Using schema.org Annotations for Training and Maintaining Product Matchers

Ralph Peeters
Data and Web Science Group
University of Mannheim
Mannheim, Germany
ralph@informatik.uni-mannheim.de

Anna Primpeli
Data and Web Science Group
University of Mannheim
Mannheim, Germany
anna@informatik.uni-mannheim.de

Benedikt Wichtlhuber
Data and Web Science Group
University of Mannheim
Mannheim, Germany
jwichtlh@mail.uni-mannheim.de

Christian Bizer
Data and Web Science Group
University of Mannheim
Mannheim, Germany
chris@informatik.uni-mannheim.de

ABSTRACT

Product matching is a central task within e-commerce applications such as price comparison portals and online market places. State-of-the-art product matching methods achieve F1 scores above 0.90 using deep learning techniques combined with huge amounts of training data (e.g >100K pairs of offers). Gathering and maintaining such large training corpora is costly, as it implies labeling pairs of offers as matches or non-matches. Acquiring the ability to be good at product matching thus means a major investment for an e-commerce company. This paper shows that the manual labeling of training data for product matching can be replaced by relying exclusively on schema.org annotations gathered from the public Web. We show that using only schema.org data for training, we are able to achieve F1 scores between 0.92 and 0.95 depending on the product category. As new products appear everyday, it is important that matching models can be maintained with justifiable effort. In order to give practical advice on how to maintain matching models, we compare the performance of deep learning and traditional matching models on unseen products and experiment with different fine-tuning and re-training strategies for model maintenance, again using only schema.org annotations as training data. Finally, as using the public Web as distant supervision carries inherent noise, we evaluate deep learning and traditional matching models with regards to their label-noise resistance and show that deep learning is able to deal with the amounts of identifier-noise found in schema.org annotations.

CCS CONCEPTS

• **Information systems → Entity resolution**; **Electronic commerce**; • **Computing methodologies → Neural networks**.

WIMS 2020, June 30-July 3, 2020, Biarritz, France

ACM ISBN 978-1-4503-7542-9/20/06…$15.00
https://doi.org/10.1145/3405962.3405964

KEYWORDS

e-commerce, product matching, schema.org, distant supervision, deep learning

ACM Reference Format:
Ralph Peeters, Anna Primpeli, Benedikt Wichtlhuber, and Christian Bizer. 2020. Using schema.org Annotations for Training and Maintaining Product Matchers. In *The 10th International Conference on Web Intelligence, Mining and Semantics (WIMS 2020), June 30-July 3, 2020, Biarritz, France.* ACM, Biarritz, France, 10 pages. https://doi.org/10.1145/3405962.3405964

1 INTRODUCTION

Product matching is the task of deciding if two offers originating from two different websites refer to the same real-world product. This is a central task for e-commerce applications such as online market places, price comparison portals, as well as for the building of product knowledge graphs [29]. Product matching is not a trivial task, because for marketing reasons merchants present products in different ways in their Web shops.

Product matching has a long history in research and practice. Early approaches to product matching applied rule- and statistics-based methods [11]. Since the early 2000s, machine learning based methods dominate product matching [17]. Due to the successes of deep learning in fields like computer vision and natural language processing, the research focus has shifted towards applying these methods also for product matching [9, 12, 21, 25]. Using labeled data from a major retailer, Mudgal et al. [21] showed that deep learning can significantly outperform traditional machine learning-based matching methods reaching a performance of >90% F1 given enough training data (>100K pairs of offers). The bottleneck for applying the proposed methods is that gathering and maintaining such large training data corpora means a major investment for e-commerce companies. More recent work focuses on using pre-trained language models based on the Transformer architecture [27] and leveraging the pre-training/fine-tuning paradigm to improve on these results [6, 19].

In this paper we show that it is possible to achieve similarly high matching performance by using the Semantic Web as single source of training data. More specifically, we show that by using only schema.org annotations from the public Web, it is possible to train deep neural networks that achieve F1 scores between 0.92

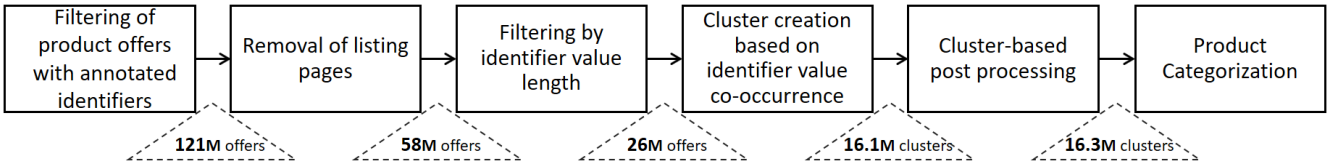


Figure 1: WDC Product Data Corpus creation pipeline.

and 0.95 depending on the product category. We rely on annotated product identifiers such as GTINs and EANs together with a specific data cleansing workflow for clustering together product offers from different shops referring to the same product. In a second step, we use this clustering as distant supervision to automatically build training sets for learning reliable product matchers.

New products appear every day. This requires product matchers to either generalize well to unseen products or be continuously maintained using training data for the new products. We show that deep learning-based matchers outperform traditional matchers for unseen products. Nevertheless, for reaching a matching performance of >90% in F1 for new products, the matchers need to be fine-tuned or re-trained using training data including offers for the new products. We show that schema.org annotations from the Web can also replace manual labeling for new products and compare different fine-tuning and re-training strategies for deep product matchers.

Finally, using schema.org product ID annotations, such as GTINs or MPNs, as distant supervision for product matching carries a certain level of inherent noise. It is therefore important to explore the label-noise resistance of the used learning method. We show that deep product matchers are able to deal with the inherent noise, but rapidly lose performance, when the level of label-noise is further increased.

