5 Must-read Papers on Product Categorization for Data Scientists

Product categorization is the organization of products into their respective departments or categories. As well, a large part of the process is the design of the product taxonomy as a whole. Product categorization was initially a text classification task that analyzed the product's title to choose the appropriate category. However, numerous methods have been developed which take into account the product title, description, images, and other available metadata. The following papers on product categorization represent essential reading in the field and offer novel approaches to product categorization tasks.

Contents

Don't Classify, Translate: Multi-Level E-Commerce Product Categorization Via Machine Translation

Completed Research Paper

Maggie Yundi Li **Stanley Kok**
School of Computing
National University of Singapore
a0131278@comp.nus.edu.sg
skok@comp.nus.edu.sg

Liling Tan
Rakuten Institute of Technology
liling.tan@rakuten.com

Abstract

E-commerce platforms categorize their products into a multi-level taxonomy tree with thousands of leaf categories. Conventional methods for product categorization are typically based on machine learning *classification* algorithms. These algorithms take product information as input (e.g., titles and descriptions) to classify a product into a leaf category. In this paper, we propose a new paradigm based on *machine translation*. In our approach, we translate a product's natural language description into a sequence of tokens representing a root-to-leaf path in a product taxonomy. In our experiments on two large real-world datasets, we show that our approach achieves better predictive accuracy than a state-of-the-art classification system for product categorization. In addition, we demonstrate that our machine translation models can propose meaningful new paths between previously unconnected nodes in a taxonomy tree, thereby transforming the taxonomy into a directed acyclic graph (DAG). We discuss how the resultant taxonomy DAG promotes user-friendly navigation, and how it is more adaptable to new products.

Keywords: E-commerce, product categorization, classification, machine translation.

1 Introduction

Product catalogs are critical to e-commerce platforms such as Alibaba, Amazon, Rakuten, and Shopee. These catalogs typically categorize millions of products into a taxonomy tree three to ten levels deep with thousands of leaf nodes [Shen et al., 2012, McAuley et al., 2015] (Figure 1), and are continually updated with millions of new products per month from thousands of merchants. Correctly categorizing a new product into the taxonomy is fundamental to many business operations, such as enforcing category-specific listing and censorship policies, extracting and presenting relevant product attributes, and determining appropriate handling and shipping fees. Further, the accuracy and coherency of the taxonomy play important roles in customer-facing services such as product search and recommendation, and user browsing and navigation of the catalog. Clearly, given the large scale of the product taxonomy, manually categorizing a new product into it is both unscalable and error-prone, and we need automated algorithms for doing so.

To date, algorithms for product categorization have largely formulated the problem as a standard machine learning *classification* task, which takes the textual description of a product as input (e.g., *"Mix Pancake Waffle 24 OZ -Pack of 6"*) and outputs a leaf node that is the product's most likely category. Because the taxonomy is a tree and each leaf node uniquely defines a path from

root to leaf, these algorithms are effectively outputting an *existing* root-to-leaf path. Modulo the addition of the product to the leaf, these algorithms do not alter the taxonomy's tree structure.

In contrast to classification-based approaches, we map the problem of product categorization to the task of machine translation (MT). An MT system takes text in one language as input (traditionally denoted as f) and outputs its translation as a sequence of words in another language (denoted as e). The input f maps to the textual description of a product, and the output e maps to the sequence of categories and sub-categories in a root-to-leaf path (e.g., `Baking Supplies` $\rightarrow$ `Flour & Dough` $\rightarrow$ `Pancake & Waffle Mixes`). By framing product categorization as an MT problem, our approach offers several operational and technical advantages over previous algorithms.

First, large e-commerce companies typically operate their sites globally in a variety of languages (e.g., *www.rakuten.com* in English and *www.rakuten.co.jp* in Japanese), and have invested heavily in their machine translation capabilities. By utilizing these existing MT systems for the task of product categorization (rather than developing a new disparate system), we are reducing the technical debt that these companies incur. They have fewer algorithms to be cognizant of, fewer systems to develop, less bugs to fix, and consequently lower maintenance cost.

