* or doing-as-business name used by the merchant for its branding and marketing.

At its core, this is a matching problem between credit card customers (users) and merchants. The goal is to identify similar or related merchants to recommend to users based on active merchant offers and user transaction history. Ideally, *merchant metadata* should include business details such as the merchant's products and services, their competitors, and their consumer-centric appeal. Historically, financial institutions have relied on business intelligence

providers (e.g., FactSet, PitchBook, etc.) that collect firmographics (e.g., industry, revenue, market share, location, years in business, etc.) to source merchant information. But firmographics are not ideal for establishing merchant similarity from a consumer-centric perspective. For example, if a small business can offer a cheaper price or a more lenient return policy, a budget-conscious user may choose the small business over a large, established business as their preferred merchant for a given transaction. Hence, user-to-merchant recommendation needs to be founded on consumer-centric merchant metadata instead of firmographics alone.

To source consumer-centric merchant metadata, we leverage LLMs for metadata generation through prompting. This approach suffices for popular merchants, as most LLMs (e.g., GPT, Claude, Gemini, etc.) are knowledgeable about popular merchants because these merchants would have been mentioned in open-source data (e.g., Wikipedia, Common Crawl, etc.) on which LLMs were trained. For lesser-known merchants, LLMs are prone to hallucinations, in which unsubstantiated metadata were generated based solely on cues from the merchant name and category. To validate the LLM-generated metadata, we implement a rigorous fact-checking mechanism as part of our metadata process. This process not only involves using external resources (e.g., Wikipedia, SerpAPI, etc.) but also human-in-the-loop to produce gold-standard data that can be fed back to finetune the system performance.

Our contribution is an end-to-end, enterprise-scale merchant metadata pipeline that ingests merchants sourced from users' transactions and offer merchants from the credit card ecosystem; performs name reconciliation and deduplication; and generates consumer-centric merchant metadata which are fact-checked and assigned a per-attribute confidence value. The metadata are organized into a knowledge graph, that can be used as features by downstream use cases (e.g., user-to-merchant recommendation or spending-aware chatbot) to develop personalization models.

2 Related Work

In recent years, the intersection of large language models (LLMs) and knowledge graphs (KGs) has garnered significant attention, particularly in the context of enhancing search systems and metadata enrichment. Knowledge graphs offer a structured way to represent and understand the context of search queries, enabling more accurate and relevant search results. Despite their potential, there has been limited research focused on leveraging LLMs to construct knowledge graphs, especially at the enterprise level.

LLMs for KG construction: Carta et al. [2] showed that iterative zero-shot prompting of LLMs can be used to construct reasonably accurate KGs. Within the enterprise setting, motivated by the need to efficiently query document data lakes, Sun et al. [9] proposed Doc2KG, a human–LLM collaborative framework designed to construct high-quality, unified knowledge graphs from heterogeneous enterprise documents. Hao et al. [3] recognized that vast amount of latent knowledge is encoded within pretrained LMs (e.g., BERT and its variants), and proposed the BertNet framework which extracts a relational KG from a pretrained LM by predicting entity pairs that satisfy a given relation, using only an initial prompt with few-shot examples of entity pairs. Hu et al. [4] started the GPTKB project, which uses GPT-4o-mini to construct a large-scale knowledge base (as of this writing, contains 100M triples for 3M entities), thus demonstrating the feasibility of extracting structured knowledge from LLMs at scale. By leveraging LLMs for the construction and maintenance of knowledge graphs, search systems can exploit the wide coverage and timely relevance of text sources while still benefiting from the structured representation of graphs.

LLMs for Metadata Enrichment: Concurrently, industries have started to embrace LLMs as a cost-effective way to enrich metadata within existing knowledge bases. Both IBM [5] and Fidelity [8] have adopted LLMs in automating the generation of metadata in data catalogs, with a focus on table-level and column-level descriptions. Reusens et al. [7] apply similar techniques to enrich metadata in museum collections to improve the accessibility of archival materials. Liao et al. [6] show that LLMs can be used to automate website information retrieval via extraction of structured product metadata (product names, categories, descriptions, and specifications) directly from company websites.

LLMs for E-Commerce Recommendation: Our work is closely related to LLM-PKG [10], which focuses on generating a product knowledge graph using LLMs for item- and user-based recommendations on the eBay shopping platform. In contrast to LLM-PKG's build-then-prune strategy for constructing the knowledge graph (i.e., prompting for product recommendations then mapping to the merchant's inventory), our method builds the graph from rigorously vetted public information about the merchants right from the start. This ensures the development of high-quality knowledge graph embeddings, enabling the identification of similar merchants with higher trustworthiness.

3 System Design and Architecture

Our metadata pipeline (cf. Figure 1) consists of a name reconciliation module (2), a metadata generation module (4b), a fact-checker module (5–9) and a knowledge graph module (10b and 11).