In comparison to our paper at the Workshop on e-Commerce and NLP (ECNLP2019) [24], which focuses on the creation of the product data corpus using schema.org annotations and only contained preliminary experimental results, the contributions of this paper are: (1) new experimental results based on a larger manually verified gold standard as well as an improved assignment of offers to product categories, (2) an analysis of the impact of different sizes of training sets, derived from schema.org annotations, on different product matching algorithms, (3) an evaluation of the performance on unseen products, (4) a comparison of three different methods of maintaining learned deep product matchers, when unseen products are introduced, (5) an evaluation of the noise resistance of different matching algorithms over increasing levels of label-noise.

The paper is organized as follows: The first part of Section 2 explains, how we used schema.org annotations for building large scale training sets for product matching as well as an evaluation gold standard. The second part details the deep and the traditional learning algorithms used as well as the experimental setup and results for training product matchers. Section 3 explores the generalization performance of the product matching models on unseen products and details several fine-tuning and re-training approaches for maintaining deep product matchers. The impact of label-noise on the product matchers is explored in Section 4. Finally, Section 5 discusses related work and Section 6 concludes the paper with a summarization of the results.

2 TRAINING PRODUCT MATCHERS USING SCHEMA.ORG ANNOTATIONS AS DISTANT SUPERVISION

The Web Data Commons project has shown that at least 850,000 websites (PLDs) use the schema.org vocabulary to annotate product offers [4]. The properties that are most widely annotated are name, description, brand and image. Interestingly and crucial for using semantic annotations from different websites to train matching methods, 30.5% of the websites annotate product identifiers, such as manufacturer part numbers (MPNs), global trade item numbers (GTINs), or stock keeping units (SKUs). Despite of carrying some noise, these identifiers allow offers for the same product from different e-shops to be grouped, which can be considered as distant supervision, for training matching methods.

We use the schema.org/Product subset provided by the Web Data Commons project (November 2017)[1]. The subset contains 809 million schema.org/Product (s:Product) and schema.org/Offer (s:Offer) entities. In the following, we describe how we cleanse the dataset, derive different size training sets for different product categories and build a gold standard for evaluating product matchers.

2.1 Data Cleansing Pipeline

Figure 1 gives an overview of the data cleansing pipeline that we use to group offers into clusters. In the following, we describe each of the cleansing steps. More details about the cleansing pipeline are found in [24].

Filtering of product offers with annotated identifiers: We aim to filter all s:Product and s:Offer entities that have the following identifier related properties: *sku, mpn, identifier, productID, gtin14, gtin13, gtin12* and *gtin8*. Considering common annotation errors such as the usage of non-existing vocabulary terms, e.g. *schema:Product/Offer/sku* [20], we extend the filtering to any entity of the subcorpus matching the following regular expression: $.*/(gtin8|gtin12|gtin13|gtin14|sku|mpn|identifier|productID)$. This first filtering step results in 121M offers.

Removal of listing pages: Product offers which are part of listing pages or advertisements are not expected to be not accompanied by useful textual description that can help the matching task. Therefore we want to remove them from the corpus. We do this by applying a heuristic based on the following features: amount of s:Offer

[1] http://www.webdatacommons.org/structureddata/2017-12/stats/schema_org_subsets.html

Table 1: Distribution of clusters, offers, and cluster group sizes per product category - English WDC Product Data Corpus

Category	% Clusters	% Offers	# Clusters of Size			
			[3-4]	[5-10]	[11-20]	[>20]
Tools_and_Home_Improvement	12.37	9.41	27,228	8,336	4,648	1,325
Home_and_Garden	10.59	8.89	33,812	16,873	5,364	1,997
Automotive	8.95	8.37	22,808	10,143	4,094	4,421
Other_Electronics	6.74	5.27	21,590	5,603	11,55	508
Clothing	5.93	6.67	40,767	32,915	3,177	1,840
Books	5.36	4.22	19,771	6,361	654	360
Watches_and_Jewelry	4.94	4.95	19,526	11,245	2,112	1,436
Shoes	4.42	4.15	27,948	9,080	1,468	381
Health_and_Beauty	4.23	3.93	22,852	7,429	1,302	537
Toys_and_Games	4.01	3.21	11,196	3,109	968	521
Office_Products	4.01	13.13	27,393	10,875	3,052	1,561
Sports_and_Outdoors	3.84	5.45	16,930	6,927	1,556	2,602
Grocery_and_Gourmet_Food	3.51	3.01	20,428	3,416	416	243
Computers_and_Accessories	3.03	4.04	12,917	6,694	3,257	4,184
Luggage_and_Travel_Gear	2.63	2.68	21,871	7,914	704	385
CDs_and_Vinyl	2.49	1.80	4,251	1,514	486	71
Camera_and_Photo	2.38	1.97	11,583	3,851	663	134
Musical_Instruments	2.26	2.01	6,218	1,176	240	180
Cellphones_and_Accessories	2.03	1.52	3,652	1,334	274	281
Others	6.28	5.33	27,781	8,561	1,972	1,378

and s:Product entities in the same url, offer description length in terms of words and attached properties denoting advertisements like s:Offer/RelatedTo and s:Offer/SimilarTo. The listing heuristic achieves 94.8% on the positive class *isListingOrAd* and reduces the corpus size to 58M offers.

Filtering by identifier value length: After manually inspecting a subset of the product identifier values, we discover three common erroneous annotation practices that lead to invalid identifiers: containment of non-alphanumeric characters and prefixes such as *ean, sku* and *isbn*, too long or too short values and values consisting only of alphabetic characters. We cleanse the identifier values by removing prefixes and non-alphanumeric characters and remove those that after cleansing do not adhere to the schema.org annotation standards and are therefore expected to be highly noisy. This cleansing step results in 26M offers.