Second, machine translation systems, through the use of deep learning [Goodfellow et al., 2016], have improved their accuracy by leaps and bounds in recent years [Kalchbrenner and Blunsom, 2013, Sutskever et al., 2014, Bahdanau et al., 2015], even to the extent of achieving human parity on some language pairs [Hassan et al., 2018]. By mapping the problem of product categorization to one of machine translation, we bring the best of MT technology to bear on the problem of product categorization in a cost-effective manner. In Section 4, we provide empirical results demonstrating that our MT approach outperforms state-of-the-art systems.

Third, machine translation systems are by nature resilient to the vagaries and noise present in language, and thus are robust to errors in a product's textual description and the varieties of ways in which a product can be specified (e.g., *"Mix Pancake Waffle 24 OZ -Pack of 6"* and *"Packet of six; waffle pancake mix; 24 ounces"* refer to the same product). This makes MT systems ideal for dealing with the uncertainties inherent in a product's natural language description.

Fourth, our MT approach not only outputs pre-existing root-to-leaf paths in a taxonomy tree, it also produces novel root-to-leaf paths that do not exist in the taxonomy. These novel paths transform the structure of a product catalog from a tree to a directed acyclic graph (DAG). This is a powerful transformation, offering potentially multiple root-to-leaf paths to a single product (rather than just one path as in previous systems). This better conforms to psychological findings that humans tend to view an object in multiple ways [Heit and Rubinstein, 1994, Ross and Murphy, 1999, Shafto and Coley, 2003, Shafto et al., 2005]. For example, a waffle is regarded as being primarily carbohydrates because it is made of flour; however, it is often also regarded as a breakfast food. The different ways of thinking about a waffle underline the different ways of thinking about food: as a system of taxonomic categories like flour and sugar, or as situational categories like breakfast foods and dinner fare. Likewise, in other product domains, items have different properties, and more than one system of categories are required to fully represent these properties. By creating multiple root-to-leaf paths, our system better caters to human intuition than previous systems, and can potentially improve customer-facing applications such as user product navigation. For example, by having both of the following paths in a product DAG, a user who predominantly views a waffle mix as baking supplies, and a user who views it as a breakfast food can both expeditiously navigate to what they need.

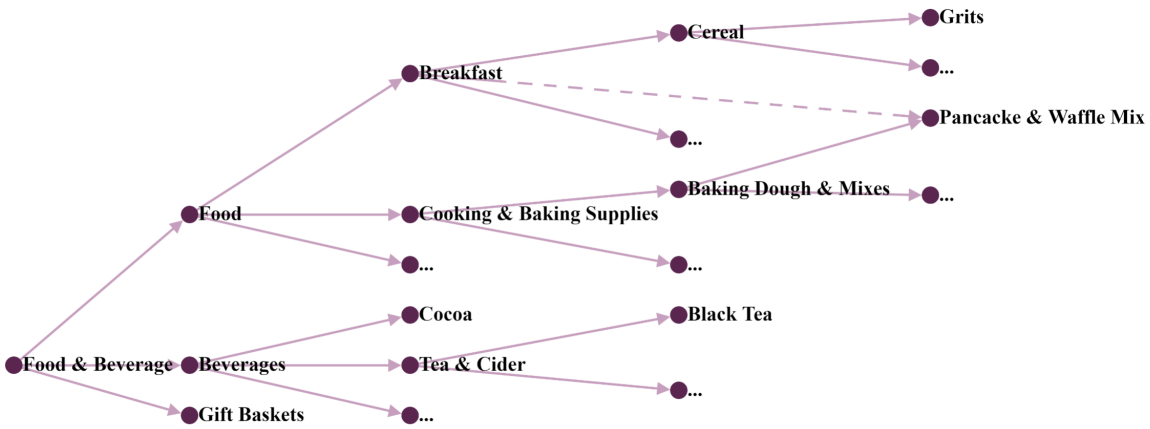

Figure 1: Example of a product catalog organized in a tree structure (solid lines and circles). Individual products are added to leaf nodes (e.g., `Grits`). The addition of the dotted edge turns the tree into a directed acyclic graph (DAG), and there are now more than one root-to-leaf paths to the leaf node `Pancake & Waffle Mix`.

- `Baking Supplies` → `Flour & Dough` → `Pancake & Waffle Mixes`
- `Breakfast Foods` → `Pancake & Waffle Mixes`

To our knowledge, we are the first to apply a machine translation approach to the problem of e-commerce product categorization.

