Attribute Similarity and Relevance-Based Product Schema Matching for Targeted Catalog Enrichment

Evan Shieh, Saul Simhon, Geetha Aluri, Giorgos Papachristoudis, Doa Yakut, Dhanya Raghu
{esshieh,ssimhon,aluriga,geopap,dyakut,dhanyr}@amazon.com
Amazon

ABSTRACT

Many eCommerce catalogs rely on structured product data to provide a good experience for customers. For large scale services, product information is provided by millions of different manufacturer and vendor schemas. Due to inherent heterogeneity of this data, unifying it to a consistent catalog schema remains a challenge. Schema matching is the problem of finding such correspondences between concepts in different distributed, heterogeneous data sources. Most approaches in automated schema matching assume either a small number of source schemas, attributes, and contexts (i.e., matching movie attributes from media knowledge bases). By contrast, schema matching in product catalogs encounter the problem of scaling across millions of noisy, heterogenous schemas spanning thousands of categories and attributes.

In this paper, we introduce a scalable schema matching framework that utilizes unsupervised domain-specific attribute representations and general attribute similarity metrics. Our method first identifies relevant attributes for a given product based on existing customer signals, and then prioritizes among candidate attributes to consolidate only those relevant product facts from multiple manufacturers and vendors with little to no labeled data. We demonstrate value by experiments that enriched catalog data containing millions of attribute enumerations sourced from tens of thousands of schemas across a wide range of product categories. Experimental results show reduced manual annotation efforts by 75% from competing schema matching efforts by automating schema matching on targeted product facts, resulting in high accuracy, precision, and recall for important attributes that contribute to customer interest. We also demonstrate performance improvements of 8% MRR using our approach compared against two well-established approaches to unsupervised schema matching.

ACM Reference Format:
Evan Shieh, Saul Simhon, Geetha Aluri, Giorgos Papachristoudis, Doa Yakut, Dhanya Raghu. 2021. Attribute Similarity and Relevance-Based Product Schema Matching for Targeted Catalog Enrichment. In *WIT @ KDD '21: WIT: Workshop On Deriving Insights From User-Generated Text @KDD2021 @ The 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 14–18, 2021, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

A key problem in knowledge systems is schema matching [1, 17]. This is especially important for systems that support eCommerce transactions over diverse datasets, such as product catalogs sourced from a variety of domain schemas and product categories [21]. Consolidating product data from different sources in a consistent, structured representation (or schema) is crucial for services that provide rich customer experiences at scale (such as enabling users to search for products and make informed purchasing decisions).

The diversity of product categories poses a challenge in schema matching at scale. For example, a schema used to model *computer monitors* may require an attribute for "`screen-size`" while for *women's swimwear* it may require an attribute for "`top-style`". Working with datasets across various domain schemas (i.e. manufacturers or vendors) further compounds the problem, where categories and attributes are represented in differing ways. For example, one data source may include an attribute "`Flash Memory: included`", another source may include "`Memory Card: Micro SD 64GB`", and third may include "`Description: Sony Cybershot, 18 MP, 16X Digital Zoom, 1 TB Micro SD`".

Many schema matching methods assume structured databases from small sources with clean data [13]. However, such methodologies are not suited to large scale eCommerce data, where the number of available domain schemas is in the order of millions when accounting for different manufacturers, vendors, and categories. More sophisticated methods learn from data [3, 15, 21] to automatically extract attributes using named entity recognition (NER) models trained to match individual attributes. However, since such models are often trained at the attribute or category level, achieving scale is difficult in settings where there exist a large number of applicable attributes and categories. Finally, we observe inherent challenges in matching across diverse product schemas that often lack grammatical structure and carry context-specific semantics.

In this work, we propose a novel approach to enrich existing non-homogenous, noisy and incomplete catalog data. Our approach involves driving a product catalog augmentation process by first identifying **customer-relevant attributes** for a product category in an unsupervised fashion, and then prioritizing a novel **attribute matching model** to help consolidate product facts from multiple sources (manufacturers and vendors). Customer-relevant attributes are defined as attributes that contribute to the customer interest about the product. We refer to this as *Targeted Catalog Enrichment*, where the focus is on back-filling and consolidating the catalog data that customers may care about and reducing information overload. While most product categories can carry several hundred or more unique attributes, it is estimated that most customers, on average, only care about less than thirty.

To address the challenge of scaling across domain schemas and attributes, our system performs schema matching with limited or no annotated training data required to map additional attributes or schemas (addressing the **cold start** problem). Furthermore, to address the challenge of context-specific attribute semantics, we learn

WIT @ KDD '21, August 14–18, 2021, Singapore, Singapore

ACM ISBN 978-1-4503-XXXX-X/18/06.
https://doi.org/10.1145/1122445.1122456

domain specific language representations from customer signals found specifically in product categories. We demonstrate that our system performs schema matching with both scale and accuracy in enriching catalog knowledge with millions of attribute enumerations sourced from tens of thousands of source schemas. In the process, we also reduce the burden of manual annotation by 75% from previous schema matching systems.

1.1 Approach Overview

We first develop an unsupervised model for predicting relevancy of product attributes. We then develop a schema matching model that matches relevant attributes across different schemas. Figure 1 shows the overall workflow for Targeted Catalog Enrichment.

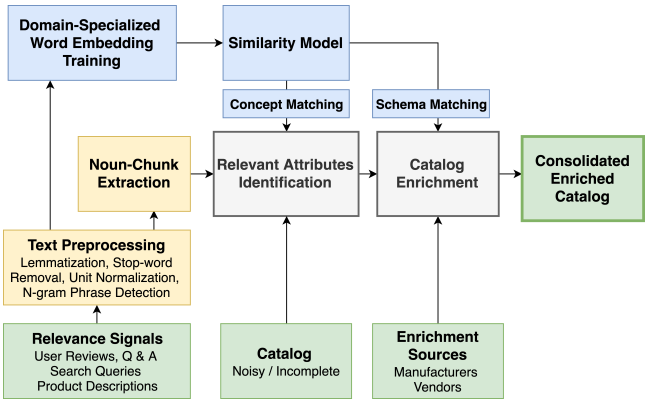

Figure 1: Relevance Based Catalog Enrichment.

