Enriching Product Ads with Metadata from HTML Annotations

Petar Ristoski[1], Peter Mika[2]

[1] Data and Web Science Group, University of Mannheim, Germany
petar.ristoski@informatik.uni-mannheim.de
[2] Yahoo Labs, London, UK
pmika@yahoo-inc.com

Abstract. Product ads are a popular form of search advertizing offered by major search engines, including Yahoo, Google and Bing. Unlike traditional search ads, product ads include structured product specifications, which allow search engine providers to perform better keyword-based ad retrieval. However, the level of completeness of the product specifications varies and strongly influences the performance of ad retrieval.
On the other hand, online shops are increasing adopting semantic markup languages such as Microformats, RDFa and Microdata, to annotate their content, making large amounts of product description data publicly available. In this paper, we present an approach for enriching product ads with structured data extracted from thousands of online shops offering Microdata annotations. In our approach we use structured product ads as supervision for training feature extraction models able to extract attribute-value pairs from unstructured product descriptions. We use these features to identify matching products across different online shops and enrich product ads with the extracted data. Our evaluation on three product categories related to electronics show promising results in terms of enriching product ads with useful product data.

Keywords: Microdata, schema.org, Data Integration, Product Data

1 Introduction

Product ads are a popular form of search advertizing[3] that are increasingly used as a replacement for text-based search ads, and are currently offered as an option by Bing, Google and Yahoo under different trade names. Unlike traditional search ads that carry only a title, link and a description, product ads are more structured. They often include further details such as the product identifier, brand, model for electronics, or gender for clothing. These details are provided as part of data feeds that merchants transmit to the search engine, and they allow search engine providers to perform better keyword-based ad retrieval, and to offer additional options such as faceted search over a set of product results.

[3] Search advertising is a method of placing online advertisements on web pages that show results from search engine queries.

The level of completeness of the product specification, however, depends on the completeness of the advertisers' own data, their level of technical sophistication in creating data feeds and/or willingness to provide additional information to the search engine provider beyond the minimally required set of attributes. As a result of this, product ads are often very incomplete when it comes to the details of the product on offer.

In this paper, we address this problem by enriching product ads with structured data extracted from HTML pages that contain semantic annotations. Structured data in HTML pages is becoming more commonplace and it obviates the need for costly information extraction. In particular, annotations using vocabularies such as schema.org (an initiative sponsored by Bing, Google, Yahoo, and Yandex) and Facebook's OGP (Open Graph Protocol) are increasingly popular. To our knowledge, ours is the first work to investigate the potential of this data for enriching product ads, and to provide a targeted solution for matching product ads to product descriptions on the Web in order to exploit this data. In this work we focus on data annotated with the Microdata markup format using the schema.org vocabulary. Recent works [10, 11] have shown that the Microdata format is the most commonly used markup format, with highest domain and entity coverage. Also, schema.org is the most frequently used vocabulary to describe products.

Our method relies on a combination of highly accurate Information Extraction from unstructured text (titles and descriptions of products) and efficient and effective instance matching (also called reconciliation or duplicate detection) between product descriptions. More precisely, we use the structured product specifications in Yahoo's Gemini Product Ads as a supervision to build two feature extraction models, i.e., dictionary-based model and Conditional Random Field tagger, able to extract attribute-value pairs from unstructured text. Later, we use these features to build machine learning models able to identify matching products. An evaluation on three categories related to electronics shows that we are able to identify matching products across thousands of online shops with high precision and extract valuable structured data for enriching product ads.

The rest of this paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we formally define the problem of enriching product ads and we introduce our methodology. In Section 4, we present the results of matching unstructured product descriptions, followed by the results of the product ads enriching in Section 5. In Section 6, we adapt the proposed methodology for the task of product categorization. We conclude with a summary and an outlook on future work.

2 Related Work

While the task of enriching product ads with features from HTML annotations hasn't been studied so far, the problem of products matching and integration on the Web has been extensively studied in the recent years.

The approach presented by Ghani et al. [6] is the first effort for enriching product databases with attribute-value pairs extracted from product descriptions

on the Web. The approach uses Naive Bayes in combination with semi-supervised co-EM algorithm to extract attribute-value pairs from text. An evaluation on apparel products shows promising results, however the system is able to extract attribute-value pairs only if both the attribute and the value appear in the text.

One of the closest works is the work by Kannan et al. [8]. The approach uses a database of structured product records to build a dictionary-based feature extraction model. Later, the features of the products are used to train Logistic Regression model for matching product offers. The approach has been used for matching offers received by Bing shopping data to the Bing product catalog.

The *XploreProducts.com* platform [16] is the first effort to integrate products from different online shops annotated using RDFa annotations. The approach is based on several string similarity functions for product matching. Once the matching products are identified, the system integrates the available ratings, offers and reviews into one system. The system is evaluated on an almost balanced set of 600 electronics product combinations. However, in real applications the problem of products matching is highly imbalanced. The approach is first extended in [1], using a hybrid similarity method. Later, the method is extended in [2], where hierarchical clustering is used for matching products from multiple web shops, using the same hybrid similarity method.