Next, we briefly review related work (Section 2). We then describe state-of-the-art machine translation systems, and how we use them for product categorization (Section 3). Next, we describe our datasets, experimental methodology, comparison systems, and empirical results (Section 4). Then, we provide a qualitative analysis of our results (Section 5). Finally, we conclude with future work (Section 6).

2 Related Work

The vast majority of product categorization systems is based on machine learning *classification* algorithms. These systems can be dichotomized into (a) those that classify a product in a single step into one of thousands of leaf nodes in a taxonomy tree, and (b) those that classify the product step-wise, first into higher-level categories, and then into lower-level subcategories. This dichotomy is due to the inherent skewness (long tail phenomenon [Anderson, 2006]) that is typical of e-commerce products – a large proportion of products are distributed over a small number of categories (leaf nodes) with the remaining fraction of products sprinkled over a large number of remaining categories. This imbalance of category sizes poses a challenge to classification algorithms, which generally require the sizes of categories to be approximately balanced. To circumvent this problem, the step-wise approach first performs classification across top-level categories, each of which aggregates products in its lower-level subcategories to ameliorate the data imbalance problem. After assigning a product to a top-level category, the step-wise approach repeats the process and performs classification across the sub-categories. Because these sub-categories belong to the same top-level category, they are likely to belong to the same class of products, and hence have

less imbalance in their sizes. However, the step-wise approach suffers from two shortcomings: (a) errors from classifiers at previous steps get propagated to classifiers at subsequent steps with no chance of recovery, and (b) the number of classifiers grows exponentially with every step. Unlike the step-wise systems, the single-step approach does not have these drawbacks, but must contend with the category imbalance problem at its full severity at the leaf nodes.

A variety of single-step classifiers have been used for product categorization. Yu et al. (2013) explore a gamut of word-level features (e.g., n-grams), and use a support vector machine (SVM; Cortes and Vapnik [1995]) as their classification algorithm. Chen and Warren (2013) sensitize the objective function of an SVM to the average revenue loss of erroneous product classifications, thereby trading high revenue-loss errors for low revenue-loss ones. Sun et al. (2014) use simple classifiers (e.g., naive Bayes, k-nearest neighbors, and perceptron), and recruit manual labor via crowdsourcing to flag their errors. Kozareva (2015) uses a variety of features (e.g., n-grams, latent Dirichlet allocation topics [Blei et al., 2003], and word2vec embeddings [Mikolov et al., 2013]) in a multi-class algorithm. Both Ha et al. (2016) and Xia et al. (2017) use deep learning to learn a compact vector representation of the attributes of a product (e.g., product title, merchant ID, and product image), and use the representation to classify the product. They differ in terms of the kinds of deep learning model used. The former uses recurrent neural networks [Hochreiter and Schmidhuber, 1997] and the latter uses convolutional neural networks [LeCun et al., 1998].

Several step-wise classifiers have also been used for product categorization. Shen et al. (2012) use simple classifiers (e.g., naive Bayes and k-nearest neighbors) in the first step, then an SVM to assign a product to a leaf node in the second step. Das et al. (2016) explore the use of gradient boosted trees [Friedman, 2000] and convolutional neural networks in each of three steps. However, they only evaluated the accuracy of their approach at the top two levels of a product taxonomy (a simpler problem because of the smaller number of categories at the top levels), and did not provide the accuracy at the leaf nodes. Cevahir and Murakami (2016) use deep belief networks (DBNs; Hinton et al. [2006]) and k-nearest neighbors (KNNs) in a two-step approach. Because the number of models grows exponentially with the number of steps, a large number of models are trained (72) even though only two steps are involved. This large number of models makes it impractical to deploy their approach in a real-world production setting. They also use a single-step approach (termed `CUDeep`) consisting of one DBN and one KNN, and found that it is competitive against the 72-model, two-step approach. With only two models, their single-step approach trains faster and is feasible for real-world deployment. In our experiments in Section 4, we compare our machine translation (MT) approaches against `CUDeep`.

