Product Classification Using Microdata Annotations

Ziqi Zhang(✉) and Monica Paramita

Information School, University of Sheffield, Sheffield, UK
{ziqi.zhang,monica.paramita}@sheffield.ac.uk

Abstract. Markup languages such as RDFa and Microdata have been widely used by e-shops to embed structured product data, as evidence has shown that they improve click-through rates for e-shops and potentially increases their sales. While e-shops often embed certain categorisation information in their product data in order to improve their products' visibility to product search and aggregator services, such site-specific product category labels are highly inconsistent and unusable across websites. This work studies the task of automatically classifying products into a universal categorisation taxonomy, using their markup data published on the Web. Using three new neural network models adapted based on previous work, we analyse the effect of different kinds of product markup data on this task, and show that: (1) despite the highly heterogeneous nature of the site-specific categories, they can be used as very effective features - even only by themselves - for the classification task; and (2) our best performing model can significantly improve state of the art on this task by up to 9.6% points in macro-average F1.

Keywords: Linked data · Product classification · Neural networks · CNN · LSTM · HAN · Machine learning

1 Introduction

Recent years have seen significant growth of semantically annotated data on the Web using markup languages such as RDFa and Microdata. Particularly in e-commerce, statistics from the WDC project[1] show that between 2017 and 2018, the number of URLs that use schema.org to embed structured product data has more than doubled; and the number of hosts has increased by over 40%.

Semantically annotated product data improve the visibility and accessibility of product information from e-shops. While different websites may sell the same products, the information about the products often differ significantly across different websites. Although product aggregator services such as Google Product Search[2] are created to integrate product information from disparate sources into a single representation, one of the first challenges they face is organising

[1] http://webdatacommons.org/structureddata/.
[2] https://www.google.com/shopping?hl=en.

C. Ghidini et al. (Eds.): ISWC 2019, LNCS 11778, pp. 716–732, 2019.
https://doi.org/10.1007/978-3-030-30793-6_41

all the products according to a universal product categorisation taxonomy. In practice, many e-shops embed category information as structured data within their web pages (to be called ‘site-specific product labels’), it is known that the categorisation schemes vary dramatically across different e-shops, even if they sell the same products [11,16].

A large number of research has emerged over the years to study the very problem of product categorisation or classification on the Web [3,5,6,9,11,12,14,19]. However, our work is different in two ways. *First*, a large number of previous work [1,3,4,7,9,11,12,14,19] looked at product classification within a single e-shop, that is assigning class labels defined by the e-shop to products listed on the same e-shop. In contrast, we explore the task in the context of product Microdata harvested from a wide range of e-shops. This is arguably more challenging as different e-shops may adopt different writing styles when publishing product information. While [16,17] also studied the task in the Microdata context, we propose new methods that obtain much better results.

Second, although it is widely recognised that site-specific product labels are hardly useful as universal categories beyond their source websites, little effort has been made to study how such heterogeneous information can assist in the task of product classification. While [16] used such information in an unsupervised approach, we carry out further studies to understand the impact of such highly heterogeneous information on supervised product classification.

Our contributions are three-fold. *First*, inspired by the deep artificial neural network (DNN) model ‘DeepCN’ used in product classification [7], we generalise and adapt it to three popular DNN architectures for a comparative analysis in this task: Convolutional Neural Networks (CNN), bidirectional Long-Short Term Memory (bi-LSTM), and Hierarchical Attention Network (HAN). We share our lessons of when different architectures work better than others in different settings. *Second*, we carry out in-depth analysis to understand the impact of the highly heteregenous site-specific product labels on the product classification task. This includes two methods of ‘pre-processing’ such labels in the hope to obtain better quality features for the task. *Third*, we empirically show a surprising insight that, despite the high level of heterogeneity in the site-specific product labels, they prove to be very useful features for the classification task, as they are more effective than traditional features widely used by previous research, as well as their ‘pre-processed’ versions. Compared to state of the art on the same dataset used for evaluation, our best peforming model obtains much better results with the highest improvement of 9.6% in macro-average F1.

The remainder of this paper is organised as follows. Section 2 discusses related work; Sect. 3 describes our methodology in details; Sect. 4 presents experiments, followed by a discussion of results in Sect. 5; finally Sect. 6 concludes this work.

2 Related Work

2.1 Text Classification

Text categorisation or classification aims to assign categories or classes to a given text according to its contents [2,18]. While early methods were largely based on manually crafted rules, current methods however, are predominantly based on statistical machine learning [18]. These methods involve a feature engineering process that represents each text by feature vectors (e.g., words, N-grams), followed by a training process where a machine learning algorithm is applied to instances labelled with appropriate classes to learn patterns for classification.

Many classic machine learning algorithms have been used in this task, such as Naive Bayes, Logistic Regression, Random Forest, and Support Vector Machines (SVM) [4,18]. Recent years have seen increasing interests in DNN models, due to their ability in automatic feature learning, which not only eliminates the feature engineering process, but also discovers very effective abstract features that traditional machine learning algorithms are unable to capture. Popular DNN models include, for example, CNN [8], LSTM [21], and HAN [20]. Using sentence-level classification for example, CNN scans a fixed-length consecutive sequences of words and transforms those into abstract features. It can be considered as a process of aggregating the meaning of the composing lexical sequences (e.g., phrases) in text reading. LSTM is a type of Recurrent Neural Network (RNN), and it captures long distance dependencies between words and simulates our reading of ordered words to incrementally develop a meaning for the sentence. HAN is based on the intuition that not all parts of a sentence are equally relevant for representing its meaning. Instead, certain sections (e.g., keywords) should be given more attention. Different adaptations of such models have been widely used in a range of text classification tasks, such as sentiment analysis [8,21], review classification [8,20], and question answering [8].