Similar to our CRF feature extraction approach, the authors in [9] propose an approach for annotating products descriptions based on a sequence BIO tagging model, following an NLP text chunking process. Specifically, the authors train a linear-chain conditional random field model on a manually annotated training dataset, to identify only 8 general classes of terms. However, the approach is not able to extract explicit attribute-value pairs.

The first approach to perform products matching on Microdata annotation is presented in [13]. The approach is based on the Silk rule learning framework [7], which is able to identify matching products based on their attributes. To do so, different combination of features from the product descriptions are used, e.g., bag of words, attribute-value pairs extracted using a dictionary, features extracted using manually written regular expressions, and combination of all. The work has been extended in [14], where the authors developed a genetic algorithm for learning regular expressions for extracting attribute-value pairs from products.

While there are several approaches concerned with products data categorization [12, 8, 15, 13, 16], the approach by Meusel at el. [11] is the most recent approach for exploiting Microdata annotations for categorization of products data. In this approach the authors exploit the already assigned *s:Category* property to develop distantly supervised approaches to map the products to set of target categories from an existing product catalog.

3 Approach

3.1 Problem Statement

We have a database A of structured product ads and a dataset of unstructured product descriptions P extracted from the Web. Every record $a \in A$ consist of

title, description, URL, and a set of attribute-value pairs extracted from the title of the ad, where the attributes are numeric, categorical or free-text attributes. Every record $p \in P$ consist of title and description as unstructured textual fields. Our objective is to use the structured information from the product ads set A as supervision for identifying duplicate records in P, or matching products from P to one or more structured ads in A. More precisely, we use the structured information as a supervision for building a feature extraction model able to extract attribute-value pairs from the unstructured product descriptions in P. After the feature extraction model is applied, each product $p \in P$ is represented as a vector of attributes $F_p = \{f_1, f_2, ..., f_n\}$, where the attributes are numerical or categorical. Then we use the attribute vectors to build a machine learning model able to identify matching products. To train the model we manually label a small training set of matching and non-matching unstructured product offers.

3.2 Methodology

The approach we propose in this paper consist of three main steps: (i) *feature extraction*, (ii) *calculating similarity feature vectors* and (iii) *classification*. The overall design of our system is illustrated in Fig. 1. The products integration workflow runs in two phases: *training* and *application*. The training phase starts with preprocessing both the structured product ads and the unstructured product descriptions. Then, we use the structured product ads to build a feature extraction model. In this work we build two strategies for feature extraction (see Section 3.3): dictionary-based approach and Conditional Random Fields tagger. Next, we manually label a small training set of matching and non-matching unstructured pairs of product descriptions. We use the created feature extraction model to extract attribute-value pairs from the unstructured product descriptions. Then, we calculate the similarity feature vectors for the labeled training product pairs (see Section 3.5). In the final step, the similarity feature vectors are used to train a classification model (see Section 3.6). After the training phase is over, we have a trained feature extraction model and a classification model.

The application phase starts with preprocessing both of the datasets that are supposed to be matched[4]. Next, we generate a set M of all possible candidate matching pairs, which leads to a large number of candidates i.e., $|M| = n * (n - 1)/2$, if we try to identify duplicates within a single dataset of n products, or $|M| = n * m$, if we try to match two datasets of products with size n and m, respectively. To reduce the search space we use the brand value for blocking, i.e., we apply the matcher only for pairs of product descriptions sharing the same brand. Then, we extract the attribute-value pairs using the feature extraction model and calculate the feature similarity vectors. In the final step we apply the previously built classification model to identify the matching pairs of products.

[4] We need to note that we apply the same approach for identifying duplicates within the dataset of unstructured product descriptions, and for identifying matches between the unstructured product descriptions and the structured product ads.

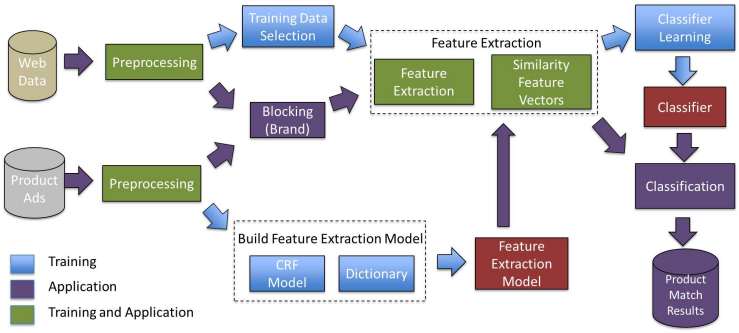


Fig. 1: System architecture overview

3.3 Feature Extraction

In this Section we describe two approaches for extracting attribute-value pairs from unstructured product title and description. In particular, both approaches take as an input unstructured text, and output a set of attribute-value pairs.

