Effective Product Schema Matching and Duplicate Detection with Large Language Models

Andrea Iovine
aiiovine@amazon.de
Amazon.com, Inc.
Munich, BY, Germany

Yunhan Huang
yunhanh@amazon.com
Amazon.com, Inc.
Seattle, WA, USA

Melvin Monteiro
montmel@amazon.com
Amazon.com, Inc.
New York, NY, USA

Mohamed Yakout
myakout@amazon.com
Amazon.com, Inc.
Seattle, WA, USA

Sedat Gokalp
sggokalp@amazon.com
Amazon.com, Inc.
Seattle, WA, USA

Abstract

Building and maintaining a rich and high-quality product schema helps customers of an e-commerce service find products based on the characteristics they desire. As the quantity of products sold on the service increases, so does the complexity of maintaining the schema. Expanding it requires finding gaps, designing new product attributes, and ensuring that they do not already exist in the schema. In this paper, we present an automated system for product schema matching, which uses a combination of semantic search and Large Language Models (LLM) in order to align the product concepts from two schemas. The approach was tested on the duplicate attribute detection task using a dataset of $1,399$ product attributes, where it achieved 90.2% F_2, outperforming humans by 8.4% on the same task. On the product schema matching task, it achieved 78.12% F_1, which is close to human-level performance. Moreover, we estimate that the system can reduce the time spent by humans reviewing new attributes by more than 90%.

ACM Reference Format:
Andrea Iovine, Yunhan Huang, Melvin Monteiro, Mohamed Yakout, and Sedat Gokalp. 2025. Effective Product Schema Matching and Duplicate Detection with Large Language Models. In *Proceedings of LLM4ECommerce Workshop at 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

1 Introduction

One of the most important challenges for e-commerce services is managing the information for the hundreds of millions of products that they offer. Products differ significantly from each other, and each one possesses unique characteristics that specific customers are looking for. Accordingly, the success of an e-commerce service is tied to how well it enables customers to find and purchase their desired products by providing complete, detailed, and reliable information [1]. As the service grows larger by offering more products and more product-level information, the difficulty of managing, storing, and reconciling this information also increases exponentially. Additionally, products gain new features over time (*e.g.*, LiDAR scanners on smartphones), which need to be captured as early as possible.

Product information is typically stored in the *catalog schema*, defined as the set of attributes, constraints, and metadata that describe a specific product. The catalog schema ensures that the data exposed by the e-commerce service is consistent across products and easy to maintain. Expanding the catalog schema requires several steps, starting from analyzing signals coming from a multitude of sources, such as manufacturers, sellers, retailers, and other services. The next step involves identifying gaps in the schema, and modeling new product attributes where necessary. A crucial part in the modeling step is ensuring that the new attributes meet a quality bar by detecting, remediating or filtering out defects before they are released. In this regard, attribute duplication is one of the most common defects, in which a new attribute describes the same information as another attribute in the schema. Duplicate and overlapping attributes pose a serious risk to any e-commerce service, resulting in redundant or even inconsistent product information being shown to customers, worsening their buying experience and compromising their trust in the service itself.

Detecting these issues is however a non-trivial task for various reasons: first, it requires gaining a holistic awareness of the entire schema, which spans hundreds of attributes for thousands of different types of products. Second, the same product information can be represented in different forms and using different levels of granularity. For example, the attribute "Image Brightness" for computer monitors may be also represented as "Minimum Brightness" and "Maximum Brightness". Finally, other defects in the attribute metadata can make it difficult to conclude whether two attributes model the same information: for instance, some attributes may not be described adequately, or feature irrelevant values. When a candidate attribute is proposed, one has to compare it against the entire set of existing attributes in the catalog schema, understand the differences, resolve any potential ambiguities, and determine whether it constitutes a net new addition to the schema. For humans, this process is very time-consuming and prone to errors. Machine learning models have been proposed and applied in both literature and practice [8, 14, 17], however they require significant amounts of

LLM4ECommerce Workshop at KDD '25, August 4, 2025, Toronto, ON, Canada
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

training data in order to achieve satisfactory performance. Given the dynamic nature of the catalog schema, both the model and its training data must be continuously updated to accommodate previously unseen entities, further increasing computational and maintenance costs. On the other hand, recent advancements in Natural Language Processing have proven that state-of-the-art Large Language Models (LLM) achieve groundbreaking performance in complex understanding and reasoning tasks with limited or no training data available. However, technical limits such as the context length and computational cost require framing the duplicate detection task in a way that plays into the strengths of LLMs while minimizing their drawbacks.

In this paper, we propose using product schema matching as a guardrail that detects duplicate attributes, and prevents them from being introduced in the catalog schema. Schema matching is responsible for building a mapping between the common elements of two or more product schemas, and is a critical step in data integration pipelines and data enrichment programs [17]. For this purpose, we have designed and implemented a product schema matching system that combines semantic search and pre-trained LLMs to match attributes that describe the same product characteristic or measurement. The system is highly configurable, and can be either used in a fully unsupervised way, or can be tuned with a small set of labeled data to identify the optimal set of parameters. We tested the system on a realistic benchmark using a purpose-built dataset of 1399 attributes, and results show that the system outperforms human annotators on the duplicate attribute detection task (90.2% F_2 vs. 81.82%). It also performs well on the schema matching task, achieving performance that is close to human level (78.12% F_1 vs. 79.75%). Through an online experiment, we discovered that our approach can reduce the time needed by humans for the duplicate attribute detection task by up to 91%.