3.1 Data

The input to the pipeline is a batch of tuples containing merchant names and categories, i.e., $x = (name,\ cat)$. As alluded to in Section 1, we have two populations of merchants: the offer merchants and the transaction merchants. For each population, the names and categories of merchants are extracted from the appropriate proprietary databases, and are prepped for ingestion by the pipeline (1). While offer merchants come from a curated offer catalog sourced by third-party partners and therefore require minimal text processing, transaction merchants generally have spurious characters in their names (introduced by payment processors as transaction metadata). Generally, each transaction maps to a single merchant and category. But if a merchant has multiple lines of business, different locations or different payment platforms (e.g., Square, Stripe, etc.), then a transaction may be attributed a different merchant category despite originating from the same merchant. In this case, a merchant may be associated with multiple categories: $\left(name, \{cat_k\}_{k=1}^{K}\right)$. To ensure coverage across the categories, we recommend expanding the categories into distinct tuples: $(name, cat_1)$, $(name, cat_2)$, ..., $(name, cat_K)$, so metadata can be generated for the merchant along its different lines of business.

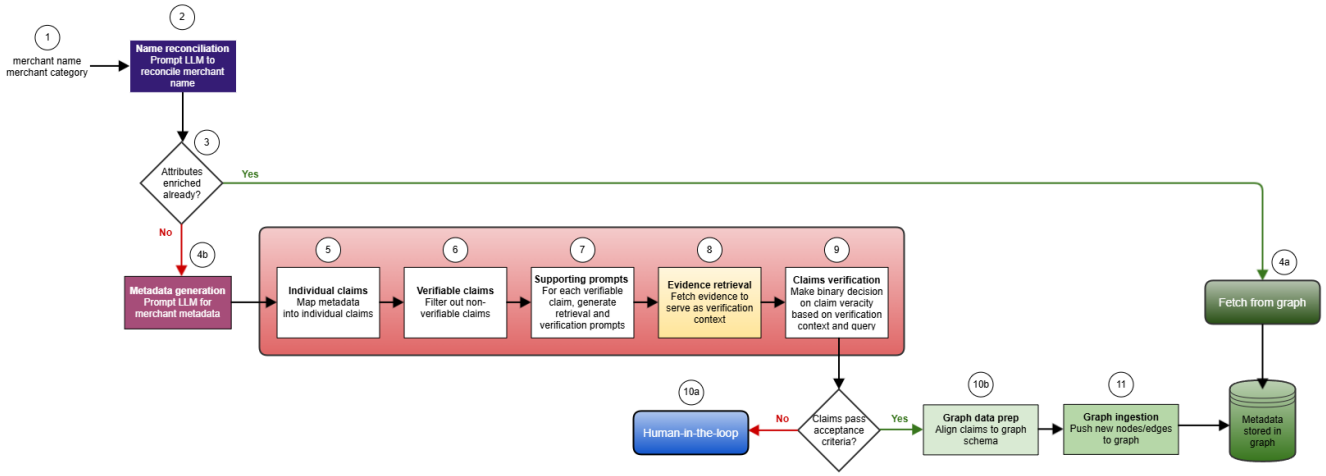

Figure 1: Merchant metadata pipeline

3.2 Name Reconciliation

To standardize the merchant names, we use LLM to map each merchant to its doing-business-as name, referred to its *dba* (2). While it is possible to look up the actual dba using tables provided by business intelligence vendors, we want the name that the LLM knows the merchant by. This step serves as a preprocessing step to filter out the merchants that are unfamiliar to the LLM and thus mitigate the risk of LLM hallucination. For each LLM call, we also elicit the confidence for the dba. In essence:

$$(name, cat) \xmapsto{\text{LLM}} (dba, conf)$$

Based on the $conf$ value, we bin the reconciled merchants accordingly to inform priority for human-in-the-loop checking.

In our current implementation, reconciled merchants with $conf \in \{high, medium\}$ are routed for metadata generation. Using the reconciled merchant's LLM-generated dba, we query the knowledge graph to check if the merchant has already been processed and its metadata saved to the graph (3). If so, the metadata is fetched and returned (4a). Otherwise, it is routed for metadata generation (4b).

3.3 Metadata Generation

In this step (4b), we prompt the LLM to generate metadata for a set of predefined attributes: $a \in \mathcal{A}$. For each of the reconciled merchants, a LLM call is invoked to generate:

$$(dba, cat) \xmapsto{\text{LLM}} \{a : (x, conf_x)\}_{a \in \mathcal{A}}$$

where a denotes the attribute, x the LLM-generated metadata for that attribute, and $conf_x$ is the LLM-generated confidence about x.

Per our business requirements, we elicit the following attributes:

- Products and their associated industries
- Services and their associated industries
- Competitors
- Target customers
- Text description of the merchant

These attributes were specifically chosen in accordance to the schema of our knowledge graph (cf. Subsection 3.5). Condensed versions of our prompts are shown in Figure 2. The prompt template for this generation task underwent the most iterations in our attempt to balance generation quality with relevance, accuracy, token count, and structured format adherence (e.g., JSON validity).