2.2 Product Classification

Product classification is typically treated as a text classification task, as the process involves assigning category labels (i.e., classes) to product instances based on their attributes - which we refer to as metadata - that are typically text-based (e.g., name, description, brands). Such labels usually reside in a categorisation taxonomy, therefore the task usually requires assigning multiple labels, one from each level of the taxonomy [14]. Below we summarise related work from different perspectives and highlight the similarity and difference in this research.

Metadata and Features. To classify products, features must be extracted from certain product metadata. Rich, structured metadata are often not available. Therefore, the majority of literature have only used product names, such as [1,4,9,11,19] and all of those participated in the 2018 Rakuten Data Challenge [14]. Several studies used both names and product descriptions [3,5,6,12,13], while a few also used other metadata such as model, brand, maker, etc., which need to be extracted from product web pages by an Information Extraction process [7,9,17]. In addition, [17] also used product images. The work by [16] used site-specific product labels found in product Microdata in an unsupervised approach based on the similarity between target class labels and other product metadata including site-specific product labels. While the authors argued that such heterogeneous labels may prove unusable for supervised learning, we specifically study the usage of such information in supervised setting and aim to understand if they need to be cleaned to support the classification task.

Generally speaking, for text-based metadata, there are three types of feature representation. The first is based on Bag-of-Words (BoW) or N-gram models, where texts are represented based on the presence of vocabulary in the dataset using either 1-hot encoding or some weighting scheme such as TF-IDF [1,3–5,7]. The second uses an aggregation of the word embeddings from the input text. For example, [11] averaged the embedding vectors of composing words from product titles, [12] summed them, while work in [8] joins word embedding vectors to create a 2D tensor to represent the text. The third applies a separate learning process to learn a continuous distributional representation of the text directly. This includes adaptation of the well-known Paragraph2Vec model such as in [6,12,17]. Our work represents input texts following the approach by [8].

Algorithms. The large majority of work has used supervised machine learning methods using popular algorithms mentioned before. These include those that use traditional machine learning algorithms [3–6,9,12,16,17], and those that apply DNN-based algorithms [7,14,19]. All DNN-based methods have used some adaptations of CNN or RNN. These include the majority of the participating systems in the 2018 Rakuten Data Challenge. Besides, [5,16] also explored unsupervised methods based on the similarity between the feature representations of a product and target classes. [1] studied product clustering, which does not label the resulting product groups. These represent unsupervised methods. Further, [13] also studied the problem as a machine translation task.

While the majority of literature either use a single type of metadata (typically product name) or concatenate several to merge into a single source of metadata, [7] argued to treat different types of metadata separately. The intuition can be that they may contribute different weights to the task. Thus they introduced the Deep Categorisation Network (DeepCN), which consists of i RNN models each taking text input from one of the i types of metadata. For example, using product name, description, brand and maker as product metadata, DeepCN combines 4 RNN models each applied to one of these input. Our work adapts DeepCN to generalise to three DNN structures including bi-LSTM, CNN and HAN, thus offers as a comparison of these popular DNN structures in this task.

Datasets. As mentioned before, the majority of methods are evaluated using datasets created within a single e-shop [3,5,6,9,11,12,14,19]. This typically involves classifying listed products on the e-shop using its own categorisation systems (i.e., their site-specific product labels). To the best of our knowledge, the work by [5,16,17] are the only ones that explored classification of products from multiple sources into a universal categorisation taxonomy, while [16,17] used the only dataset built on product Microdata. Furthermore, while the majority of datasets are proprietary, the only publicly available ones include that released in [14] and [16]. Among the two, [14] only contains product names (no site-specific product labels as the data were harvested from a single website), while [16] contains names, descriptions, and site-specific product labels. Our work uses the dataset by [16].

3 Methodology

We firstly describe the DNN models proposed for the task and the set of features to be used with them (Sect. 3.1). We then introduce two methods aimed at 'standardising' the heterogenous site-specific product labels. One uses clustering (Sect. 3.2), while the other is based on a set of heuristics (Sect. 3.3).

3.1 Models and Features

We propose to adapt the DNN architecture 'DeepCN' in [7]. DeepCN uses multiple RNNs each dedicated to a different type of product metadata for generating features. These then feed into fully connected layers that learn to integrate these features for the classification task. We generalise DeepCN (to be called 'GN-DeepCN') by replacing the RNNs with any kind of DNN structures. An adapted model is illustrated in Fig. 1.

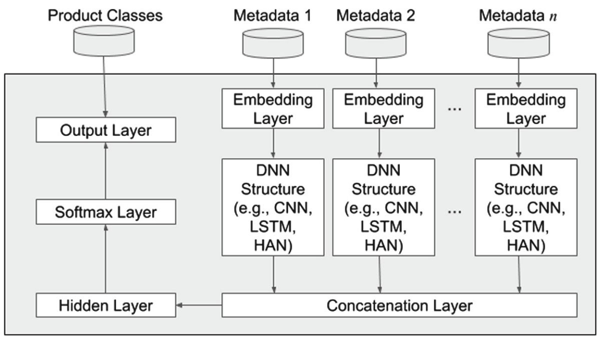

Fig. 1. Architecture of the proposed GN-DeepCN.