The paper is structured as follows: Section 2 frames our work within the current literature; Section 3 defines the product schema matching task; Section 4 presents our schema matching system, and Section 5 describes the experimental setup and the results. Finally, Section 6 presents the conclusions and outlines future work.

2 Related Work

Our work draws inspiration from three lines of work: Ontology Alignment (OA), Schema Matching (SM), and Entity Matching (EM).

Ontology Alignment. Ontology Alignment [5] (also known as Ontology Matching) is defined as the identification of relations between common elements of multiple ontologies. In the literature, OA approaches can be divided into four macro-categories: Terminological, Structural, Extensional, and Semantic [13, 19]. Our approach mostly falls within the Terminological macro-category, and specifically, Language-based approach, since it involves reasoning over textual content. Recent studies have investigated the application of LLMs for the OA task. BERTMap [8] uses the BERT language model as a base, and performs pre-training and fine-tuning to adapt it to the OA task. He et al. [9] tested the pre-trained Flan-T5-XXL and GPT-3.5-turbo models on the Ontology Alignment Evaluation Initiative (OAEI) Bio-ML dataset, showing predictive performance that is close to BERTMap, despite being used in a zero-shot setting. Hertling and Paulheim [10] present OLaLa, which uses Sentence-BERT embeddings to represent and retrieve concepts, and a LLM prompt to classify them. The authors also compared two styles of prompting: *multiple-choice* and *single-choice*, concluding that the latter obtains higher F_1. In this paper, we explore the application of LLM-based OA in the product schema domain in a zero-shot scenario. Similarly to [10], we rely on a single-choice prompt design.

Schema Matching. Given two schemas as input, the goal of Schema Matching is to produce a mapping between elements of the two schemas that correspond semantically to each other [16]. Shieh et al. [17] learns a vector representation for source and target attributes, and uses configurable thresholds to classify attribute pairs based on cosine similarity. We build upon this work by leveraging the most recent sentence embedding models such as Cohere, and LLMs from the Claude and Mistral families for retrieval and classification respectively. Liu et al. [12] presents GRAM, a schema matching system based on generative AI solutions. It uses a text classifier based on RoBERTa to partition source and target attributes into logical categories. It also features a retrieval component that selects relevant target attributes, as well as labeled examples from a knowledge base. Finally, Flan-T5 is used to select the matching target attribute, given the output of the retrieval component, using a multiple-choice prompt similar to the one from [10]. In our paper, we apply a similar solution for the specific goal of identifying duplicate attributes before they are integrated into our product catalog schema. While it features similar architectural decisions, our approach does not rely on dividing attributes into orthogonal groups, nor it requires any model training, aside from optional parameter tuning. Recently, Chugh and Zambre [4] presented ASTRA, which frames the SM task as a machine translation task, and relies on fine-tuning mT5 and mBART models to generate the matching target attribute given an input attribute. One drawback of this approach is that the model has to be re-trained each time the target schema is updated. On the other hand, our approach does not rely on fine-tuning, and is better suited to a scenario in which the target schema is constantly evolving.

Entity Matching. This task, also known as Entity Resolution, consists of determining whether two entities refer to the same real-world object or not [6]. EM draws some similarities with OA, however, it focuses specifically on equivalence relationships between named entities. Deep learning approaches are used extensively in the EM literature, which includes DITTO [11], BERT [14], and multi-tasking models [18]. More recently, Peeters et al. [15] showed that LLMs can achieve similar performance as fine-tuned models with limited or no training data in the EM task. Fan et al. [6] introduce the concept of *batch prompting*, in which multiple pairs of entities are sent to the LLM for review in the same prompt, which improves performance while reducing cost. It also investigates the use of in-context learning, by selecting pertinent labeled examples to integrate into the prompt text to assist the LLM. Our approach implements the batch prompting strategy described in the work above: specifically, we batch together all candidate target schema attributes for a given attribute in the source schema.

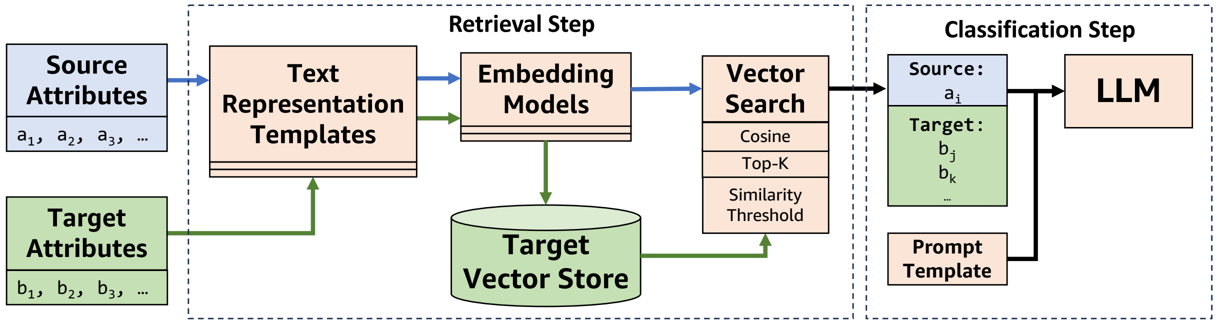

Figure 1: Architecture of the proposed approach. Blue denotes the data related to the source schema, while green represents the target schema.

