Aligning Product Categories using Anchor Products

Varun R Embar
University of California, Santa Cruz
vembar@ucsc.edu

Golnoosh Farnadi
University of California, Santa Cruz
gfarnadi@ucsc.edu

Jay Pujara
University of Southern California
jay@cs.umd.edu

Lise Getoor
University of California, Santa Cruz
getoor@soe.ucsc.edu

ABSTRACT

E-commerce sites group similar products into categories, and these categories are further organized in a taxonomy. Since different sites have different products and cater to a variety of shoppers, the taxonomies differ both in the categorization of products and the textual representation used for these categories. In this paper, we propose a technique to align categories across sites, which is useful information to have in product graphs. We use *breadcrumbs* present on the product pages to infer a site's taxonomy. We generate a list of candidate category pairs for alignment using *anchor products* - products present in two or more sites. We use multiple similarity and distance metrics to compare these candidates. To generate the final set of alignments, we propose a model that combines these metrics with a set of structural constraints. The model is based on probabilistic soft logic (PSL), a scalable probabilistic programming framework. We run experiments on data extracted from Amazon, Ebay, Staples and Target, and show that the distance metric based on products, and the use of PSL to combine various metrics and structural constraints lead to improved alignments.

1 INTRODUCTION

The widespread adoption of e-commerce sites such as Amazon and Ebay has led to the emergence of product search sites such as Google Shopping[1] and Bing Shopping[2]. These sites allow users to search for products, compare prices across e-commerce sites, and recommend related products. In order to provide these services, search sites often make use of knowledge graphs of products called *product graphs*.

Subsumption relationships among product categories are an important component of product graphs. One way to infer these relationships is by using *breadcrumbs* present on the product pages. Breadcrumbs show the path in an e-commerce site's taxonomy, from the root to the parent category of a product. In this paper, we represent breadcrumbs as a list of categories separated by the character '>'. For example, the page for `Apple watch` on Amazon has the following breadcrumb:

`Electronics` > `Wearable Technology` > `Smart Watches`

From the above breadcrumb, we can infer that "`Smart Watches`" are a type of "`Wearable Technology`", which is itself a type of "`Electronics`".

[1] https://www.google.com/shopping
[2] www.bing.com/shop

KBCOM'18, Feb 2018, Los Angeles, California USA
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Since each e-commerce site offers a different set of products and caters to a variety of shoppers, sites' product taxonomies differ both in categorization of products and in their textual representation. As a result, there are multiple textual representation for the same category across breadcrumbs from different sites. For instance, products related to 3D printing are present in a category called "`3D Printing & Supplies`" on Ebay, while the same products are present in a category called "`Additive Manufacturing Products`" on Amazon. Another challenge is the use of same textual representation to to denote multiple categories. The textual representation "`Accessories`", for e.g., occurs in more than 25 contexts on Amazon, as a child category of strollers, video games, car seats and so on.

Thus, in addition to incorporating subsumption relationships from breadcrumbs extracted from various sites, it is useful to add information about similar categories across sites' taxonomies. This problem can also be viewed as aligning similar nodes across product taxonomies.

One of the early approaches to product taxonomy alignment was proposed by Park and Kim[7]. In their work, they find a set of synonyms for each category using Wordnet, and align categories using two measures computed on the paths in the taxonomy. Following this, many other approaches have been proposed to align product taxonomies [1, 2]. All of these approaches make use of Wordnet [5] hierarchies to overcome the challenges of synonyms and polysemy. While this technique helps in the case of categories such as "`notebooks`" and "`televisions`", it fails to find a match for categories such as "`playstation`" and "`xbox`", which are not found in Wordnet. Another drawback of the existing techniques is that they search through all possible categories in a taxonomy to find an alignment. This is expensive and cannot scale to large taxonomies such as Amazon, which have tens of thousands of categories.

In contrast, our proposed technique makes use of products present under each category to help in alignment. This is similar to instance-based matching approaches used in the closely related problem of ontology matching [4, 8]. We use products in two ways. First, we use them to prune the search space by generating a set of *anchor products*. This is similar to the concept of anchors proposed by Noy and Musem[6] in the context of ontology alignment. Anchor products are products that are have two or more breadcrumbs, from different sites, associated with them. Given the set of anchor products along with associated breadcrumbs, we only consider categories that appear in these breadcrumbs for possible alignment. This results in a significant reduction in the search space. Second, we use products to compute a distance metric between categories. The metric uses tf-idf vectors generated from product titles present

under each category. Our experiments show that this improves the performance of the alignments.