All of these classification systems assign a product into an existing leaf node (which is equivalent to a unique existing root-to-leaf path). Unlike them, our machine translation approach is able to create both existing root-to-leaf paths and novel non-existing paths for a product, thereby presenting a richer representation of a product to both downstream business operations and customer-facing applications. In addition, our MT approach outperforms classification algorithms in terms predictive accuracy (results in Section 4).

3 Machine Translation Systems

A machine translation (MT) system takes as input a sentence $\mathbf{f} = f_1 f_2 \dots f_m$ with m tokens in a *source* language, and finds its best translation $\mathbf{e} = e_1 e_2 \dots e_n$ with n tokens in a *target* language

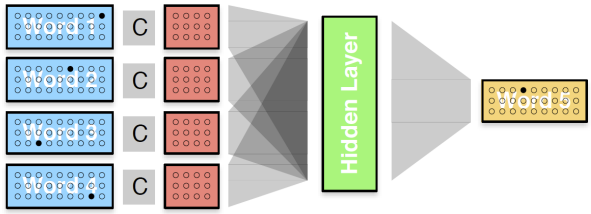

Figure 2: A language model encoded by a feed forward neural network. Previous 4 words are represented as 1-of-N vectors (blue), compressed into continuous vectors (red) using the same matrix C for all words, passed through a hidden layer, and then used to predict the next word as a 1-of-N vector (yellow) (Image from [Koehn, 2017])

(this is expressed mathematically as $\mathbf{e} = \text{argmax}_{\mathbf{e}} P(\mathbf{e}|\mathbf{f})$).

In the past, the predominant MT approach was phrase-based machine translation (PBMT), which is grounded in information theory and statistics. Though moderately successful in its heyday, it has recently been eclipsed by neural machine translation (NMT) approaches that utilize deep learning [Goodfellow et al., 2016]. Deep learning contributes to NMT by incrementally building more sophisticated models of languages, and then linking them to models of the translations.

First, deep learning compresses the common 1-of-N representation of a word (i.e., an N-dimensional vector with a single 1 at the index corresponding to the word and 0's everywhere else) into a smaller continuous-valued feature vector (also known as an embedding) [Mikolov et al., 2013]. Intuitively, this vector provides a distributed continuous representation of an input word, with the vector's continuous values varying gradually among similar words and differing greatly among dissimilar ones. Such vectors are then used to represent a probability distribution over the words they represent. The continuity in the vectors automatically smoothens the distribution and alleviates the data sparsity problem (this occurs when the vocabulary of a language is large, and too few occurrences of various words appear in a corpus).

Second, deep learning allows complex features to be learned automatically from text. Building upon the vector embeddings of words, we could connect these to another layer of vectors that are collectively termed *hidden* layers, and in turn, connect those to other hidden layers. As more hidden layers are added (one on top of another) to form a *feedforward neural network*, they can model more complex interactions and features among words in the input text. Feedforward neural networks can model a language by using the previous n words in a sentence to predict the current word (Figure 2). This way a feedforward neural network encodes the probability distribution of a next word given its previous words as context.

Third, more powerful language models can be built using recurrent neural networks (RNNs) [Hochreiter and Schmidhuber, 1997]. An RNN is similar to a feedforward neural network in having an input layer of words that is connected to a hidden layer, which in turn is connected to an output layer representing a probability distribution over words. It differs by linking the hidden layer back to itself with recurrent connections, which propagate information across a sequence of words in an RNN. Conceptually, when an RNN is "unrolled" it is equivalent to a feedforward neural network with an infinite number of connected hidden layers stacked on top of one another. Because of this depth of hidden layers, it can potentially learn complex dependencies among words (Figure 3).

Fourth, Cho et al. (2014b) extended RNNs for machine translation by creating the encoder-

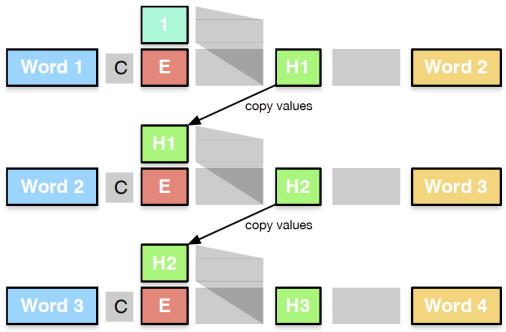

Figure 3: A language model encoded as a recurrent neural network. After predicting Word 2 (yellow), we reuse the hidden layer H1 (green) together with the correct Word 2 (blue) to predict Word 3 (yellow). (Image from [Koehn, 2017])