Adaptations. We introduce three adaptations to the original model. First, the authors used 1-hot encoding to represent each type of input metadata. We instead use the approach by [8] to use an embedding layer to represent input text as a 2D-tensor in the shape of $[words, dimensions]$, where $words$ is the number of tokens expected in the input text and will depend on the type of metadata (see below in 'Features'); and $dimensions$ is the dimension of the continuous vector representation of each word. In this work, we use the GloVe word embedding vectors pre-trained on the Common Crawl corpus[3] with 300 dimensions. Since we are dealing with content from e-commerce web pages, we consider GloVe vectors a better option than, e.g., Word2Vec that is trained on news corpora.

Second, for the internal DNN structure, the authors used a stack of two RNN layers to process each type of input metadata. We propose to compare three popular architecture in text classification, including bi-LSTM, CNN and HAN. We detail their parameter settings below.

Third, the original model consists of two hidden layers, one stacked on another. We use only one as the authors showed in their results that the contribution of an additional hidden layer to learning accuracy was rather insignificant. The hidden layer has 600 neurons, same as the best performing setting in [7].

DNN Structures. For the bi-LSTM structure, we use a single bi-directional LSTM layer with 100 neurons. For the CNN structure, we concatenate three convolutional layers each of which has 100 filters and uses a window size of 2, 3, and 4 respectively. Their concatenated output is then pooled by a max pooling layer with a pool size of 4. For the HAN architecture, we adapt the model by [20]. The original work has a two-level attention mechanism that simulates attention to particular sentences within a document, and then particular words within sentences. We simplify this structure to a one-level attention network that only encodes word-level attention. This is because product metadata are usually short-text. All other implementations remain the same as [20].

Features. We use three types of product metadata that are more available than the rest in the product Microdata, these include product name, description, and site-specific product labels. When product labels are not available, we use the breadcrumbs instead (if available). Breadcrumbs records the navigational path of the current web page, and in the context of e-shops, may indicate the site-specific category path. Texts of these metadata are then normalised by lowercasing, removal of non-alphanumeric characters, and lemmatisation. They are then passed into the internal DNN branches for feature extraction. We constrain product name or labels to a maximum of 20 words and product description to 100 words, truncating longer texts and pad shorter texts with zeros.

In order to study the impact of different metadata on the classification task, we evaluate different combinations of input metadata. Therefore, in some settings, the model may not take all three input metadata. In particular, we test each type of input metadata separately (i.e., using only name, description or site-specific product label). To the best of our knowledge, all previous work have used either name alone, or some combinations of name and other metadata. No work has studied how the highly heterogenous site-specific product labels alone can be used in the product classification task.

Also, for site-specific product labels, we test three different ways of using them in order to study the impact of the heterogeneity in the metadata. These include: (1) using the normalised labels as-is (to be referred to as the 'original' site-specific product labels); (2) clustering normalised labels and then using the cluster membership associated with a product's label as feature (Sect. 3.2, to

[3] http://nlp.stanford.edu/data/glove.840B.300d.zip.

be referred to as 'product label clusters'); and (3) 'standardised' labels by a 'cleaning' processs that aims to reduce the level of heterogeneity (Sect. 3.3, to be referred to as 'cleaned product labels').

Parameter Optimisation. We use the categorical cross entropy loss function and the Adam optimiser to train all models with an epoch of 20 using a batch size of 100. The categorical cross entropy loss function is empirically found to be more effective on classification tasks than other loss functions [15]. The Adam optimiser is designed to improve the classic stochastic gradient descent (SGD) optimiser and in theory combines the advantages of two other common extensions of SGD (AdaGrad and RMSProp) [10].

3.2 Clustering Site-Specific Product Labels

While different e-shops may categorise their products using different labels, we expect their labels to be semantically similar for products that belong to the same categories. For example, given `'USA Curling > USA Curling Sweatshirts & Fleece'` from an e-shop and `'Apparel & Accessories > Clothing > Shirts & Tops > T-Shirts'`, from another, we can anticipate both products to belong to the category of 'clothing', or 'top clothing'. While this category may not match those in the gold standard per se, they may be useful features for classification.

To capture this information, we propose to apply clustering to split all unique site-specific product labels in a dataset into groups. Thus given a set of unique site-specific product labels, we firstly represent each product label as a fixed-length feature vector. To do so, we treat each label as a document and calculate the TF-IDF weights of its composing words. Next, given a product label, we find the pre-trained GloVe embedding vectors of its composing words. Then we represent each product label as the TF-IDF weighted sum of its composing word embedding vectors. A similar approach is used in [11] with product descriptions.

Finally, we apply agglomerative clustering to the feature vectors of product labels to split them into k groups (settings of k to be detailed in Sect. 4). Each product is then assigned a cluster number associated with its site-specific product label. This cluster number is used as a feature for product classification.

3.3 Cleaning Site-Specific Product Labels

As discussed before, site-specific product labels vary widely as e-shops adopt categorisation systems that use different vocabulary and granularity. For example, some items may be categorised based on their types (e.g., `'Audio & Video > Cables & Adapters'`), while others (e.g., clothing items) can be categorised based on the brands they represent (e.g., `'NFL > New Orleans Saints > New Orleans Saints Sweatshirts & Fleece'`), or by their prices (e.g., `'Dallas Mavericks > Dallas Mavericks Ladies > less than $10'`). Some of these labels are not informative of the types of products and we expect them to be unuseful in the product classification task. Therefore, we propose to clean site-specific product labels to reduce the level of heterogeneity.