3.4 Metadata Fact-checker

Using the confidence values, the LLM-generated metadata are filtered to keep only those with high confidence:

$$\{(a : x^*)\} \underset{conf_x = high}{\subseteq} \{a : (x, conf_x)\}$$

The filtered metadata $\mathcal{X}^* = \{x^*\}$ is subsequently fact-checked against open-source data. Each attribute x^* in structured format is mapped to a claim c (5), which constitutes a simple sentence:

$$c : \underset{dba}{\text{<subject>}} + \underset{verb(a)}{\text{<verb>}} + \underset{x}{\text{<object>}}$$

where the subject is generally the merchant's reconciled name dba; the verb is selected to be semantically equivalent to the attribute a; and the direct object is the attribute's metadata x.

Each claim c is analyzed (via LLM) for subjectivity, in order to determine whether it is factually verifiable. The claims that are not verifiable are discarded (6):

$$\{c^*\} \underset{verifiable}{\subseteq} \{c\}$$

Each verifiable claim c^* is used to generate an evidence query q_e and a verification query q_v (7). The evidence query is tailored based on whether Retrieval-Augmented Generation (RAG) systems or real-time APIs are used for fact-checking.

To retrieve evidence (8), one can utilize real-time APIs (e.g., SerpAPI) or offline data dumps. We recommend using real-time APIs if external API calls can be made from the production environment, as this minimizes the overhead associated with managing offline dumps. If real-time API access is not feasible, business-relevant data dumps can be curated from open sources like Wikipedia and Wikidata. These offline data dumps can be integrated into a RAG system. In the evidence query q_e, "documents" can refer to either

the responses from real-time APIs or the processed chunks from curated data dumps. The verification query q_v is instantiated with the retrieved evidence e, and used to prompt a LLM to verify the claim as a structured response $(verdict, score)$ (9). Using the confidence $score$, the claim c will be accepted, rejected or be manually re-examined by human annotators (10a).

In summary, a high-confidence attribute x^* is transformed into a simple-sentence claim c. A factually-verifiable claim c^* is fact-checked using supporting queries (q_e, q_v) to fetch evidence e for determining a verification verdict $(verdict, score)$:

$$\begin{array}{lllll} x^* & \mapsto & c & \rightsquigarrow & c^* \quad \longmapsto \quad (q_e, q_v) \\ & & & & q_e \quad \xmapsto{\text{Retriever}} \quad e \\ & & & & q_v(e) \quad \xmapsto{\text{LLM}} \quad (verdict, score) \end{array}$$

3.5 Knowledge Graph

The claims determined to be "Supported" (with sufficiently high confidence) are propagated as verified claims $\{c^{\checkmark}\}$:

$$C^{\checkmark} = \left\{c^{\checkmark}\right\} \underset{verdict^+}{\subseteq} \left\{c^*\right\}$$

The verified claims are mapped back to the structured metadata ($\{a : x\}$) to generate the tuples for graph database ingestion:

$$\underbrace{(dba)}_{\text{CompanyNode}} \; - \; \underbrace{[a]}_{\text{Relationship}} \; - \; \underbrace{(x)}_{\text{AttributeNode}}$$

These tuples are aggregated into dataframes for incremental loading into the graph. For now, we use an embedded graph database, like Kuzu[1], to reduce latency and simplify our architecture without the overhead of maintaining a database server.

The conversion from verified metadata to graph-ready tuples can be streamlined if one starts with highly structured metadata covering similar concepts. In our case, we started with the graph schema, and used the graph relationships to come up with the merchant attributes to include in our metadata generation prompt. We iterated between generation and schema refinement, incorporating feedback from our human annotators that some metadata would be difficult to verify and thus, we would be wasting resources in generating subjective metadata that would be discarded. As a result, our knowledge graph schema was also refined in the process based on the quality of data that we were able to source through the LLM and fact-checked using our available data sources.

If one does not have a predefined graph schema, the LangChain `LLMGraphTransformer`[2] library can be used to extract entities and relationships from text. We found this to be useful in brainstorming an initial graph schema but impractical for mass conversion of unstructured text to graph, as it is prompted to extract as many entities and relationships as possible, resulting in spurious entities and relationships that are irrelevant.

4 Implementation

The financial industry is one of the most heavily regulated sectors globally, subject to a myriad of compliance procedures designed to ensure the stability and integrity of the financial system, protect consumers from fraud and malpractice, and prevent systemic risks that could lead to economic crises. It follows that financial institutions deploying systems on the cloud must adhere to stringent regulatory compliance requirements, such as ensuring data encryption and implementing robust access controls to protect sensitive financial information.

Our system is designed with these considerations in mind (cf. Figure 3). Our system is deployed on AWS, where we have separate zones to ensure isolation of our users' transactional data from any external connections. External to our system, user transaction histories are pulled into a s3 bucket at regular cadence. The merchant names and categories from these transactions are saved to a separate Transactions s3, for ingestion by our system.

