Undefined 0 (2016) 1–0
IOS Press

A Machine Learning Approach for Product Matching and Categorization

Use case: Enriching Product Ads with Semantic Structured Data

Petar Ristoski [a], Petar Petrovski [a], Peter Mika [b], Heiko Paulheim [a],
[a] *Data and Web Science Group, University of Mannheim, B6, 26, 68159 Mannheim*
E-mail: petar.ristoski@informatik.uni-mannheim.de,petar@informatik.uni-mannheim.de, heiko@informatik.uni-mannheim.de
[b] *Yahoo Labs, London, UK*
E-mail: pmika@yahoo-inc.com

Abstract. Consumers today have the option to purchase products from thousands of e-shops. However, the completeness of the product specifications and the taxonomies used for organizing the products differ across different e-shops. To improve the consumer experience, e.g., by allowing for easily comparing offers by different vendors, approaches for product integration on the Web are needed. In this paper, we present an approach that leverages neural language models and deep learning techniques in combination with standard classification approaches for product matching and categorization. In our approach we use structured product data as supervision for training feature extraction models able to extract attribute-value pairs from textual product descriptions. To minimize the need for lots of data for supervision, we use neural language models to produce word embeddings from large quantities of publicly available product data marked up with Microdata, which boost the performance of the feature extraction model, thus leading to better product matching and categorization performances. Furthermore, we use a deep Convolutional Neural Network to produce image embeddings from product images, which further improve the results on both tasks.

Keywords: Product Data, Data Integration, Vector Space Embeddings, Deep Learning, Microdata, schema.org

1. Introduction

In recent years, with the advancements of the Internet and e-business services, the amount of products sold through e-shops has grown rapidly. A recent study estimates that the total e-commerce retail sales for the fourth quarter of 2015 in the USA only were 89.1 billion dollars [5]. Yet, there still is one big issue in the process of product search and purchase that consumers have to deal with. The same product may be found on many different e-shops, but the information about the products offers greatly differ across different e-shops. Furthermore, there are no global identifiers for products, and offers are most often not interlinked among each other. Therefore, there is no simple way for the consumers to find all the necessary information and best prices for the products they search for. To offer a better user experience, there are many products aggregators, like Google Product Search[1], PriceGrabber[2], and Shopzila[3], trying to integrate and categorize products from many e-shops and many different merchants. However, the task of identifying offers of the same product across thousands of e-shops and integrating the information into a single representation is not trivial. Furthermore, to allow users better navigation and product search, product aggregators have to deal with the task of organizing all the products according to a product taxonomy.

[1] http://www.google.com/shopping
[2] http://www.pricegrabber.com/
[3] http://www.shopzilla.com/

To support the integration process, e-shops are increasingly 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 that leverages neural language models and deep learning techniques in combination with standard classification approaches for product matching and categorization from HTML annotations. We focus on data annotated with the Microdata markup format using the schema.org vocabulary. Recent works [24,25] 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. Although considered *structured* annotations, the empirical studies [24,31] have shown that the vast majority of products are annotated only with a name and a textual description, i.e., they are rather unstructured than structured data. This helps identifying the relevant parts on the Website, but leads to rather shallow textual data, which has to be tackled with methods for unstructured data.

In a previous work [33], we have proposed an approach for enriching structured product ads with 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 identified the Microdata dataset as a valuable source for enriching existing structured product ads with new attributes.

In this paper, we enhance the existing approach using neural language models and deep learning techniques. We inspect the use of different models for three tasks: (1) Matching products described with structured annotations from the Web, (2) Categorizing products, and (3) enriching an existing product database with product data from the Web. For those tasks, we employ Conditional Random Fields for extracting product attributes from textual descriptions, as well as Convolutional Neural Networks to produce embeddings of product images. We show that neural word embeddings outperform baseline approaches for product matching and produce results comparable to those of supervised approaches for product categorization. Furthermore, we show that image embeddings can only provide a weak signal for product matching, but a strong signal for product classification.

The rest of this paper is structured as follows. In section 2, we give an overview of related work. In section 3 and section 4, we introduce our methodology for product matching and categorization, respectively. In section 5, we present the results of matching unstructured product descriptions, followed by the evaluation of the product ads enrichment with metadata extracted from HTML annotations in section 6. In section 7 we present the results of the product categorization approach. We conclude with a summary and an outlook on future work.

2. Related Work

Both product matching and product categorization for the Web have been explored with various approaches and methods within the last years.

2.1. Product matching

Since there are no global identifiers for products, and links between different e-commerce Web pages are also scarce, finding out whether two offers on different Web pages are referring to the same product is a non-trivial task. Therefore, *product matching* deals with identifying pairs or sets of identical products.

Ghani et al. [7] first presented enriched product databases with attribute-value pairs extracted from product descriptions on the Web, by using Naive Bayes in combination with a 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 name (e.g., "color") and the attribute value (e.g., "black") appear in the text.

The *XploreProducts.com* platform detailed in [36] integrates products from different e-shops annotated using RDFa annotations. The approach is based on several string similarity functions for product matching. The approach is extended by using a hybrid similarity method and hierarchical clustering for matching products from multiple e-shops [2].

Kannan et al. [12] use the Bing products catalog to build a dictionary-based feature extraction model. Later, the features of the products are used to train a Logistic Regression model for matching product offers to the Bing shopping data. Another machine learning approach for matching products data is proposed in [15]. First, several features are extracted from the title and the description of the products using manually written regular expressions. In contrast, named

entity recognition based feature extraction models are developed in [23] and [33]. Both approaches use a CRF model for feature extraction, however [23] has a limited ability to extract explicit attribute-value pairs, which is improved upon in [33]. Comparably, in this paper we enhance the CRF model with continues features, thus boosting the CRF's ability to recognize a larger number of attribute-value pairs.

The first approach to perform products matching on Microdata annotations is presented in [31], based on the Silk rule learning framework [11]. To do so, different combinations of features (e.g. bag of words, dictionary-based, regular expressions etc.) from the product descriptions are used. The work has been extended in [32], where the authors developed a genetic algorithm for learning regular expressions for extracting attribute-value pairs from products.

The authors of [29] perform product matching on a dataset of the Bing search engine. In their approach, the authors use historical knowledge to generate the attributes and to perform schema matching. In particular, they visit the merchant's web page to compare the values of the products in the catalog with the values on the web page, converting the problem to a standard table schema matching problem. Next, the authors use instance-based schema matching to align the attributes' names in the catalog to the ones on the merchant's web page.

In [8] the authors propose an unsupervised web-based enrichment approach for the purpose of product matching. They start with enriching the title of the product with tokens retrieved using a web search engine, i.e., they use the title of the product as a query to a search engine, and the top K returned tokens are used for the enrichment. To identify the relevance of each token in the title, they again use web search, i.e., the token that returns more results is more relevant for the given product. The pairwise product matching is performed based on cosine similarity, using prefix filtering techniques as a blocking approach.

A similar approach for enriching product descriptions with tokens using web search engine is proposed in [20]. The authors propose a similar approach to [8]. The approach first enriches the offer's title with tokens using web search engine. Then, it uses Community Detection for an approximate matching approach which is responsible for computing the distance between the pairs, computed based on the strength of the constructed "social" network between the private tokens. The approach is evaluated on the Abt-Buy dataset[4] and a small custom dataset.

While the discussed studies above have implemented diverse approaches (classifiers, genetic programming, string similarities), the feature extraction techniques used are mostly dependent on a supervised dictionary-based approach.

To reduce the labelling effort required by supervised approaches, in this paper we rely on neural word embeddings and CNNs, however, a full review of deep learning and neural networks in general is beyond the scope of this paper; please see [34]. Instead, we provide a review of the most relevant work for product feature extraction that use neural word embeddings and deep learning.

Recently, a handful of approaches employ word embeddings for getting features from product data for the problem of matching, as well as other problems concerning product data. The approach by Grbovic et al. [9] discusses the problem of product recommendations as a part of online advertisements. To perform recommendations, the authors use word2vec [27] to create product embeddings from product titles for product-to-product predictions, as well as paragraph2vec [18] to create user embeddings for user-to-products predictions. Similarly, in [39], the authors present a product recommendation system for microblogging websites where the main idea is that users and products can be represented in the same feature space by employing word2vec to calculate the feature vectors.