We use the probabilistic soft logic (PSL)[3] framework to jointly align all categories across taxonomies, combining multiple distance and similarity metrics, and structural constraints to find consistent alignments.

In this paper, we propose an approach that aligns similar product categories in breadcrumbs extracted from product pages. Since the space of possible alignments is large, we use anchor products to prune the search space. We propose multiple similarity and distance metrics to align the candidate pairs. To fuse these metrics along with various structural constraints, we use PSL. To evaluate the effectiveness of various components of our approach, we perform an ablation study on data extracted from Amazon, Ebay, Staples and Target.

2 SIMILARITY & DISTANCE METRICS

In this section, we describe various similarities and distance metrics that we use in our approach. First, we present our approach for generating candidate category pairs for alignment using anchor products in Section 2.1. Next, in Section 2.2 we present a similarity metrics based on the position of categories in the breadcrumbs. In Section 2.3 and Section 2.4, we introduce two distance metrics based on the textual representation of product categories, and products present in each category.

2.1 Anchor Products

Anchor products are products that are on sale on multiple sites. Given a set of products on sale across various e-commerce sites, we define a *matching function* to find anchor products. The matching function could look for product IDs such as Universal Product Code(UPC) and International Standard Book Number(ISBN). In this paper, we use a simple approach where we consider two products to be the same only if their titles match exactly. Once we have the set of anchor products, we create the set of candidate alignment pairs by generating all possible pairs of categories that occur in the breadcrumbs. For example, the product `lego creator corner deli 31050` has the following breadcrumbs:

Amazon.com – [toys & games; building toys]

Target.com – [toys; building sets & blocks; building sets & kits]

From this anchor product, we generate the following 6 candidate category pairs where the first category is from Amazon and the second one is from Target: [(toys & games, toys), (toys & games, building sets & blocks), (toys & games, building sets & kits), (building toys, toys), (building toys, building sets & blocks), (building toys, building sets & kits)]

2.2 Breadcrumb Similarity

In the above example of `lego creator corner deli 31050`, the correct alignment for the category "`building toys`" is "`building sets & kits`". Since both "`building toys`" and "`building sets & kits`" are present at lower positions, we could infer the alignment based on the position of the categories in the breadcrumb. We call this breadcrumb similarity. However, since the breadcrumbs often have different lengths, and taxonomies differ in their granularity, a simple one-to-one matching strategy does not work. To overcome this, for each anchor product, we assign indices to the categories starting with 0 for the category at the bottom of the breadcrumb, and assign higher indices as we go up. Given the indices, the per-product anchor similarity score for a candidate category pair is given by:

$$PerProductBreadcrumbSim(C_1, C_2) = max(0, 1 - \alpha * |i - j|) \quad (1)$$

where i and j are the indices of categories C_1 and C_2 in anchor product p. α is a hyperparameter that varies the similarity of categories at different depths.
In the example, the pair (`building toys`, `building sets & kits`) has a score of 1 and the pair (`building toys`, `toys`) has a score of 0.8 ($\alpha = 0.1$).

The overall breadcrumb similarity is given by taking the average of all per-product breadcrumb similarity scores.

$$BreadcrumbSim(C_1, C_2) = \frac{\sum_{p \in A} PerProductBreadcrumbSim(C_1, C_2)}{\sum_{p \in A} 1}$$

where p iterates over all anchor products that contain categories C_1 and C_2.

2.3 Name Distance

Name distance is based on the textual representation of categories. One of the ways to compute this could be to use Jaccard Similarity on the tokens in the textual representation. However, to take into account the rarity of occurrence of tokens, we propose the following approach. We first tokenize the category names and compute the idf scores. We then represent each category by an idf vector computed from the tokens in textual representation of the category and normalize the 2-norm. The name distance between two categories is given by:

$$NameDist(C_1, C_2) = 1 - {v_1}^T v_2 \quad (2)$$

where C_1, C_2 are two categories and v_1, v_2 are their category vectors.