1.1.1 Semantic Attribute Similarity. Both the attribute relevancy model as well as the schema matching model are based on an underlying model for *semantic attribute similarity*, which involves identifying and filtering salient candidate tokens from a phrase that describe product attributes (sourced from either corpus data or from schema data), and then applying a language model to compute the semantic similarity between relevant tokens.

In our context, it is important that language models are trained to distinguish semantics that vary based on the product category. For example, the token "hp" may represent horsepower in the context of *lawnmowers*, or Hewlett Packard in the context of *printers*. Such relationships can be learned by training word embeddings on a suitable product corpus. In our work, we use product descriptions and customer reviews from products grouped by product category in our target schema.

1.1.2 Attribute Relevance. Knowledge of what attributes are relevant to customers for products within a certain category or context allows us to prioritize attributes for enrichment through schema matching. However, inferring relevant attributes manually is inherently subjective and laborious when spanning a large number of diverse categories.

We formulate computing saliency measures for product attributes as an information retrieval problem, where the prevalence and likelihood of corresponding concepts found in customer signals (ex. search queries, customer reviews, or customer Q&A) is considered as the metric for attribute relevance. We begin with a set of attributes sourced from an incomplete and noisy catalog, and match concepts found in the signals to the attributes in the target schema. The frequencies of such matched concepts in the signals is then combined with heuristics to compute an overall relevancy score for each attribute in a product category.

1.1.3 Schema Matching. Finally, we enrich our product catalog by integrating multiple sources for product data. In order to support enrichment, schema matching attempts to consolidate attributes across different schemas while addressing the problem of *impedance mismatch* [7]. Our approach relies on the same underlying mechanism for semantic matching of product concepts, however, we apply a variant of attribute similarity that leverages sequence information to improve precision. We augment our similarity metric using ontology-specific normalization, and build models to schema match through ranking and classification.

2 RELATED WORK

In traditional schema matching where full access to the source schema and attribute relationships are available, [13] present a seminal approach to schema match based on query discovery. However, this is rarely applicable in online settings where schemas are implicitly represented (as key-value pairs, in flat tables, or in freetext as is common with product schemas).

Modern approaches to schema matching address this problem by leveraging deep learning techniques to process schemas as sequences of tokens from which relevant phrases are extracted from noisy text and then matched as attributes. [21] poses attribute extraction as a sequence tagging problem and applies a BiLSTM-CRF network with attention layers to perform schema matching from freetext. An added benefit of learning on sequences is that such approaches may extend to perform schema matching and entity matching simultaneously, as [15] does with a CNN-based attentional model *Seq2SeqMatcher* and [3] does with variants of *BERT* [5]. The main drawback that render such approaches infeasible in our problem space is their need for large training datasets. Both [15] and [3] are trained on tens of thousands of training examples given relatively few attributes (ranging from 3 to 8). [21] tries to address this using an active learning mechanism (similar to [19]) that reduces the burden of annotation to several hundred examples per attribute and product category, but this does not scale in our setting where our target scale is tens of thousands of attributes and categories. Nevertheless, these approaches all point to the success of modeling attributes as sequences of tokens with varying relevance. We leverage this intuition in our work while reaching scale with unsupervised methods.

For unsupervised approaches to schema matching, [10] introduces a clever procedure that clusters attributes using self-organizing maps and uses cluster membership to train a feedforward neural network to perform schema matching. [20] extends this general approach to the web service setting. However, this style of schema matching requires re-training the clusters for each additional schema, which does not scale in our problem space with hundreds of thousands of potential schemas. To address this problem, [16] and [9] leverage semantic similarity from word embeddings to perform schema matching. These approaches scale in our scenario,

but at the expense of discarding valuable sequence signals through Bag-of-Words (BOW) models that perform poorly in noisy settings. We build upon these semantic embeddings approaches for unsupervised schema matching while preserving the sequence-based structure that deep learning schema matching models employ. To our knowledge, our work is the first to combine both approaches to schema matching.

3 SEMANTIC ATTRIBUTE SIMILARITY

The task of computing attribute similarity is a challenging one since product data lack regular grammatical structure and tend to be skewed towards technical and abbreviated language. In addition, word pairs with large edit distance (ex. "`centimeters`", "`feet`") might still refer to the same underlying concept. Expanding on the example in Sec. 1.1.1, different words within any given category may also represent the same concept, for example, *spring mattress* and *coil mattress* are synonymous to the same type of mattress.

To address these problems we learn semantic word embeddings by training on data within specified product categories. We leverage customer product reviews and product descriptions as they are readily found on product listings and are a rich source of natural text for training across many categories. We use FastText as it handles both whole words and character n-grams [2] that better match on abbreviated language common in product corpora.

We denote the normalized embedding of word w by e_w. We are interested in either computing similarities between an attribute and free text as well as matching an attribute from a source domain to its most similar attribute from a target domain. Let $\mathcal{A}_z = \{a_1^z, a_2^z, \ldots, a_n^z\}$ denote the set of attributes under reference domain Z. For the sake of convenience, we will drop symbol z unless the domain under which attributes are defined is unclear. Each attribute a_i is defined as $a_i := \langle k(a_i), \mathcal{V}(a_i)\rangle$, where $k(a_i)$ is the attribute name and $\mathcal{V}(a_i)$ is the set of unique attribute values which we refer to as *enumerations*. For example, an attribute that represents color can take the form $a = \langle$"`color`", {"`black`", "`brown`", ..., "`yellow`"}$\rangle$.