ID-cluster creation: We exploit the co-occurrence of identifier values and group the offers into ID-clusters. Each ID-cluster represents the same real-world product. After grouping the clusters we apply a post-processing cleansing step to identify and remove category specific identifiers, such as UNSPSC. The removal of the UNSPSC values splits some of the ID-clusters and creates 199K additional ones. After this final cleansing step the corpus contains 26.5M product offers from 79K distinct PLDs. The resulting offers have 18.5M distinct identifier values and are grouped in 16.3M ID-clusters. We filter the English offers based on the top-level-domain of the url of each offer and derive the English WDC Product Data Corpus with 16.4M offers from 43K e-shops grouped in 10M clusters.

Categorization: The ID-clusters are categorized into a product categorization schema suggested by a publicly available Amazon product dataset[2]. We train a one-vs-rest ensemble of Logistic Regression classifiers for categorizing product offers. For training and testing we manually annotate 2,115 clusters while considering an annotation limit of at least 50 clusters per product category. The ensemble achieves 85% F_{1micro} on the test set over all categories. All experimental details and statistics on the gold standard are reported on the WDC Product Data Corpus Categorization Web page[3]. The learned model is applied to the English corpus and assigns a product category to each ID-cluster. Table 1 presents the distribution of clusters, offers and cluster group sizes per category.

Quality of the WDC Product Data Corpus: Despite the cleansing steps, it is still expected that the WDC Product Data Corpus is subject to remaining noise. We evaluate the corpus quality by manually inspecting 900 pairs of offers from the same ID-clusters and verifying if they describe the same real-world product. The sampled pairs derive from ID-clusters of different product categories and cluster sizes: small (3-5 offers), medium (6-80 offers), and large (>80 offers). The evaluation results show that 94% of the sampled offer pairs are true matches, 3% are wrongly matched offers and for 3% of the cases both annotators involved in the evaluation quality process were unsure on the correct label, mostly because of short titles and descriptions. Further analysis on the wrongly clustered offers distinguished two main types of errors: (1) wrong identifier values, i.e. the identifier is the same, but the offers refer to different products and (2) grouping errors caused by the design decision of grouping based on identifier value co-occurrence. Further details about the error analysis can be found in [4].

[2]http://jmcauley.ucsd.edu/data/amazon/
[3]http://webdatacommons.org/categorization/

2.2 Deriving Evaluation Gold Standards and Training Sets

We derive gold standards for evaluating product matching methods from the English WDC Product Data Corpus. Version 2 of the gold standards, which we released in October 2019, consists of 1100 pairs of offers from each of the following four product categories: *Computers & Accessories*, *Camera & Photo*, *Watches* and *Shoes*. Each gold standard covers 150 products (ID-clusters) from a category with positive and negative pairs. Overall offers for more than these 150 products are contained in the gold standards, serving only as negative correspondences to the 150 products without positives of their own. For each of the 150 products the gold standard contains 2 matching pairs of offers (positives) and 5 or 6 non-matching pairs of offers (negatives). All pairs of offers were manually reviewed. The aim was to include a diverse selection of products as well as a good mixture of difficult and easy to match pairs of offers into the gold standard. To this end, similarity metrics are applied to different attributes to sort pairs and collect hard matching cases (positives and negatives) for the gold standard, while also including randomly selected offer pairs. The clustering of semantically annotated ID-values serves as distant supervision, separating matches (intra-cluster) from non-matches (inter-cluster).

Table 2: Gold standard profiling

Category	#Products	# Pos. Pairs	# Neg. Pairs	% Density T	D	B	S
Computers	150(745)	300	800	100	82	42	22
Cameras	150(563)	300	800	100	73	25	7
Watches	150(617)	300	800	100	71	15	7
Shoes	150(563)	300	800	100	70	8	2
All	600(2485)	1,200	3,200	100	74	23	10

Table 2 shows statistics of the gold standard, including the percentage of pairs per category, where both offers contain a value for the respective attribute. We choose the four textual attributes title (T), description (D), brand (B) and specification table content (S) for our experiments. Furthermore, we focus mostly on the Computers category, representing more structured products, as well as the set combining all categories for the experiments in this paper.

To build training sets, the ID-clusters of all products contained in the respective gold standard are used to automatically build positive pairs inside clusters and negative pairs across clusters using a heuristic similar to the one used for building the gold standards. These pairs are not checked manually. The training pairs represent the main result of the application of semantic annotations described in this paper. Table 3 shows the statistics of the training sets. More details about the heuristics used for creating the gold standards and training sets can be found on the project's website[4].

2.3 Learning Product Matchers

In order to demonstrate the utility of the training sets for learning product matchers, we run experiments for all deep learning models that are part of the *Deepmatcher*[5] framework [21]. These are

[4]http://webdatacommons.org/largescaleproductcorpus/v2/
[5]https://github.com/anhaidgroup/deepmatcher

Table 3: Training set profiling

Category	# Pos. Pairs	# Neg. Pairs	% Density T	D	B	S
Small						
Computers	722	2,112	100	51	34	21
Cameras	486	1,400	100	53	21	4
Watches	580	1,675	100	43	15	7
Shoes	530	1,533	100	49	8	0
All	2,318	6,720	100	49	21	10
Medium						
Computers	1,762	6,332	100	51	34	20
Cameras	1,108	4,147	100	57	22	4
Watches	1,418	4,995	100	44	14	6
Shoes	1,214	4,591	100	49	7	0
All	5,502	20,065	100	50	20	9
Large						
Computers	6,146	27,213	100	51	31	18
Cameras	3,843	16,193	100	60	25	3
Watches	5,163	21,864	100	45	13	6
Shoes	3,482	19,507	100	51	6	0
All	18,634	84,777	100	51	19	8
XLarge						
Computers	9,690	58,771	100	50	30	16
Cameras	7,178	35,099	100	66	29	3
Watches	9,264	52,305	100	50	11	5
Shoes	4,141	38,288	100	53	5	0
All	30,273	184,463	100	54	19	7

binary classifiers that compare two products by first converting all word tokens into an embedding space attribute-wise, e.g. using *fastText* embeddings [5]. The embedding sequences of each attribute of the two products are then summarized and aggregated before concatenation of the resulting per-attribute embeddings as input to a classification module. The models in the Deepmatcher framework mainly differ on the complexity of the summarization step. The *SIF* model aggregates embedding sequences by simply averaging them using smooth inverse frequency [2]. The *RNN* uses a bi-GRU, while the *Attention* model employs *decomposable attention* [22]. Finally, *Hybrid* is a combination of the latter two models and the most complex. For more in-depth information about these models, we refer the reader to the original publication [21].