3 Task Definition

The product schema matching task is defined as follows: Given two product schema ontologies, respectively called A (also referred to as *source schema*) and B (also referred to as *target schema*), their respective sets of product attributes $A = \{a_1, ..., a_n\}$ and $B = \{b_1, ..., b_m\}$, the goal is to define a set of equivalence relationships $R = \{(a_i, b_j) | a_i \in A, b_j \in B\}$ so that each pair of attributes refers to the same quality, measurement, or characteristic of the product, and both attributes in each pair can be used interchangeably without affecting the product information. For example, suppose we are looking to align two product schemas that describe the properties of backpacks, with the attribute "Outer Material" appearing in the first one, and "Exterior Fabric" in the second. These two attributes are very likely describing the same feature, that is, the material of which the external part of the backpack is made of. In order to determine whether two product attributes are a match or not, we rely on their *metadata*: this often includes a name, a short description explaining the scope of the attribute, a *data type* indicating the type of values to be used, and/or a list of possible values that the attribute can accept.

Product schema matching poses some unique challenges: for instance, the product schema is generally very wide and shallow. In fact, e-commerce services need to provide a wide range of information about the products they sell, in order to ensure an adequate experience for shoppers. On the other hand, a product schema only features few relationships between nodes, making it unsuitable for matching approaches that rely on structural data. The size also comes with an additional challenge: this task requires checking all pairs of attributes from the source and target schemas, which equates to a $n \times m$ matching task. As the number of pairs increases quadratically with the size of the schemas, reviewing each pair quickly becomes intractable.

Differences between product attributes can be very nuanced, especially when dealing with different modeling choices and levels of granularity, which often lead two attributes to be considered *partial matches* instead, *e.g.*, the attribute "Is Waterproof" only captures part of the information conveyed by the more fine-grained attribute "Water Resistance Level". In other cases, two attributes may have some minor differences that do not preclude semantic equivalence: in the domain of furniture tables, the attributes "Height" and "Table Height" are describing the same measurement, with the only difference being that the latter is explicitly scoped for tables. Finally, more complex relationships with a *1:n* or *m:n* cardinality are also possible, especially when an attribute is partially matching with two or more attributes. This phenomenon is similar to the complex ontology alignment task described in Amini et al. [2]. Referring back to the backpack domain example, one product schema may contain an attribute called "Number of Pockets", while another schema may distinguish between the "Number of Interior Pockets" and "Number of Exterior Pockets". While the approach described in this paper can be configured to cover both partial semantic matches and complex relationships, we decide to evaluate it on its ability to detect full semantic equivalence relationships, for two reasons: *(i)* it minimizes the risk of inconsistent labeling of the ground truth dataset due to differing interpretations of "partial match"; *(ii)* some amount of information overlap can be allowed, to align with customer expectations, fulfill legal requirements which require the service to display sensitive product information, or to power specific user experience elements.

4 Proposed Approach

In this paper, we present a system for product schema matching, the architecture of which is shown in Figure 1. The system can be divided into two main functional components: *retrieval* and *classification*. The retrieval component uses semantic search to identify the pairs of attributes from the source and target schemas that are most likely to be equivalent. These pairs are then input to the classification component, which relies on a pre-trained LLM together with a prompt template built by domain experts to determine whether each attribute pair is semantically equivalent or not. The architecture is highly customizable and configurable, and supports a wide range of embedding models and LLMs. Another advantage is that it does not make assumptions on the underlying metadata available for each schema, and can in fact be adapted to work with different types of information. The system can be used either in a fully unsupervised way, or it can use of a small development dataset to optimize its parameters for a specific task, allowing for maximum flexibility. Because it does not require supervised training, it can adapt to new schemas or to updates to an existing schema with minimal cost. The following sections will describe each component in detail.

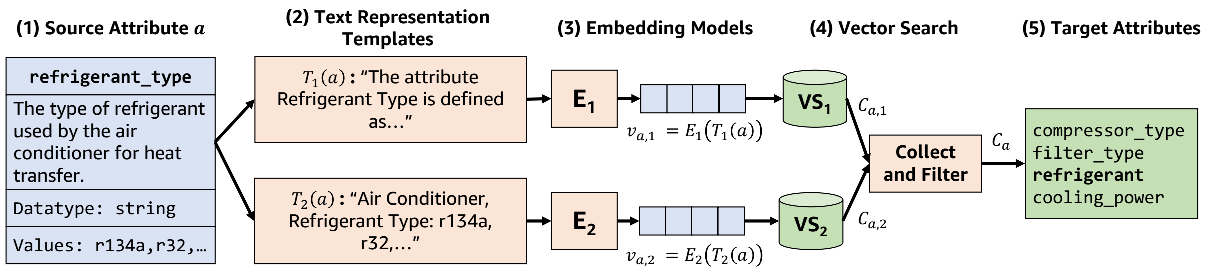

Figure 2: Running example of the retrieval step for the source attribute *refrigerant_type.*

4.1 Retrieval Step

As mentioned in Section 3, product schema matching theoretically requires processing all concept pairs. Despite this, we can expect that only a small fraction of pairs will actually represent a match. This means that an exhaustive search of the entire set of pairs is not only economically unfeasible, but also unnecessary. For this reason, we implemented a retrieval step, whose goal is to identify a subset of attribute pairs $(a_i, b_j) \in A \times B$ that are most likely to be semantically equivalent. This approach is also known as *blocking* in the literature for both OA and EM systems [3, 11], and can also be considered as an application of Retrieval-Augmented Generation (RAG) [7]. The system offers a *multi-retrieval* feature that can use multiple ways to represent each attribute at the same time, and even customize the representation based on attribute metadata.