A scheduler (1) would trigger our system to ingest the data from this s3. Within our internal VPC, the prompt generator (2) would read from the Merchants s3 and instantiate the prompts using predefined prompt templates that have been cleared by our governance body. A prompt checker (3) applies proprietary, context-sensitive, content filtering to ensure that the instantiated prompts contain no sensitive data or unintended information that could be shared with the external API providers. The proprietary filtering is achieved through libraries that are managed by our risk team and are rigorously vetted per compliance regulations.

Isolation between the internal data store and external calls is accomplished via a VPC that communicates with the internal VPC through a publish-subscribe model implemented using Kafka[3]. Within the external VPC, a request scheduler (5) consumes data from Kafka and construct the appropriate payloads for interfacing with external APIs (e.g., Azure[4], Bedrock[5], Wikipedia/Wikidata[6], and SerpAPI[7]). Under the hood, it handles API authentication (refreshing API keys as needed) and creates authorization headers, as well as tracks API usage to schedule jobs within service limits (e.g., requests per minute).

The API calls are routed through a proxy, ensuring secure and controlled access, while the responses are similarly retrieved via the proxy to maintain consistency and security in data handling. A response checker (6) checks the responses for errors, routing back unsuccessful requests to the request scheduler as needed. The successful responses are processed by the content checker (7) which performs another pass of content filtering to ensure safety and compliance in the responses. The responses from both LLMs and fact-checking data APIs are combined to generate and fact-check the merchant metadata (as described in Section 3) within the metadata curator (8). Processed merchants are now *enriched* by their corresponding metadata.

Finally, the curated metadata would be queued for "pushing" back into the internal VPC by Kafka. Within the internal VPC, the metadata would be picked up by a metadata consumer (9) which would register the metadata in an internal data catalog before saving to the Curated Metadata s3. It would prepare the metadata into graph-ready tuples (cf. Section 3.5) for ingestion into the Kuzu graph

[1] https://kuzudb.com
[2] https://python.langchain.com/docs/how_to/graph_constructing
[3] https://kafka.apache.org
[4] https://azure.microsoft.com/en-us/products/api-management
[5] https://aws.amazon.com/bedrock
[6] https://www.wikidata.org
[7] https://serpapi.com

LLM Prompt: Name Reconciliation Template

```
Identify the company for the transaction labeled "[insert name]" in the "[insert category]" merchant category. Provide
the information in JSON format:
{
"company": "Common name, include establishment type if eatery, or leave blank if unknown",
"confidence": "high|medium|low|idk"
}
```

LLM Prompt: Metadata Generation Template

```
Provide factual information on "[insert dba]" in the "[insert category]" category in JSON format.
{
        "Company": "Company",
        "Products": {
                "list": "<Product>: <Industry> or blank",
                "confidence": "high|medium|low|idk"
        },
        "Services": {
                "list": "<Service>: <Industry> or blank",
                "confidence": "high|medium|low|idk"
        },
        "Competitors": {
                "list": "Company names or blank",
                "confidence": "high|medium|low|idk"
        },
        "Target Customers": {
                "list": "Descriptive list or blank",
                "confidence": "high|medium|low|idk"
        },
        "Description": {
                "text": "Brief summary or blank",
                "confidence": "high|medium|low|idk"
        }
}
```

Retriever Query: Evidence Retrieval Template

```
Retrieve evidence from the documents to verify: [Insert claim]
```

LLM Prompt: Verification Template

```
Verify the claim and return a JSON response:
**Claim:** [Insert claim]
**Evidence:** [Insert retrieved evidence]
**Output Format:**
{
“verdict”: “supported|unsupported”,
“confidence”: “A confidence score (0 – 100%)”,
“supporting_evidence”: “Phrases from the evidence that substantiates the claim”,
“contradictory_evidence”: “Phrases from the evidence that refutes the claim”
}
```

Figure 2: Simplified templates for name reconciliation, metadata generation and fact-checking