Experimental Setup: In preparation for the experiments, punctuation is removed, all word tokens are lower-cased and stopwords are removed using the *NLTK*[6] stopword list. We run experiments for all product categories as well as their combination as described in Section 2 for all four training set sizes and all four Deepmatcher models. As input embeddings we choose character-based fastText embeddings pre-trained on Wikipedia[7], as these have shown to perform well for the task of product matching [21]. Additionally, we use the brand, title and description attributes of the WDC Product Data Corpus to derive self-trained fastText embeddings as a

[6]https://www.nltk.org/
[7]https://fasttext.cc/docs/en/pretrained-vectors.html

comparison. These are trained using the default fastText parameters. All training sets are split into training and validation sets using a stratified 80:20 split. All Deepmatcher experiments are run three times for 15 epochs each and results are averaged. All models use default parameters apart from pos-neg ratio, which allows to penalize errors on the minority class more severely in imbalanced datasets. The pos-neg ratio is set to the actual ratio found in each training set. Once training is completed the model is evaluated on the corresponding gold standard.

In a second step, we repeat these experiments, while allowing the model to propagate gradients back to the embedding layer, which is not the case for the standard Deepmatcher implementation that uses fixed embeddings. This *end-to-end training* essentially allows the model to fine-tune the embeddings for the product matching task.

Finally, for both, standard Deepmatcher and the end-to-end variant, we select the Computers XLarge training set and optimize the parameters of the best performing model (RNN) using random search to showcase possible performance gains of an optimized model. The adjusted parameters include learning rate, learning rate decay, mini batch size as well as pos-neg ratio. Every selected parameter combination is trained three times and the results are averaged to find the best performing combination.

Baselines: The *Magellan* [16] framework and a simple baseline using *word co-occurrence* are trained using the same training and validation sets, to allow a comparison to traditional methods. Magellan offers automatic feature creation using string-based similarity metrics like Levenshtein and Jaccard similarity. These features are then used as input to scikit-learn [23] classifiers. The word co-occurrence method uses a binary feature vector of words occurring in the attributes of both products without regard for attribute borders. The vocabulary of words is derived using the words occurring in the training set. The resulting feature vector is then used as input to scikit-learn classifiers. For both Magellan and word co-occurrence hyperparameters are tuned using automatic grid-search. All experiments are run using different combinations of the features title, description, brand and specification table content.

Results: Table 4 shows the results of the experiments on the largest of the four training sets for each product category. Magellan performs the worst overall with F1 values between 60 and 69%. The word co-occurrence baseline reaches up to 84% F1 with significantly higher precision than Magellan. All of the Deepmatcher approaches achieve at least 90% F1 while peaking at nearly 96% F1 for the Watches category. Overall the RNN-based model has proven to consistently perform the best for the product matching task on this dataset. These results show that the derivation of training data using schema.org annotations allows training product matchers that can achieve a high level of performance comparable to using manually created training sets.

When comparing standard Deepmatcher to the end-to-end variation, it is evident that fine-tuning of the embedding layer can improve performance by up to 4% F1, which can be mostly credited to improved recall. Parameter optimization leads to nearly 2% higher F1 for the Computers set and standard RNN Deepmatcher, while improving mostly on precision. The end-to-end RNN parameter optimization did not lead to improved performance.

The comparison of pre-trained and self-trained embeddings shows that self-training can marginally increase the performance for some product categories, but overall pre-trained embeddings lead to higher results on average. Overall, fine-tuning pre-trained embeddings apparently offers significant performance gains without the overhead of self-training on large corpora. An inspection of the training times of the Computers RNN and its end-to-end equivalent shows an increase of 6% on average for the latter, which is acceptable considering the possible gains in performance.

Figure 2 shows the learning curves for the four Deepmatcher models as well as the end-to-end variant of the RNN on the Computers and combined gold standards. This is representative of the learning curves of all product categories with only minor differences among them. The curve shows that the RNN model significantly surpasses the other models for training sets of the size *large*. The overall improvement from *large* to *XLarge* was up to 3% F1 in our experiments, thus showing that training sets of the size of *large* are already close to the maximum observed performance and may be preferable for practical applications due to the significantly lower amount of required training labels. The end-to-end variant of Deepmatcher surpasses the standard for medium sized training sets and thus should be the preferred way of training if a medium-sized training set is available.

Figure 3 shows the learning curves of the best performing Deepmatcher model (end-to-end RNN) compared to those of the traditional baselines on the Computers and combined gold standards. Whereas Magellan does not improve with training sets of larger size, both word co-occurrence and Deepmatcher show a strong increase with training sets of size *medium*. The *large* set marginally improves the performance of word co-occurence by around 3% F1 while Deepmatcher gains nearly 10%. Surprisingly, for all product categories the performance of Deepmatcher on the smallest training set is greater than that of the traditional methods, which makes it the preferred method if resources for training and maintaining deep learning models are available. Nevertheless, practitioners looking to apply these models should aim for training sets of at least size *medium* for good performance.

Overall the results demonstrate that using schema.org annotations and distant supervision for building training sets allows to learn models that can achieve 90% and more F1 over several product categories and consequently are a viable alternative to manually labeled datasets, especially when considering the required size and the human workload this would incur.