Table 1. Different separators between category levels

Example of categories/breadcrumbs	Separators
`Hardware > hardware accessories > cabinet hardware > ...`	`>`
`Clothing, dresses, sale, the LBD, clothing, mini dress`	,
`Beeswax products|beeswax polish`	|
`Combination\\|acne-prone\\|sensitive\\|normal\\|clinicians\\|...`	\\|
`Home  hunting  wildlife feed & feeders  deer feeders ...`	`[multi-spaces]`
`Women/clothing/leggings`	`/`
`Home/gear/accessories/books/videos/stickers/books`	`[space]/[space]`
`Toe rings >> double toe rings`	`>>`
Note: This is not an exhaustive list of possible separators.	

We firstly pre-process the site-specific product labels by normalising the different separators that are used in the category/breadcrumbs values across different sites (shown in Table 1) by replacing them with '`>`'[4]. We refer to each segment separated by '`>`' a product label at a different level. We also filter out meaningless category values by regular expression (e.g., blank nodes such as `'node35ea8dc879ed1d78...'` and URIs). Next, we follow the steps below to further clean the labels:

1. Divide the label values using the '`>`' to result in different levels.
2. If the top (parent) level label is written in all capital letters, remove it. This is because the top level labels written as uppercase are rare (about 4% in our experimental dataset) and are often generic (e.g., '`HOME`') or non-product related (e.g., '`NBA`', '`NFL`'). As an example: '~~NFL~~ > Carolina Panthers > Carolina Panthers Shoes & Socks > Carolina Panthers Shoes & Socks Ladies > $20 to $40'.
3. If the parent label exists as part of sub-level labels, remove the parent label from all these levels. E.g., '~~Carolina Panthers~~ > ~~Carolina Panthers~~ Shoes & Socks > ~~Carolina Panthers~~ Shoes & Socks Ladies > $20 to $40'.
4. Remove labels that include the word 'Sale', 'Deals' or contain prices, using regular expression. E.g., 'Shoes & Socks > Shoes & Socks Ladies > ~~$20 to $40~~'.

[4] This separator is chosen as it is the most commonly used in the dataset (described in Sect. 4).

5. For the remaining labels at each level, compute their TF-IDF scores and keep only the one with the highest TF-IDF score (i.e., 'cleaned product labels'). The idea is to find the most specific label that's likely to be relevant. Specifically, we group the original, uncleaned labels by their source websites, then treat each label as a document. Next we calculate the TF-IDF scores of words from these 'documents' within the context of each website. The TF-IDF score of the remaining labels at each level is the average TF-IDF score of their composing words. E.g., as a result of this process, we may obtain '`~~Shoes & Socks~~ > Shoes & Socks Ladies`'.

4 Experiment

4.1 Dataset

Currently, the only publicly available product classification dataset containing site-specific product labels and based on Microdata is that created by [16]. The dataset[5] contains 8,361 product instances randomly sampled from 702 hosts. It is annotated using the GS1 Global Product Classification system[6]. Categories from the top three levels of the classification taxonomy are used to label each product instance. We refer to these annotations as level 1, 2, and 3 GS1 annotations. The number of target classes for level 1, 2, and 3 are 37, 76, and 289. The distribution of instances over these classes are largely imbalanced. For example, the largest class all three levels has over 3,000 instances; and in extreme cases, some small classes have only a handful of instances.

As mentioned before, we use three types of product metadata found in the dataset, including name (`sg`[7]`:Product/name`), description (`sg:Product/descri-ption`), and original site-specific product labels (`sg:Product/category`). When product labels are not available, we use the breadcrumbs instead (`sg:breadcrumb`). Overall, all instances have name, 8,072 instances have description, 7,181 instances have site-specific label, and 1,200 instances have breadcrumbs. When both site-specific product labels and breadcrumbs are considered, there are in total 4,111 (or 2,866 after lemmatisation, which is also used to standardise input text to all models) unique values distributed over all instances, indicating the extreme high level of heterogeneity in the categorisation systems used by different websites.

4.2 Model Variants

To compare the contribution of different product metadata on this task, particularly the impact of the highly heterogenous site-specific product labels, we create models that use different combination of product metadata as input.

[5] http://webdatacommons.org/structureddata/2014-12/products/data/goldstandard_eng_v1.csv.
[6] https://www.gs1.org/standards/gpc.
[7] sg: http://schema.org/.

Let $\boldsymbol{n}$, $\boldsymbol{d}$, $\boldsymbol{c}$ denote product name, description and original site-specific labels respectively, we firstly test our models using each of these input metadata only. We then use combined inputs of $\boldsymbol{n+c}$, $\boldsymbol{n+d}$, and $\boldsymbol{n+d+c}$[8]. Further, we replace the original site-specific labels with cleaned product labels (*c.clean*) and product label clusters (*c.cluster*) respectively, to create ***n+c.clean***, ***n+c.cluster***, ***n+d+c.clean*** and ***n+d+c.cluster***. We experiment with a range of cluster numbers including $k \in \{25, 50, 100, 200\}$.