In [35], the authors present a system that automatically estimates the quality of machine translated segments of product offers. The authors again use word embeddings, specifically paragraph2vec [18], to learn feature vectors from the product title. These vectors are then used to that predict post-edition effort (HTER) on products from three different categories.

To the best of our knowledge, product matching based on image features has been applied in couple of domains including apparel and interior design. In [14], the authors propose an approach that matches clothing in two different settings: street photos vs. online shop photos. Specifically, similarly to our approach for extracting image features they use a CNN to learn image embeddings which then are used as an input in a binary classification task. Another approach tackling the same problem is introduced in [38]. Differently from

[4] http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution

[14], the authors propose the usage of Siamese Deep Networks (SDN) to bridge the domain gap between the user photos and the online product images. Additionally, the authors propose an alternative of the popular contrastive loss used in SDN, namely robust contrastive loss, where the penalty on positive pairs is lowered to alleviate over-fitting.

It is worth mentioning that with the rapid development of deep learning and neural nets for image processing, there are several approaches to recommend clothing based on images [21].

In [1], the authors propose a similar approach to the one in [38] in the domain of interior design. Namely, to obtain the image embeddings, they use SDN on pairs of images. Several training architectures including repurposing object classifiers, using siamese networks, and using multitask learning are proposed.

Since the product characteristics in the domains covered by the above-mentioned works are primarily visual, it is not surprising that image-based matching methods work considerably well. This, however, cannot be easily assumed for all domains. For example, product characteristics in domains like electronics are mainly textual (description, specification, technical fact sheets), and therefore, in this paper we cannot limit our methods to purely learning image features.

2.2. Product classification

Product classification deals with assigning a set of labels from a product hierarchy to a product. Since not all web sites use a hierarchy, and those who use one are unlikely to use the same, a unified classification of products from different web sites is needed to provide the user with useful browsing and searching functionalities.

While there are several approaches concerned with product data categorization [29,12,31,36], the approach by Kozareva [16] is one of the only few approaches that use neural text embeddings as features for the classification task. Specifically, the author learn a linear classification model and use a commination of lexical features, LDA topics and text embeddings as the input feature vector. The approach is extended in [10] where the authors, propose usage of a two-level ensemble instead the linear classification model from [16].

Meusel at el. [25] is the most recent approach for exploiting Microdata annotations for categorization of product data. In that approach, the authors exploit the already assigned `s:Category`[5] property to develop distantly supervised approaches to map the products to set of target categories from an existing product catalog.

Table 1

An example of a structured product record

Product Title	Apple iPhone 6s 64GB 9.3 iOS gold	
Product Description	Combining style and functionality, the iPhone 6s sports an elegant design and is available in gold. It has a 4.7-inch...	
Product URL	http://www.example.com/item-iphone6s	
	Attribute Name	Attribute Value
Attributes	Brand	Apple
	Phone Type	iPhone 6s
	Memory	64GB
	Operating System	9.3 iOS
	Color	gold

Although there are a lot of approaches for products categorization based on text features, only a few are using image features for the given task. Kannan et al. [13] proposed one of the first approaches for product categorization that besides text features uses image features. The approach is based on Confusion Driven Probabilistic Fusion++, which is cognizant of the disparity in the discriminative power of different types of signals and hence makes use of the confusion matrix of dominant signal (text) to prudently leverage the weaker signal (image), for an improved performance. In our paper, we follow the same setup for building the classifier, however, we use a different image feature extraction technique, i.e., we use deep neural nets image embeddings, while they use simple spatial image features.

3. Product Matching Approach

In our approach for product matching, we use various feature extraction methods to derive a set of useful features for the product matching task, i.e., for identifying identical products.

3.1. Problem Statement

We define the problem of product matching similarly to the problem statement defined in Kannan et al [12]. We have a database A of structured products and a dataset of unstructured product descriptions P extracted from the Web. Every record $a \in A$ consists

[5] In this paper, `s` is used as a shorthand notation for the `http://schema.org/` vocabulary

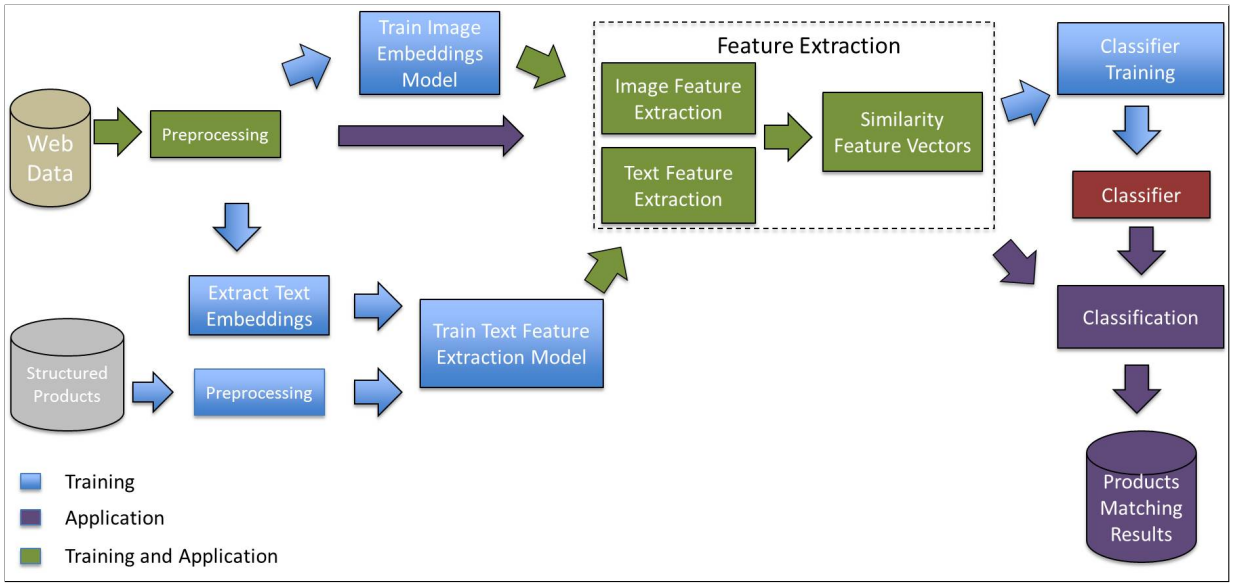


Fig. 1. System architecture overview for the product matching task

of a title, description, URL, and a set of attribute-value pairs extracted from the title of the product, where the attributes are numeric, categorical, or free-text (an example of such a product record is given in Table 1). Every record $p \in P$ consists of a title and a description as unstructured textual fields, and a product image. Our objective is to use the structured information from the product set A as supervision for identifying duplicate records in P, in combination with neural text embeddings extracted from all the records in P. 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

Our approach for products matching consists 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 workflow runs in two phases: *training* and *application*. The training phase starts with preprocessing both the structured and the unstructured Web product descriptions. Then, we build four feature extraction models as follows:

Dictionary-Based We build a dictionary of the product attributes and their values present in the structured product descriptions.

Conditional Random Field (CRF) We build a CRF model with a set of discrete features.

CRF with Text Embeddings In order to handle the dynamic text patterns in product descriptions we enhance the training of the preceding CRF model with text embedding features. This approach is detailed in section 3.3.3.

Image Feature Extraction Model In addition to the textual features, we furthermore build an image embeddings model.

It is important that while the feature extraction part contains many time-consuming parts, like, training the CRF and image embeddings models, those steps are only taken once, and can be performed in an offline pre-processing step. Thus, at run-time, products can be matched fast, since the pre-trained CRF and embeddings models only need to be applied at that stage. The approaches are detailed in section 3.3.

Next, we manually label a small training set of matching and non-matching unstructured pairs of product descriptions. Subsequently, we calculate the similarity feature vectors for the labeled training product pairs (section 3.4). In the final step, the similarity

feature vectors are used to train a classification model for distinguishing matching and non-matching pairs (section 3.5). After the training phase is over, we have a trained feature extraction model and a classification model.