3.1 Semantic Similarity

The semantic similarity between two words w_i, w_j is defined as the cosine similarity of the corresponding word embeddings or more simply as the dot product of the normalized word embeddings:

$$\text{sim}_w(w_i, w_j) = e_{w_i}^T e_{w_j} \tag{1}$$

3.2 Sequence Similarity

A string sequence can represent many different things, from attribute values to customer reviews or product titles (such as "`travel backpack with USB port and silver straps`"). Although measuring similarity over such sequences can be done by aggregating words using mean embeddings [18], this approach does not represent product information well as it treats each word equally in addition to discarding word order. For example sequences like "`navy blue backpack, silver straps`", and "`silver backpack, navy blue straps`" would have the same mean embedding but a completely different meaning.

To account for that we consider two alternatives that take word importance and structure into account. The first one computes the similarity of each token in sequence s_x against its best matching token in sequence s_y weighted by tf-idf, while the other considers the harmonic mean of the similarities of all subsequences starting from the beginning of the sequences. The first approach has increased entropy and is used for relevance computation, while the second approach is biased towards more strict matching across text fragments and is used for precise schema matching.

In the first approach, we define the similarity between two sequences s_x, s_y as:

Initialization: $\text{sim}_s(s_x, s_y) = 0, \text{norm}_s(s_x, s_y) = 0$
while $s_x \neq \emptyset$ **do**
  $(w_i^*, w_j^*) \leftarrow \underset{w_i \in s_x, w_j \in s_y}{\arg\max}\ \text{sim}_w(w_i, w_j)$
  $\text{sim}_s(s_x, s_y)$ += tf-idf$(w_i^*) \cdot$ tf-idf$(w_j^*) \cdot \text{sim}_w(w_i^*, w_j^*)$
  $\text{norm}_s(s_x, s_y)$ += tf-idf$(w_i^*) \cdot$ tf-idf(w_j^*)
  Remove w_i^* from s_x, w_j^* from s_y
end while
$\text{sim}_s(s_x, s_y)$ /= $\text{norm}_s(s_x, s_y)$

where $\text{sim}_w(w_i, w_j)$ is defined in (1) and tf-idf(w_i^*), tf-idf(w) is the tf–idf score of word w. That is, we compute the similarities of all token pairs between s_x, s_y. We keep the largest similarity and remove the corresponding tokens from s_x, s_y. We continue by keeping the second largest similarity, while removing the corresponding tokens from s_x, s_y until we exhaustively remove all tokens from s_x.

In the second approach which we use for matching, we consider contiguous subsequences of each sequence and measure similarity as the harmonic mean of all position-wise word similarities. This is a departure from BOW approaches used in similar applications [9, 16] that we introduce in order to encode sequential information crucial for matching attributes. We call this *Maximum Contiguous Subsequence Similarity (MCSS)*, defined as:

$$\text{sim}_s(s_x, s_y) = \gamma(s_x, s_y) \cdot \max_{s'_x, s'_y} \frac{n}{\sum_{i=1}^{n} \text{sim}_w(s'_x[i], s'_y[i])^{-1}} \tag{2}$$

where $n = \min(|s_x|, |s_y|)$ is the number of words in the shorter sequence, and s'_x, s'_y are n-length subsequences of s_x, s_y respectively. $\gamma(s_x, s_y) = \min(1, |s_x|/|s_y|)$ penalizes similarity when the enumeration sequence from source domain x is shorter than the one from target domain y. We use the harmonic mean for aggregation to favor cases where similarity is consistently high across all words in the sequence. In simple terms, we compute similarity between two enumerations by comparing their most similar text fragments equal in length to the shorter enumeration. For example, given $s_x =$ "`Length: 15 centimeters diagonal`", $s_y =$ "`18 cm`", MCSS will compare text fragments that produce the maximum position-wise similarity (in this case, "`15 centimeters`").

3.3 Attribute Similarity

To define similarity between two attribute enumeration sets, we simply take the weighted average MCSS similarity across all enumeration pairs in sets $\mathcal{V}(a_x), \mathcal{V}(a_y)$:

$$\text{sim}_{\mathcal{V}}(\mathcal{V}(a_x), \mathcal{V}(a_y)) = \frac{\sum_{s_x} \sum_{s_y} p(s_x) p(s_y) \text{sim}_s(s_x, s_y)}{\sum_{s_x} \sum_{s_y} p(s_x) p(s_y)} \tag{3}$$

where $p(s_z)$ is the probability that attribute a_z has value s_z in a given product in schema z.

The attribute similarity is then simply the harmonic mean of name and enumeration similarities.

$$\text{sim}_a(a_x, a_y) = \frac{2}{\text{sim}_s(k(a_x), k(a_y))^{-1} + \text{sim}_{\mathcal{V}}(\mathcal{V}(a_x), \mathcal{V}(a_y))^{-1}} \tag{4}$$

4 RELEVANT ATTRIBUTES IDENTIFICATION

The problem of computing attribute relevance is cast as an information retrieval problem. Our core assumption is that product concepts that contribute to customer interest are co-located in customer signals and catalog data for a particular product category with higher frequency than across arbitrary products. We define a *product class* as a set of products that share the same form and utility. Let the domain $z = \{z_1, z_2, \ldots, z_k\}$ represent a set of products for a product class z, and let $\mathcal{A}_z = \{a_1^z, a_2^z, \ldots, a_n^z\}$ denote the set of attributes that are applicable to all products in the class. For all products in the class, there must exist a valid enumeration value set $\mathcal{V}(a_i^z)$ for all attributes a_i^z. A product class may also include optional attributes $\mathcal{A'}_z$, and a *coherent* product class is one where $\|\mathcal{A}_z\| \gg \|\mathcal{A'}_z\|$. We use the terms product class and product category interchangeably.