2.4 Product Distance

Intuitively, categories that have very different products under them should not be aligned. We formalize this notion by computing a distance metric called product distance based on the products present in each category. As in the case of name distance, we use a tf-idf based score. We first tokenize the product titles and compute the idf scores for each token in the dataset. We then represent each product by a tf-idf vector computed from its title. We generate the category vector by summing all the product vectors that belong to that category and normalize its 2-norm.

The product distance between two categories is given by:

$$ProdDist(C_1, C_2) = 1 - {v_1}^T v_2$$

where C_1, C_2 are two categories and v_1, v_2 are their category vectors.

3 ALIGNING PRODUCT CATEGORIES

In this section, we present our proposed framework in which we combine different metrics introduced in Section 2, and strucutral

constraints using PSL. We first review PSL in Section 3.1 and then introduce our proposed PSL model in Section 3.2.

3.1 Probabilistic Soft Logic

PSL is a statistical-relational learning framework for defining hinge-loss markov random fields (HL-MRFs), a class of undirected probabilistic graphical models, that supports modeling of relational data. A PSL model consists of weighted logical clauses that encode statistical dependencies and structural constraints. Given a set of random variables, some of which are observed, a PSL model defines a probability distribution over the unobserved variables. PSL supports efficient maximum a posteriori (MAP) inference.

A PSL model consists of a set of rules. For instance the rule:

$$\lambda : \textsc{Child}(C_1, C_2) \wedge \textsc{Align}(C_1, C_3) \implies \neg\textsc{Align}(C_2, C_3)$$

is a PSL rule which suggests that the categories (C_2, C_3) are unlikely to be aligned if categories (C_1, C_3) are aligned and $C1$ is the child of $C2$ in the site's taxonomy. Observe that this rule makes the random variables corresponding to $\textsc{Align}(C_1, C_3)$ and $\textsc{Align}(C_2, C_3)$ dependent, and during inference they are inferred jointly. The rule is weighted by a weight λ, that could either be set by the user, using his domain knowledge, or can be learned from the data, and denotes the importance of the rule.

3.2 PSL Model

In this section, we discuss the various rules that are present in our PSL model.

3.2.1 Similarity & Distance Rules. We combine the various similarity and distance metrics and compute a combined similarity between two categories. The combined similarity between two categories is given by the predicate $\textsc{Similar}(C_1, C_2)$. We add the following rules to the PSL model to infer the value of $\textsc{Similar}(C_1, C_2)$.

$$\lambda_{AncSim} : \textsc{BreadcrumbSim}(C_1, C_2) \implies \textsc{Similar}(C_1, C_2)$$
$$\lambda_{NameDist} : \textsc{NameDist}(C_1, C_2) \implies \neg\textsc{Similar}(C_1, C_2)$$
$$\lambda_{ProdDist} : \textsc{ProdDist}(C_1, C_2) \implies \neg\textsc{Similar}(C_1, C_2)$$

The above rule assigns higher value to Similar if the BreadcrumbSim is high and NameDist and ProdDist is low.

3.2.2 Alignment Rules. To infer if two categories should be aligned, we define a predicate Align, and add the following rules to the PSL model.

$$\lambda_{Align} : \textsc{Align}(C_1, C_2) \implies \textsc{Similar}(C_1, C_2)$$
$$\lambda_{Align} : \textsc{Similar}(C_1, C_2) \implies \textsc{Align}(C_1, C_2)$$

The rules state that categories with high similarity should align, and categories that align must have high similarity.

3.2.3 Structural Rules. Structural rules consider the structure of the taxonomy and create dependencies between various inferred alignments. As a result, the alignments are jointly inferred.

We represent the paths in the site's taxonomy using the predicate Child. $\textsc{Child}(C_1, C_2)$ is set to 1 if the category C_1 is a child of C_2 in the site's taxonomy and set to 0 otherwise.

One of the issues in aligning product categories is that, often, the parent and the child categories have similar names and similar products. As a result, it is hard to find the right level of generality in aligning categories. For example, consider the category "books" and "children's books" from Amazon, and "books" and "kid's books" from Target. All four categories have the word *books* in them. There is also some overlap in the products as all of them deal with books. The model might propose, based on the previous metrics, to have all possible combinations of alignments, i.e, "books" and "children's books" from Amazon aligned to both "books" and "kids books" from Target. In order to bias the alignments to pick one of the two categories, we add the following rules to the PSL model.