In the application phase, we generate a set *M* of all possible candidate matching pairs, which leads to a large number of candidates i.e., $|M| = |P| * (|P| - 1)/2$. 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.

3.3. Feature Extraction

We pursue different approaches for extracting features from the structured and unstructured product descriptions.

3.3.1. Dictionary-Based Approach

To implement the dictionary-based approach we were motivated by the approach described by Kannan et al [12]. We use the database *A* of structured products to generate a dictionary of attributes and values. Let *F* represent all the attributes present in the product database *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*. In the example depicted in table 1, *f(gold)* would yield *color*, *f(apple)* would yield *brand*, etc.

Then, to extract features from a given product description $p \in P$, we generate all possible token 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, and ties with multiple n-grams of the same maximal lengths are resolved by random selection.

3.3.2. Conditional Random Fields

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[6] in order to train product specific CRF models [6]. To train the CRF model, we use a comprehensive set of discrete features that comes from the standard distribution of the Stanford NER model: 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).

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

3.3.3. Conditional Random Fields with Continuous Features

While CRF delivers rather good results, it requires a lot of labeled and diverse data. This becomes a challenge when new products are emerging on the market everyday, and merchants are changing the textual pattern of the product description. To address this challenge, we make use of the available unstructured data. Specifically, we use neural language modeling to extract word embeddings from the unstructured product description. The goal of such approaches is to estimate the likelihood of a specific sequence of words appearing in a corpus, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent. Examples for such approaches are hierarchical log-bilinear models [28], SENNA [4], GloVe [30] and word2vec [26,27]. In our approach we use the well-established word2vec model, which is the most popular and widely used approach for word embeddings. word2vec is a particularly computationally-efficient two-layer neural net model for learning word embeddings from raw text. There are two different algorithms, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. The Skip-Gram model predicts contextual words given window of size n, while CBOW predicts the current word, given the context in the window.

Projecting such latent representation of words into a lower dimensional feature space shows that semantically similar words appear closer to each other (see section 5.3). Meaning that values of different product attributes are represented close to each other in the latent feature space. Therefore, we follow the approach presented in [37] to implement a CRF model that beside the previously described discrete features, includes the word embeddings as features for the CRF model. Training the CRF model using word embeddings makes the model more robust, meaning it is able to detect attributes that have not been seen during the training phase with higher precision. For example, given the same structured product description from section 3.1, "*Apple[Brand] iPhone 6s[phone type] 64GB[Memory] 9.3 iOS[Operating System] gold[Color]*", the word embeddings can assign a priori labels on the words in a completely un-

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

Table 2

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
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

seen product description by the CRF model, e.g., given the product description "New Samsung Galaxy S4 GT-19505 16GB 5.0 inches Android White", the word embedding model shows high semantic similarity between the words "Samsung" and "Apple", therefore we can a priori assign the label *brand* to the word "Samsung". Same applies for the rest of the attributes.

3.3.4. 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 use the same attribute normalization pipeline presented in [33], i.e., attribute type detection, string normalization, number and number with unit of measurement normalization.

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

3.3.5. Image Feature Extraction

Many e-shops use the same or similar image for identical products. Therefore, the image can be used as an indicator for identifying matching products. In this work, we use one of the most popular image processing techniques, i.e., deep Convolutional Neural Networks (CNNs) [19]. Usually, CNN models consist of several convolutional layers connected in a row, followed by fully connected layers. In each convolutional layer, several small filters are convolved on the input image. The weights of each filter are randomly initialized, thus, different filters get triggered by different features in the image, e.g., some might get triggered by a specific shape in the image, others by a specific color, etc. As this might produce a large number of features, each convolutional layer is usually connected to a pooling layer, which reduces the number of features by subsampling. The output of the convolutional layer is connected to a standard Feed-Forward Neural Net to solve a particular task.

In this work, we adopt the architecture proposed in [17]. The architecture of the model is shown in Figure 3. The network consists of five convolutional layers, followed by three fully connected layers. The number and size of the used filters in each convolutional layer is shown in the Figure. All neurons have a Rectified Linear Unit (ReLU) activation function to accelerate the learning process. The output of the last fully-connected layer is fed to a N-way softmax, where N is the number of labels in the dataset.

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

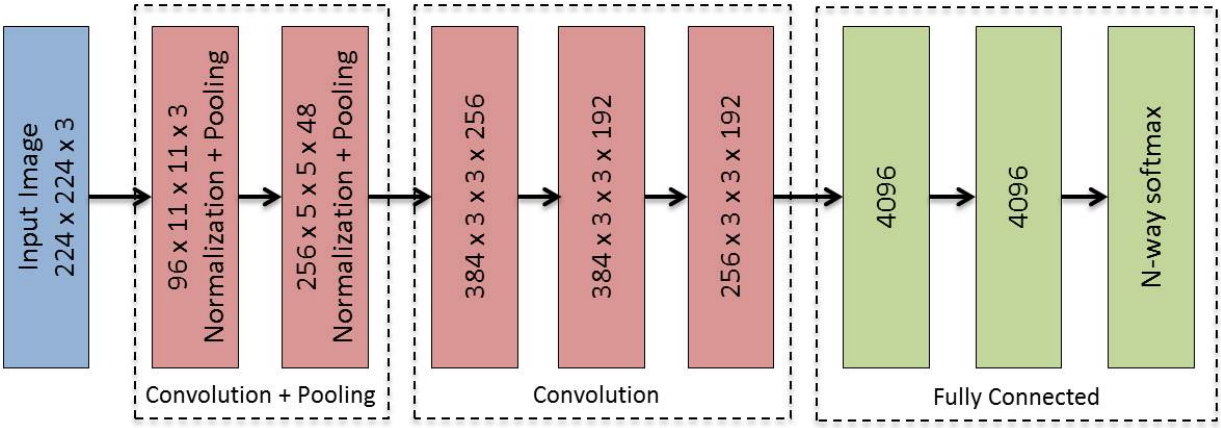


Fig. 3. Convolutional Neural Nets architecture

3.4. Calculating Similarity of 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, including the image vectors. 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 by the attribute vectors $F_{p1} = \{f_1v_1, f_2v_1, ..., f_nv_1\}$ and $F_{p2} = \{f_1v_2, f_2v_2, ..., f_nv_2\}$, 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 using a mixed measure based on the attribute data type as shown in Equation 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.

3.5. Classification Approaches

We treat the product matching problem as a two-class classification problem: given a pair of products, they are either matching or non-matching. Since there are far more non-matching pairs than matching pairs, this classification problem is heavily imbalanced.

Once the similarity feature vectors are calculated, we train four different classifiers that are commonly used for such a task: (i) Random Forest, (ii) Support Vector Machines (SVM), (iii) Naive Bayes and (iv) Logistic Regression. We try to find the most suitable classifier able to address the problem of high class imbalance in the product matching task, i.e., there only a few matching product pairs (positive class), and many non-matching product pairs (negative class).

4. Product Categorization Approach

The approach for product categorization consists of two steps: (i) feature extraction and (ii) classification. We use supervised and unsupervised approaches for feature extraction.

4.1. Feature Extraction

We use similar feature sets as we use for product matching.

4.1.1. Supervised Text-based Feature Extraction

For the task of product categorization we use a dictionary-based approach for feature extraction from product descriptions [33][7]. The dictionary-based approach starts by building a dictionary of all attribute-value pairs used in the structured product dataset. 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 eliminated tokens shorter than 3 characters. The terms of each feature are concatenated with the feature name e.g. for the value *blue* for the feature *color*, the final value will be *blue-color*.

[7] 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 product dataset should be trained.

4.1.2. Unsupervised Text-based Feature Extraction

Similarly to section 3.3.3, we use neural language modeling to extract text embeddings from the unstructured product descriptions in order to overcome the challenge of the diversity of new products and their ever changing description. Since our product descriptions represent whole documents for the classification task, we construct text embeddings given the whole document. The most prominent neural language model for text embedding on a document level is paragraph2vec [18], an extension to word2vec. As with word2vec, paragraph2vec relies on two algorithms: Distributed Memory (DM), which corresponds to CBOW, and Distributed Bag-of-Words (DBOW), which corresponds to the skip-gram model. paragraph2vec is based on the same computationally-efficient two-layer neural network architecture. However, to be able to represent document embeddings paragraph2vec maps each document to a unique paragraph vector. In DM, this paragraph vector is averaged/summed with the word vectors, making the paragraph vector a memory of what is missing from the current context. Contrary to DM, DBOW ignores the context words in the input and instead forms a classification task given the paragraph vector and randomly selected words from a text window sample.