Dictionary-Based Approach: To implement the dictionary-based approach we were motivated by the approach described by Kannan et al [8]. We use the set of product ads in A to generate a dictionary of attributes and values. Let F represent all the attributes present in the product ads A. The dictionary represents an inverted index D from A such that $D(v)$ returns the attribute name $f \in F$ associated with a string value v. Then, to extract features from a given product description p, we generate all possible n-grams ($n \leq 4$) from the text, and try to match them against the dictionary values. In case of multiple matches, we choose the longest n-gram.

Conditional Random Fields: As the dictionary-based approach is able to extract only values that were seen, we need to use more advanced approach that is able to extract unseen attribute-value pairs. A commonly used approach for tagging textual descriptions in NLP are conditional random field (CRF) models. A CRF is a conditional sequence model which defines a conditional probability distribution over label sequences given a particular observation sequence. In this work we use the Stanford CRF implementation[5] in order to train product specific CRF models [5]. To train the CRF model the following features are used: current word, previous word, next word, current word character n-gram (n $\leq$ 6), current POS tag, surrounding POS tag sequence, current word shape, surrounding word shape sequence, presence of word in left window (size = 4) and presence of word in right window (size = 4).

To train the CRF model we use the structured product ads from database A. That means that the model is able to extract only attribute names that appear in the database A, but it can tag values that don't appear in the database.

Custom Feature Extraction: Beside the supervised extraction approaches, we extract several more features for all unstructured products. We use the product Web domain name, and the product URL (both are considered a long string

[5] http://nlp.stanford.edu/software/CRF-NER.shtml

Fig. 2: Example of attribute extraction from a product title

Table 1: Attributes and values normalization

Attribute Name	Attribute Value	Normalized Attribute Value	Attribute Data type
Brand	Samsung	samsung	string
Phone type	Galaxy S4	galaxy s4	string
Product code	GT-19505	gt-19505	string
Memory	16GB	1.6e+10 (B)	unit
Size	5.0 inches	0.127 (m)	unit
Compatible computer operating system	Android	android	string
Phone carrier	Sprint	sprint	string
Color	White Frost	white frost	string
Tagline	New Smartphone with 2-Year Contract	new smartphone with 2 year contract	long string

in the following section). The rationale for using these two fields is that often important keywords can be found in the product URL, and the domain might be a good indicator about the type of the product.

Furthermore, we noticed that in some of the product title and/or description a so called *product code* is present, which in many cases uniquely identifies the product. For example, *UN55ES6500* is a unique product code for a *Samsung* TV. This attribute has a high relevance for the task of product matching. To extract the product code from the text we use a set of manually written regular expressions across all categories.

3.4 Attribute Value Normalization

Once all attribute-value pairs are extracted from the given dataset of offers, we continue with normalizing the values of the attributes. To do so, we first try to identify the data type of each of the attributes, using several manually defined regular expressions, which are able to detect the following data types: string, long string (string with more than 3 word tokens), number and number with unit of measurement. Additionally, the algorithm uses around 200 manually generated rules for converting units of measurements to the corresponding base unit (metric system), e.g. 5" will be converted to 0.127 meter. In the end, the string values are lower cased, stop words and some special characters are removed.

Example Tagging In Fig. 2 we give an example of feature extraction from a given product title. The extracted attribute-value pairs are shown in Table 1, as well as the normalized values, and the detected attribute data type.

3.5 Calculating Similarity Feature Vectors

After the feature extraction is done, we can define an attribute space $F = \{f_1, f_2, ..., f_n\}$ that contains all of the extracted attributes. To measure the similarity between two products we calculate similarity feature vector $F(p_i, p_j)$ for each candidate product pair. For two products p_1 and p_2, represented with the

Table 2: Datasets used in the evaluation

Dataset	#products	#matching pairs	#non-matching pairs
Televisions	344	236	58,760
Mobile Phones	225	467	24,734
Laptops	209	146	25,521

attribute vectors $F_{p1} = \{f_1v, f_2v, ..., f_nv\}$ and $F_{p2} = \{f_1v, f_2v, ..., f_nv\}$, respectively, we calculate the similarity feature vector $F(p_1, p_2)$ by calculating the similarity value for each attribute f in the attribute space F. Let p_1.val(f) and p_2.val(f) represent the value of an attribute f from p_1 and p_2, respectively. The similarity between p_1 and p_2 for the attribute f is calculated based on the attribute data type as shown in Equation 1.

$$f(p_1, p2) = \begin{cases} 0, & if \quad p_1.val(f) = 0 \quad OR \quad p_2.val(f) = 0 \\ JaccardSimilarity(p_1.val(f), p_2.val(f)), & if \quad f \quad is \quad string \quad attribute \\ CosineSimilarity(p_1.val(f), p_2.val(f)), & if \quad f \quad is \quad long \quad string \quad attribute \\ p_1.val(f) == p_2.val(f) \quad ? \quad 1 : 0, & if \quad f \quad is \quad numeric \quad or \quad unit \quad attribute \end{cases} \tag{1}$$

The Jaccard similarity is calculated on character n-grams (n $\leq$ 4), and the Cosine similarity is calculated on word tokens using TF-IDF weights.