3 MAINTAINING MATCHERS TO COVER NEW PRODUCTS

New products appear on the market every day. For e-commerce applications it is thus crucial to be able to cluster offers for new products and correctly distinguish them from offers for known products. In the following, we investigate to which extent matchers that were trained for one set of products are able to generalize to new, unseen products. Afterwards, we experiment with different approaches for fine-tuning matchers using offers for previously unseen products that we again gather from the Web.

Experimental Setup: In order to evaluate how matchers generalize to unseen products, we randomly select 30 out of the 150

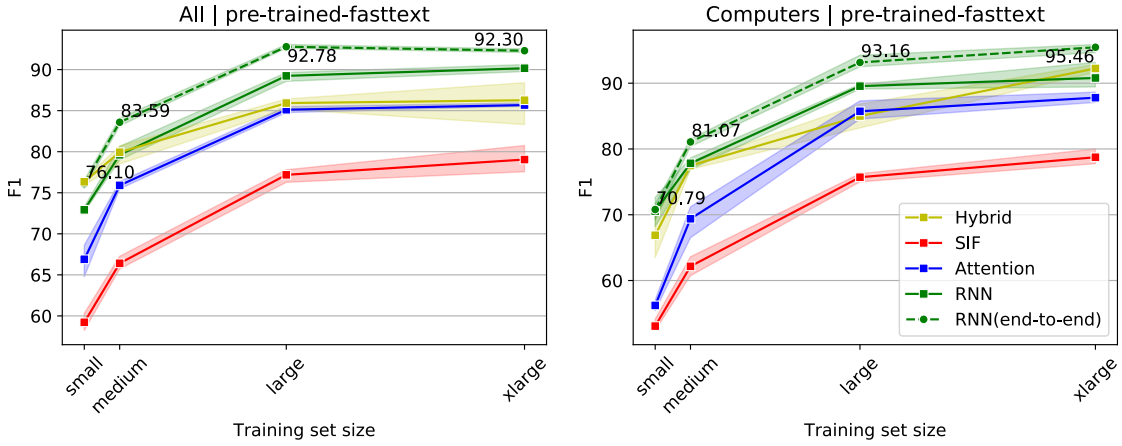
Table 4: Results on training set XLarge

Category	Model	Features	P	R	F1	σ
Magellan						
Computers	XGBoost	T+D+B+S	72.32	65.33	68.65	-
Cameras	XGBoost	T+D+B+S	73.87	54.67	62.84	-
Watches	XGBoost	T+D+B+S	74.15	50.67	60.20	-
Shoes	RandomForest	T+D+B+S	51.33	83.67	63.62	-
All	RandomForest	T+D+B+S	48.68	78.42	60.07	-
Word Co-Occurrence						
Computers	LinearSVC	T+D+B+S	86.74	80.67	83.59	-
Cameras	LinearSVC	T+D+B+S	81.51	64.67	72.12	-
Watches	LinearSVC	T+D+B+S	88.14	69.33	77.61	-
Shoes	LogisticRegression	T	86.14	58.00	69.32	-
All	LogisticRegression	T+D+B+S	85.97	66.92	75.26	-
Deepmatcher - Pre-Trained fastText						
Computers	RNN	T+D+B	89.75	91.89	90.80	0.79
Computers(opt.)	RNN	T+D+B	93.07	92.11	92.56	0.76
Cameras	RNN	T+D+B+S	89.22	89.22	89.21	1.68
Watches	RNN	T+D+B+S	92.73	94.22	93.45	0.93
Shoes	RNN	T	93.16	92.11	92.61	1.05
All	RNN	T+D	92.04	88.36	90.16	0.43
Deepmatcher - Self-Trained fastText						
Computers	RNN	T+D+B	89.66	94.22	91.88	0.79
Cameras	RNN	T+D	89.74	85.44	87.53	1.34
Watches	RNN	T+D	93.66	91.78	92.70	0.95
Shoes	RNN	T	90.65	90.45	90.54	0.85
All	RNN	T+D	90.98	87.31	89.10	0.39
Deepmatcher(End-to-End) - Pre-Trained fastText						
Computers	RNN	T	94.00	97.00	95.46	0.70
Computers(opt.)	RNN	T+D+B	93.76	96.78	95.25	0.31
Cameras	RNN	T	91.44	93.00	92.18	0.70
Watches	RNN	T+D	94.58	96.89	**95.72**	0.50
Shoes	RNN	T	95.58	93.78	**94.67**	0.54
All	RNN	T+D+B	91.25	93.38	92.30	0.31
Deepmatcher(End-to-End) - Self-Trained fastText						
Computers	RNN	T+D+B	93.88	97.16	**95.50**	0.35
Cameras	RNN	T	93.16	92.11	**92.63**	0.27
Watches	RNN	T+D	94.84	95.17	95.00	1.45
Shoes	RNN	T	93.25	91.56	92.38	0.25
All	RNN	T+D+B	92.40	93.62	**92.98**	0.11

products having positive and negative pairs in the gold standard. We treat these 30 products as *unseen* and split the gold standard of the category Computers as well as the Computers XLarge training set by removing all pairs containing any of these 30 products from both sets into a new training set and gold standard. Table 5 provides statistics about the resulting training and test sets for *seen* and *unseen* products.