4.1.3. Unsupervised Image-based Feature Extraction

We use the same CNN model for extracting image embeddings as described in section 3.3.5. In this case, we use the image vectors as such, i.e., we use the complete image vectors for the task of image classification.

4.2. Classification

For each of the feature extraction approaches in the end we use the feature vectors to build a classification model, i.e., Naive Bayes (NB), Support Vector Machines (SVM), Random Forest (RF) and k-Nearest Neighbors (KNN), where k=1.

While we treated product matching as a binary classification problem, the problem of product categorization is a multi-class problem, since the number of product categories is much larger than two.

5. Products Matching Evaluation

In this section, we evaluate the product matching pipeline described in section 3. We start by describing the used datasets 5.1, i.e., the database of structured products used for supervision, and the dataset of unstructured products descriptions. Next, as the text feature extraction model is the core module of our pipeline, in section 5.2, we evaluate the performance of our CRF model built only on discrete features and the CRF model built with discrete features and continuous features from word embeddings. In section 5.3, we analyze the semantics of the word vector representations in order to give deeper insights of their relevance for training the CRF model. In section 5.4, we describe the experiment setup, followed by the final results of the entire product matching pipeline in section 5.5. In the final section we perform error analysis of the proposed approach.

Table 3
Datasets used in the evaluation

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

5.1. Datasets

For the evaluation, we use Yahoo's Gemini Product Ads (GPA) for supervision[8], and we use a subset of the WebDataCommons (WDC) extraction[9].

5.1.1. Product 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 3,330 laptops, 3,476 TVs, and 3,372 mobile phones. 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 and the CRF feature extraction models.

5.1.2. Unstructured Product Offers - WDC Microdata Dataset

We use a recent extraction of WebDataCommons, which 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[10]. In this dataset, we focus on product entities annotated with Microdata using the schema.org vocabulary. To

[8] Note: we could use any database of structured products for the given task

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

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

Table 4
CRF evaluation on GPA data

				CRF			CRFemb			
Dataset	#training	#test	#atts.	P	R	F1	P	R	F1	ΔF1
Laptops	2,330	1,000	27	94.81	93.35	**94.08**	93.67	93.2	93.43	-0.65
Televisions	2,436	1,040	35	96.2	94.31	95.25	96.41	94.85	**95.62**	0.37
Mobile Phones	2,220	1,010	35	97.62	96.13	**96.87**	96.72	95.84	96.27	-0.6

Table 5
CRF evaluation on WDC data. The best results are marked with bold

		CRF			CRFemb			
Dataset	#test	P	R	F1	P	R	F1	ΔF1
Laptops	50	57.71	53.95	55.76	71.91	64.19	**67.83**	12.06
Televisions	50	84.62	56.57	67.81	88.24	77.14	**82.32**	14.51
Mobile Phones	50	72.83	60.12	65.87	85.03	64.25	**73.2**	7.33

do so, we use a sub-set of entities annotated with `s:Product`. The dataset contains *288,082,823* entities in total, or *2,829,523,589* RDF quads. In our approach, we make use of the properties `s:name` and `s:description` for extracting attribute-value pairs, and the `s:Product/image` for image embeddings. As discussed above, although WDC is usually considered *structured* data, most of our entities are only described by those three attributes (i.e., name, description, and image), and is thus rather weakly structured.

To evaluate the approach, we have built a gold standard from the WDC dataset on three categories in the *Electronics domain*, i.e., TVs, mobile phones and laptops. We have imposed some constraints on the entities we select: (i) the products must contain an `s:name` and an `s:description` property in English language[11], (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., it contains enough information to pinpoint the exact product. The dataset can be found online[12].

The gold standard has been 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 dataset is annotated by two annotators independently, where the conflicts have been resolved jointly. The number of entities, as well as the number of matching and non-matching pairs for each of the datasets, is shown in Table 3.

To build the text embeddings models, we select two subsets of the unstructured data: (i) the subset containing only the `s:name`, (ii) and the subset containing the `s:name` and `s:description`. Additionally, we preprocess the WDC dataset even further, i.e., we tokenize the input such that complex product IDs (ex. G70-35-80Q5002SGE) are considered as one word, and we apply a WordNet[13] lemmatizer.

5.2. CRF Evaluation

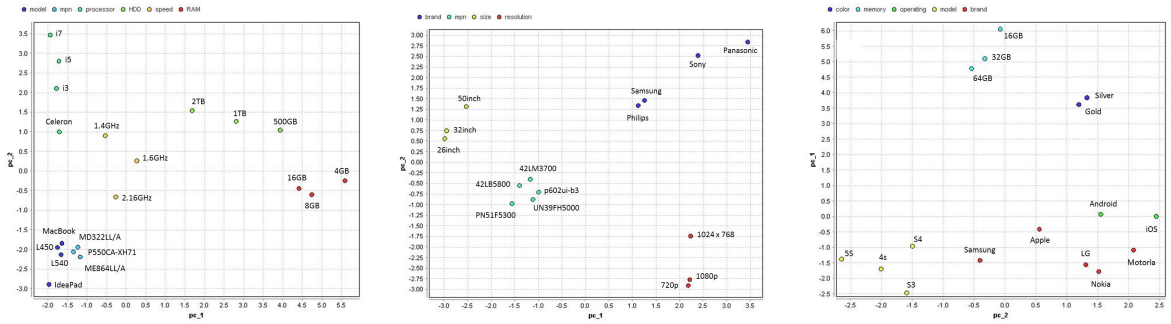
Extracting correct attribute-value pairs with high coverage from the product text descriptions is essential for the task of products matching. Thus, we compare the performances of the CRF model built only on discrete features (just CRF in the following) with the CRF model built on discrete and continuous features from word embeddings (CRFemb in the following). To train the CRFemb model, we build both CBOW and Skip-Gram neural models for word embeddings. We train the models on the complete WDC and GPA datasets. We experimented with different parameters for the models, to finally select the following parameters' values: window size = 5; number of iterations = 10; negative sampling for optimization; negative samples = 10; with average input vector for CBOW; vectors size = 200. We used the *gensim* implementation[14] for model training. All models, as well as an extended

[11] All the product descriptions extracted from English top-level domains are considered to be in English language, i.e., "com", "org", "net", "eu" and "uk".

[12] `http://data.dws.informatik.uni-mannheim.de/gpawdc/Gold/`

[13] https://wordnet.princeton.edu/

[14] `https://radimrehurek.com/gensim/`



a) *Laptop attributes* **b)** *TV attributes* **c)** *Mobile attributes*

Fig. 4. Two-dimensional PCA projection of the 200-dimensional Skip-Gram vectors for different product attributes, extracted from the product titles.

overview of the parameters are available for download online[15].

We evaluate both CRF models on the database of structured product ads in a split validation setting. 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 testing, and the number of attributes are shown in Table 4.

The results show that there is no significant difference in the performance of the two models. However, we believe that the models might be overfitted on the structured product ads. The reason for this assumption is that the structured product ads are rather clean, and ads coming from the same retailer follow similar patterns. Consequently, we are not able to observe significant difference in performance between the two models. Therefore, we manually labeled all the attribute-value pairs in 50 randomly selected products of each category on WDC products[16]. Again, the dataset has been annotated by two annotators independently, where the conflicts were jointly resolved. The evaluation results of both models on the new data are shown in Table 5. The results, indeed, show significant difference in performance between the two models. The classifiers' overall accuracy results are significantly different according to the McNemar's test [22], with a significance level $p < 0.001$. The reason for such difference in performance is that the CRFemb model is more robust because of the word embedding features, allowing it to identify attribute-value pairs in "dirty" data from the Web.