In practice, the problem of identifying the ontology of product classes and the corresponding set of applicable attributes is the problem we aim to approximate by inferring only the relevant attributes that contribute to customer interest. We assume that we can approximate the class of products from an existing noisy catalog, and aggregate a superset of candidate attributes $\mathcal{A}_{z'}$ that may or may not be applicable to that class z. Given this initial approximation from noisy catalog data, we then aim to compute a relevancy score for attributes with respect to some relevance context c as defined by the customer signals. We then rank candidate attributes in $\mathcal{A}_{z'}$ and consider very low ranking ones as noise to be removed and low ranking ones as not relevant.

4.1 Predicting Attribute Relevance

Given a catalog with a noisy schema definition and both incomplete and noisy product data, our objective is to estimate a probability distribution $p(a, z, c)$ from source signals corresponding to relevance context c. The likelihood that a_i is a relevant attribute is then estimated by computing the relative probabilities of finding concepts corresponding to attributes a_i in the source signals (associated to the product class z and the relevance context c). The corresponding likelihood of each concept fragment found the corpus then provide the relevance scores for the attributes:

$$p(a_i|z, c) = \frac{\sum_{j,x} p(c_{j,x}|a_i, z) * p(a_i|z)}{\sum_{j,x} p(c_{j,x}|z)} \tag{5}$$

where $c_{j,x}$ is a text fragment x from source signals j corresponding to relevance context c, and $p(c_j, x|a_i, p)$ is the probability of the text fragment given attribute a_i and product class z. The likelihood for a text fragment given the attribute can be estimated as the likelihood that both the text fragment and the attribute correspond to the same concept s, i.e., semantic similarity. This likelihood is computed using the semantic similarity metrics described in Sec. 3.2. The probability of an attribute a_i given a product class z is inferred from statistics in catalog data. All scores are normalized across the corpus.

In practice, we also compute attribute relevance scores by normalizing each signal separately using the highest scoring attribute for that signal. This allows us to combine scores from different signals while avoiding scenarios where one dominant customer overshadows all of the others. This is further discussed in Sec. 4.2.

Once relevance scores are computed, we rank the attributes according to their relevance and empirically set a threshold to the maximum number of relevant attributes that are desired for schema enrichment in conjunction with a threshold for the the minimum relevance score. These thresholds are empirically determined by evaluating results for several hundred product classes.

4.2 Relevance Context

We define three relevance contexts for different use cases and compute an overall relevance score as a uniform combination of the three contexts. Below is a list of the relevance contexts and the corresponding source signals used for each.

- *Product Compare*: attributes relevant for comparing products for purchasing decisions. Source signals include *customer reviews*, *customer Q&A*, and *vendor product descriptions*.
- *Product Title*: attributes relevant for display on titles. Source signals include *vendor product titles*.
- *Product Discovery*: attributes relevant for product search and filtering. Source signals include *customer search queries*.
- *Overall Relevance*: a uniform average of the relevance scores of all the above.

We justify that customers discuss in reviews and Q & A forums important product features for making product purchasing decisions. We also justify that vendors or manufacturers emphasise what they believe customers care about in their product descriptions and overviews. (Note that customer reviews and product descriptions are also used to train our word embeddings model).

The Overall Relevance context is used to drive Schema Enrichment in our experiments. However, our approach generalizes to help prioritize catalog enrichment for other aspects of relevancy. Further, relevant attributes for each context may be labeled in the schema for their contextual usage, such as for display priority customer application, or for other system workflows.

4.3 Preprocessing Corpora and Catalog Data

Each data set is preprocessed using a language model based on work from Kiperwasser et. al. [8]. The corpus is segmented, tagged by POS tags, lemmatized, stop words are removed and units and numbers are normalized. POS tagging is excluded on schema data, where the words found in a semi-structured model generally do not exhibit grammatical structure.

Bigrams and trigrams (e.g., screen size -> screen_size) are automatically detected based on high-frequency co-located tokens following the approach outlined by Mikolov, et al. [11]. We train an unsupervised ngram phrase detector based upon the continuous skip-gram model. As input, we use the same data corpora used to train the word embeddings, and configure a low score threshold (0.25) to encourage phrase formation. Tokenization based on phrase detection produces n-grams of similar length from customer signals, manufacturer/vendor values and catalog values. Distributional differences can then be accounted for by measuring semantic

similarity against these phrases as described in Sec. 3. We apply the same phrase model on all domains to avoid introducing bias.

We further consider that all product concepts found in the corpus are nouns. We construct a histogram dictionary comprising of only *noun chunks* extracted from the data after POS tagging. Given the nature of the corpus, we expect this dictionary to predominantly contain product facts. This dictionary is then the source of text fragments $c_{j,x}$ for computing relevance in Equation 5.

5 SCHEMA MATCHING: ATTRIBUTE RANKING AND CLASSIFICATION

Given a set of relevant attribute candidates $\mathcal{A}_y$ representing the target schema, we define attribute schema matching to be the problem of selecting attributes from the target schema that best represent a given source attribute a_x for a given product category y. In the target schema, we have an index of relevant attribute candidates by product category produced in Sec. 4. On average, we produce roughly $|\mathcal{A}_y| = 30$ relevant attribute candidates for a given product category. This is far below the average number of attributes found on product categories, which ranges at approximately 400.

Manually schema matching attributes is laborious in our setting due to scale, with millions of possible source schemas matching to thousands of target attributes (see Sec. 6). At the same time, our problem space requires highly precise matching in order to maintain quality catalog data. We build two approaches to scale schema matching without compromising precision. Given the large number of source schemas, neither approach relies on training models specific to a given source.