To begin, text representation templates are used to convert all attributes from the source and target schema into standardized textual forms. The templates organize attribute metadata like its name, example values, and description into a sequence of words. We then use sentence embedding models to convert these sequence of words into vectors. Target schema embeddings are stored in a vector store, for reuse across source schemas and for efficient cosine similarity calculations. We also include metadata for each attribute (e.g. product type, data type) to reduce the candidate space before similarity calculations are done. At inference time, the attributes from the source schema are embedded, and used to query the vector store to retrieve the top K most similar target attributes via cosine similarity. Additionally, a minimum similarity threshold can be set, in order to filter out low-similarity pairs from the result set.

Thanks to multi-retrieval, the system can use multiple combinations of text representation templates and embedding models, and it can even be configured to use different templates for different types of attributes, combining the advantages of each model/template. For each combination k, we define the corresponding template T_k, embedding model E_k, target vector store $VS_k = \{v_{b_j,k} | b_j \in B\}$, candidate threshold K_k, and minimum similarity threshold S_k. Accordingly, given a source attribute $a \in A$, we define the corresponding text representation as $T_k(a)$, and the vector representation as $v_{a,k} = E_k(T_k(a))$. The set of candidate target attributes $C_{a,k}$ is then defined as follows:

$$\{b_j \in B | j \in Top_{K_k}(sim(v_{a,k}, v_{b_j,k})) \wedge sim(v_{a,k}, v_{b_j,k}) \geq S_k\} \quad (1)$$

Once the results are retrieved from each combination of template and embedding model, they are collected and merged. Assuming that we are using $k = 1...p$ templates and embedding models, the final set of candidates C_a is defined as follows:

$$C_a = \bigcup_{k=1}^{p} C_{a,k} \quad (2)$$

Figure 2 shows an example of the multi-retrieval approach in action, applied to the attribute *refrigerant_type* defined in the context of air conditioners: in step (2), the source attribute is converted into text using two templates T_1 and T_2, each using different metadata: the first template uses the attribute name and description, while the second one uses the attribute name and example values. In step (3), the sentences representing the source attribute are processed separately by embedding models E_1 and E_2 respectively, resulting in two distinct vector representations. These are then used to query the vector stores VS_1 and VS_2, representing the target schema attributes using the corresponding text representation templates and embedding models. Finally, in step (4), the final set of candidate target attributes is collected by merging the vector store outputs $C_{a,1}$ and $C_{a,2}$.

Our hypothesis is that the multi-retrieval approach enhances the diversity and coverage of the retrieval step output by utilizing various metadata in the semantic similarity calculation. For example, it can be used to select target attributes that either have similar descriptions, values, or both. The system can use the development set to get the optimized combination of the following parameters: the quantity and type of text representation templates and embedding models, the value of K_k, and the minimum similarity thresholds S_k.

As for the implementation, we use Jinja[1] to power the templates that convert attribute metadata into text. We chose it because it can produce highly flexible templates with limited need for configuration. We decided to employ embedding models offered by AWS Bedrock[2], which is a service that offers easy integration with state-of-the-art language models. For the vector store, we rely on the Amazon Relational Database Service (RDS) [3], which simplifies storage of vector-based data, and also provides efficient vector search functionalities.

[1]https://jinja.palletsprojects.com/en/stable/
[2]https://aws.amazon.com/bedrock/
[3]https://aws.amazon.com/rds/

4.2 LLM Classification Step

The classification step employs a LLM prompt to determine whether each pair of attributes is semantically equivalent or not. The component is made up of three parts: the *prompt template*, *input data*, and *output format*. The prompt template is the section of the prompt that contains the definition of semantic matching (see Section 3) and a set of instructions, both of which are crafted by our team of expert ontologists. We decided to follow this strategy since the definition itself is very specific to the e-commerce domain, but also relatively stable across different types of products. The full version of the prompt used by the LLM classifier is available in Figure 3.

The second component is the input data, which collects the attribute pairs and the metadata needed to perform the classification. Following the insights found in the literature, we rely on *batch prompting* [6] to group together multiple pairs of attributes in the same prompt execution. This approach was shown to reduce overall cost and increase classification stability, thanks to the additional amount of contextual information given to the LLM in each execution. Differently from the existing implementation, we group all pairs that share the same source attribute in a single prompt. We leave the experimentation of other batching strategies for future work. For source and target attributes, we provide the metadata that describes them, such as the name, description, data type, and values. In most cases, all available metadata is included in the prompt, however, data sparsity and quality issues may cause some parts of the metadata to decrease matching performance rather than increase it. For this reason, the system can be configured to include or exclude each metadata element from the LLM prompt.

The third and final element of the classification step is the output format, which defines the structure that the LLM must follow when providing answers. This is a unique challenge that arises when using a generative model to perform classification: unlike a traditional machine learning classifier, where the model output is specifically constrained to the task at hand, such as a leaf node for decision trees, or a probability vector for neural networks, generative models provide their answer as a sequence of words, which is then interpreted by the user. LLMs can be instructed to provide this answer in potentially infinite ways, and due to the way they function, the chosen answer format can affect the performance of the generative classifier itself.