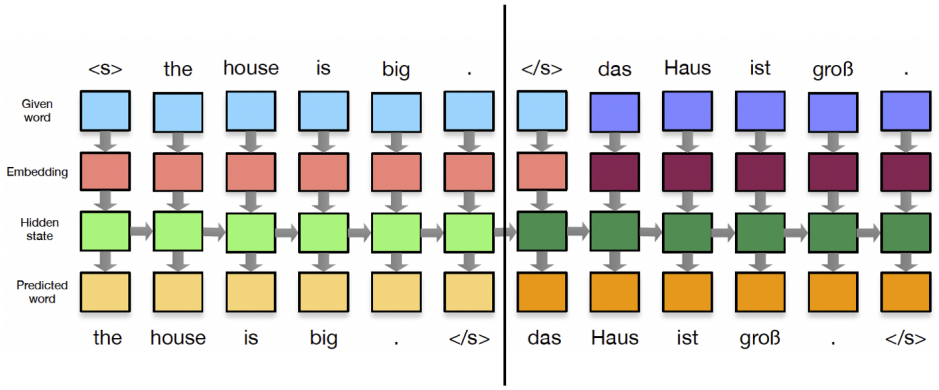

Figure 4: The encoder-decoder model, also known as the sequence-to-sequence model (Seq2Seq). The light green boxes to the left of the vertical line make up the encoder RNN. The first dark box to the right of the vertical line encodes the entire source sentence. As we move from left-to-right, this source encoding is use to generate words in the target language. (Image from [Koehn, 2017])

decoder model (also known as the sequence-to-sequence model (Seq2Seq)). This model concatenates two RNNs together, one that encodes the source language, and one that decodes the target language. In Figure 4, the light green boxes to the left of the vertical line make up the encoder RNN, which encodes the words in the source sentence. The first dark green box to the right of the vertical line encodes the entire source sentence. As we move from left-to-right, this source encoding is used to generate words in the target language.

Fifth, a major advancement in NMT occurred through the use of a memory mechanism to align the source sequence positions to the target sequence state. Cho et al. [2014a] observe that Seq2Seq models deteriorate quickly as the input sequence length increases. This is because the decoder is forced to make a hard decision to predict a target word at every state. Bahdanau et al. [2015] propose an attention mechanism that allows the Seq2Seq model to focus on a set of positions from the source sentence to form a context vector that are most relevant to the current state in the target sequence. It uses the context vector from the attention mechanism to predict the current word. Luong

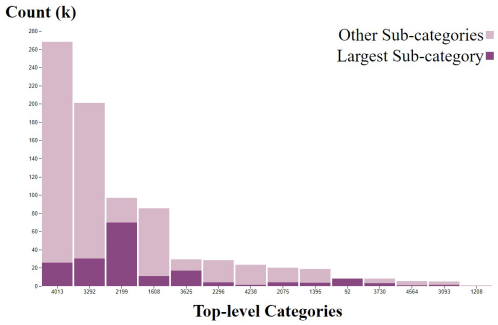

Figure 5: Skewed product distribution in RDC.

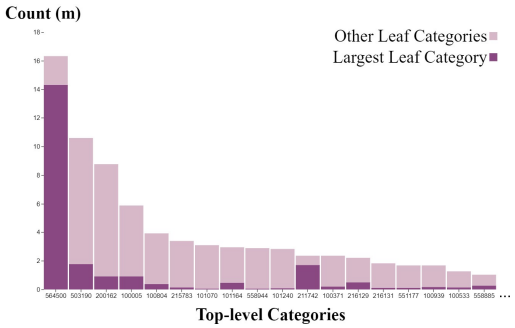

Figure 6: Skewed product distribution in Ichiba.

et al. [2015] extends the attention mechanism by introducing both global and local attention mechanisms. The global attention mechanism functions as described above by considering all positions in a source sentence; the local attention mechanism restricts its focus to the vicinity of source positions that best correspond to the target position that is to be predicted. Attentional Seq2Seq models are among the best performers on standard machine translation benchmarks. Hence, we employ one such model [Luong et al., 2015] for our experiments in Section 4.