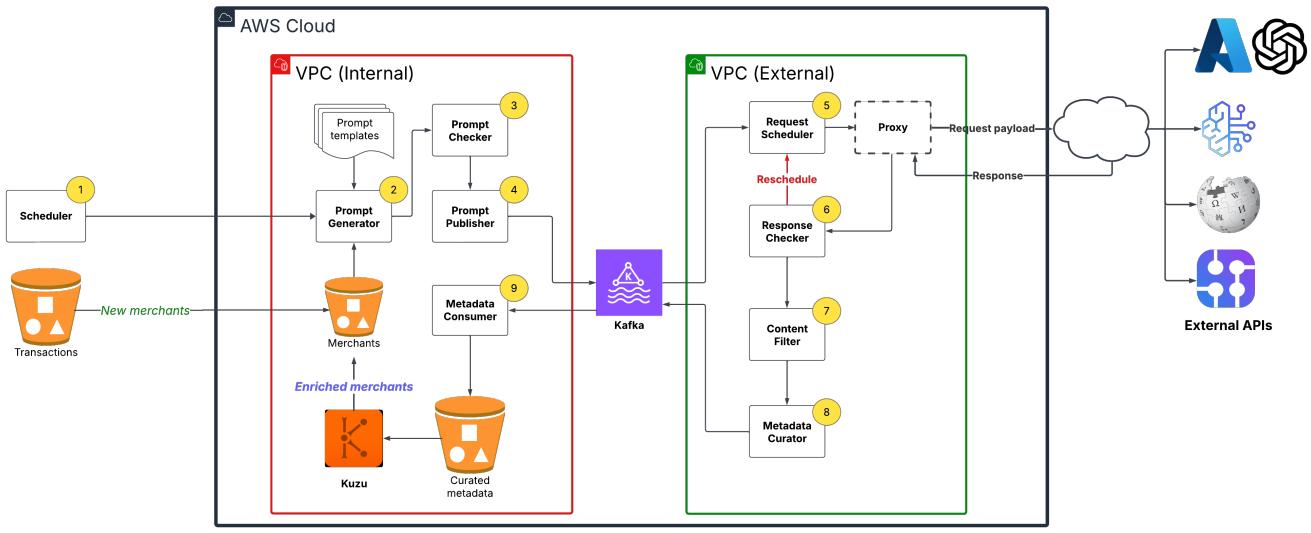

Figure 3: Metadata pipeline implementation

database. The ingestion process will add nodes and edges (corresponding to the enriched merchants) to the knowledge graph. Once ingested, the enriched merchants would be temporarily removed from the Merchants s3 until they are due for metadata refresh.

5 Evaluation

To evaluate the curated metadata, we apply them in a downstream task of merchant recommendation. To recap, we have two populations of merchants: the offer merchants and the transaction merchants. The task is to determine which offer (associated with an offer merchant) to recommend to a user. Given an offer merchant, we retrieve similar merchants from the population of transaction merchants. If these retrieved merchants exist within a user's recent history, then the offer merchant may be recommended to the user. (This approach protects user information as user information remains inside our internal VPC. LLMs are only being used to generate the merchant metadata and their text embeddings. No user information is passed to external LLMs.)

Within a typical 90-day period, we can have tens of thousands of offer merchants and millions of transaction merchants. For this offline evaluation study, we sampled 20K merchants: 10K from each population, extracted during the same timeframe of February to May 2025. As the merchants proceed through the pipeline, they are dedupped using the LLM-generated dbas' (cf. Section 3.2) and pruned if metadata generation is insufficient (cf. Section 3.3). By the time the metadata undergoes fact-checking (cf. Section 3.4), the offer merchant population has been reduced by 6% and the transaction merchant population by 22%. During fact-checking, only metadata that are vetted by supporting evidence are kept, thus further reducing the merchant population to those that have public records or online presence (~25% reduction). Finally, the evaluation knowledge graph is constructed from the remaining merchants, and ends up having ~30K entities and 7 relationships, as follows:

- $(Company) - [Offers] - (Product)$
- $(Company) - [Offers] - (Service)$
- $(Product) - [BelongsTo] - (Industry)$
- $(Service) - [BelongsTo] - (Industry)$
- $(Company) - [CompetesWith] - (Company)$
- $(Company) - [Targets] - (Customer)$

5.1 Embeddings

The metadata are used to generate text and knowledge graph embeddings. In this study, we derive three sets of metadata-derived embeddings, and compare against the baseline embedding:

- **base**: Using the merchant name and category, a sentence is formed and embedded using `text-embedding-ada-002`[8] with embedding dimension of 1536.
- **meta-ada**: Using the LLM-generated metadata, the text description is embedded using `text-embedding-ada-002` with embedding dimension of 1536.
- **meta-TransE**: Using the evaluation knowledge graph (built from metadata), `PyKEEN`[9] was used to train a `TransE`[10] model with embedding dimension of 150.
- **meta-ComplEx**: Using the evaluation knowledge graph, `PyKEEN` was used to train a `ComplEx`[11] model with embedding dimension of 200.

Both `TransE` and `ComplEx` are popular embedding models used in knowledge graph representation learning, with `TransE` being computationally efficient and `ComplEx` being more suitable for modeling complex and asymmetric relationships between entities. For both models, the hyperparameters have been tuned using `PyKEEN`'s

[8] https://platform.openai.com/docs/models/text-embedding-ada-002
[9] https://github.com/pykeen/pykeen
[10] https://pykeen.readthedocs.io/en/stable/api/pykeen.models.TransE.html
[11] https://pykeen.readthedocs.io/en/stable/api/pykeen.models.ComplEx.html