Each type of the input metadata combination is then used on each of the three GN-DeepCN networks described before. As an example, we use $\mathbf{CNN_{n+c}}$ to denote the CNN version of our GN-DeepCN network using product names and original site-specific labels as input. All models are evaluated in a 10-fold cross validation experiment. We measure classification accuracy (Acc), precision (P), recall (R), and F1. For P, R, and F1, we compute micro-average, macro-average, as well as weighted macro-average. Our results are fully reproducible as our datasets and code (implemented in Python) are shared online[9].

4.3 Methods for Comparison

We compare our methods against those reported in [16] and [17], both of which used the same datasets. [16] used an unsupervised approach based on similarities between the target categories and product metadata. Their best performing model uses the combination of product name, description, site-specific product label and/or breadcrumbs, as well as distributional statistics computed using an external corpus. Results are reported in classification accuracy, macro-average precision and F1.

[17] evaluated a large number of different methods. In terms of machine learning algorithms, they used SVM, Naive Bayes (NB), Random Forest (RF) and K-Nearest Neighbour (KNN). Each of these models are then used with several different types of features. These include: (1) a TF-IDF weighted Bag-of-Words representation of the concatenated product names and description (*bow*); (2) product attribute-value pairs extracted from their name and descriptions using a separate Information Extraction process (*dict* as these can be consider a 'dictionary' of product attributes); (3) product embedding vectors trained using the Paragraph2Vec model applied to the concatenated product names and descriptions (*Par2Vec*); (4) two variants of product image vectors (*ImgEmb* and *ImageNet*); and (5) the concatenated vectors of Par2Vec and ImageNet. Results are reported in classification accuracy, macro-average precision, recall

[8] The literature has mostly used name + other features, which we also do in this work. Also, as we shall show in the results, among n, d, and c alone, d generally performs the worst. So we do not report results based on description + other features.

[9] https://github.com/ziqizhang/wop.

and F1[10]. For each GS1 level of annotation, we only compare against the best result obtained from all of their models.

5 Results and Discussion

We firstly present our results in Sect. 5.1 and show (1) surprisingly, the original site-specific product labels can be very effective for the classification task, while the 'processed' features based on cleaning or clustering are not; (2) when different GN-DeepCN architectures perform better depending on availability of metadata; (3) our proposed methods outperforms state of the art significantly. We then present our analyses to understand the product label clustering and cleaning quality and why they did not help the task (Sects. 5.2 and 5.3).

5.1 Detailed Results

Due to space limitations, in Tables 2 and 3 we only show our results in macro-average F1 and classification accuracy, as they are used in [16,17]. Further, for our experiments on product label clusters, we only include the best results obtained with $k = 50$. The full results however, can be found online[11]. We ran statistical significance test on every pair of configurations using the K-fold cross-validated paired t-test, and we are able to confirm that the results obtained by different configurations are statistically significant.

Table 2. Macro-average F1 of the proposed model variants. The best result on each level are highlighted in **bold**.

Input metadata	Lvl.1			Lvl.2			Lvl.3		
	bi-LSTM	CNN	HAN	bi-LSTM	CNN	HAN	bi-LSTM	CNN	HAN
n	53.3	54.3	53.2	41.9	40.7	40.9	28.1	27.2	28.5
c	57.6	59.3	56.6	42.4	43.0	43.2	27.1	27.9	28.5
d	52.4	51.0	51.5	39.6	38.4	38.4	25.3	23.3	24.7
n+c	65.7	66.3	64.9	50.8	52.3	51.2	34.8	35.1	34.8
n+d	61.8	61.1	59.5	48.9	48.0	45.2	30.8	29.2	31.1
n+d+c	**67.4**	67.3	65.4	**52.9**	51.2	51.4	35.2	33.5	**35.4**
n+c.clean	61.9	61.7	59.5	50.0	49.2	47.8	32.4	33.7	33.3
n+d+c.clean	66.4	64.4	64.3	53.7	50.2	50.7	34.4	31.9	34.5
n+c.cluster = 50	54.1	53.6	52.6	41.0	41.9	39.8	26.9	27.5	28.6
n+d+c.cluster = 50	61.7	60.7	59.8	46.7	46.4	46.7	30.8	29.0	32.2

[10] This is an assumption based on our observation, as we have been unable to confirm this with the authors despite our efforts. However, we assume this is the truth as our macro-average results are the closest, while our micro-and weighted macro-average results are significantly higher (in the range between 70 and 90).

[11] https://github.com/ziqizhang/wop/tree/master/iswc2019_results.

Table 3. Classification accuracy of the proposed model variants. The best results on each level are highlighted in **bold**.

Input metadata	Lvl.1			Lvl.2			Lvl.3		
	bi-LSTM	CNN	HAN	bi-LSTM	CNN	HAN	bi-LSTM	CNN	HAN
n	80.6	81.6	81.3	79.4	80.0	79.5	72.7	73.2	72.6
c	80.7	80.9	79.9	79.3	79.0	78.5	70.0	68.5	69.1
d	80.4	79.8	79.7	79.0	77.9	77.9	70.0	68.7	70.0
$n{+}c$	87.3	88.0	87.2	85.7	86.4	85.8	78.9	79.9	78.8
$n{+}d$	85.4	85.7	85.3	84.0	84.0	83.5	76.6	76.5	76.4
$n{+}d{+}c$	**89.0**	88.6	88.0	**86.8**	86.6	86.6	**80.3**	79.5	79.6
$n{+}c.clean$	85.6	86.2	85.5	84.8	84.7	84.1	77.1	78.1	77.8
$n{+}d{+}c.clean$	87.8	87.5	87.3	86.7	85.9	85.9	79.2	78.2	79.0
$n{+}c.cluster = 50$	81.1	81.4	81.1	79.5	80.4	79.7	72.6	73.6	73.1
$n{+}d{+}c.cluster = 50$	85.9	85.3	85.2	83.3	84.0	84.0	76.9	76.1	76.7