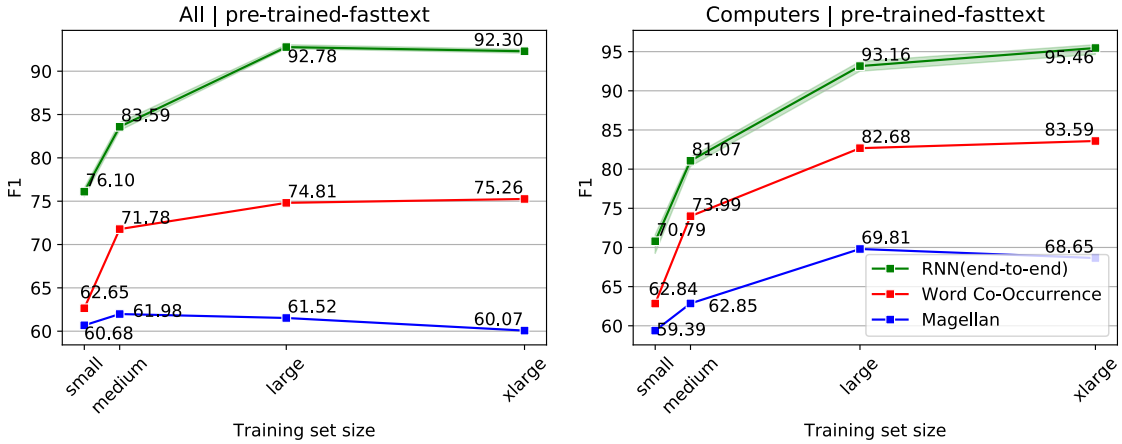
Generalization to Unseen Products: In a first experiment, we train different matching models using the training data for the seen products and subsequently evaluate the models only using the gold standard for unseen products. Table 6 shows the results for the evaluation of the trained model on seen and unseen products for the Deepmatcher as well as the Magellan and word co-occurrence baselines. The performance of Deepmatcher on the seen products is comparable to the results obtained in Table 4 for the Computers XLarge training set. It is evident that performance on unseen products drops considerably for Deepmatcher and word co-occurrence with losses of 16% and 40% F1 respectively. Magellan is impacted the least with an absolute loss of 4% in F1. However, the F1 performance for unseen products of all three models is clearly below 0.9 and thus likely insufficient for e-commerce applications. Thus, the continuous maintenance of product matchers to cover new products is necessary.

Fine-tuning Matchers for New Products: By regularly recrawling e-shops that are known to annotate product identifiers using schema.org terms, it is possible to monitor the appearance of new



(a) All **(b) Computers**

Figure 2: Comparison of Deepmatcher models



(a) All **(b) Computers**

Figure 3: Comparison Deepmatcher to baselines

Table 5: Datasets for seen and unseen new products

Category	# Pos. Pairs	# Neg. Pairs	% Dens. T	% Dens. D	% Dens. B
Gold Unseen	59	154	100	89	57
Gold Seen	241	646	100	90	56
Train Unseen	264	3,562	100	66	49
Train Seen	7,488	43,454	100	69	49
Train Comb.	7,752	47,016	100	69	49

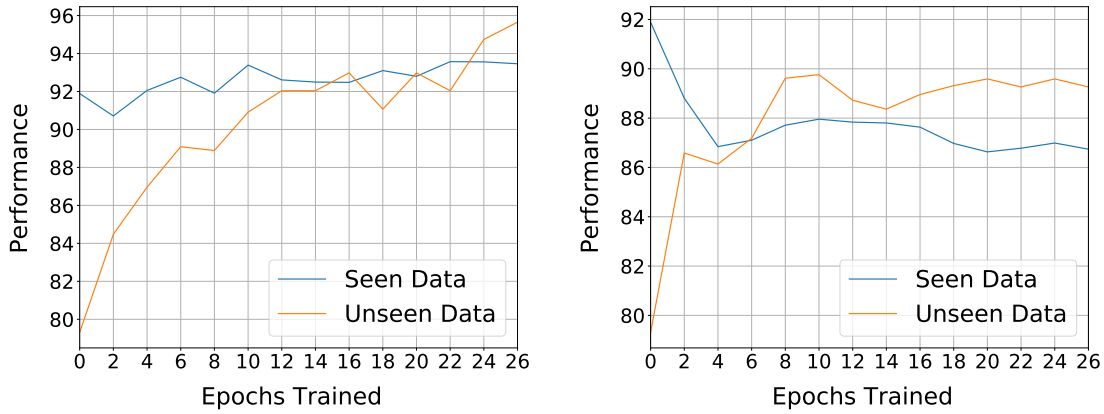
products on the Web (= products having IDs which have not been contained in previous crawls) as well as to gather heterogeneous offers for these products. The crawled offers for new products can subsequently be used for the maintenance of the existing product matchers.

In the following, three methods of adapting the product matcher to previously unseen products are explored. Firstly, the trained model is fine-tuned by continuing training using just the small training set containing the unseen products from Table 5. Secondly, the model is fine-tuned using the training sets for seen as well as previously unseen products, i.e. all available training data. Both fine-tuning approaches are evaluated after every two trained epochs. The optimizer for the model is reset before fine-tuning. Finally, a model is trained from scratch using all of the training data for comparison.

Figure 4 shows the results of the two fine-tuning approaches continuously evaluated on seen and unseen products for up to 26 epochs. Fine-tuning the model using only training data for the unseen products achieves around 90% F1 after 8 epochs, while performance on the seen products drops to around 89% F1, probably due to the network *forgetting* patterns learned for them because of noisy weight updates. When training using the combined training

Table 6: Results for seen and unseen new products

Model	Seen				Unseen				
	P	R	F1	σ	P	R	F1	σ	$\Delta F1$
Word Co-Occurrence	84.9	81.7	83.3	-	72.0	30.5	42.9	-	40.4
Magellan	73.1	64.3	68.4	-	70.0	59.3	64.2	-	4.2
Deepmatcher	92.8	91.0	91.9	0.5	76.2	75.7	75.9	2.9	12.6



(a) Fine-tuning on full set **(b) Fine-tuning on new data only**

Figure 4: Fine-tuning product matchers

sets, the performance on the seen products does not experience this drop and can even further increase by a small amount. For the unseen products, good performance of 90% F1 is achieved after 10 epochs. This further increases until it can match the performance on seen products after 16 epochs.