LLM Prompt: LLM-as-a-judge for recommendation grading

```
You are a shopping assistant and your job is to recommend stores that I would be interested in. Recently, I shopped at
[Insert offer merchant's dba]. [Insert offer merchant's description]
For each store in this list, determine its relevance.
===== START list =====
* [Insert retrieved transaction merchant1's dba and identifier]: [Insert retrieved transaction merchant1's description]
* [Insert retrieved transaction merchant2's dba and identifier]: [Insert retrieved transaction merchant2's description]
...
===== END list =====
Return your response as JSON:
{
[Insert retrieved transaction merchant1's identifier]: "high|medium|low|idk",
[Insert retrieved transaction merchant2's identifier]: "high|medium|low|idk",
...
}
```

Figure 4: Simplified template for LLM-as-a-judge to evaluate merchant recommendations

	Taste of Belgium Restaurant *(Lowest Precision)*	Georgetown Liquor Company *(Low Precision)*
base	Versailles Restaurant, Friendly's Restaurant, Burgerville Restaurant, Dig Restaurant, Taco Bell Restaurant, The Melt Restaurant, **Eataly Restaurant**, Yours Truly Restaurants, Nando's Restaurant	**Admiral Beverage**, Liquid I.V., **The Coca-Cola Company**, **Coca-Cola**, Foods Co, **Village Tavern**, Trader Joe's, Foodland, Food City
meta ada	**La Madeleine French Bakery & Café**, **La Madeleine - French Bakery & Cafe**, First Watch Restaurant, Bagel Street Cafe, Common Bond Bistro, Au Bon Pain Bakery & Café, Sant Ambroeus Restaurant, 85°C Bakery Cafe, Waffle House Restaurant, Biscuits Cafe	**Parakeet Cafe**, **Boise Co-op**, **Common Bond Bistro**, **Aladdin's Eatery Restaurant**, **Juiceland Juice Bar**, **Boise Co-op Village**, Burgerville Restaurant, **Velvet Taco Restaurant**, **Tavern In the Square Restaurant**, Gourmet Garage
meta Trans	Maple Street Biscuit Co Restaurant, Original Pancake House, Hash House A Go Go Restaurant, Biscuits Cafe, Keke's Breakfast Cafe, Yours Truly Restaurants, Breakfast Republic Restaurant, Black Bear Diner Restaurant, Waffle House, Black Bear Diner	**Brothers Bar & Grill**, **Hampton Social Restaurant**, **Rock Bottom Restaurant & Brewery**, **Olive & Ivy Restaurant**, **Lazy Dog Restaurant & Bar**, **Twin Peaks Sports Bar & Restaurant**, **Tommy's Tavern & Tap Restaurant**, **Yard House Restaurant**, The Capital Grille Steakhouse
meta Com-plEx	Maple Street Biscuit Company, Cupitol Coffee & Eatery, Hash House A Go Go Restaurant, Common Bond Bistro, Costa Coffee - Coffee Shop, Daily Provisions Bakery & Cafe, 7 Leaves Cafe, Frothy Monkey Coffeehouse & Eatery, Broken Yolk Cafe - Eatery, Breakfast Republic Restaurant	Duke's Waikiki Restaurant, Seasons 52 Restaurant, **Old Ebbitt Grill Restaurant**, **ML Rose Craft Beer**, Versailles Restaurant, **Black Tap** , **Chuy's Tex-Mex Restaurant**, Bosa Donuts - Donut Shop, **Tommy's Tavern & Tap Restaurant**
	Solidcore *(Moderate Precision)*	Rocket Express Car Wash *(Highest Precision)*
base	Shell, Nordstrom, Dermstore, **Nordic Naturals**, **Designs for Health**, **Thorne Research**, Key Food, SalonCentric, Staples	**Car Wash USA Express**, **Go Car Wash**, **Mister Car Wash**, **Crew Car Wash**, **Mike's Car Wash**, **Zip's Car Wash**, **Mike's Carwash**, **Autobell Car Wash**, **WOW Carwash**, **Brown Bear Car Wash**
meta ada	**Pure Barre**, **Club Pilates**, **SoulCycle**, **Equinox Fitness**, **Powerhouse Gym**, **Lifetime Fitness**, **Barry's Bootcamp**, **Planet Fitness**, **CycleBar**	**Car Wash USA Express**, **Tommy's Express**, **Go Car Wash**, **Mister Car Wash**, **Everwash**, **Crew Car Wash**, **Zip's Car Wash**, **WOW Carwash**, **Autobell Car Wash**, **Take 5 Car Wash**
meta Trans	**Barry's Bootcamp**, **SoulCycle**, **CycleBar**, **Pure Barre**, **Club Pilates**, **Powerhouse Gym**, **Vasa Fitness**, **EOS Fitness**, **24 Hour Fitness**, **LA Fitness**	**Go Car Wash**, **Cobblestone Carwash**, **Car Wash USA Express**, **Mike's Carwash**, **Cobblestone Car Wash**, **Take 5 Car Wash**, **Mister Car Wash**, **Cobblestone Auto Spa**, **Brown Bear Car Wash**, **Crew Car Wash**
meta Com-plEx	**Barry's Bootcamp**, **Equinox Fitness**, **Lifetime Fitness**, **Planet Fitness**, **Retro Fitness**, **Thorne Research**, **24 Hour Fitness**, **Club Pilates**, Spoiled Child Wellness	**Cobblestone Auto Spa**, **Cobblestone Carwash**, **Cobblestone Car Wash**, **Brown Bear Car Wash**, **Delta Sonic**, **Car Wash USA Express**, **Autobell Car Wash**, **Crew Car Wash**, **Take 5 Car Wash**, **Go Car Wash**