In *attribute ranking*, our model produces schema matching suggestions to reduce the cost of human annotation in a semi-automated system. Our model output is an ordering over $\mathcal{A}_y$ for each source attribute, of which the top suggested attributes are considered for final matching. In *attribute classification*, we build models to directly classify common attributes (or attribute types) across product categories. In both scenarios, we account for challenges such as addressing noise or variability by building a common approach to normalize enumerations and extract phrases that are comparable across various schemas.

5.1 Ontology-based Normalization

One challenge specific to product attribute data is the relative lack of regular grammatical structure. This limits the potency of word embeddings that are primarily trained on natural language expressions. Word senses in the attribute value space tend to be skewed towards technical and abbreviated language. For example, in our setting the word *ton* is far more likely to refer to a unit of measure than a vernacular expression.

We also observe a large frequency of high-precision numerical enumerations. Small differences in numerical value are not captured well by traditional natural language embeddings, which tend to underestimate similarities in enumerations such as *5.4 inches* and *5.8 inches*. For our setting, differences between these two values are minimal as our objective is to capture overall attribute similarity.

To address both problems, we adopt a simple word ontology-based approach to normalize numeric enumerations with units. For identifying unit synonyms we leverage WordNet [12], a well-established lexical database that contains ontological word relationships as well as hand-crafted synonyms. Using WordNet, we construct a reverse index of synonyms for all accepted units in the catalog, relying on the ontology to restrict synonym sets to units of measure only (such as hyponyms of `definite_quantity.n.01`). This index of unit synonyms is also augmented with symbols for common units (such as " for inches).

Our normalization heuristic uses regex matching for detecting numeric enumerations followed by potential units. If unit synonyms are matched, then we normalize the unit to the canonical representation in the target catalog. We round numeric values to one significant figure before converting to word representations.

Finally, we observe large distributional differences in length and structure between source attribute enumerations and attribute enumerations in the catalog schema. This may arise due to varying strictness in schema datatypes, where for example the source schema may represent a given attribute as TEXT (or VARCHAR) but the corresponding attribute in the target is modeled with a compound datatype. We account for this by adopting our similarity metric to consider relevant subsequences in both domains, and during normalization we identify comparable phrases to be grouped together as n-grams (as in Sec. 4.3). For instance, we may measure similarity of phrases "`diagonal`" and "`corner to corner`".

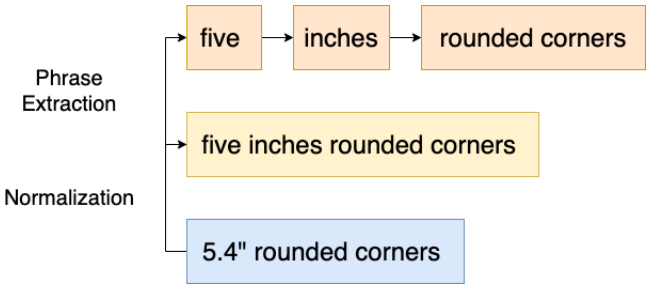

Figure 2: Ontology-based normalization.

5.2 Schema Matching as Ranking

While it is possible to estimate similarity on sets of phrases, we observe that taking mean similarity with a BOW approach discards structure that is important for characterizing enumerations. One such example is the product dimensions attribute, which is often depicted as an ordered tuple of numeric values (ex. "`9.9 in. x 6.8 in. x 0.3 in.`"). A robust similarity metric for matching needs to preserve such structure.

To address this, we introduce the *MCSS* metric (2) defined in Sec. 3.2 that models enumerations as ordered sequences. We choose the most relevant subsequences to score enumeration similarity, with the intuition that enumerations of the same attribute tend to have at least one highly similar ordered subsequence of phrases (for example, a numeric quantity followed by a unit of measure). This also makes MCSS robust to distributional differences across schemas (in length, additional metadata, etc.).

We schema match in an unsupervised manner by computing attribute similarity $\text{sim}_a(a_x, a_y)$ for a given source attribute a_x

and any candidate target attribute a_y generated in Sec. 4 based on the product category of a_x. Target attributes a_y that share a high similarity are surfaced as ranked suggestions. Taking such a schema-independent approach to schema matching allows our system to generalize to match new schemas rapidly. Rankings may be surfaced for human review for semi-automated matching, or used as in Sec. 5.3 as input to a fully automated schema matching system.

5.3 Attribute-Specific Classification

While ranking generalizes well to unseen source schemas and attributes, by itself it still necessitates some human judgment to select final schema matches. We therefore explore ways to remove the manual burden of matching attributes with models that classify with high precision across schemas and product classes.

Supervised classification of attributes is difficult in our setting where the total number of labelled matches is smaller than the number of possible attribute labels. At the same time, we observe that product attribute frequency follows a power-law distribution, as does the distribution of attribute enumerations. Leveraging this, we extend attribute similarity to construct simple binary classification heuristics for the most frequently occuring attributes.

For a given target attribute a_y, we construct a high-quality enumeration set $\mathcal{V}(a_y)$ by either selecting the most frequently occuring enumerations in the catalog (after normalization described in 5.1) or by leveraging domain experts to provide sample enumerations. We take this practical approach in order to reduce risks due to noise and boost precision at the cost of losing recall.

Given a restricted enumeration set $V(a_y)$, we then perform binary classification by simply thresholding on attribute similarity sim_a (4), where the threshold is a hyperparameter than can be optimized per-attribute during evaluation. Despite the simplicity of this approach for schema matching [16], we find that this method performs surprisingly well for common product attributes; meeting precision bars while classifying with reasonable recall.