Sixth, Vaswani et al. [2017] create an NMT model called Transformer that dispenses with RNNs. RNNs requires a time-consuming, left-to-right, word-by-word traversal of the entire input sentence in order to model the full span of a sentence. However, such a traversal is not parallelizable and severely slows down model training. By discarding RNNs, the Transformer model becomes highly parallelizable, and it retains the ability to model the entire span of a sentence through the use of *self-attention*. In an attentional Seq2Seq model, the attention mechanism models the association between an output word with every input word. In self-attention, we compute the association between each input word and every other input word, thereby disambiguating an input word using other input words as context. Further, the Transformer uses *multi-head* self-attention, i.e., it applies self-attention in multiple representation spaces (e.g., one that captures the syntax of a language, and another that captures the morphology) to enrich the representation of a word. The Transformer model is among the best performers on standard machine translation benchmarks, and we use it for our experiments in Section 4.

4 Experiments

4.1 Datasets

We used the following two e-commerce datasets for our experiments (Table 1 provides a summary of their characteristics).

- **Rakuten Data Challenge (RDC)**[1]. This dataset from *www.rakuten.com* contains 800,000 product titles in English with their respective multi-level category labels. We lowercase all product titles and tokenize them with the Moses tokenizer [2].

[1] https://sigir-ecom.github.io/data-task.html

[2] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

- **Rakuten Ichiba**[3]. This dataset from *www.rakuten.co.jp* consists of 280 million products listed by over 40,000 merchants and has 28,338 categories. We remove duplicate product listings and those in the '`Others`' category that are erroneously assigned by merchants. After this, we are left with about 100 million Japanese product titles that are paired with their multi-level category labels. We tokenize the product titles with the MeCab Japanese segmenter [Kudo et al., 2004].

	Rakuten Data Challenge (RDC)	**Ichiba**
Language	English	Japanese
Source	`www.rakuten.com`	`www.rakuten.co.jp`
Data size	800,000	100,436,907
No. of First-Level Categories	14	35
No. of Unique Categories	3,008	21,819
Tokenization	Moses tokenizer + lowercase	MeCab

Table 1: Summary of Rakuten Data Challenge (RDC) and Ichiba Datasets

In both datasets, the products are assigned to the leaf nodes of a taxonomy tree. For each product, we have its title (e.g., *"Mix Pancake Waffle 24 OZ -Pack of 6"*) and its root-to-leaf path (e.g., `Baking Supplies` → `Flour & Dough` → `Pancake & Waffle Mixes`). All the models in Section 4.2 take the product title as input to predict its associated root-to-leaf path (we term this path the *label* of the product).

The distribution of products across categories in both datasets is skewed towards the most popular categories as is usually the case in e-commerce domains [He and McAuley, 2016, Xia et al., 2017]. Figure 5 and 6 show the number of products in each category at the top-level of the taxonomy tree (each vertical bar reflects the number of products in that category). These figures show that a majority of products is assigned to a few categories, and the rest are spread across the remaining categories in a long tail. Further, the dark-colored portion of each bar represents the sub-category with the highest count within the top-level category, and the lighter tip of the bar represents all other sub-categories within that top-level category. As can be seen, the distribution within some top-level categories may also be skewed.

We randomly split both the RDC and Ichiba datasets in a stratified manner into their respective training, validation and test sets in the proportion of 80/10/10. The validation set was used to determine the early stopping criteria for our NMT models.

4.2 Models

For our neural machine translation (NMT) models, we use the attentional Seq2Seq model of Luong et al. [2015] and the Transformer model of Vaswani et al. [2017] as implemented in the Fairseq toolkit (commit `5d99e13`)[4] (see Section 3 for their descriptions). The hyperparameters of our

[3] `https://rit.rakuten.co.jp/data_release/`

[4] `https://github.com/pytorch/fairseq/tree/master/fairseq`

models are given in Table 2. Note that the recurrent neural network (RNN) hidden layer size is specific to the attentional Seq2Seq model while the feedforward network (FFN) hidden layer and attention heads hyperparameters are only used in the Transformer model. We also ensemble the attentional Seq2Seq model and the Transformer model together by averaging their decoder outputs.