Effect of Different Product Metadata. Looking at each of three kinds of metadata lone, Table 2 upper section shows that we obtain the best F1 using the original site-specific product labels c, regardless of the DNN models or categorisation levels. In many cases, the difference from model variants that use only n or d is quite significant (e.g., CNN$_c$ outperforms CNN$_n$ by 5% points at Lvl. 1). It is also worth to note that product descriptions d appear to be the least useful metadata. This is rather surprising as no previous work has considered using site-specific product labels alone in this task, but has all exclusively focused on product names and descriptions. Yet we show c to be the most useful for this task, despite the high level of heterogeneity in the metadata.

Again inspecting the three kinds of metadata separately, Table 3 upper section shows that generally n contributes to the best classification accuracy. However, the difference from c or d alone is rather small.

For both F1 and accuracy, we can see that c can consistently improve performance when it is combined with any other metadata (e.g., $n + c$ against n), regardless of model variants or categorisation levels. The improvements are in many cases, quite significant. For example, HAN$_{n+c}$ improves HAN$_n$ by 10.3 points in F1 at Lvl. 2; CNN$_{n+c}$ improves CNN$_n$ by 7.4 points in accuracy at Lvl. 1. The majority of DNN models achieved their best results with $n + d + c$ (except CNN which works better more often with $n{+}c$ instead), but also achieved comparable results with $n + c$ only. This is a useful finding, as n and c are much shorter than d and therefore, models using them as input can be more efficient.

Effect of Different GN-DeepCN Structures. Overall, the best F1 and accuracy scores are obtained by bi-LSTM in 5/6 cases, with the sixth case (i.e., F1 on $n + d + c$ at Lvl.3) being an extreme close-call. There is also a notable tendency for bi-LSTM to work better than CNN or HAN (in terms of either F1 or accuracy) when d is included in the input metadata. This could be because bi-LSTM captures more useful dependency information when long text input is used. Therefore, when product description is available, it may be beneficial to use recurrent network based models. Otherwise, CNN or HAN may suffice.

Effect of Pre-processing Site-Specific Product Labels. Neither product label clusters or cleaned labels help with the task, as shown in the lower sections of both Tables 2 and 3. In fact, when compared against the corresponding model variants using the original site-specific product labels (e.g. bi-LSTM$_{n+c.clean}$ v.s. bi-LSTM$_{n+c}$), the cleaned labels damage performance slightly, while the clusters harm performance quite signficantly. This may suggest that the original site-specific product labels could have provided useful contextual information which the supervised models have managed to capture. Either cleaning or clustering will cause this information to be lost instead. We carry out further analysis on this and discuss them in Sects. 5.2 and 5.3.

Comparison Against State of the Art. In Table 4 we compare our results obtained using the following input metadata against those reported in [16,17]: $n+c$ as this is more efficient than using d and led to very competitive results; $n+d$ which is used in [17]; and $n+d+c$ which is used in [16]. Overall, in terms of F1, our methods perform much better at Lvl.2 and Lvl.3, which are arguably more difficult because of the increasing sparsity in data. Using F1 as example, at Lvl.2, our best F1 is 9.6 points higher than the best state of the art result [17] when using $n+d+c$ with the bi-LSTM model (i.e., Lvl.2 in Table 4, 52.9 v.s. 43.3), or 5.6 points (i.e., 48.9 v.s. 43.3) higher when using the same $n+d$ input metadata as [17]. Correspondingly at Lvl.3, the highest F1 improvements are 8.5 and 4.2 (by HAN). Notice that the results cited from [17] at Lvl.2 and 3 are based on a model that uses both text and image inputs. Our models use only texts, but prove to be more effective. Our best F1 at Lvl.1 is comparable to that in [17], which however, requires a separate Information Extraction process to pre-process the data. Again it is worth to mention that our more efficient models variants using only $n+c$ have achieved very competitive results (better than our $n+d$ variants), which are much better than state of the art at Lvl.2 and 3, in both accuracy and F1.

Table 4. Comparison against results reported in the state of the art. The highest accuracy and macro-average F1 on each level are highlighted in **bold**. 1 - best results were obtained by their SVM model with *dict* features; 2 - best results were obtained by their KNN model with *Par2vec+ImageNet* features. Recall is not reported in [16]