$$\lambda_{Exclusive} : \textsc{Child}(C_1, C_2) \wedge \textsc{Align}(C_1, C_3) \implies \neg\textsc{Align}(C_2, C_3)$$
$$\lambda_{Exclusive} : \textsc{Child}(C_1, C_2) \wedge \textsc{Align}(C_2, C_3) \implies \neg\textsc{Align}(C_1, C_3)$$

Consider the categories "baby products" from Amazon and "baby" from Target. The category "baby products" has a child node "baby monitors" and the the category "baby" has a child node "monitors". If we align the categories "baby" and "baby products", it is likely that their children "monitors" and "baby monitors" should also be aligned. However, another child of "baby" is "diapering", which is not related to "baby monitors" and should not be aligned. To encourage similar children of aligned categories to align, we add the following rules to the PSL model:

$$\lambda_{Child} : \textsc{Child}(C_1, C_2) \wedge \textsc{Child}(C_3, C_4) \wedge \textsc{Align}(C_2, C_4) \wedge \textsc{Similar}(C_1, C_3) \implies \textsc{Align}(C_1, C_3)$$
$$\lambda_{Child} : \textsc{Child}(C_1, C_2) \wedge \textsc{Child}(C_3, C_4) \wedge \textsc{Align}(C_2, C_4) \wedge \neg\textsc{Similar}(C_1, C_3) \implies \neg\textsc{Align}(C_1, C_3)$$

When there is no evidence to suggest alignment of two categories, we should not align the categories. This is enforced by the prior rules.

$$\lambda_{Prior} : \neg\textsc{Align}(C_1, C_2)$$
$$\lambda_{Prior} : \neg\textsc{Similar}(C_1, C_2)$$

4 EXPERIMENTS

We performed our experiments on data extracted from four websites - Amazon, Ebay, Staples and Target. We used the Diffbot[3] KB Search API to get the set of products and their breadcrumbs. Certain textual representations such as "accessories", "safety" occur multiple times in the taxonomies representing different categories. A few of the contexts in which the word "accessories" occurs are:

`cell phones & accessories > accessories`

`video games > nintendo ds > accessories`

`clothing shoes & jewelry > men > accessories`

`baby products > strollers & accessories > accessories`

To distinguish between categories with different ancestors, we represent them as $\texttt{accessories}_1$, $\texttt{accessories}_2$ and so on. We split the data into train and test, where the train split contains products and categories related to electronics and the test split contains all other products and categories. Table 1 presents the number of products and categories across splits for each site.

[3] https://diffbot.com

Site	Train		Test	
	# of prod	# of cat	# of prod	# of cat
Amazon	44919	801	441677	18853
Ebay	2385	870	58526	21024
Staples	8971	111	32912	1427
Target	2188	184	32846	2315

Table 1: Data Statistics

We generated anchor products from this data where we considered two products to be the same if their titles were same. We found 389 anchor products in the training data and 3269 anchor products in the test. From these anchor products we generated the set of candidate alignment pairs. There were 1022 pairs in the training data and 9446 pairs in the test set. We then computed various similarity and distance metrics for these pairs.

For the baseline, we considered all category pairs where the $NameDist$ is below a threshold. We call this approach $Name_\theta$, where θ is the threshold. In our experiments we set $\theta = 0.8$.

We ran the alignment process using PSL models consisting of all similarity and distance rules mentioned in 3.2.1 (PSL_{Sim}), and the full PSL model consisting of all similarity and structural rules (PSL_{Full}). In our experiments we used the following parameters - $\lambda_{AncSim} = 25$, $\lambda_{NameDist} = 25$, $\lambda_{ProdDist} = 35$, $\lambda_{Align} = 50$, $\lambda_{Exclusive} = 35$, $\lambda_{Child} = 20$, $\lambda_{Prior} = 2$, $\alpha = 0.1$.

After the PSL models generated truth values for the candidate category pairs, we labeled all pairs with truth value greater than 0.5 as either correct or incorrect. In Table 2 we give the number of suggested alignments, number of correct alignments, and precision for all three models.