5.3. Semantics of Vector Representations

To analyze the semantics of the word vector representations, and get deeper insights of their relevance for training the CRFemb model, we employ Principal Component Analysis (PCA) to project the"high"-dimensional attribute vectors in a two dimensional feature space.[17] We select several attribute-value pairs for each category and plotted them on a 2D scatter plot, as shown in Figure 4. The figure illustrates the ability of the model to automatically organize values of different attributes, i.e., values of the same attributes are usually positioned close to each other. Also, the model was able to automatically order semantically similaar ttributes (for instance processor is closest to speed shown in Figure 4a).

5.4. Experiment Setup

To evaluate the effectiveness of the product matching approach we use the standard performance measures, i.e., Precision (P), Recall (R) and F-score (F1). The precision states the percent of correct matching product pairs predictions made by the classier; The recall states the percent of the matching product pairs that the classifier identifies; The F-score measures the trade off between the precision and the recall, calculated as a harmonic mean of the both. 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.[18]. All experiments were run on a Mac-

[15] http://data.dws.informatik.uni-mannheim.de/gpawdc/DL

[16] The dataset can be found online `http://data.dws.informatik.uni-mannheim.de/gpawdc/Gold/`

[17] Please note that we employ PCA solely to create the plots depicted in Fig. 4. PCA has *not* been used anywhere in the product matching and/or categorization pipelines.

[18] `http://www.rapidminer.com/`

Table 6

Product matching baseline results using cosine similarity on TF-IDF, paragraph2vec and Silk. The best results for each dataset are marked in bold.

	Laptops			Television			Mobile Phones		
Approach	P	R	F1	P	R	F1	P	R	F1
Cosine similarity TF-IDF (title)	29.6	39.7	33.9	29.9	21.9	25.3	37.5	38.3	37.9
Doc2Vec title (DBOW 50)	35.8	32.9	34.3	31.2	23.4	26.7	73.0	32.8	45.3
Silk	28.4	80.8	**42.0**	50.1	91.1	**64.6**	40.6	84.0	**54.7**

Table 7

Product Matching Performance. The best results for each dataset are marked in bold.

Model	Dictionary			CRF			CRFemb			CRFemb + Image emb.		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Laptops												
Random Forest	74.1	49.81	59.56	81.52	51.29	62.96	72.85	71.33	72.08	81.82	66.47	**73.35**
SVM	72.49	50.29	59.56	69.42	56.48	62.28	84.99	54.76	66.61	79.50	60.62	68.79
Naive Bayes	11.61	55.95	19.23	6.20	75.81	11.46	7.11	83.57	12.87	6.31	84.29	11.56
Logistic Regression	56.06	47.52	51.44	65.05	51.02	57.18	51.00	79.20	62.05	53.00	78.30	63.21
Televisions												
Random Forest	80.53	74.13	77.20	92.10	73.90	82.00	83.65	82.74	83.19	93.06	78.16	**84.96**
SVM	62.20	62.97	62.58	84.90	63.90	72.90	84.90	71.68	77.73	85.71	73.03	78.86
Naive Bayes	7.43	94.12	13.74	7.26	89.37	13.36	5.45	95.78	10.28	6.31	93.66	11.72
Logistic Regression	23.22	81.13	36.14	51.92	76.93	61.94	52.74	77.62	62.80	51.59	79.82	62.67
Mobile Phones												
Random Forest	38.6	65.92	48.68	88.49	75.60	81.53	89.42	77.01	82.75	90.56	77.06	**83.27**
SVM	41.60	16.26	23.38	43.95	45.65	44.78	44.81	45.58	45.19	46.46	45.67	46.06
Naive Bayes	10.20	35.36	15.83	15.31	63.06	24.63	15.46	73.23	25.53	15.37	76.74	25.61
Logistic Regression	27.85	32.50	29.99	41.34	48.93	44.82	43.36	47.29	45.23	44.66	47.4	45.98

Book Pro with 8GB of RAM and 2.3 GHz Intel Core i7 CPU processor. We compare the results of the product matching approach when using the Dictionary approach, the CRF and the CRFemb model for feature extraction. Furthermore, we add image embeddings as additional features for the given task. To do so, we train the CNN model described in section 3.3.5, using a dataset of 8,362 products images, labeled with 303 different labels from the third level of the GS1 product catalog (see section 7.1). We use this CNN model as an image feature extraction model. To do so, for each image we use the output of the second fully connected layer, which results in a feature vector of length 4,096. Such image vectors can be used in any classification algorithm. For each pair of products we calculate the cosine similarity on the image vectors, which is included in the feature similarity vector for the given pair of products.

We compare our approach to three baselines. As a first baseline, we match the products based on a Bag-of-Words TF-IDF cosine similarity, reporting the best score on different levels of matching thresholds, i.e., we iterate the matching threshold 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. We calculated the similarity based on different combination of title and description, but the best results were delivered when using only the product title.

As a second baseline, we use the document vectors generated as explained in section 4.1.2. Moreover, we build both DM and DBOW models for each of the datasets. We experiment with different vectors size, i.e., 50, 200, 500 and 1000. We calculate the cosine similarity between each pair of vectors, and we report the best score on different levels of matching thresholds.

As a third baseline, we use the Silk Link Discovery Framework [11], an open-source tool for discovering links between data items within different data sources. The tool uses genetic programming 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.

5.5. Results

The results for the baselines are shown in Table 6. For all three approaches the best results are achieved when using only the title of the products. The best results for the paragraph2vec approach are achieved

a) *iPhone 4 16GB* **b)** *iPhone 5 16GB* **c)** *Samsung Galaxy S4 Mini*

Fig. 5. Images for Products

when using the DBOW method with 50 latent features. We can see that the documents embeddings outperform the standard BOW TF-IDF, and the Silk framework outperforms both. However, the F1-measure is not too high in all cases, limiting the utility of those baselines in many real-world applications.

Next, we show the results for products matching using the CRFemb for attribute extraction, compared to the standard CRF and the Dictionary model. The results are given in Table 7. We can note that all three approaches outperform the baseline approaches. The classifiers' overall accuracy results scores are significantly different according to the McNemar's test with $p < 0.001$. The Random Forest classifier delivers the best result for all three categories using the CRFemb feature extraction approach. 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 noteworthy that the results for the *laptops* datasets are significantly better when using the CRFemb approach, compared to the CRF and the Dictionary approach. The results are statistically significant according to the McNemar's test with $p < 0.001$. We already showed in [33] that the matching task for laptops is more challenging, because it needs more overlapping features to conclude that two products are matching[19]. The improvement of the results confirms that the CRFemb is more robust than the CRF and the Dictionary approach, and it is able to extract attributes with higher precision. Furthermore, 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 phones that were advertised in the year 2015, while the WDC dataset contains phones that were popular in 2014. Therefore, the dictionary-based approach fails to extract many attribute-value pairs, and overfits to the products contained in the GPA dataset.

From the last column of the table, we can conclude that using the image embeddings slightly improves the results. Furthermore, the feature relevance analysis using information gain shows a rather high relevance of the image similarity feature for the given task, and it is comparable with the relevance of the *brand* attribute. The image cannot be used as a strong signal for the task of product matching because a lot of similar products of the same brand have the same or similar image. For example, "iPhone 6s 16GB gold" and "iPhone 6s 64GB gold" have the same image across many e-shops. However, using such image features it is possible to distinguish products of different brands. For example, in Fig 5a is given an *iPhone 4 16GB*, in Fig 5b is given an *iPhone 5 16GB* and in Fig 5c is given an *Samsung Galaxy S4 Mini*. The cosine similarity of the image features between the first two is 0.76, while the cosine similarity between the first and third mobile phone is 0.35.

5.6. Error Analysis

In order to gain additional insights into the performance of our method, we performed a manual error analysis of its output. More precisely, we manually investigate the false positives and false negatives to identify the different categories of errors. We identify two main error categories:

[19]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.

Errors in the Approach In this category, we subsume all the errors caused by any of the modules in the approach pipeline: (i) Inability to extract relevant attribute-value pairs from the product description, because of complex description structure, typos, or abbreviations; (ii) Wrong normalization of attributes and values: inability to detect the correct attribute data type, inability to normalize the numerical and unit values; and (iii) Inability of the classification models to detect all matching pairs.