In our system, we decided to generate answers in the form of a JSON array, with one item for each attribute pair from the input data. We then ask the LLM to provide the semantic equivalence level for each pair as a 5-point scale (*very low, low, medium, high, very high*), and set a cutoff value to map each value to a binary {*match, no match*} decision, *e.g.*, consider *high* and *very high* answers as a *match*, everything else as *no match*. While this choice may seem counter-intuitive, the experimental results show that it performs better than using a simple binary answer format, as it allows the LLM to ignore some of the irrelevant differences between attributes mentioned in Section 3. We also instruct the LLM to write a short paragraph explaining the given answer, that we use to investigate and remediate errors. Similarly to the retrieval step, the system can use the development set to choose the optimal combination of prompt template, attribute metadata, output format, and LLM.

Table 1: Dataset statistics

	Development Set	Test Set
# Source Attributes	400	999
# Source Attributes w/match	123 (30.8%)	342 (34.2%)
# Matching Pairs	129	354
# Product Types	81	85

The LLM services are provided through the AWS Bedrock service. Jinja is again used as the main templating engine for injecting the input data into the prompts, as well as processing the LLM answers. Prompt templates are persisted in Amazon DynamoDB[4] for analytics and reporting purposes.

5 Experiment

5.1 Dataset

We created a new product schema matching and duplicate detection dataset for the specific purpose of evaluating our system on data that is close to production. The dataset is made up of $1,399$ source attributes, describing properties of 85 different types of products, ranging from electronics (*e.g.,* computers and air conditioners), clothing (*e.g.,* sweaters), food, and furniture, with approximately 16.5 source attributes per product type. The dataset is split into two sets: a *development set* containing 400 source attributes, which we use to tune the model parameters, and a *test set* with 999 source attributes, used to measure the performance. Each source attribute in the dataset was manually mapped with the semantically equivalent attributes in our catalog (target) schema, if any exist, or is left empty in case no suitable match is found. The target schema contains on average $\approx$ 190 attributes per product type.

The source attributes come from an initiative to expand our catalog schema, during which new candidate attributes were proposed and modeled with the necessary metadata. We sampled this set using stratified sampling to ensure that all product types and data types are represented. We then manually labeled the attributes, following a two-step strategy: first, the source attributes were sent to our organization's data labeling team, who were asked to match each source attribute against the semantically equivalent attributes in the target schema, according to the definition described in Section 3. Then, the labeled data was manually inspected and reviewed with domain and data modeling experts, and the labels were corrected in case of errors. We chose this strategy because it represents a good compromise between labeling effort and quality, and also allows us to use the original labels as an additional baseline for our approach. Finally, we split the data in development and test sets using the same stratification setup as before. A training split is not necessary in this case, since our approach does not need model training.

Detailed dataset statistics are presented in Table 1, which describes the total number of source attributes, the number of source attributes with at least one matching attribute in our catalog schema,

[4]https://aws.amazon.com/pm/dynamodb/

The attribute {source attribute } for {product type} product type is described as: {source attribute description }. It has the following enumeration values: {...}. As an ontologist :

1. Carefully review the attribute description of {source attribute } against every target attribute (enclosed in the target_attributes XML tag)
2. Determine if any of the target attributes are an exact semantically equivalent match for the {source attribute } given the attribute description and the product type {product type}.

For attributes to be considered semantically equivalent they must have the following properties :

1. The attributes can be used interchangeably to express the exact same characteristic considering the product type {product type}.
2. The attribute must be at the same level of granularity and not be a more specific concept for the item. For example, an attribute describing a characteristic of a specific part of an item (e.g. pocket_material) would not be an exact match with an attribute that is either describing the whole product or is unclear about what part of the product it is describing (e.g. material).

Answer as a list of JSON objects, with each struct containing the following fields the input_attribute field , the target_attribute field , a semantic_equivalence_level field and a reasoning field (string) explaining your reasoning for the selecting the level. Ensure that the response is a list of JSON structs and encode quotes within JSON text. If no target attributes are available, provide an empty list ([]).

Apply the below rules when constructing the response:

1. semantic_equivalence_level is based on degree of semantic equivalence (very_high, high, medium, low and very_low) with very_high meaning exact semantic equivalence.

```
<target_attributes>
    <target_attribute>
        <name>...</name><description>...</description><values>...</values>
    </target_attribute>...
</target_attributes>
```

Figure 3: Full prompt used in the LLM classification step

the number of individual matching (source, target) attribute pairs, and the number of product types included in each split.

5.2 Experimental Setup

We evaluate the performance of our approach in a realistic setting by building a benchmark environment that is close to real operating conditions. The main task consists of processing new candidate product attributes proposed for a given product type, and determine whether they already exist in our catalog schema, or alternatively, if they are *net-new*. In this case, we are using product schema matching to intercept duplicate attributes and prevent them from being added to our schema: given a source attribute, if a semantically equivalent attribute is found in the schema, it is marked as a duplicate. If no match is detected by the system, the source attribute is considered as net-new.

The first stage of the setup is *tuning*, during which we identify the optimal combination of configuration parameters for our approach by running a grid-search on the development set. Specifically, we select the combination of parameters that obtains the best development set performance *wrt.* our goal metric. For the retrieval step, we adjust the quantity and content of the text representation templates T_k, the quantity and type of embedding models E_k, the number of candidate target attributes K_k, and the minimum similarity threshold S_k. For the classification step, we adjust the amount of attribute metadata provided to the LLM (*name+description* vs. *name+description+example values*) and the cutoff level for LLM answers (*very high* vs. *high*). Finally, we run the model with the best combination of parameters on the test set, in order to measure its unbiased performance. We repeat this process across five foundation models available in the Bedrock service: Claude Sonnet 3, 3.5, 3.5 v2, Mistral Large, and Mistral Large 2, in order to further prove the generality of the results. We chose these models as they are the largest and more recent currently offered by the Bedrock service. For all models, we set a temperature of 0, and a top-P of 0.999, to minimize the risk of inconsistencies in the answers.