The two fine-tuning approaches differ heavily on the size of the training set used (3,826 vs 50,942 pairs) and subsequently on training time per epoch. In our experiments, one epoch of fine-tuning on the large set took approximately 30 times longer than training one epoch on the small set, making the latter approach more relevant in time-constrained scenarios. Fully re-training the model for 15 epochs on the combined training set achieves 91% F1 on the seen and 90% F1 on the unseen products, which is marginally less performant than fine-tuning an existing model with the same training set over the same amount of epochs, making this the least desirable approach for matcher maintenance.

4 IMPACT OF NOISY TRAINING DATA

The usage of schema.org data from a large number of websites as distant supervision implies a certain level of noise in the derived labels. The results of Section 2.3 show that the 6% label-noise inherent to the WDC Product Data Corpus, as explained in Section 2.1, still allows the learning of high-performance product matchers.

In order to investigate the impact of further label-noise on the performance of deep learning and traditional methods, we perform a set of experiments with additional label-noise. For this purpose the labels of 5, 10, 20, 30 and 40% of labels in the Computers XLarge training set are randomly flipped. Both positive and negative labels are affected equally. Since the WDC Product Data Corpus contains approx. 6% inherent label noise, this serves as the minimum possible noise value for the experiments. We use the optimized standard Deepmatcher RNN model on the Computers XLarge training set for this experiment. The model is trained for 15 epochs using the same train/validation split as in the experiments presented in Section 2.3.

Table 7 shows the results of this experiment. The Magellan and word co-occurrence models are capable of handling 5% in additional noise with only marginal losses in performance. Deepmatcher already loses 10% F1 in this scenario. The addition of another 5% does impact all of them significantly, with Magellan losing 4%, word co-occurrence 5% and Deepmatcher a further 5% F1. With increasing label-noise, all models continue to drop in performance until rapid deterioration happens around the mark of 30% and 40% noise for all models. Interestingly, Deepmatcher still performs comparably better, but the difference in performance to the traditional methods is only marginal. Overall, the non deep learning models can handle up to 10% label-noise with marginal loss in performance, while Deepmatcher takes a large hit with any increase in label noise. With further addition of label-noise all models rapidly decrease in performance. For the process of automatically deriving training labels using distant supervision it is thus very important to make sure label-noise is minimized as much as possible. In our work we showed that a noise level of approx. 6% in the training data allows learning high-performance deep product matchers. Traditional methods seem to be more noise-resistant up until around 11% label-noise.

Table 7: Impact of Noisy Training Data

	Magellan			Word Co-Occur.			Deepmatcher			
Noise %	P	R	F1	P	R	F1	P	R	F1	σ
inherent	74.13	64.00	68.69	84.91	80.67	82.74	93.12	93.11	93.11	0.58
+5	74.31	63.67	68.59	85.20	78.67	81.80	84.75	83.11	83.90	0.43
+10	68.66	61.33	64.79	79.42	73.33	76.26	79.82	77.56	78.66	0.36
+20	48.24	91.33	63.13	71.38	69.00	70.17	71.83	75.67	73.69	1.48
+30	36.40	92.33	52.21	48.12	85.33	61.54	45.49	71.67	55.63	2.98
+40	30.65	91.33	45.90	31.52	63.67	42.16	35.59	62.45	45.29	1.59

5 RELATED WORK

Entity Resolution/Product Matching: Entity Resolution has a research history of more than 50 years [11]. It aims to solve the problem of deciding if two entity mentions refer to the same real-world entity. Approaches to solving the problem can be broadly divided in rule-based, crowd-based and machine learning-based, further divided into unsupervised, weakly supervised and supervised methods [7, 10]. Recent unsupervised methods in this area are AutoER [28] which relies on Generative Modelling and a graph-based approach from Zhu et al. [31]. The Magellan framework represents an end-to-end implementation of a supervised approach using attribute-wise similarity metrics as features [16]. Product Matching is a domain-specific application of Entity Resolution and has been researched for the past 30 years [17]. Some recent unsupervised matching methods to solve the problem focus on clustering [1, 13]. Other researchers frame the problem as a classification task and focus on supervised learning using e.g. attribute similarities as features [14, 18].

Deep Learning for Entity Resolution/Product Matching: Deep Learning methods started to move into the focus of the research community in 2018 with *DeepER* [9], which is considered a pioneering work in the area of entity matching with deep learning, introducing two DL networks inspired by architectures from the NLP area of research. The *Deepmatcher* framework which is used in this paper incorporates variants of these models as described by Mudgal et al. [21], who have shown that deep learning methods outperform symbolic methods on product data with highly textual attributes, which we are able to replicate. Shah et al. [25] experiment with deep learning for product matching, employing fastText for product classification, while framing the problem as multi-class classification, as well as binary classification with deep siamese networks. Barbosa [3] experimented with a combination of symbolic and sub-symbolic entity representations. More recently, Fu et al. [12] developed a combination of sub-symbolic embedding and symbolic string/number based methods that learns to choose among a set of metrics using either subsymbolic or symbolic features depending on the attribute. Kasai et al. [15] and Thirumuruganathan et.al [26] focus on transfer learning for Entity Resolution using neural approaches, the former in combination with Active Learning. More recent work follows the trend of using fully attention-based approaches to capture contextual information [30]. Especially language models following the pre-training/fine-tuning paradigm based on the Transformer architecture [27] like BERT [8] show promising results for the tasks of entity resolution and product matching. Brunner and Stockinger [6] do a preliminary analysis of the suitability of different Transformer models and Li et al. [19] propose an entity resolution system using pre-trained language models in combination with advanced pre-processing and data augmentation techniques. They evaluate their system on the datasets presented in this paper among others and show small improvements (+1%) over the end-to-end variant of Deepmatcher presented in this paper for two product categories, as well as substantial improvements (>5%) when using the small and medium training sets. They did not investigate the performance of their system for new products as well as its noise resistance.