	Suggested	Correct	Precision	Recall*	F1*
$Name_{0.8}$	621	242	0.389	0.98	0.558
PSL_{Sim}	492	227	0.461	0.922	0.615
PSL_{Full}	393	206	0.524	0.837	0.644

Table 2: Alignment Metrics - Recall* and F1* are recall and F1 computed using the correct pairs present in the union of alignments suggested by all the models

We first observed that the number of suggested alignments decreases as we go from $Name_{0.8}$ to PSL_{Full}. We also observed that the alignments suggested by PSL_{Sim} were a superset of alignments suggested by PSL_{Full}. This is because most of the additional rules have ¬Align and ¬Similarity in their head which reduce the truth values of alignments below the threshold. However, we observe that the precision improves as we add more rules. This suggests that most of the alignments that are removed are incorrect alignments.

Since the entire set of correct alignments in the dataset is large and difficult to get, we cannot compute recall for all models. Instead, we compute the recall and f1 score for the models using the set of all correct alignments present in the union of alignments suggested by all three models. We observe, from Table 2, that the f1 score for PSL_{Full} is higher than that of PSL_{Sim} which in turn higher than that of $Name_{0.8}$.

We briefly explain our hypothesis for the above finding. Unlike $Name_{0.8}$, PSL_{Sim} considers the products under each category. As a result, it is able to distinguish between categories that have similar words in their textual representation but mean slightly different things. For example, while $Name_{0.8}$ suggests alignment between "`bicycle stands & storage`" and "`storage & home organization`", due the common word *storage*, PSL_{Sim} does not. As a result PSL_{Sim} has better metrics than $Name_{0.8}$.

PSL_{Full} is able to correctly align the categories to the right generality due the structural rules. For example, while PSL_{Sim} aligns "`beauty & personal care`" to both "`beauty`" and "`hair care`", PSL_{Full} suggests aligning it to only "`beauty`" as "`hair care`" is a child of "`beauty`". As a result PSL_{Full} performs better than PSL_{Sim}.

We also observed that quite few of the ambiguous categories such as "accessories" were correctly aligned. For example, "$accessories_{21}$" of Amazon, which occurs in the context of "`baby products`" > "`car seats & accessories`" > "`accessories`", was correctly aligned to "`car seat accessories`" of Target. Similarly "$\texttt{training aids}_2$" of Ebay, which occurs in the context of "`sporting goods`" > "`team sports`" > "`soccer`" > "`training aids`", was correctly aligned to "$\texttt{training equipment}_5$" of Amazon, which has the following context "`sports & outdoors`" > "`sports & fitness`" > "`team sports`" > "`soccer`" >"`training equipment`".

5 CONCLUSION & FUTURE WORK

In this paper we have proposed a method to align similar categories present in breadcrumbs extracted from various e-commerce sites. We make use of products present under each category to generate a set of anchor products which in turn reduces the search space for possible alignments. We also make use of products to generate a distance metric that improves the accuracy of the alignment. We use PSL to combine various similarity and distance metrics, and to also incorporate various structural features. In our experiments, we show that this increases the performance of alignments.

In our future work, we plan to explore better matching functions that can generate a larger set of anchor products. Another direction is to explore other similarity and distance metrics that can improve the performance this technique. Finally, we plan to run the alignment technique on a larger dataset involving a large number of e-commerce websites, and provide a more comprehensive evaluation of recall.

REFERENCES

[1] Steven S Aanen, Lennart J Nederstigt, Damir Vandić, and Flavius Frăsincar. 2012. SCHEMA-an algorithm for automated product taxonomy mapping in e-commerce. In *Extended Semantic Web Conference*. Springer, 300–314.

[2] Steven S Aanen, Damir Vandic, and Flavius Frasincar. 2015. Automated product taxonomy mapping in an e-commerce environment. *Expert Systems with Applications* 42, 3 (2015), 1298–1313.

[3] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-Loss Markov Random Fields and Probabilistic Soft Logic. *Journal of Machine Learning Research (JMLR)* 18 (2017), 1–67.

[4] Sabine Massmann and Erhard Rahm. 2008. Evaluating Instance-based Matching of Web Directories.. In *WebDB*.

[5] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[6] Natalya F Noy and Mark A Musen. [n. d.]. Anchor-PROMPT: Using non-local context for semantic matching.

[7] Sangun Park and Wooju Kim. 2007. Ontology mapping between heterogeneous product taxonomies in an electronic commerce environment. *International Journal of Electronic Commerce* 12, 2 (2007), 69–87.

[8] Andreas Thor, Toralf Kirsten, and Erhard Rahm. 2007. Instance-based matching of hierarchical ontologies.. In *BTW*, Vol. 103. 436–448.