3.6 Classification Approaches

Once the similarity feature vectors are calculated, we train four different classifiers that are commonly used for the given task: (i) Random Forest, (ii) Naive Bayes, (iii) Support Vector Machines (SVM) and (iv) Logistic Regression.

As the training dataset contains only a few matching pairs, and a lot of non-matching pairs, the dataset is highly imbalanced. To address the problem of classifying imbalanced datasets we use two sampling approaches [4]: (i) *Random Under Sampling (RUS)*: removes samples from the majority class until the number of the samples of the minority class equals the number of samples of the majority class; (ii) *Random Over Sampling (ROS)*: randomly samples instances from the minority class until the number of the samples of the minority class equals the number of samples of the majority class.

4 Evaluation

In this Section, we evaluate the extent to which we can use the dataset of structured product ads for the task of matching unstructured product descriptions.

4.1 Datasets

For the evaluation we use Yahoo's Gemini Product Ads (GPA) for supervision, and we use a subset of the WebDataCommons (WDC) extraction[6].

[6] http://webdatacommons.org/structureddata/index.html

Prouduct Ads - GPA dataset For our experiments, we are using a sample of three product categories from the Yahoo's Gemini Product Ads database. More precisely, we use a sample of 3,476 TVs, 3,372 mobile phones and 3,330 laptops. There are 35 different attributes in the TVs and mobile phones categories, and 27 attributes in the laptops category. We use this dataset to build the dictionary-based and the CRF feature extraction models.

Unstructured Product Offers - WDC Microdata Dataset The latest extraction of WebDataCommons includes over 5 billion entities marked up by one of the three main HTML markup languages (i.e., Microdata, Microformats and RDFa) and has been retrieved from the CommonCrawl 2014 corpus[7]. From this dataset we focus on product entities annotated with Microdata using the schema.org vocabulary. To do so, we use a sub-set of entities annotated with *http://schema.org/Product*. The dataset contains *288,082,823* entities in total, or *2,829,523,589* RDF quads. *89,608* PLDs (10.9%) annotate at least one entity as *s:Product* and *62,849* PLDs (7.6%) annotate at least one entity as *s:Offer*. In our approach, we make use of the properties *s:name* and *s:description* for extracting attribute-value pairs.

To evaluate the approach, we built a gold standard from the WDC dataset on three categories in the *Electronics domain*, i.e., TVs, mobile phones and laptops. We set some constraints on the entities we select: (i) the products must contain s:name and s:description property in English language, (ii) the s:name must contain between 3 and 50 words, (iii) the s:description must contain between 10 and 200 words, (iv) ignore entities from community advertisement websites (e.g., gumtree.com), (v) the product can be uniquely identified based on the title and description i.e., contains enough information to pinpoint the exact product.

The gold standard is generated by manually identifying matching products in the whole dataset. Two entities are labeled as matching products if both entities contain enough information to be uniquely identified, and both entities point to the same product. It is important to note that the entities do not necessarily contain the same set of product features. The number of entities, the number of matching and non-matching pairs for each of the datasets is shown in Table 2.

4.2 Experiment Setup

To evaluate the effectiveness of the approach we use the standard performance measures, i.e., Precision (P), Recall (R) and F-score (F1). The results are calculated using stratified 10-fold cross validation. For conducting the experiments, we used the RapidMiner machine learning platform and the RapidMiner development library.

We compare our approach with two baseline methods. First, we try to match the products based on TF-IDF cosine similarity. We report the best score on different levels of matching thresholds, i.e., we iterate the matching threshold

[7] http://blog.commoncrawl.org/2015/01/december-2014-crawl-archive-available/

Table 3: Products matching baseline results using cosine similarity and Silk

	Cosine similarity TF-IDF (title)			Silk (CRF features)		
Dataset	P	R	F1	P	R	F1
Television	0.299	0.219	0.253	0.501	0.911	0.646
Mobile Phones	0.375	0.383	0.379	0.406	0.840	0.547
Laptops	0.296	0.397	0.339	0.284	0.808	0.420