5.2.1 *Metrics.*

We measured the performance on two tasks: *product schema matching*, and *duplicate detection.*

Product Schema Matching: Given a source attribute and a product type, the goal is to find all and only the attributes in the target schema that are semantically equivalent to it. Aside from finding duplicates, schema matching can be used also in other ways, such as consolidating the product information spread across multiple schemas. Performance is reported using the F1-score derived from the *information retrieval* definition of precision and recall:

$$P = \frac{|M_{pred} \cap M_{gt}|}{|M_{pred}|}, \quad R = \frac{|M_{pred} \cap M_{gt}|}{|M_{gt}|}, \quad F_1 = \frac{2PR}{P+R}, \qquad (3)$$

where M_{pred} is the set of attribute pairs that are predicted as semantically equivalent by the system, and M_{gt} is the set of ground truth matching pairs from the dataset.

Duplicate Detection: given a source attribute and a product type, the goal is to determine whether it is net-new, or it is a duplicate, *i.e.*, a semantically matching attribute exists in the target schema. This is simplification of schema matching, equivalent to a binary classification task, in which the positive class is assigned to duplicate attributes, and the negative class is assigned to net-new ones. We chose to report the binary classification F_2-score as our main metric, since false negatives (*i.e.,* introducing a duplicate attribute) pose a significantly higher risk compared to false positives (*i.e.,* rejecting a net-new attribute). This is the main metric for which the system is optimized during the tuning stage.

5.2.2 *Experimental Configurations.*

The *full version* of our approach, featuring the multi-retrieval and 5-point answer format, is compared against two variants: *single-retrieval*, and *binary-answer.* The *single-retrieval* variant only allows the use of one text representation

Table 2: Detailed results of the experiments on the test set

Model	Full Approach		*single-retrieval*		*binary-answer*		*human*	
	DD-F_2	PSM-F_1	DD-F_2	PSM-F_1	DD-F_2	PSM-F_1	DD-F_2	PSM-F_1
Claude Sonnet 3	84.44%	70.35%	83.47%	71.90%	85.48%	64.62%	81.82%	79.75%
Claude Sonnet 3.5	88.51%	74.14%	84.19%	75.58%	82.27%	75.80%	-	-
Claude Sonnet 3.5 V2	88.15%	74.42%	85.14%	76.46%	84.92%	77.32%	-	-
Mistral Large	**90.20%**	76.26%	86.02%	76.75%	86.26%	**78.12%**	-	-
Mistral Large 2	87.55%	67.94%	85.33%	71.29%	85.71%	74.87%	-	-
Avg. Difference	-	-	+2.94%	-1.77%	+2.85%	-1.52%	-	-

template and embedding model during the retrieval phase. Specifically, we use the template and embedding model that achieves the best performance on the development set. The *binary-answer* variant of the approach replaces the answer format of the LLM classifier with a binary one. Finally, the *human* baseline is obtained from the raw annotations produced by our internal data labeling team, as described in Section 5.1, which approximates the average performance of non-domain expert humans.

5.3 Experimental Results

Table 2 describes the results of the experiment on the test set, reporting, for each approach and foundation model, the performance achieved for both tasks described above. Specifically, the *DD-F_2* columns report the F2-score of the Duplicate Detection task, and the *PSM-F_1* columns report the F1-score for the Product Schema Matching task. The *Avg. Difference* row compares the average score of the full approach against the variants. Of course, the *model* column does not apply to the *human* baseline. We also measured the execution time of our approach on the test set: on average, the retrieval step was completed in 2 minutes, while the classification step was completed in 20 minutes.

5.3.1 Duplicate Detection Task. The results show that our approach demonstrated good levels of performance for the Duplicate Detection task, especially in conjunction with the Mistral Large model, which scored the highest overall *DD-F_2* (90.2%). In fact, this configuration was able to correctly identify more than 95% of duplicate source attributes in the test set. Compared to the *single-retrieval* and *binary-answer* variants, the full version of our approach is able to optimize our main goal metric, with an average uplift of $\approx$ 2.9% in F_2 across the five LLMs.

The multi-retrieval strategy allows our approach to retrieve more relevant target attributes, while the 5-point answer format induces the LLM to be more flexible in assessing whether two attributes are semantically matching, compared to a purely binary answer format. In fact, during the tuning phase we observed that setting the cutoff answer to *high* almost always yields better performance than *very high*, meaning that the LLM can ignore negligible differences between matching attributes.

The most surprising result however is that our approach achieved higher *DD-F_2* than the *human* baseline in all configurations. The main reason is the low recall of the human annotators, who missed a large amount of matching attribute pairs. This is understandable: the schema matching task involves exhaustively comparing a large amount of attribute pairs, and there is a non-negligible risk that humans can miss some of them. On the other hand, our automated solution is able to thoroughly navigate the search space and systematically make thousands of comparisons.