As mentioned, each product is associated with its title and root-to-leaf path. Our NMT models consider the title to be a sentence in a source language (English for RDC and Japanese for Ichiba), and translate it into a sequence of tokens corresponding to the nodes in a root-to-leaf path.

	Attentional Seq2Seq	**Transformer**
Input/Output Embedding Dimension	512	512
RNN Hidden Layer Size	1,024	-
FFN Hidden Layer Size	-	2,048
Stacked Layers	1	6
Dropout	0.2	0.2
Attention Heads	-	8
No. of Parameters	7,5435,103	99,105,792

Table 2: Hyperparameters of attentional Seq2Seq and Transformer Models

We compare our machine translation models with a traditional classification-based system `CUDeep` [Cevahir and Murakami, 2016] that achieved state-of-the-art performance on the Ichiba dataset (this model is described in Section 2). `CUDeep` trains a deep belief network using a stacked restricted Boltzmann machine architecture [Hinton and Salakhutdinov, 2006], and learns an encoder that embeds a product title into a vector representation. Next, it trains a feedforward neural layer to map the vector representation to a predicted product category. Henceforth, we will term this model DBN. Aside from using deep belief networks, `CUDeep` also uses K-nearest neighbors (KNN) [Cover and Hart, 1967] to predict product categories by mapping a product title to the 1-nearest neighbor's category that is seen in the training data. We also combined the outputs of the DBN and KNN models to form a DBN+KNN ensemble and averaged the probabilities of their category predictions to re-rank the predictions.

4.3 Evaluation Metrics

Cevahir and Murakami [2016] previously used n-best accuracy as the evaluation metric for the Ichiba dataset, and Lin et al. [2018] applied the support weighted F-score as the metric to evaluate the Rakuten Data Challenge (RDC) dataset. To keep the comparisons consistent across datasets, we opted for the single metric of weighted F-score. This metric weighs the accuracy in each leaf node by its number of products, and is thus better suited for multi-class prediction in skewed datasets. The multi-class F1 score is computed as follows:

$$\begin{aligned} TP_c &= |\hat{y}_c \cap y_c| \qquad P_c = \frac{TP}{|\hat{y}_c|} \qquad R_c = \frac{TP}{|y_c|} \\ F_c &= \frac{2 \cdot P_c \cdot R_c}{P_c + R_c} \qquad F_c^{\,weighted} = \frac{1}{|C|} \sum_{c \in C} TP_c \cdot F_c \end{aligned} \tag{1}$$

where C represents all possible categories/labels, $\hat{y}_c$ are the products that are labeled by the system as c, y_c are products with c as the true labels, and $TP_c, P_c, R_c, F_c, F_c^{weighted}$ are respectively the true

positives, precision, recall, F-score, and weighted F-score for label c. Due to the highly skewed category distribution, we use the *weighted* variant of the precision, recall and F-score [5], where the scores across labels are summed and weighted by their true positive values. (NB: For weighted variants of precision, recall, and F-score, the F-score may not lie between precision and recall.)

4.4 Results

Table 3 reports the weighted precision (P), weighted recall (R), and weighted F-scores (F) on the test sets of the RDC and Ichiba datasets. These scores only deem a label (i.e., a predicted root-to-leaf path) to be correct if it is an exact match to the ground truth. As long as one node in the path is wrong (even when the leaf node is correct), the prediction is deemed wrong. Note that this penalizes our NMT models because they can predict novel root-to-leaf paths that do not exist in a taxonomy tree, and can thus arrive at the correct leaf nodes via multiple paths (and not only through the unique root-to-leaf path in the taxonomy). Even though CUDeep also predicts a root-to-leaf path, that path is an existing one in the taxonomy tree and is uniquely determined by the leaf node. To allow for a consistent comparison with CUDeep, we decided to determine correctness by the full root-to-leaf path. If we consider the correctness of the leaf nodes only, our results will surpass those shown below.