5.4 Classifying by Attribute Datatype

An alternative approach to classification is to perform supervised learning based on *attribute datatype* in the catalog schema. This approach benefits from the number of attribute types being much lower than the number of distinct attributes, hence reducing size of the target label space. In our experiments, we train a binary classifier for attributes describing quantities (ex. number of pieces in a puzzle). We construct training data from historical schema matches by labelling source attributes with a 1 if they match to quantity attributes or 0 otherwise. Note that using MCSS to directly match such attributes is challenging due to the strictness of the target datatype that only accepts integers (and leaves room for false positives when comparing sequences such as "`iPhone 3`" and "`3`").

Instead, we construct features that concatenate mean embeddings for product category, name, and enumeration, along with positional features. These are then passed to a logistic regression model that learns to distinguish attributes of the quantity type.

This model is applied to classify attributes according to the following procedure that leverages the relevant attributes candidate generation described in Sec. 4 along with name similarity (2):

Require: t: name similarity threshold
for $a_x \in \mathcal{A}_x$ **do**
  if ClassifyQuantity(a_x) **then**
    $\mathcal{A}_y \leftarrow$ RelevantAttributes(a_x)[Type = Quantity]
    $a_y^* \leftarrow \arg\max_{a_y \in \mathcal{A}_y} \text{sim}_s(k(a_x), k(a_y))$
    Classify a_y^* **as** $\text{sim}_s(k(a_x), k(a_y^*)) > t$
  end if
end for

6 EXPERIMENTS AND CASE STUDIES

Datasets: We demonstrate the effectiveness of the proposed solutions on real-world eCommerce data. For attribute ranking and classification, our unlabelled data is comprised of 150 thousand attributes of various product categories that span over 14 thousand source schemas. We evaluate our algorithms on a set of 911 attributes from 698 source schemas that are matched to equivalent attributes in our target schema (totaling 235 unique target attributes). Each source attribute is associated with a set of value enumerations (one enumeration per product). There are roughly 7500 unique value enumerations in our dataset.

Evaluation Metrics: We evaluate our schema matching solution for ranking accuracy using **Mean Reciprocal Rank (MRR)**. MRR is a standard measure for ranking tasks where the number of relevant results is small. MRR is defined by

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} \tag{6}$$

where $rank_i$ is the minimum rank of the suggested attributes for the i-th attribute list and N is the total number of attributes in the evaluation data set. We also capture additional metrics that reflect the practical value of our algorithm within a semi-automated system where the top 5 suggestions are surfaced: **Accuracy@n** (defined to be the event that the best attribute appears in the top n suggestions) and **Mean Rank**. For the classification tasks, we use precision, recall, and sensitivity as our evaluation metrics.

6.1 Attribute ranking for schema matching

We compare our attribute similarity metric against two established models for scalable unsupervised schema matching. [16] compute a mean attribute vector as a frequency-weighted sum of the embeddings for words in the attribute text, and match attributes using cosine similarity of attribute vectors. In DeepAlignment [9], this Bag-of-Words approach is extended using a modification of the *Dual Embedding Space Model* [14] to compare attributes as sets of word vectors. Their model considers attribute distance $dis(Q, S)$ between two attribute word vector sets Q and S as:

$$dis(Q, S) = max(\delta(Q, S), \delta(S, Q)) \tag{7}$$

$$\delta(Q, S) = \frac{1}{|Q|} \sum_{q_i \in Q} d(q_i, \bar{S}) \tag{8}$$

$$\bar{S} = \frac{1}{|S|} \sum_{s_j \in S} \frac{s_j}{\|s_j\|} \tag{9}$$

where d is cosine distance between embeddings. By contrast, our approach accounts for order of tokens in attribute enumerations by modeling similarity over sequences.

The results are presented in Table 1. To scope the comparison to semantic similarity metrics only, we keep our embeddings and normalization logic constant across all three approaches below.

Table 1: Evaluation of proposed attribute similarity metric (MCSS) against comparison ranking measures.

Similarity Metric	MRR	Acc@5	Mean Rank
Mean Attribute Vector [16]	0.7316	0.8778	3.787
DeepAlignment [9]	0.7296	0.8352	5.063
MCSS Attribute Similarity	**0.8156**	**0.9147**	**2.801**

The results show that MCSS schema matches with a +8% increase in MRR compared to the best alternative method. For our semi-automated schema matching system which uses top-5 ranked attributes, we consider accuracy@5 as our operational metric. There the performance increase is non-trivial +3.7% compared to the second best approach on our dataset (attribute vector). MCSS suggests the best schema match nearly 91.5% of the time. Combined with a mean rank of 2.801, this suggests that the algorithm will perform suitably well for suggesting schema matches from even a large pool of candidates (roughly 30 on average in our dataset).

6.1.1 Ablation study. We test our method with different configurations to establish the proposed method provides consistent improvement over baselines. The three impact points we consider are 1) representation - pre-trained embeddings (FastText) versus WordNet-only similarity - 2) normalization - ontology-normalized numeric and unit values versus non-normalized - and 3) representation aggregation - sliding-window embeddings in MCSS versus mean embedding vector. Results are shown in Table 2.

Table 2: Effect of design considerations (1) representation, (2) normalization, and (3) representation aggregation.

	representation	normalization	aggregation
MRR	+0.11	+0.07	+0.09
Acc@5	+0.09	+0.06	+0.04
Rank	-2.6	-1.7	-1.0
	(FastText)	(Normalized)	(Sliding-window)

The comparative advantage of MCSS attribute similarity over comparison approaches such as mean embedding similarity is reflected across all three of our metrics. This provides validation for our approach in normalization, phrase extraction, and subsequence comparison. Through these stages, our model considers both semantics and structure of attribute names and enumerations that prove to be assets in performing schema matching across a diverse set of target attributes on a challenging dataset.