5.3.2 Product Schema Matching Task. In this case, results show that the *binary-answer* and *single-retrieval* variants often perform better than the full approach, with the best overall *PSM-F_1* being 78.12%. Looking at the results in detail, we observed that the full approach tends to detect more attribute pairs as matching compared to the other variants, leading to a 1.77% decrease in *PSM-F_1*. This is consistent with the finding that the multi-retrieval strategy and 5-point answer format optimized for *DD-F_2* performance. The *binary-answer* variant obtained the best overall performance, being only 1.6% below the *human* baseline, proving that our system can also perform competitively with human annotators when configured accordingly. By further analyzing the results, we however discovered that the precision-recall balance of the two differs significantly: as previously mentioned, human annotators lack in recall (72.31%), but make up with very high precision (88.89%). On the contrary, our best result from the *binary-answer* variant leads in recall (82.21%) but suffers from lower precision (74.43%). The full approach features even wider gaps between precision and recall, which explains the lower F_1.

In summary, the results prove that our approach performs admirably well even in the more complex product schema matching task, but also that there is still a wide margin for improvement, especially in reducing false matches. The most common cause of false positives can be attributed to partial information overlap being mistaken by the LLM as full duplication (*e.g.,* confusing the primary product material with the material of one of its components). An example is found in the air conditioner domain, where the system matched the net-new source attribute *Energy Efficiency Ratio* with the target attribute *Seasonal Energy Efficiency Ratio*, which are two separate albeit related measurements of the efficiency of the product. Because the LLMs used in this experiment are trained on general-purpose tasks, they can sometimes overlook nuanced modeling differences such as the one above, even when domain-specific instructions are provided as part of the prompt text.

5.3.3 Comparison across LLMs. As expected, the results of the experiment are relatively stable across the five foundation models included in the experiment: the full approach and the *binary-answer* variant report the highest *DD-F_2* and *PSM-F_1* respectively in four out of five models. Indeed, Claude Sonnet v3 represents an outlier, sporting slightly lower performance than the others, which is understandable since it is a significantly older model. On the other hand, the gap between Claude Sonnet v3.5 and Mistral Large is much closer, with the latter being the best performer for both tasks. Overall, these results further support the validity and generality of the findings presented in this section, and also confirm that state-of-the-art pre-trained LLMs are capable of understanding the concepts behind product catalog schemas, and detecting semantically-rich relationships between them.

5.4 Online Experiment

The experimental results described in Section 5.3 were instrumental in selecting the configuration of the product schema matching system with the highest duplicate attribute detection performance. Accordingly, we decided to proceed with the full version of our approach, *i.e.* with multi-retrieval and 5-point answer format, using Mistral Large as the LLM for the classification step. We conducted an online experiment, during which the approach was made available to internal users, to estimate how well the system can reduce the time and effort required by human experts in making schema-level decisions, thus improving efficiency. According to one of our senior ontologists in fact, it takes on average 10 minutes to check a new proposed attribute against the existing catalog schema, and determine whether it is a duplicate.

The online experiment involved an unlabeled set of 687 candidate attributes, proposed for 85 product types, as well as a group of ontologists from our organization. The system was able to determine that 319 (46.4%) of these candidate attributes already exist in our schema, taking less than one hour of total processing time. Because the system meets our objective recall level for automation ($> 95\%$), the only remaining step is to confirm that the duplicate attribute pairs are correct, which humans can perform more efficiently and accurately. During the experiment, we measured that it takes approximately two minutes for an ontologist to verify that a pair of attributes is truly semantically equivalent.

Accordingly, we estimate that our product schema matching system would reduce the amount of ontologist time from $687 \times (10 \text{ minutes}) = 114.5$ hours to only $319 \times (2 \text{ minutes}) \approx 10.6$ hours, *i.e.,* a 91% reduction. This is an excellent result, which proves that LLMs can help domain experts make faster and more accurate decisions, reducing the time needed to filter out duplicate content, and discover net new attributes from other schemas. Furthermore, this would enable an e-commerce service to react to changes in the product space much faster, resulting in a more comprehensive, accurate, and enjoyable shopping experience for buyers.

6 Conclusion and Future Work

In this paper, we presented an application that combines semantic search and large language models for the product schema matching and duplicate detection tasks. Through an experiment, we evaluated the performance of our approach on a real e-commerce problem, *i.e.,* identifying and filtering out duplicate attributes, using a purpose-built ground truth dataset to simulate realistic operating conditions on a multitude of product types. Experimental results show that our approach outperforms human annotators in the duplicate attribute detection task with 90.2% F_2, and also achieves near-human performance in the product schema matching task with 78.12% F_1. Features such as multi-retrieval and the fine-grained answer format allowed the system to identify and remove even more duplicate attributes, although with a small cost in terms of precision. Through an online experiment, we found that our approach can reduce the time that ontologists need to dedicate to duplicate detection by up to 91%.

However, there are still some limitations that need to be pointed out. As mentioned in Section 5.3, the system outperforms humans in terms of recall, but precision still poses a significant obstacle, as it tends to miss the more nuanced differences between non-matching attributes. Of course, partial matches and other cases of complex relationships described in Section 3 are also often mistaken as duplicates, which is expected, and highlights the difficulty of the product schema matching task. Therefore, solving the precision problem constitutes an interesting avenue for future work, for which we propose the integration of *in-context learning* strategies, such as those discussed in [6, 12, 15]. In-context learning can be especially useful in teaching the LLM how to handle the most complex cases by dynamically retrieving relevant labeled examples from a knowledge base. Integrating a confidence scoring mechanism can also help further reduce the level of human intervention required, by sending only the attribute pairs with low confidence to manual review.