Table 1: A comparison of merchant recommendations (from lowest to highest) across different embedding sets. Bold merchants are retrieved transaction merchants that are flagged as highly relevant by LLM-as-a-judge.

Hyperparameters	TransE	ComplEx
Embedding dimension	150	200
Scoring function norm	2	N/A
Loss margin	1.00775	1.07732
Learning rate	0.00129	0.00880
Number of negatives per positive	4	96
Batch size	1024	128
Number of epochs	100	300

Table 2: Hyperparameters for `TransE` and `ComplEx` training

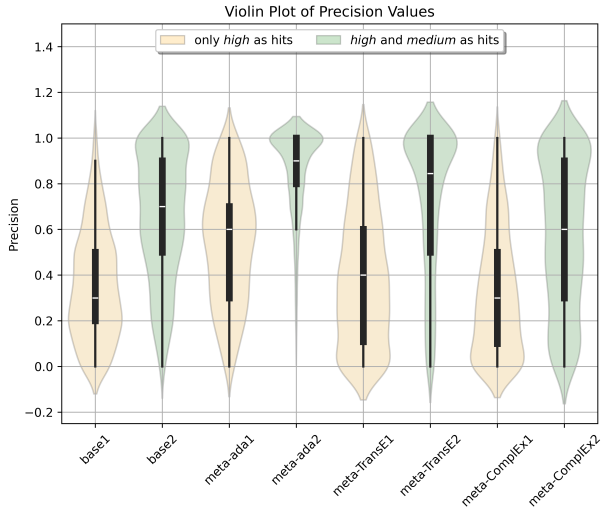

Figure 5: Violin plot of precisions comparing the metadata-derived embeddings against the baseline

hyperparameter optimization pipeline which leverages `Optuna`[12]. The final `TransE` and `ComplEx` models are instantiated with margin ranking loss along with tuned hyperparameters (cf. Table 2), then trained over 100 and 300 epochs, respectively, with early stopping. (Since `ComplEx` learns complex-valued embeddings, it requires more epochs for convergence.)

5.2 Setup

Since the merchant populations have been subsampled, we leverage a combination of `g4dn.12xlarge` and `t3.medium` AWS instances, with the former for GPU-enabled training of knowledge graph embeddings and the latter for interfacing with external APIs. We use a combination of `gpt-4o-2024-11-20`[13] and `o3-mini-2025-01-31`[14] in our metadata pipeline.

Each embedding model (**base**, **meta-ada**, **meta-TransE**, **meta-ComplEx**) generates its own set of embeddings, which is indexed by a separate FAISS vector store. For each set, we apply FAISS to retrieve the top-10 most relevant transaction embeddings for a given (query) offer embedding. For `ComplEx`'s complex-valued embeddings, the real and imaginary parts were extracted and concatenated as real-valued embeddings (effectively doubling the embedding dimension) before they are indexed by FAISS.

To determine the relevance of the retrieved transaction merchants, we use `anthropic-claude-v2-1`[15] as LLM-as-a-judge because it has been found to align well with human judgment [1]. The retrieved merchants from all four embedding sets are dedupped and combined into one LLM-as-a-judge call to ensure consistency across relevance grading and reduce costs (e.g., number of tokens and requests). The LLM-as-a-judge prompt is given in Figure 4, and its response is used to compute the retrieval precision.

5.3 Results

Using the LLM-as-a-judge annotations that label each retrieval as one of {*high*, *medium*, *low*, *idk*}, we compute the precision (*P*@10) values and display them as a violin plot in Figure 5. We consider both scenarios:

- Group1 (in orange): only *high* as relevant
- Group2 (in green): both *high* and *medium* as relevant

The mean precisions for Group2, ranked from best to worst, are as follows: 0.859700 for meta-ada2, 0.724367 for meta-TransE2, 0.651589 for base2, and 0.572917 for meta-ComplEx2. Similarly, Group1 precisions are 0.540667 for meta-ada1, 0.391147 for meta-TransE1, 0.354756 for base1, and 0.322814 for meta-ComplEx1. In both groups, the ranking remains consistent: **meta-ada** is the best, followed by **meta-TransE**, **base**, and **meta-ComplEx** as the worst. **meta-ComplEx** may have been a poor fit as the metadata might have lacked non-translational relationships. Notably, both **meta-ada** and **meta-TransE** outperform the baseline, highlighting the potential of LLM-generated metadata to enhance merchant recommendations. The strength of **meta-ada** lies in that the embeddings are based on the company descriptions that emphasize the products and services – likely the most crucial features for this recommendation task. In contrast, **meta-TransE** encapsulates a broad spectrum of relationships, including competitors and target customers, while achieving storage efficiency at one-tenth the size of **meta-ada**.