6 CONCLUSION

Schema.org annotations in Web pages are widely used by major search engines for rendering rich snippets in search results. This paper described an alternative use of the annotations: By using schema.org annotations as training data for learning product matchers, it becomes possible to completely replace manual labeling while maintaining a matching performance above 0.9 F1. This means that the costly task of manually labeling product offers as matches or non-matches can be automated using the Semantic Web.

New products appear on the Web every day. It is thus important for product matchers to either generalize well to them or to be easily adaptable making them maintainable. To adapt trained deep product matchers, two fine-tuning methods were compared with regards to performance on old and new products, trading re-training time for F1 performance. In a real-world scenario, either of them or a hybrid may be applied depending on the use-case.

Finally, we have shown that deep product matchers can handle the noise inherent to datasets created using schema.org annotations for distant supervision. Adding more label-noise leads to quick deterioration of performance.

The utility of schema.org annotations from large numbers of websites as training data for machine learning tasks is not restricted to the use case of product matching. Other use cases include sentiment analysis, information extraction and taxonomy matching [4]. The product corpus, the training sets, the gold standards[8] as well as the code[9] for replicating the experiments presented in this paper are available for public download and we invite interested researchers to use them to compare their approaches to the results presented in this paper.

[8] http://webdatacommons.org/largescaleproductcorpus/v2/
[9] https://github.com/Weyoun2211/wdc-lspc-v2

The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant INST 35/1134-1 FUGG.

REFERENCES

[1] L. Akritidis and P. Bozanis. 2018. Effective Unsupervised Matching of Product Titles with K-Combinations and Permutations. In *2018 Innovations in Intelligent Systems and Applications (INISTA)*. 1–10.

[2] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2019. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017*.

[3] Luciano Barbosa. 2019. Learning Representations of Web Entities for Entity Resolution. *International Journal of Web Information Systems* 15, 3 (2019), 346–358.

[4] Christian Bizer, Anna Primpeli, and Ralph Peeters. 2019. Using the Semantic Web as a Source of Training Data. *Datenbank-Spektrum* 19, 2 (2019), 127–135.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (Dec. 2017), 135–146.

[6] Ursin Brunner and Kurt Stockinger. 2020. Entity Matching with Transformer Architectures - a Step Forward in Data Integration. In *International Conference on Extending Database Technology, 30 March-2 April 2020*.

[7] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2019. End-to-End Entity Resolution for Big Data: A Survey. *arXiv:1905.06397 [cs]* (2019).

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the Association for Computational Linguistics*. 4171–4186.

[9] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proc. VLDB Endow.* 11, 11 (2018), 1454–1467.

[10] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1 (2007), 1–16.

[11] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[12] Cheng Fu, Xianpei Han, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. End-to-End Multi-Perspective Matching for Entity Resolution. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 4961–4967.

[13] Vishrawas Gopalakrishnan, Suresh Parthasarathy Iyengar, Amit Madaan, Rajeev Rastogi, and Srinivasan Sengamedu. 2012. Matching Product Titles Using Web-Based Enrichment. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. 605–614.

[14] Anitha Kannan, Inmar E. Givoni, Rakesh Agrawal, and Ariel Fuxman. 2011. Matching Unstructured Product Offers to Structured Product Specifications. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 404–412.

[15] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-Resource Deep Entity Resolution with Transfer and Active Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5851–5861.

[16] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *Proc. VLDB Endow.* 9, 12 (2016), 1197–1208.

[17] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.* 3, 1-2 (2010), 484–493.

[18] Hanna Köpcke, Andreas Thor, Stefan Thomas, and Erhard Rahm. 2012. Tailoring entity resolution for matching product offers. In *Proceedings of the 15th International Conference on Extending Database Technology*. 545–550.

[19] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *arXiv:2004.00584 [cs]* (April 2020).

[20] Robert Meusel and Heiko Paulheim. 2015. Heuristics for Fixing Common Errors in Deployed Schema.Org Microdata. In *Proceedings of the 12th European Semantic Web Conference - Volume 9088*. 152–168.

[21] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.

[22] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2249–2255.

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2011. Scikit-Learn: Machine Learning in Python. *Journal of machine learning research* 12 (2011), 2825–2830.

[24] Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC training dataset and gold standard for large-scale product matching. In *Workshop on e-Commerce and NLP (ECNLP2019), Companion Proceedings of WWW*. 381–386.

[25] Kashif Shah, Selcuk Kopru, and Jean David Ruvini. 2018. Neural Network Based Extreme Classification and Similarity Models for Product Matching. In *Proceedings of the 2018 Conference of the Association for Computational Linguistics, Volume 3 (Industry Papers)*. 8–15.

[26] Saravanan Thirumuruganathan, Shameem A. Puthiya Parambath, Mourad Ouzzani, Nan Tang, and Shafiq Joty. 2018. Reuse and Adaptation for Entity Resolution through Transfer Learning. *arXiv:1809.11084 [cs, stat]* (2018).

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 6000–6010.

[28] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2019. AutoER: Automated Entity Resolution Using Generative Modelling. *arXiv:1908.06049 [cs]* (Aug. 2019).

[29] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Product Knowledge Graph Embedding for E-Commerce. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20)*. 672–680.

[30] Dongxiang Zhang, Yuyang Nie, Sai Wu, Yanyan Shen, and Kian-Lee Tan. 2020. Multi-Context Attention for Entity Matching. In *Proceedings of The Web Conference 2020 (WWW '20)*. 2634–2640.

[31] Linhong Zhu, Majid Ghasemi-Gol, Pedro Szekely, Aram Galstyan, and Craig A. Knoblock. 2016. Unsupervised Entity Resolution on Multi-Type Graphs. In *The Semantic Web – ISWC 2016*. 649–667.