6.2 Attribute classification for schema matching

We demonstrate the effectiveness of our attribute classification approach that applies attribute similarity with thresholding to fully automate schema matching. Our analysis covers both models that classify individual attributes (for frequently occuring attributes) as well as models that classify attributes belonging to the same attribute type.

We evaluate the former approach with a model for classifying *color*, which is a common attribute that empirically represents challenges in our scope including high variance, noisy enumerations, and cross-attribute conflation challenges in many source schemas (ex. `"color": "dark wood"`). Our *color* model consists of less than 20 labelled high-quality enumerations $\mathcal{A}_y$ (independent of schema or product category) that is used during classification. A given source attribute is mapped to color if the MCSS attribute similarity to this set is higher than a tunable threshold that we vary as a hyperparameter.

We evaluate the latter on a classification model for types of numbered attributes referring to quantities (such as *item quantity*), another common family of attribute that varies greatly in verbosity and structure when represented across different schemas. First, we train a binary logistic regression on a dataset of 10 thousand historical attributes, applying a label of 1 if it is a numbered quantity attribute and 0 otherwise. Then, we use this classifier to restrict our candidate target attribute set during schema matching before applying thresholding in a similar manner described in (5.4).

In addition to precision and recall as standard classification metrics, we also consider specificity (true negative rate) in our scenario due to the large quantity of potential target attribute labels (n=238 in our evaluation set). Effectively, specificity measures the ability for such classification models to co-exist when deployed to perform schema matching in parallel (or in future ensemble approaches) when the number of output classes is high, imbalancing any specific attribute model. Results are shown in Table 3.

Table 3: Performance of classification models for schema matching

Model	Precision	Recall	Specificity
Color	0.9615	0.7143	0.9981
Quantity Attributes	0.9578	0.2966	0.9451

Results on our *color* classifier demonstrate high precision and specificity (> 95%) at a reasonable recall of 71%. The simplicity of the model paired with its minimal dependence on labelled training data shows that it can achieve good performance that can scale across new schemas and categories with minimal to no effort. In our attribute-type model for quantity attributes, we attain a similar precision and specificity that is nearly as high, but our recall is lower (at 30%). This is likely due to higher variability in both attribute name and enumeration, as over a hundred attributes fall under this type in our target schema. Nevertheless, in a fully-automated setting both classification models demonstrate promise in schema matching at scale that results in a significantly reduced need for human annotation, as we discuss in Section 6.4.

6.3 Relevant Attribute Candidates and Concept Matching

We demonstrated that the proposed schema matching approaches are effective at mapping across schemas with noisy data. Now, we exemplify qualitatively results of domain specific word embeddings, which are able to capture the association of tokens to concepts. We also demonstrate both qualitatively and quantitatively the results of the attribute ranking model.

6.3.1 Semantic Similarity. Table 4 shows a similarity matrix for phrases found under the Laptop category, where the word embeddings for those phrases have been learned from the category specific corpus. Note how both synonyms as well as non-synonyms relate to suitable concepts, e.g., `8gb` and `128gb` are semantically similar to `memory`, though `8gb` is more closely related to `RAM` and `128Gb` is more closely related to `storage`.

Table 4: Similarity matrix of test phrases for laptops.

	ram	memory	storage	CPU	processor
ram	1				
memory	.83	1			
storage	.47	.65	1		
cpu	.43	.40	.28	1	
processor	.51	.49	.34	.71	1
graphics_card	.35	.36	.22	.33	.49
8gb	.66	.62	.38	.28	.36
128gb	.37	.45	.60	.17	.23

To demonstrate the domain-specific nature of the learned embeddings across different domains, we present results of clustered tokens corresponding to attributes in different categories. We show "operating system" for two different domains (product categories): "Laptops" and "Mattress" in Table 5. While "operating system" is not applicable to Mattresses, note the suitably of each token to the attribute concept as learned from the corpus.

Table 5: Clusters of tokens for Laptops and Mattresses.

Laptop Graphics Card	Laptop Operating System	Mattress Operating System
video_card	os	operate
geforce_gtx_980	operate_system	system
gpu	windows_10	electronic
integrated_gpu	op_system	battery
nvidia	windows_8	control
750_ti	os_x	mechanism
ge_force	chrome_os	cord
...	...	...

6.3.2 Relevant Attributes. Table 6 shows the computed relevance ranking of attributes for Televisions. For brevity, we only show the overall rank (used to drive Catalog Enrichment) and the discovery rank. Additionally, we show examples of attributes that are detected as redundant (using attribute similarity to self-consolidate a noisy schema). It is easy to see qualitatively how the top ranked attributes are relevant for making product purchasing decisions.

Table 6: Relevance Ranked attributes for Televisions.

Attribute	Overall Rank	Discovery Rank
Brand	1	1
Resolution	2	3
Connection Technology	3	6
Display Technology	4	5
Size	5	2
Supported Apps	6	4
Contrast Ratio	7	18
Number of HDMI Ports	8	51
Refresh Rate	9	22
Speaker	10	26
...	...	...
Hardware Interface	redundant	redundant
Internet Services	redundant	redundant
Frequency	redundant	redundant
...	...	...

6.3.3 Relevant Attributes: Quantitative Metric. To quantitatively evaluate the overall ranking model, we collect ground truth data from humans. Humans were tasked to research product categories and identify what are the relevant attributes they consider important for searching or making product purchasing decisions (a binary decision, not a ranking). While this process may be subjective, the results are aggregated over multiple raters for each category. Relevance labeling was performed for 100 product categories. Table 7 shows the results, including precision recall of our threshold-based relevant attributes identification model (from Sec. 4.1).

Table 7: Results for Classification of Relevance Attributes for 100 Product Categories.