6 Conclusion

Understanding customer intent is crucial for personalizing experiences across e-commerce applications. Given that we have access only to the merchants that our users transacted at, we focus on gaining insights about merchants by curating merchant metadata. To achieve this, we have developed an enterprise-scale, LLM-enabled merchant metadata pipeline that generates high-quality metadata organized into a knowledge graph. Our evaluation indicates that both text and TransE embeddings derived from this merchant metadata can enhance merchant recommendations. In this work, we have leveraged LLMs for content generation, fact-checking and LLM-as-a-judge to verify merchant recommendations. For future, we plan to incorporate more LLM-as-a-judge for internal quality controls, as well as LLM finetuning to improve metadata generation.

References

[1] Rewina Bedemariam, Natalie Perez, Sreyoshi Bhaduri, Satya Kapoor, Alex Gil, Elizabeth Conjar, Ikkei Itoku, David Theil, Aman Chadha, and Naumaan Nayyar.

[12] https://optuna.org
[13] https://platform.openai.com/docs/models/gpt-4o
[14] https://platform.openai.com/docs/models/o3-mini
[15] https://www.anthropic.com/news/claude-2-1

2025. Potential and Perils of Large Language Models as Judges of Unstructured Textual Data. arXiv:2501.08167 [cs.CL] https://arxiv.org/abs/2501.08167

[2] Salvatore Carta, Alessandro Giuliani, Leonardo Piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. 2023. Iterative Zero-Shot LLM Prompting for Knowledge Graph Construction. *CoRR* abs/2307.01128 (2023). doi:10.48550/ARXIV.2307.01128 arXiv:2307.01128

[3] Shibo Hao, Bowen Tan, Kaiwen Tang, Bin Ni, Xiyan Shao, Hengzhe Zhang, Eric P. Xing, and Zhiting Hu. 2023. BertNet: Harvesting Knowledge Graphs with Arbitrary Relations from Pretrained Language Models. arXiv:2206.14268 [cs.CL] https://arxiv.org/abs/2206.14268

[4] Yujia Hu, Shrestha Ghosh, Tuan-Phong Nguyen, and Simon Razniewski. 2024. GPTKB: Building Very Large Knowledge Bases from Language Models. *ArXiv* abs/2411.04920 (2024). https://api.semanticscholar.org/CorpusID:275213655

[5] Corey Keyser. 2024. Enrich Your Data with Metadata Enrichment Powered. https://community.ibm.com/community/user/blogs/corey-keyser/2024/06/19/enrich-your-data-with-metadata-enrichment-powered. Accessed: 2025-05-19.

[6] Chen Liao, Gang Cheng, Shilei Huang, and Lin Yao. 2024. LLM-Based Automating Product Information Retrieval for Industry Analysis: A Real-World Application. In *Cognitive Computing - ICCC 2024: 8th International Conference, Held as Part of the Services Conference Federation, SCF 2024, Bangkok, Thailand, November 16–19, 2024, Proceedings* (Bangkok, Thailand). Springer-Verlag, Berlin, Heidelberg, 116–128. doi:10.1007/978-3-031-77954-1_8

[7] Manon Reusens, Amy Adams, and Bart Baesens. 2025-02-27. Large Language Models to make museum archive collections more accessible.

[8] Mayank Singh, Abhijeet Kumar, Sasidhar Donaparthi, and Gayatri Karambelkar. 2025. Leveraging Retrieval Augmented Generative LLMs For Automated Metadata Description Generation to Enhance Data Catalogs. arXiv:2503.09003 [cs.IR] https://arxiv.org/abs/2503.09003

[9] Qiang Sun, Yuanyi Luo, Wenxiao Zhang, Sirui Li, Jichunyang Li, Kai Niu, Xiangrui Kong, and Wei Liu. 2025. Docs2KG: A Human-LLM Collaborative Approach to Unified Knowledge Graph Construction from Heterogeneous Documents. In *Companion Proceedings of the ACM on Web Conference 2025* (Sydney NSW, Australia) *(WWW '25)*. Association for Computing Machinery, New York, NY, USA, 801–804. doi:10.1145/3701716.3715309

[10] Menghan Wang, Yuchen Guo, Duanfeng Zhang, Jianian Jin, Minnie Li, Dan Schonfeld, and Shawn Zhou. 2024. Enabling Explainable Recommendation in E-commerce with LLM-powered Product Knowledge Graph. arXiv:2412.01837 [cs.IR] https://arxiv.org/abs/2412.01837

Received xx May 2025