Output consistency is another limitation observed during our experiments: small differences in the wording of a prompt can sometimes induce LLMs to change their answer unexpectedly, which is a consequence of the probabilistic nature of their output. The problem is amplified by the use of batch prompting, which forces the LLM to reason over multiple attribute pairs at once, meaning that the answer given to a pair also is influenced by the other pairs in the same batch. As future work, we propose to investigate approaches to reduce the inconsistency of the model output, *e.g.*, by using *ensemble* models, or employing dedicated models fine-tuned for the product schema matching task.

Finally, we will continue to iterate with more comprehensive experiments, which will include a larger variety of retrieval and classification parameters, as well as text representation templates, embedding models, and LLMs. We will also consider expanding the scope of the system to match other languages and/or types of product entities. Another avenue for future work is the detection of complex relationships, such as partial overlaps between attributes.

Acknowledgments

We would like to thank Eric Kauz for sharing his expertise in the product schema domain, which was instrumental in writing the definition, prompt engineering, and reviewing the dataset. We are also thankful to Simon Hartmann, Klevis Ramo, Jona Neumeier, Arsh Kumar, and Hasan Turalic for the design and implementation of the overall infrastructure, which was used during the online experiment, as well as all the ontologists from the Ontology team that participated. A special mention must also be given to the Global Catalog Operations team, which annotated the dataset used in the experiment. Finally, we thank Grant Galloway, Arjun Bakshi, Sharadha Viswanathan, Tarang Chugh, and Tarik Arici for providing valuable feedback during the writing of this paper.

References

[1] Ali, A., Rasool, G., and Pathania, A. Antecedents for success of e-commerce platforms: an investigative approach. *Int. J. Inf. Technol. Manag. 16*, 4 (2017), 376–390.

[2] Amini, R., Norouzi, S. S., Hitzler, P., and Amini, R. Towards Complex Ontology Alignment using Large Language Models, July 2024. arXiv:2404.10329.

[3] Chen, J., Dong, H., Chen, J., and Horrocks, I. Ontology Text Alignment: Aligning Textual Content to Terminological Axioms. In *Frontiers in Artificial Intelligence and Applications*. IOS Press, Oct. 2024.

[4] Chugh, T., and Zambre, D. ASTRA: Automatic Schema Matching using Machine Translation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024* (2024), Association for Computational Linguistics, pp. 1237–1244.

[5] Ehrig, M. *Ontology Alignment: Bridging the Semantic Gap.* Springer Science & Business Media, Dec. 2006. Google-Books-ID: nxzBZonEF50C.

[6] Fan, M., Han, X., Fan, J., Chai, C., Tang, N., Li, G., and Du, X. Cost-Effective In-Context Learning for Entity Resolution: A Design Space Exploration. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024* (2024), IEEE, pp. 3696–3709.

[7] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., and Wang, H. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR abs/2312.10997* (2023). arXiv: 2312.10997.

[8] He, Y., Chen, J., Antonyrajah, D., and Horrocks, I. BERTMap: A BERT-Based Ontology Alignment System. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022* (2022), AAAI Press, pp. 5684–5691.

[9] He, Y., Chen, J., Dong, H., and Horrocks, I. Exploring Large Language Models for Ontology Alignment. In *Proceedings of the ISWC 2023 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 22nd International Semantic Web Conference (ISWC 2023), Athens, Greece, November 6-10, 2023* (2023), vol. 3632 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[10] Hertling, S., and Paulheim, H. OLaLa: Ontology Matching with Large Language Models. In *Proceedings of the 12th Knowledge Capture Conference 2023* (Pensacola FL USA, Dec. 2023), ACM, pp. 131–139.

[11] Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow. 14*, 1 (2020), 50–60.

[12] Liu, X., Wang, R., Song, Y., and Kong, L. GRAM: Generative Retrieval Augmented Matching of Data Schemas in the Context of Data Security. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024* (2024), ACM, pp. 5476–5486.

[13] Otero-Cerdeira, L., Rodríguez-Martínez, F. J., and Gómez-Rodríguez, A. Ontology matching: A literature review. *Expert Syst. Appl. 42*, 2 (2015), 949–971.

[14] Peeters, R., and Bizer, C. Dual-Objective Fine-Tuning of BERT for Entity Matching. *Proc. VLDB Endow. 14*, 10 (2021), 1913–1921.

[15] Peeters, R., Steiner, A., and Bizer, C. Entity Matching using Large Language Models. In *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025* (2025), OpenProceedings.org, pp. 529–541.

[16] Rahm, E., and Bernstein, P. A. On matching schemas automatically. *VLDB journal 10*, 4 (2001), 334–350.

[17] Shieh, E., Simhon, S., Aluri, G., Papachristoudis, G., Yakut, D., and Raghu, D. Attribute Similarity and Relevance-Based Product Schema Matching for Targeted Catalog Enrichment. In *2021 IEEE International Conference on Big Knowledge, ICBK 2021, Auckland, New Zealand, December 7-8, 2021* (2021), IEEE, pp. 261–270.

[18] Tu, J., Fan, J., Tang, N., Wang, P., Li, G., Du, X., Jia, X., and Gao, S. Unicorn: A Unified Multi-tasking Model for Supporting Matching Tasks in Data Integration. *Proc. ACM Manag. Data 1*, 1 (2023), 84:1–84:26.

[19] Univ Ctr of El Bayadh, Inst. Science and Technology, Algeria, Ardjani, F., Bouchiha, D., and Malki, M. Ontology-Alignment Techniques: Survey and Analysis. *International Journal of Modern Education and Computer Science 7*, 11 (Nov. 2015), 67–78.