Table 4: Products Matching Performance - Televisions

		CRF			Dictionary		
Model	Sampling	P	R	F1	P	R	F1
Random Forest	ROS	0.882	0.765	0.819	0.829	0.741	0.783
	RUS	0.697	0.826	0.756	0.663	0.779	0.716
	No sampling	0.921	0.739	**0.820***	0.805	0.741	**0.772**
Naive Bayes	ROS	0.069	0.911	0.128	0.074	0.941	0.137
	RUS	0.069	0.898	0.128	0.043	0.932	0.081
	No sampling	0.072	0.893	0.133	0.046	0.932	0.088
SVM	ROS	0.629	0.682	0.655	0.070	0.114	0.087
	RUS	0.679	0.660	0.669	0.622	0.630	0.626
	No sampling	0.849	0.639	0.729	0.708	0.431	0.536
Logistic Regression	ROS	0.506	0.762	0.608	0.232	0.811	0.361 6
	RUS	0.486	0.742	0.587	0.134	0.821	0.231
	No sampling	0.519	0.769	0.619	0.219	0.806	0.345

starting from 0.0 to 1.0 (with step 0.01) and we assume that all pairs with similarity above the threshold are matching pairs[8].

As a second baseline we use the Silk Link Discovery Framework [7], an open-source tool for discovering links between data items within different data sources. The tool uses genetic algorithm to learn linkage rules based on the extracted attributes. For this experiment, we first extract the features from the product title and description using our CRF model, and then represent the gold standard in RDF format. The evaluation is performed using 10-fold cross validation.

4.3 Results

The results for both baseline approaches are shown in Table 3. We might conclude that both baseline approaches deliver rather poor results.

Table 4 shows the results of our approach on the TVs dataset, using both CRF and Dictionary feature extraction approach. The best score is achieved using the CRF feature extraction approach, and Random Forest classifier without sampling. We can see that the same classifier performs a little bit worse when using the dictionary-based feature extraction approach.

Table 5 shows the results on the mobile phones dataset. As before, the best score is achieved using the CRF feature extraction approach, and Random Forest classifier using ROS sampling. We can note that the results using the dictionary-based approach are significantly worse than the CRF approach. The reason is that the GPA dataset contains a lot of trendy phones from 2015, while the WDC dataset contains phones that were popular in 2014, therefore the dictionary-based approach fails to extract many attribute-value pairs.

[8] We tried calculating the similarity based on different combination of title and description, but the best results were delivered when using only the product title.

Table 5: Products Matching Performance - Mobile Phones

		CRF			Dictionary		
Model	Sampling	P	R	F1	P	R	F1
Random Forest	ROS	0.885	0.756	**0.815***	0.374	0.659	0.477
	RUS	0.814	0.744	0.777	0.337	0.624	0.438
	No sampling	0.894	0.742	0.811	0.386	0.659	**0.487**
Naive Bayes	ROS	0.153	0.631	0.246	0.102	0.354	0.158
	RUS	0.125	0.575	0.205	0.102	0.325	0.155
	No sampling	0.128	0.535	0.206	0.102	0.310	0.153
SVM	ROS	0.002	0.007	0.003	0.416	0.163	0.234
	RUS	0.440	0.457	0.448	0.398	0.150	0.218
	No sampling	0.340	0.430	0.380	0.385	0.143	0.208
Logistic Regression	ROS	0.413	0.489	0.448	0.258	0.323	0.287
	RUS	0.388	0.457	0.420	0.279	0.325	0.300
	No sampling	0.407	0.472	0.437	0.247	0.319	0.278

Table 6: Products Matching Performance - Laptops

		CRF			Dictionary		
Model	Sampling	P	R	F1	P	R	F1
Random Forest	ROS	0.687	0.530	0.598	0.702	0.484	0.573
	RUS	0.588	0.619	0.603	0.535	0.560	0.547
	No sampling	0.815	0.513	**0.630***	0.741	0.498	**0.596**
Naive Bayes	ROS	0.050	0.778	0.094	0.095	0.567	0.163
	RUS	0.062	0.758	0.115	0.116	0.560	0.192
	No sampling	0.061	0.758	0.113	0.105	0.560	0.177
SVM	ROS	0.676	0.529	0.594	0.172	0.585	0.265
	RUS	0.690	0.531	0.600	0.725	0.503	0.594
	No sampling	0.694	0.565	0.623	0.858	0.434	0.576
Logistic Regression	ROS	0.643	0.504	0.565	0.538	0.433	0.480
	RUS	0.438	0.587	0.502	0.561	0.475	0.514
	No sampling	0.651	0.510	0.572	0.301	0.543	0.387

Table 6 shows the results of our approach on the Laptops dataset. Again, the best score is achieved using the CRF feature extraction approach, and Random Forest classifier without sampling. We can observe that for this dataset the results drop significantly compared to the other datasets. The reason is that the matching task for laptops is more challenging, because it needs more overlapping features to conclude that two products are matching [9].

The results show that our approach clearly outperforms both baseline approaches on all three categories. The Random Forest classifier delivers the best result for all three categories. We can observe that the other classifiers achieve high recall, i.e., they are able to detect the matching pairs in the dataset, but they also misclassify a lot of non-matching pairs, leading to a low precision. It is also interesting to observe that the RUS sampling performs almost as good as the other sampling techniques, but it has considerably lower runtime.