Data Errors Errors that exist in the Web data itself: (i) incomplete or simplified values; (ii) misaligned product names and descriptions; (iii) wrong product picture; and (iv) multiple products descriptions in a single description field.

6. Use Case: Enriching Product Ads with Semantic Structured Data

Product ads are a popular form of search advertizing[20] 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. An example of search advertizing is shown in Fig. 6. 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. 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. Incomplete product descriptions directly influence the performance of the advertizing engine, leading to presenting less relevant products to the user, which leads to revenue loss. Therefore it is crucial to extract all possible features

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

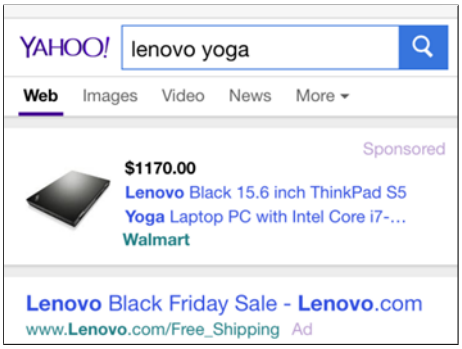


Fig. 6. An example for search advertizing for the query "lenovo yoga"

from the product ads descriptions submitted by the merchants or from external data sources.

In this section, we apply the previously built models on the whole WDC and GPA products datasets, in order to identify product matches for the products in the GPA dataset, and extract additional attributes from the WDC products. In this section we perform two experiments: (i) Identifying duplicate products within the WDC dataset for top 10 TV brands (section 6.1); (ii) Identifying matching products in the WDC dataset for the product ads in the GPA dataset in the TV category, i.e., enriching GPA product ads with features extracted from products from the WDC dataset (section 6.2). For both experiments we use the best performing models evaluated in the previous section, i.e., a Random Forest model built on CRF features, and a Random Forest model built on CRFemb features. We do not include the dictionary approach, because the results in the initial experiments were not too promising.

6.1. Integrating Unstructured Product Descriptions

In the first experiment, we apply the models 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 5, 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 and CRFemb feature extraction approaches to extract the features separately, and we tune the Random Forest model in a way that we increase the precision, at the cost of lower recall, i.e., a candidate product pair is considered to be positive matching pair if the classification confidence of the model is above *0.8*.

Table 8
Discovered matching products in the WDC dataset

		CRF		CRFemb	
Brand	#WDC products	#Matches	Precision	#Matches	Precision
sony	3,673	926	96.00	973	95.58
lg	14,764	734	94.00	758	94.19
samsung	4,864	567	88.18	587	89.43
rca	3,961	385	93.55	401	93.51
vizio	563	296	94.59	271	97.78
panasonic	1,696	160	93.75	173	94.21
philips	1,466	44	95.45	49	95.91
magnavox	141	29	100.00	39	97.43
nec	23,845	18	100.00	26	100
proscan	30	7	100.00	9	100

Table 9
Discovered matching products in the WDC dataset for product ads in the GPA dataset

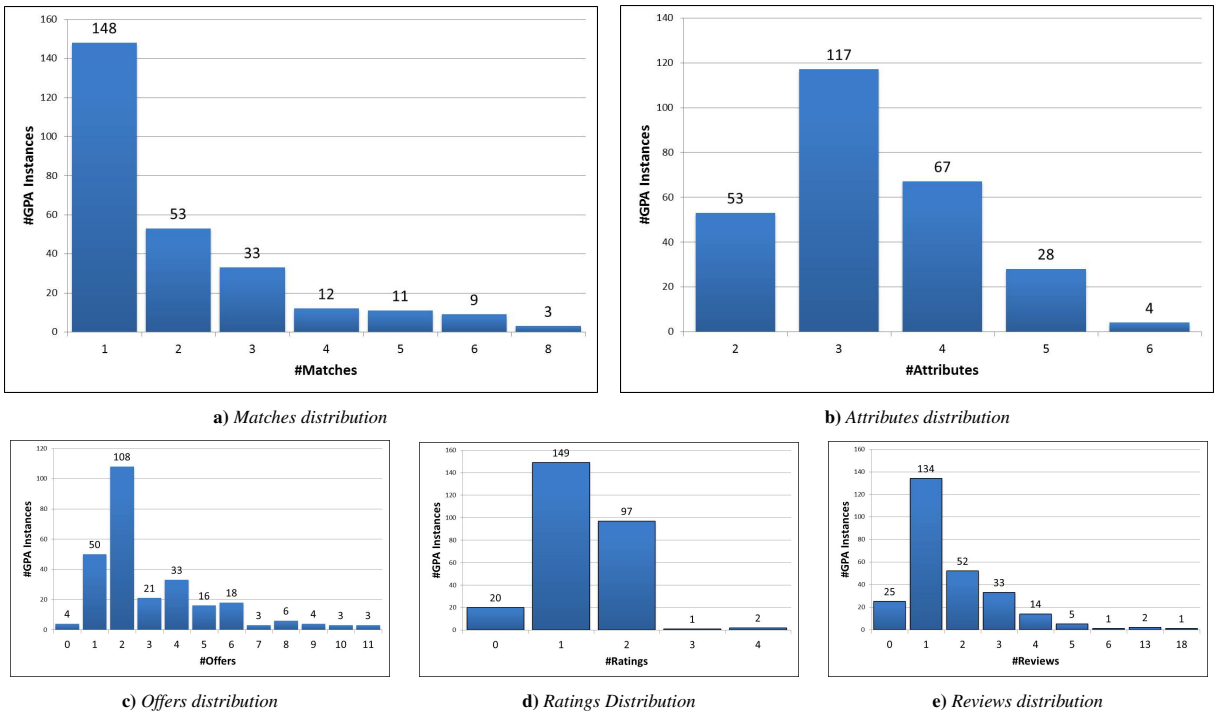
			CRF		CRFemb	
Brand	#GPA products	#WDC products	#Matches	Precision	#Matches	Precision
samsung	560	4,864	202	80.85	217	82.02
vizio	253	563	123	91.80	161	92.54
lg	288	14,764	102	89.24	124	91.93
rca	10	3,961	67	79.10	79	81.01
sony	102	3,673	40	97.50	70	94.28
proscan	18	30	28	100.00	29	100.00
nec	22	23,845	21	85.70	33	87.87
magnavox	28	141	12	100.00	27	100.00
panasonic	41	1,696	6	100.00	10	100.00
philips	11	1,466	2	100.00	10	100.00

We report the number of discovered matches for each of the TV brands in Table 8. The second column 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 for both the CRF and the CRFemb approach. The results show that we are able to find a large number of matching products with high precision. The number of discovered matches when using the CRFemb approach is slightly higher compared to the CRF approach, while the precision remains in the same range. By relaxing the selection constraints of product candidates the number of discovered matches would increase, but it might also reduce the precision.

6.2. 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 and the CRFemb feature extraction models, and for the GPA products, we use the already existing features provided by the merchants. To identify the matches we apply the respective Random Forest models for the CRF and the CRFemb approach. The results are shown in Table 9. The second column reports the number of products of the given brand in the GPA dataset, and the third column in the WDC dataset.

The results show that we are able to identify a small number of matching products with high precision. Again, the number of discovered matching products is slightly higher when using the CRFemb approach compared to the CRF approach, i.e., using the CRF approach we are able to find at least one match for 269 products in the GPA dataset, while using CRFemb for 310 products, and using the CRF approach there are 534 correct matches in total, while with CRFemb there are 676 correct matches. However, we have to note 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 using the CRF approach is shown



a) *Matches distribution*

b) *Attributes distribution*

c) *Offers distribution*

d) *Ratings Distribution*

e) *Reviews distribution*

Fig. 7. Distribution of newly discovered matches and attributes per product ad using the CRF approach

in Fig. 7. The results show that for each of the product ads that we found a matching product description, at least two new attribute-value pairs were discovered, while for some products, up to six new attribute-value pairs can be added. The same distributions can be observed when using the CRFemb approach are shown in Fig. 8. With that approach, we are able to identify more matches, thus more attribute-value pairs, offers, ratings, and reviews.