Metric	Value
Average # of Attributes per Category:	426
Average # of Relevant Attributes per Category:	30
Precision:	67%
Recall:	83%

6.4 Effect on manual labelling efforts

Empirical data collected over experiments in the last year demonstrates the promise of our schema matching systems in increasing productivity and scale of catalog enrichment. To estimate impact at scale, we run our experiments on a significant dataset of 50 million attribute enumerations from 150 thousand customer-relevant attributes across 14 thousand schemas and product categories.

We conducted an experiment using unsupervised ranking algorithms for schema matching in a semi-automated prototype to provide attribute suggestions to manual annotators. As none of

our approaches require labelling data per additional schema, product category, or attribute, our approach is an effective solution for the **cold start problem** of understanding new schemas rapidly. Through benchmarking, we estimated a reduction of 75% in the manual effort required to enrich and backfill a new product schema end-to-end, saving many hours per schema compared to our existing approach.

We also conducted experiments exploring the feasibility of using our classifiers in a fully-automated system that schema matches common attributes or attribute types with minimal need for additional labelling of data (being able to leverage existing attribute enumerations). Collectively, we estimate our impact in automating up to 15% of schema matches to all targeted attributes of high customer interest.

7 FUTURE WORK

Our experiments point to the promise of learning to classify attributes or attribute types in a few-shot setting. Future explorations in this domain may involve extending our similarity metrics with meta-learning approaches such as Induction Networks [6] or multi-task learning to improve embeddings through transfer, as with the Universal Sentence Encoder [4].

Regarding the adaptation of embeddings to the ontology or schema setting, we find the counter-fitting method (with antonym repel and synonym attract) explored in [9] to be a principled strategy that would pair well with our approach of adapting category-specific word vectors.

8 CONCLUSION

In this work, we introduced a novel attribute-based scalable schema matching system for highly heterogeneous large-scale eCommerce data (*Targeted Catalog Enrichment*). Our method identifies context-specific customer relevant attributes and prioritizes them by consolidating product data from a multitude of manufacturers and vendors. Given that our proposed method can be trained with little to no training data, it scales easily and can be applied to new domain schemas or attributes. Qualitative evaluations also verify the attribute relevancy method retains domain-specific meanings. We show that attribute similarity defined in this way can be easily extended to create simple thresholding-based classifiers that schema match with high precision and reasonable recall.

Across a wide-range of product categories and over a year's worth of real-world eCommerce data, we show that Targeted Catalog Enrichment significantly outperforms two other baseline methods and reduces the burden of manual labelling efforts. Overall, our results demonstrate the importance of considering both relevance semantics and attribute structure in designing automating schema matching to achieve scalable impact in enriching large catalog schemas.

REFERENCES

[1] Nils Barlaug and Jon Atle Gulla. 2020. Neural Networks for Entity Matching. arXiv:2010.11075 [cs.DB]

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2016. Enriching Word Vectors with Subword Information. *CoRR* abs/1607.04606 (2016). arXiv:1607.04606 http://arxiv.org/abs/1607.04606

[3] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures - a step forward in data integration. In *Proceedings of the 23rd International Conference on Extending Database Technology*. OpenProceedings.

[4] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 169–174.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[6] Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction Networks for Few-Shot Text Classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3895–3904.

[7] Christopher Ireland, David Bowers, Michael Newton, and Kevin Waugh. 2009. A Classification of Object-Relational Impedance Mismatch. In *Proceedings of the 2009 First International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA '09)*. IEEE Computer Society, USA, 36–43. https://doi.org/10.1109/DBKDA.2009.11

[8] Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics* 4 (2016), 313–327. https://doi.org/10.1162/tacl_a_00101 arXiv:https://doi.org/10.1162/tacl_a_00101

[9] Prodromos Kolyvakis, Alexandros Kalousis, and Dimitris Kiritsis. 2018. DeepAlignment: Unsupervised Ontology Matching with Refined Word Vectors. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 787–798. https://doi.org/10.18653/v1/N18-1072

[10] Wen-Syan Li and Chris Clifton. 2000. SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks. *Data Knowl. Eng.* 33, 1 (April 2000), 49–84.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781

[12] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. https://doi.org/10.1145/219717.219748

[13] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández. 2000. Schema Mapping as Query Discovery. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 77–88.

[14] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A Dual Embedding Space Model for Document Ranking. https://www.microsoft.com/en-us/research/publication/a-dual-embedding-space-model-for-document-ranking/

[15] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. Deep Sequence-to-Sequence Entity Matching for Heterogeneous Entity Resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 629–638. https://doi.org/10.1145/3357384.3358018

[16] K. Nozaki, T. Hochin, and H. Nomiya. 2019. Semantic Schema Matching for String Attribute with Word Vectors. In *2019 6th International Conference on Computational Science/Intelligence and Applied Informatics (CSII)*. 25–30. https://doi.org/10.1109/CSII.2019.00012

[17] Erhard Rahm and Philip A. Bernstein. 2001. A Survey of Approaches to Automatic Schema Matching. *The VLDB Journal* 10, 4 (Dec. 2001), 334–350. https://doi.org/10.1007/s007780100057

[18] Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 440–450.

[19] Ning Wang, Minnan Luo, Kaize Ding, Lingling Zhang, Jundong Li, and Qinghua Zheng. 2020. Graph Few-Shot Learning with Attribute Matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 1545–1554. https://doi.org/10.1145/3340531.3411923

[20] Junming Zhang, Jinglin Li, Shangguang Wang, and Jiali Bian. 2014. A Neural Network Based Schema Matching Method for Web Service Matching. In *Proceedings of the 2014 IEEE International Conference on Services Computing (SCC '14)*. IEEE Computer Society, USA, 448–455. https://doi.org/10.1109/SCC.2014.66

[21] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. OpenTag: Open Attribute Value Extraction from Product Profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1049–1058.