CRF evaluation: We also evaluate the Conditional Random Field model on the database of structured product ads. For each of the three product categories we select 70% of the instances as a training set and the rest as a test set. The results for each category, as well as the number of instances used for training and

[9] For example, two laptops might share the same brand, same CPU, and same HDD, but if the memory differs, then the laptops are not the same.

Table 7: CRF evaluation on structured product ads data

Dataset	#training	#test	#attributes	P	R	F1
Televisions	2,436	1,040	35	0.962	0.9431	0.9525
Mobile Phones	2,220	1,010	35	0.9762	0.9613	0.9687
Laptops	2,330	1,000	27	0.9481	0.9335	0.9408

Table 8: Discovered matching products in the WDC dataset

Brand	sony	lg	samsung	rca	vizio	panasonic	philips	magnavox	nec	proscan
#WDC products	3,673	14,764	4,864	3,961	563	1,696	1,466	141	23,845	30
#Matches	926	734	567	385	296	160	44	29	18	7
Precision	95	94	88.18	93.55	94.59	93.75	95.45	100	100	100

testing, and the number of attributes are shown in Table 7. The results show that the CRF model is able to identify the attributes in the text descriptions with high precision.

5 Data Fusion

As the evaluation of the approach showed that we are able to identify duplicate products with high precision, we apply the approach on the whole WDC and GPA products datasets. First, we try to identify duplicate products within the WDC dataset for top 10 TV brands. Then, we try to identify matching products in the WDC dataset for the product ads in the GPA dataset in the TV category.

Integrating Unstructured Product Descriptions: In the first experiment we apply the previously trained Random Forest model to identify matching products for the top 10 TV brands in the WDC dataset. To do so, we selected a sub-set of products from the WDC dataset that contain one of the TV brands in the *s:name* or *s:description* of the products. Furthermore, we apply the same constraints described in Section 4, which reduces the number of products. We use the brand name as a blocking approach, i.e., we generate candidate matching pairs only for products that share the same brand. We use the CRF feature extraction approach to extract the features and we tune the Random Forest model in a way that we increase the precision, on the cost of lower recall, i.e., a candidate products pair is considered to be positive matching pair if the classification confidence of the model is above *0.8*.

We report the number of discovered matches for each of the TV brands in Table 8. The second row of the table shows the number of candidate product descriptions after we apply the selection constraints on each brand. We manually evaluated the correctness of the matches and report the precision. The results show that we are able to find a large number of matching products with high precision. By relaxing the selection constraints of product candidates the number of discovered matches would increase, but it might also reduce the precision.

Furthermore, we try to identify how many matches and new attributes can be identified for single products. We randomly chose 5 different TVs and counted the number of discovered matches, *s:offers*, *s:reviews* and *s:aggregatedRating* properties from the WDC dataset, and how many new attribute-value pairs we discover from the *s:name* and *s:description* using the CRF model.

Table 9: Extracted attributes for TV products

Product	#matches	#offers	#reviews	#ratings	#attributes
Vizio TV E241I-A1 24”	10	15	13	2	7
RCA TV 24G45RQ 24”	10	11	2	3	8
Samsung TV 55” UN55ES6500	8	12	2	1	14
LG TV 42LN5300 42”	6	8	4	0	6
Panasonic TV TH-32LRH30U 32”	4	6	0	1	6

Table 10: Discovered matching products in the WDC dataset for product ads in the GPA dataset

Brand	samsung	vizio	lg	rca	sony	proscan	nec	magnavox	panasonic	philips
#GPA products	560	253	288	10	102	18	22	28	41	11
#WDC products	4,864	563	14,764	3,961	3,673	30	23,845	141	1,696	1,466
#Matches	202	123	102	67	40	28	21	12	6	2
Precision	80.85	91.80	89.24	79.10	97.50	100.00	85.70	100.00	100.00	100.00

The results are shown in Table 9. The results show that we are able to identify a number of matches among products, and the aggregated descriptions have at least six new attribute-value pairs in each case.