7. Product Categorization Evaluation

In a third series of experiments, we evaluate the quality of our approach for product categorization introduced in section 4. The objective of the evaluation is to categorize Web product descriptions into an existing target product catalog.

7.1. Dataset

For our experiments we use the GS1 Product Catalog (GPC)[21] as a target product categorization 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 (77 used) and the third level 783 categories (303 used).

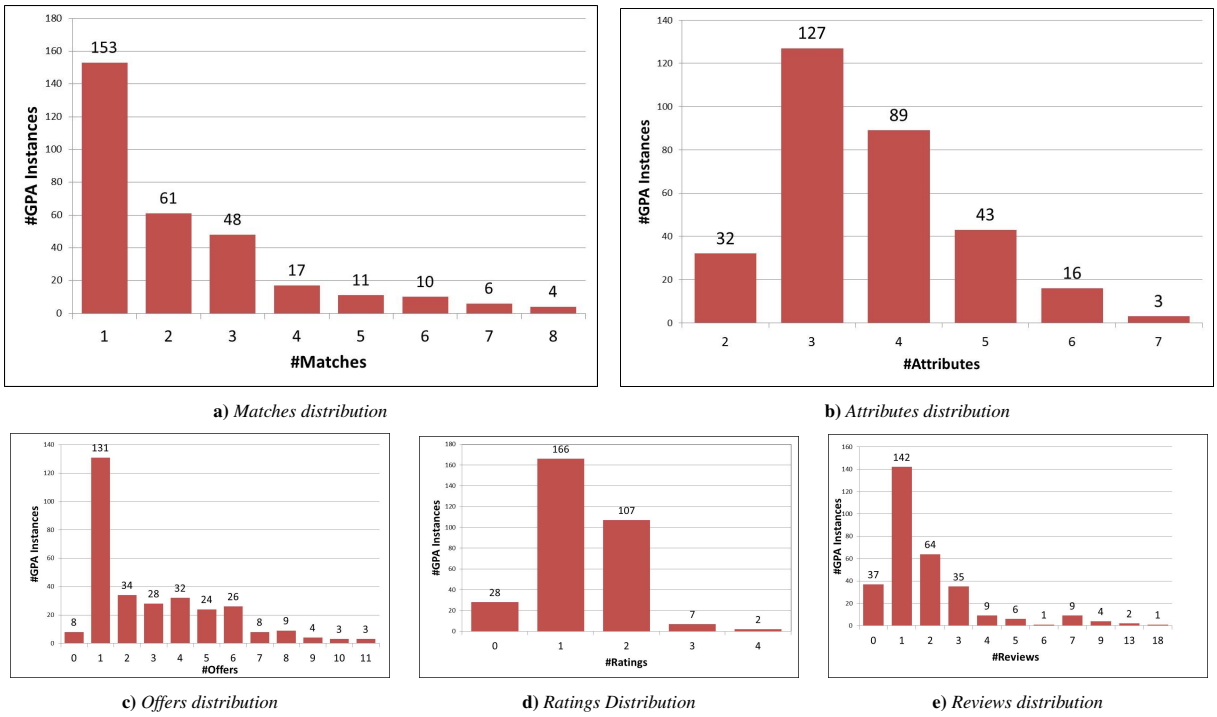
To evaluate the proposed approach we use the Microdata products gold standard developed in [25][22]. We removed non-English instances from the dataset, resulting in 8,362 products. In our evaluation we use the *s:name* and the *s:description* properties for generating the text embeddings, and the *s:Product/image* for generating the image embeddings.

7.2. Experiment Setup

The evaluation is performed using 10-fold cross validation. We measure accuracy (Acc), Precision (P), Recall (R) and F-score (F1). Here, we evaluate both supervised and unsupervised feature extraction for product categorization. Moreover, we compare the dictionary approach (Dict.) to the unsupervised paragraph2vec feature extraction (Par2vec), and the unsupervised image feature extraction model (ImgEmb).

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

[22] http://webdatacommons.org/structureddata/2014-12/products/gs.html



a) *Matches distribution*

b) *Attributes distribution*

c) *Offers distribution*

d) *Ratings Distribution*

e) *Reviews distribution*

Fig. 8. Distribution of newly discovered matches and attributes per product ad using the CRFemb approach

As for the product matching task (see section 5.4), we build both DM and DBOW models with vector size of 50, 200, 500 and 1000.

For the image feature extraction we use a CNN model trained on the labels from the third level of the GS1 catalog (ImgEmb). Furthermore, we compare our CNN model to an existing CNN model. In particular, we used a Caffe reference model[23], which has been pre-trained on 1.2 million ImageNet (ILSVRC2012 challenge) images[24]. We use the output of the second fully connected layer, which results in a feature vector of length 4,096. Finally, we do a combination of both the unsupervised text feature extraction model and the image embedding model, by concatenating the output vectors of both models (Par2Vec + ImageNet).

We compare all of the approaches with a baseline approach based on a Bag-of-Words TF-IDF cosine similarity. Same as before, for conducting the experiments, we used the RapidMiner machine learning platform and the RapidMiner development library. All experiments were run on a MacBook Pro with 8GB of RAM and 2.3 GHz Intel Core i7 CPU processor.

7.3. Results

The results for each of the three levels are shown in Table 10. All the experiments that did not finish within ten days, or that have run out of memory are marked with "\". We show only the best performing results of all the paragraph2vec models, where on each of the three levels the best results were achieved using the DBOW model with 500 dimensions, trained on both the title and description of the products. The complete results can be found online[25].

We can observe that the supervised dictionary-based approach outperforms the rest on the first level. However, on the second and the third level the combination of the document and image embedding approach outperforms the others. Furthermore, the documents embedding approach alone outperforms the baseline on the second and third level, and gives comparable results on the first level. It is interesting to note that the image embeddings approach alone performs rather well on all three levels, where the model trained only on product images performs slightly worse than the model built on the ImageNet dataset. The reason is that

[23] bvlc_reference_caffenet from `caffe.berkeleyvision.org`
[24] `http://image-net.org/challenges/LSVRC/2012/`

[25] `http://data.dws.informatik.uni-mannheim.de/gpawdc/SWJ`

Table 10
Product Categorization results. The best results are marked in bold.

		GS1 Level 1				GS1 Level 2				GS1 Level 3			
Features	Model	ACC	P	R	F1	ACC	P	R	F1	ACC	P	R	F1
BoW TF-IDF	NB	80.87	61.26	51.64	56.04	78.81	42.21	37.78	39.87	69.87	20.47	22.70	21.52
	K-NN	77.79	58.35	48.42	52.92	78.29	40.15	38.42	39.27	65.99	20.67	19.18	19.89
	SVM	86.06	71.50	55.62	62.57	\	\	\	\	\	\	\	\
	RF	73.58	55.52	36.74	44.22	73.00	39.43	27.96	32.72	65.35	20.12	16.87	18.35
Dict.	NB	81.98	66.43	51.68	58.13	79.87	46.31	40.08	42.97	70.90	25.45	24.36	24.89
	K-NN	76.32	54.68	46.28	50.13	76.44	40.15	39.65	39.90	65.66	21.93	20.14	20.99
	SVM	**88.34**	74.11	64.78	**69.13**	\	\	\	\	\	\	\	\
	RF	79.07	66.73	49.09	56.57	73.74	41.75	31.93	36.19	65.82	21.87	18.61	20.10
Par2Vec	NB	67.01	46.39	44.90	45.63	65.59	34.73	33.57	34.14	65.82	20.52	21.40	20.95
	K-NN	70.11	48.13	35.82	41.07	79.44	42.96	42.00	42.47	76.64	26.35	27.06	26.70
	SVM	80.24	57.89	52.92	55.29	\	\	\	\	\	\	\	\
	RF	58.17	41.40	16.50	23.60	68.82	31.80	25.69	28.42	62.63	16.15	14.80	15.45
ImgEmb	NB	56.85	34.04	23.69	27.94	55.39	21.92	13.97	17.06	51.26	14.72	8.31	10.62
	K-NN	66.26	39.38	30.78	34.55	66.34	30.77	21.28	25.16	59.76	21.59	16.98	19.01
	SVM	70.29	48.65	32.20	38.75	\	\	\	\	\	\	\	\
	RF	65.65	50.26	27.19	35.29	63.96	32.18	16.70	21.99	58.38	18.43	14.42	16.18
ImageNet	NB	61.66	31.49	25.96	28.46	60.94	20.24	15.30	17.43	53.73	9.63	7.48	8.42
	K-NN	76.02	50.14	47.36	48.71	74.80	32.19	33.63	32.89	68.32	20.22	21.46	20.82
	SVM	76.69	55.66	46.89	50.90	\	\	\	\	\	\	\	\
	RF	64.09	39.21	23.60	29.47	66.51	28.44	17.67	21.80	60.41	14.40	12.05	13.12
Par2vec + ImageNet	NB	64.3	28.2	24.5	26.22	63.41	16.79	13.25	14.81	55.68	8.40	6.04	7.03
	K-NN	75.99	46.65	37.1	41.33	**83.77**	43.85	42.76	**43.30**	**77.76**	26.60	27.15	**26.87**
	SVM	84.82	60.37	53.04	56.47	\	\	\	\	\	\	\	\
	RF	69.98	46.66	23.27	31.05	70.44	28.67	27.55	28.10	65.21	17.94	15.16	16.43