	Lvl.1				Lvl.2				Lvl.3			
	Acc.	P	R	F1	Acc.	P	R	F1	Acc.	P	R	F1
bi-LSTM$_{n+c}$	87.3	67.0	64.8	65.7	85.7	52.5	49.8	50.8	78.9	36.2	34.7	34.8
CNN$_{n+c}$	88.0	69.4	64.1	66.3	86.4	54.7	51.1	52.3	79.9	36.8	34.9	35.1
HAN$_{n+c}$	87.2	67.8	63.0	64.9	85.8	52.2	50.9	51.2	78.8	36.3	34.6	34.8
bi-LSTM$_{n+d}$	85.4	64.2	60.3	61.8	84.0	51.1	48.1	48.9	76.6	32.1	30.8	30.8
CNN$_{n+d}$	85.7	65.8	58.5	61.1	84.0	53.0	46.1	48.0	76.5	30.6	29.1	29.2
HAN$_{n+d}$	85.3	60.5	58.7	59.5	83.5	47.3	44.7	45.2	76.4	32.7	31.0	31.1
bi-LSTM$_{n+d+c}$	**89.0**	70.4	65.8	67.4	**86.8**	54.4	52.3	**52.9**	**80.3**	36.5	35.2	35.2
CNN$_{n+d+c}$	88.6	70.4	65.1	67.3	86.6	54.8	49.6	51.2	79.5	35.8	33.0	33.5
HAN$_{n+d+c}$	88.0	66.2	64.9	65.4	86.6	53.1	50.3	51.4	79.6	36.4	35.7	**35.4**
[17], using n+d	88.3^1	74.1	64.8	**69.1**	83.8^2	43.9	42.8	43.3	77.8^2	26.6	27.2	26.9
[16], using n+d+c	47.9	49.9	-	48.9	38.0	39.5	-	38.7	25.8	26.9	-	26.3

5.2 Analysis of Product Label Clusters

We undertake further analysis to investigate why clustering did not work. Essentially, we would like an ideal clustering algorithm to place as many instances belonging to the same category as possible within the same cluster, and not include instances of too many different categories. We refer to such an ideal cluster as a 'high purity cluster'. Thus using the 50 clusters created before as example, for each cluster, we map its instances to their category labels at each of the three levels in the gold standard, and count the number of unique labels within each cluster. Figure 2 ranks these clusters by the number of unique labels for the three levels. Apparently, clustering generated too many clusters (over 50%) of very low purity (over 10 different labels) at all levels. As a result, it does not create useful features for classification, but loses contextual information that can be otherwise useful to the classifiers.

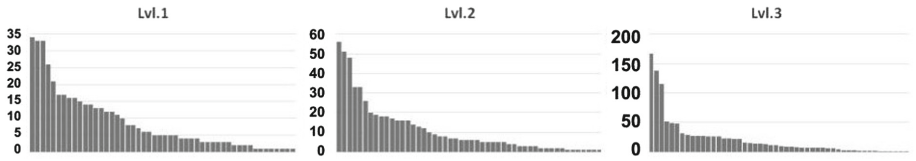

Fig. 2. Distribution of gold standard labels across each cluster (cluster number = 50) for the three classification levels. y-axis: number of unique gold standard labels. Each bar represents a separate cluster.

5.3 Analysis of Cleaned Product Labels

The cleaning process described before reduced the number of unique site-specific product labels from 4,111 to 2,394. Some examples of the different labels that were merged are shown in Table 5 (the number of different labels merged into the particular cleaned label is shown in the brackets). Although this process was shown to produce less heterogenous category labels, the cleaned labels were not shown to help with the product classification task. This might be due to a few reasons.

First, different websites may have used different names to refer to the same categories that our cleaning process failed to capture, e.g., `'Lawn & Garden'`, `'Lawn & Patio'` and `'Lawn Ornaments'`. Second, similar items (e.g., a ladies' jacket) may be assigned labels of different granularities across different sites (e.g., '`Ladies`', '`Jackets`' or '`Ladies Jackets and Coats`'). Since they do not necessarily contain lexical overlap, further method is therefore required to identify that these labels are related. Third, whilst a manual inspection seems to suggest that the cleaning process worked well with clothing items (e.g., in Table 5), in many other cases, this resulted in producing labels that are rather too specific, or less relevant. For example:

Table 5. Category cleaning results

Site-specific product labels	
Original	Cleaned
`NHL > New York rangers > New York rangers mens`	`Mens` (126)
`College > Boston college eagles > Boston college eagles mens`	
`San Jose sabercats > San Jose sabercats mens > sale items > $10 to $20`	
`Chicago bulls > Chicago bulls mens > $20 to $40`	
`ACC gear > ACC gear t-shirts`	`T-shirts` (114)
`Duke blue devils > duke blue devils t-shirts`	
`College > Florida gators > Florida gators t-shirts`	
`Dallas mavericks > Dallas mavericks home office & school`	`Home office & school` (16)
`Auburn tigers > Auburn tigers home office & school > $40 to $60`	
`NFL > Tampa bay buccaneers > Tampa bay buccaneers home office & school > NFL accessories`	

1. `‘Hardware > Tools > Carving Tools > Chisels’` was cleaned into `‘Chisels’`
2. `‘Home > Ammunition > Pistol > Rifle Ammo > Shop Centerfire Ammo by Caliber > 5 mm - 7 mm > 6.5 X 55 SWEDISH’` was cleaned into `‘6.5 X 55 SWEDISH’`
3. ‘`Electronics > Audio > Car` was cleaned into ‘`Car`’

These examples also confirmed our thoughts that the cleaning process might have caused useful contextual information from higher level labels to be lost (e.g., example 1 and 2). And it has been very challenging to identify the level of labels potentially most appropriate for classification. In certain cases, it may be beneficial to keep labels from more than one levels (e.g., example 1). We also found that 2,146 cleaned labels (89.64%) only contained one product each, which indicates that the cleaning process does not efficiently reduce the number of unique labels for most products in the dataset. One way to address this is to discard step 5 from the cleaning process, or revise it to allow multiple levels of labels to be selected (e.g., based on the ranked TF-IDF score). We will investigate this in the future.