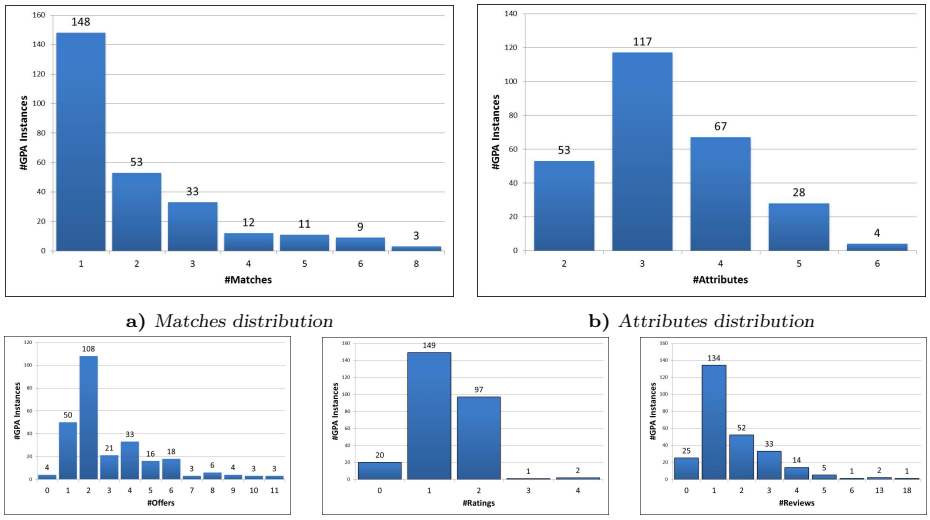
Enriching Product Ads: In this experiment we try to identify matching products in the WDC dataset for the product ads in the GPA dataset. Similarly as before, we select WDC products based on the brand name and we apply the same filtering to reduce the sub-set of products for matching. To extract the features for the WDC products we use the CRF feature extraction model, and for the GPA products we use the already existing features provided by the merchants. To identify the matches we apply the same Random Forest model as before. The results are shown in Table 10. The second row reports the number of products of the given brand in the GPA dataset, and the third row in the WDC dataset.

The results show that we are able to identify small number of matching products with high precision. We have to note again that we are not able to identify any matches for the products in the GPA dataset that are released after 2014, because they do not appear in the WDC dataset.

Furthermore, we analyzed the number of new attributes we can discover for the GPA products from the matching WDC products. The distribution of matches, newly discovered attribute-value pairs, offers, ratings and reviews per GPA instance is shown in Fig. 3. The results show that for each of the product ads that we found a matching product description, at least 1 new attribute-value pair was discovered. And for some product ads even 8 new attribute-value pairs were discovered.

6 Product Categorization

Categorization of products in a given product catalog is an important task. For example, online shops categorize products for easier navigation, and in the case of product ads, it allows easier ad retrieval and better user targeting. Also, identifying the category of the products before applying the matching approach might be used as a blocking approach. Here we examine to which extent we can use



a) *Matches distribution* **b)** *Attributes distribution*

c) *Offers distribution* **d)** *Ratings Distribution* **e)** *Reviews distribution*

Fig. 3: Distribution of newly discovered matches and attributes per product ad

a structured database of product ads to perform categorization of unstructured products description. Again we use the database of structured product ads to extract features from unstructured product descriptions, which are then used to build a classification model

In this approach we use only the dictionary-based feature extraction approach as described in Section 3.3[10]. To build the dictionary, we use all product ads across all categories in the database of product ads. To generate the feature vectors for each instance, after the features from the text are extracted, the value of each feature is tokenized, lowercased, and removed tokens shorter than 3 characters. The terms of each feature are concatenated with the feature name e.g. for a value *blue* for the feature *color*, the final value will be *blue-color*.

Following Meusel et al. [11], in order to weigh the different features for each of the elements in the two input sets, we apply two different strategies, Binary Term Occurrence (BTO) and TF-IDF. In the end we use the feature vectors to build a classification model. We have experimented with 4 algorithms: Naive Bayes (NB), Support Vector Machines (SVM), Random Forest (RF) and k-Nearest Neighbors (KNN), where k=1.

Gold Standard: For our experiments we use the GS1 Product Catalogue (GPC)[11] as a target hierarchy. The hierarchy is structured in six different levels, but in our experiments we try to categorize the products in the first three levels of the taxonomy: Segment, Family and Class. The first level contains 38 different categories, the second level 113 categories and the third level 783 categories.

[10] We were not able to build a sufficiently good CRF model that is able to annotate text with high precision because of the many possible attributes across all categories. A separate CRF model for each category in the structured database of product ads should be trained.

[11] http://www.gs1.org/gpc

Table 11: The best categorization results using the dictionary-based approach and baseline approach

GS1 Level	Features	Model	ACC	P	R	F1
1	Dictionary (BTO)	SVM	88.34	74.11	64.78	69.13
2	Dictionary (TF-IDF)	NB	79.87	46.31	40.08	42.97
3	Dictionary (TF-IDF)	NB	70.9	25.45	24.36	24.89

To evaluate the proposed approach we use the Microdata products gold standard developed in [11]. We removed non-English instances from the dataset, resulting in 8,362 products. In our evaluation we use the *s:name* and the *s:description* properties for generating the features.

Evaluation: The evaluation is performed using 10-fold cross validation. We measure accuracy (Acc), Precision (P), Recall (R) and F-score (F1). We compare our approach to a TF-IDF and BTO baseline, where the text is preprocessed as before, but no dictionary is used.