the number of labeled products images we used is significantly lower that the dataset used for training the ImageNet model. Also, we have to note that we did not apply any preprocessing or filtering on the images[26].

8. Conclusion

section

In this paper, we have proposed an approach that focuses on two tasks of the product integration pipeline, i.e., product matching and categorization. Our approach introduces the usage of unsupervised feature extraction for product data by using neural language modeling (word2vec and paragraph2vec) and deep learning models (CNN). The highlights of this paper include: (i) word embeddings help the CRF model training significantly for "dirty" web data, (ii) text embeddings improve product categorization considerably, and (iii) image embeddings can be used as a weak signal for product matching and strong signal for product categorization. Moreover, we provide a thorough product specification fusion as a part of our use case of enriching product ads with semantic structured data.

Besides integrating products, our approach could be used for search query processing, product recommendation and information retrieval, which would improve the shopping experience for users [3].

In this paper, we focus on product data, but the proposed approach could be easily extended to other areas. As shown in [24], HTML annotations are used in many other areas, for example, the Microdata markup format and schema.org are widely used to describe persons, events, job postings, hotels, places, organizations, recipes, movies, books, music and many other types of data. All these entities contain textual description and images. To build interesting application on top of this data, again feature-extraction approach needs to be applied on the text and image description of the entities. Our approach could be adapted and tuned for that purpose. Once the attribute-value pairs are extracted various applications can be built on top, e.g., address books, travel agents, search engines, recommender systems and aggregators for a large variety of entity types.

[26] Many images do not directly depict the product, but, e.g., a vendor logo, or are of a bad quality.

References

[1] Bell, S., Bala, K.: Learning visual similarity for product design with convolutional neural networks. ACM Trans. Graph. 34(4), 98:1–98:10 (Jul 2015), http://doi.acm.org/10.1145/2766959

[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 WWW. pp. 317–326. ACM (2011)

[4] Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning. pp. 160–167. ACM (2008)

[5] DeNale, R., Weidenhamer, D.: Quarterly retail e-commerce sales 4th quarter 2015. US Census Bureau News (2015)

[6] 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 (2005)

[7] 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)

[8] Gopalakrishnan, V., Iyengar, S.P., Madaan, A., Rastogi, R., Sengamedu, S.: Matching product titles using web-based enrichment. In: 21st ACM international conference on Information and knowledge management. pp. 605–614 (2012)

[9] Grbovic, M., Radosavljevic, V., Djuric, N., Bhamidipati, N., Savla, J., Bhagwan, V., Sharp, D.: E-commerce in your inbox: Product recommendations at scale. In: Proceedings of the 21th ACM SIGKDD. pp. 1809–1818. ACM (2015)

[10] Gupta, V., Karnick, H., Bansal, A., Jhala, P.: Product classification in e-commerce using distributional semantics. CoRR abs/1606.06083 (2016), http://arxiv.org/abs/1606.06083

[11] Isele, R., Bizer, C.: Learning linkage rules using genetic programming. In: Proceedings of the International Workshop on Ontology Matching. pp. 13–24 (2011)

[12] Kannan, A., Givoni, I.E., Agrawal, R., Fuxman, A.: Matching unstructured product offers to structured product specifications. In: 17th ACM SIGKDD (2011)

[13] Kannan, A., Talukdar, P.P., Rasiwasia, N., Ke, Q.: Improving product classification using images. In: Proceedings of the 2011 IEEE 11th International Conference on Data Mining. pp. 310–319. IEEE Computer Society (2011)

[14] Kiapour, M.H., Han, X., Lazebnik, S., Berg, A.C., Berg, T.L.: Where to buy it: Matching street clothing photos in online shops. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 3343–3351 (Dec 2015)

[15] Köpcke, H., Thor, A., Thomas, S., Rahm, E.: Tailoring entity resolution for matching product offers. In: Proceedings of the 15th International Conference on Extending Database Technology. pp. 545–550. ACM (2012)

[16] Kozareva, Z.: Everyone likes shopping! multi-class product categorization for e-commerce. In: The 2015 Annual Conference of the North Americal Chapter for the ACL. pp. 1329–1333 (2015)

[17] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

[18] Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053 (2014)

[19] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation 1(4), 541–551 (1989)

[20] Londhe, N., Gopalakrishnan, V., Zhang, A., Ngo, H.Q., Srihari, R.: Matching titles with cross title web-search enrichment and community detection. Proceedings of the VLDB Endowment 7(12), 1167–1178 (2014)

[21] McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: Proceedings of the 38th International ACM SIGIR Conference. pp. 43–52. ACM (2015)

[22] McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika 12(2), 153–157 (1947)

[23] 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)

[24] Meusel, R., Petrovski, P., Bizer, C.: The webdatacommons microdata, rdfa and microformat dataset series. In: The Semantic Web–ISWC, pp. 277–292 (2014)

[25] Meusel, R., Primpeli, A., Meilicke, C., Paulheim, H., Bizer, C.: Exploiting microdata annotations to consistently categorize product offers at web scale. In: Proceedings of EC-Web. Valencia, Spain (2015)

[26] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[27] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

[28] Mnih, A., Hinton, G.E.: A scalable hierarchical distributed language model. In: Advances in neural information processing systems. pp. 1081–1088 (2009)

[29] 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)

[30] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. vol. 14, pp. 1532–1543 (2014)

[31] Petrovski, P., Bryl, V., Bizer, C.: Integrating product data from websites offering microdata markup. In: Proceedings of the companion publication of the 23rd international conference on World wide web companion. pp. 1299–1304 (2014)

[32] Petrovski, P., Bryl, V., Bizer, C.: Learning regular expressions for the extraction of product attributes from e-commerce microdata (2014)

[33] Ristoski, P., Mika, P.: Enriching product ads with metadata from html annotations. In: Proceedings of the 13th Extended Semantic Web Conference. (2016)

[34] Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks 61, 85 – 117 (2015), //www.sciencedirect.com/science/article/pii/S0893608014002135

[35] de Souza, J.G., Federico, M., Sawaf, H.: Mt quality estimation for e-commerce data. In: Proceedings of MT Summit XV (2015)

[36] Vandic, D., Van Dam, J.W., Frasincar, F.: Faceted product search powered by the semantic web. Decision Support Systems 53(3), 425–437 (2012)

[37] Wang, M., Manning, C.D.: Effect of non-linear deep architecture in sequence labeling. In: IJCNLP. pp. 1285–1291 (2013)

[38] Wang, X., Sun, Z., Zhang, W., Zhou, Y., Jiang, Y.G.: Matching user photos to online products with robust deep features. In: Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval. pp. 7–14. ICMR '16, ACM, New York, NY, USA (2016), http://doi.acm.org/10.1145/2911996.2912002

[39] Zhao, W.X., Li, S., He, Y., Chang, E., Wen, J.R., Li, X.: Connecting social media to e-commerce: Cold-start product recommendation on microblogs. IEEE Transactions on Knowledge and Data Engineering PP(99), 1–1 (2015)