To summarise, our current cleaning method might help with reducing the heterogeneity in the different categorisation systems used by some websites, but does not appear to be generalisable. Especially with websites that use very specific categorisation systems (which may be indicated by a larger number of levels of labels), it may produce labels too specific to be useful for classification. More work needs to be done to study if and how we can better clean the labels into an appropriate level of specificity. For example, can we use external knowledge resources (e.g., WordNet) to determine the relative level of specificity of labels at different levels? Could the target classification taxonomy be used to 'guide' the cleaning in an unsupervised way (e.g., by measuring similarity between elements in the taxonomy and labels at different levels)? If so, can and how can this information improve the cleaning process to be useful for product classification?

6 Conclusion and Future Work

This work studied product classification using product linked data on the Web. In particular, we investigated the effect of different kinds of product metadata on the classification performance. We showed that, although site-specific categorisation labels are highly inconsistent across different e-shops and therefore, cannot be directly used as product categories beyond the e-shops themselves, they can be very effective features when used for automated product classification tasks; even more so than other metadata widely used in the previous methods. By comparing three popular DNN architectures for classification tasks, we showed that when long product descriptions are available, RNN-based architectures work better; otherwise, CNN-or HAN-based architectures are more effective. Our new DNN architectures also significantly outperformed state of art on the same datasets. As future work, we will explore other strategies for cleaning the site-specific labels and their effect on the product classification task. In addition, we will also investigate two directions. First, site-specific product labels are not always available in the product linked data. Therefore, we will investigate if it is possible to generate 'fuzzy' labels based on product names or descriptions when they are absent. Second, we will look into research on mining product taxonomy from such highly heterogenous site-specific labels.

References

1. Akritidis, L., Fevgas, A., Bozanis, P.: Effective unsupervised matching of product titles with k-combinations and permutations. In: IEEE 30th International Conference on Tools with Artificial Intelligence, pp. 213–220 (2018)
2. Altınel, B., CanGaniz, M.: Semantic text classification: a survey of past and recent advances. Inf. Process. Manag. **54**(6), 1129–1153 (2018)
3. Cevahir, A., Murakami, K.: Large-scale multi-class and hierarchical product categorization for an e-commerce giant. In: Proceedings of the 26th International Conference on Computational Linguistics (COLING): Technical Papers, pp. 525–535. The COLING 2016 Organizing Committee (2016)
4. Chavaltada, C., Pasupa, K., Hardoon, D.R.: A comparative study of machine learning techniques for automatic product categorisation. In: Cong, F., Leung, A., Wei, Q. (eds.) ISNN 2017. LNCS, vol. 10261, pp. 10–17. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59072-1_2

5. Ding, Y., et al.: Automated classification of product data in e-commerce. In: Proceedings of the Business Information Systems Conference (2002)
6. Gupta, V., Karnick, H., Bansal, A., Jhala, P.: Product classification in e-commerce using distributional semantics. In: Proceedings of COLING 2016: Technical Papers, pp. 536–546. The COLING 2016 Organizing Committee (2016)
7. Ha, J.W., Pyo, H., Kim, J.: Large-scale item categorization in e-commerce using multiple recurrent neural networks. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD), pp. 107–115. ACM (2016)
8. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751. Association for Computational Linguistics (ACL) (2014)
9. Kim, Y., Lee, T., Chun, J., Lee, S.: Modified Naïve Bayes classifier for e-catalog classification. In: Lee, J., Shim, J., Lee, S., Bussler, C., Shim, S. (eds.) DEECS 2006. LNCS, vol. 4055, pp. 246–257. Springer, Heidelberg (2006). https://doi.org/10.1007/11780397_20
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the 3rd International Conference for Learning Representations (2015)
11. Kozareva, Z.: Everyone likes shopping! multi-class product categorization for e-commerce. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pp. 1329–1333. ACL (2015)
12. Lee, H., Yoon, Y.: Engineering doc2vec for automatic classification of product descriptions on O2O applications. Electron. Commer. Res. **18**(3), 433–456 (2018)
13. Li, M.Y., Kok, S., Tan, L.: Don't classify, translate: multi-level e-commerce product categorization via machine translation. CoRR. http://arxiv.org/abs/1812.05774
14. Lin, Y.C., Das, P., Datta, A.: Overview of the SIGIR 2018 eCom Rakuten data challenge. In: eCom Data Challenge at SIGIR 2018. ACM (2018)
15. McCaffrey, J.D.: Why you should use cross-entropy error instead of classification error or mean squared error for neural network classifier training. https://jamesmccaffrey.wordpress.com. Accessed Jan 2018
16. Meusel, R., Primpeli, A., Meilicke, C., Paulheim, H., Bizer, C.: Exploiting microdata annotations to consistently categorize product offers at web scale. In: Stuckenschmidt, H., Jannach, D. (eds.) EC-Web 2015. LNBIP, vol. 239, pp. 83–99. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27729-5_7
17. Ristoski, P., Petrovski, P., Mika, P., Paulheim, H.: A machine learning approach for product matching and categorization. Seman. Web **9**(5), 707–728 (2018)
18. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**(1), 1–47 (2002)
19. Xia, Y., Levine, A., Das, P., Di Fabbrizio, G., Shinzato, K., Datta, A.: Large-scale categorization of Japanese product titles using neural attention models. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL), vol. 2, Short Papers, pp. 663–668. Association for Computational Linguistics (2017)
20. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of NAACL 2016, pp. 1480–1489. Association for Computational Linguistics (2016)
21. Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B.: Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. In: Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers, pp. 3485–3495. The COLING 2016 Organizing Committee (2016)