Due to space constraints we show only the best performing results for each of the three levels. The complete results can be found online[12]. The results show that the dictionary-based approach can be used for classification of products on different level of the hierarchy with high performance. Also, the results show that it outperforms the baseline approaches for all three levels of classification for both accuracy and F-score. Again, we have to note that the gold standard contains products that appear in 2014, while the GPA dataset contains relatively up to date products from 2015.

7 Conclusion

In this paper, we proposed an approach for enriching structured product ads with structured data extracted from HTML pages that contain semantic annotations. The approach is able to identify matching products in unstructured product descriptions using the database of structured product ads as supervision.

We identify the Microdata dataset as a valuable source for enriching existing structured product ads with new attributes. We showed that not only we could integrate some of the existing Microdata attributes, like *s:offers*, *s:aggregateRating* and *s:review*, but with our approach we can extract valuable attribute-value pairs from the textual information of the products that do not exist in the structured product ads. In future work, to validate the value of the new attributes we need to evaluate the influence of the new attributes on the ads ranking algorithm. We could also include other schema.org product properties in the approach, like *s:mpn*, *s:model* and *s:gtin*, which might be useful for identity resolution. Additionally, mining search engine query logs we could extract valuable features for identifying matching products.

Besides integrating products over different online shops and product categorization, our approach could be used for search query processing, which would undoubtedly improve the shopping experience for users [3].

[12] http://data.dws.informatik.uni-mannheim.de/gpawdc/categorization/results/FullResults.xlsx

Acknowledgements We would like to acknowledge Roi Blanco (Yahoo Labs) and Christian Bizer (University of Mannheim) for their helpful comments to our work. We would also like to acknowledge the support, help and insights of the Yahoo Gemini Product Ads engineering and the Yahoo Labs Advertising Sciences teams, in particular Nagaraj Kota and Ben Shahshahani.

References

1. de Bakker, M., Frasincar, F., Vandic, D.: A hybrid model words-driven approach for web product duplicate detection. In: Advanced Information Systems Engineering. pp. 149–161. Springer (2013)
2. van Bezu, R., Borst, S., Rijkse, R., Verhagen, J., Vandic, D., Frasincar, F.: Multi-component similarity method for web product duplicate detection (2015)
3. Bhattacharya, S., Gollapudi, S., Munagala, K.: Consideration set generation in commerce search. In: Proceedings of the 20th International Conference on World Wide Web. pp. 317–326. WWW '11, ACM, New York, NY, USA (2011), http://doi.acm.org/10.1145/1963405.1963452
4. Chawla, N.V.: Data mining for imbalanced datasets: An overview. In: Data mining and knowledge discovery handbook, pp. 853–867. Springer (2005)
5. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. pp. 363–370. Association for Computational Linguistics (2005)
6. Ghani, R., Probst, K., Liu, Y., Krema, M., Fano, A.: Text mining for product attribute extraction. ACM SIGKDD Explorations Newsletter 8(1), 41–48 (2006)
7. Isele, R., Bizer, C.: Learning linkage rules using genetic programming. In: Proceedings of the International Workshop on Ontology Matching. pp. 13–24 (2011)
8. Kannan, A., Givoni, I.E., Agrawal, R., Fuxman, A.: Matching unstructured product offers to structured product specifications. In: 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 404–412 (2011)
9. Melli, G.: Shallow semantic parsing of product offering titles (for better automatic hyperlink insertion). In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1670–1678. ACM (2014)
10. Meusel, R., Petrovski, P., Bizer, C.: The webdatacommons microdata, rdfa and microformat dataset series. In: The Semantic Web–ISWC, pp. 277–292 (2014)
11. Meusel, R., Primpeli, A., Meilicke, C., Paulheim, H., Bizer, C.: Exploiting microdata annotations to consistently categorize product offers at web scale. In: Proceedings of the 16th International Conference on Electronic Commerce and Web Technologies (EC-Web2015/T2). Valencia, Spain (2015)
12. Nguyen, H., Fuxman, A., Paparizos, S., Freire, J., Agrawal, R.: Synthesizing products for online catalogs. Proceedings of the VLDB Endowment 4(7), 409–418 (2011)
13. Petrovski, P., Bryl, V., Bizer, C.: Integrating product data from websites offering microdata markup. In: Proceedings of the 23rd international conference on World wide web companion. pp. 1299–1304 (2014)
14. Petrovski, P., Bryl, V., Bizer, C.: Learning regular expressions for the extraction of product attributes from e-commerce microdata (2014)
15. Qiu, D., Barbosa, L., Dong, X.L., Shen, Y., Srivastava, D.: Dexter: large-scale discovery and extraction of product specifications on the web. Proceedings of the VLDB Endowment 8(13), 2194–2205 (2015)
16. Vandic, D., Van Dam, J.W., Frasincar, F.: Faceted product search powered by the semantic web. Decision Support Systems 53(3), 425–437 (2012)