		RDC			Ichiba		
		P	**R**	**F**	**P**	**R**	**F**
CUDeep	**Deep Belief Net (DBN)**	72.19	74.72	72.86	78.09	78.29	77.52
	K-Nearest Neighbors (KNN)	71.14	72.10	70.94	79.24	78.69	78.66
	DBN+KNN	73.46	75.57	73.85	82.65	82.27	82.05
Our NMT Models	**Attentional Seq2Seq**	74.03	73.43	72.50	84.70	82.39	82.08
	Transformer	74.44	75.25	73.83	83.79	83.59	**84.74**
	Seq2Seq+Transformer	**75.22**	**75.65**	**74.19**	**85.08**	**84.31**	84.26

Table 3: Results of our NMT Systems vs CUDeep Classification Systems

	RDC					
	p5			**p95**		
	P	**R**	**F**	**P**	**R**	**F**
DBN+KNN	73.08	75.17	73.32	74.11	75.98	74.22
Seq2Seq+Transformer	74.81	75.23	73.68	75.76	76.05	74.57

Table 4: Confidence Interval of 1000 iterations of Bootstrap Resampling on the Best Performing Models on RDC dataset

[5] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

	Ichiba					
	p5			**p95**		
	P	**R**	**F**	**P**	**R**	**F**
DBN+KNN	82.64	82.24	82.00	82.71	82.30	82.07
Seq2Seq+Transformer	85.07	84.28	84.22	85.13	84.35	84.28

Table 5: Confidence Interval of 1000 iterations of Bootstrap Resampling on the Best Performing Models on Ichiba dataset

From Table 3, we see that our Transformer model outperforms both CUDeep single models (DBN and KNN) on both datasets (weighted F-scores of 73.83 and 84.74 on the RDC and Ichiba test sets respectively). The Transformer also outperforms the DBN+KNN ensemble on the Ichiba dataset, and is competitive on the RDC dataset. Our attentional Seq2Seq model has mixed results on RDC, but outperforms all CUDeep models for all metrics on the larger Ichiba dataset. Our Seq2Seq+Transformer ensemble is the best performer across the board. It is better than both our single models and all CUDeep models. The only exception is that the weighted F-score of our Seq2Seq+Transformer model is marginally lower than that of our Transformer model.

To further confirm that our system outperforms CUDeep, we conducted 1000 iterations of bootstrap resampling on the best performing model in each system to find out the 95% confidence interval of their performance scores. From Tables 4 and 5, we see can conclude that our attentional Seq2Seq+Transformer ensemble surpasses CUDeep's DBN+KNN ensemble.

	80-10-10	**60-10-30**	**40-10-50**	**20-10-70**
DBN+KNN	73.85	74.08	71.24	61.27
Seq2Seq+Transformer	74.19	74.94	73.77	69.58

Table 6: Effects of Data Size with Respect to Systems' Weighted F-score

We investigated the effect of training data size on the performances of the systems. Table 6 presents the F-scores of the ensembled systems with respect to various train-validation-test sizes. For instance, "60-10-30" indicates that a model was trained, validated and tested on 60%, 10% and 30% of the data respectively.

We note that the 60-10-30 split has higher F-scores than the 80-10-10 split for both ensembled systems. This is due to the random split of the 80-10-10 data giving its test set a higher proportion of classes with one instance (i.e., these classes do not appear in the training set). The instances of such classes are impossible to correctly predict for both systems. From Table 6, we see that our machine-translation-based Seq2Seq+Transformer ensemble is consistently more robust to reductions in data sizes than the DBN+KNN ensemble. Even with only 20% of the training data, the performance of our Seq2Seq+Transformer ensemble does not degrade as much as that of the DBN+KNN model. Further, our Seq2Seq+Transformer ensemble consistently outperforms the DBN+KNN model across data sizes.

	RDC	Ichiba
RNN	106	5,183
Transformer	76	113,287
RNN + Transformer	124	48,455

Table 7: Count of Full-Path Categories Created

5 Analysis

As discussed in previous sections, our NMT models generate root-to-leaf paths based on the vocabulary of categories. This generation allows new paths to be created based on product titles. Although such system-created paths utilize *existing* nodes in a product taxonomy tree, the paths (which are permutations of nodes) need not pre-exist in the tree. When the paths are added to the tree to form new edges between nodes, they transform the tree into a DAG, which offers a richer representation of the products.

We present the count of novel categorization paths created by each of our models in Table 7. In this section, we qualitatively analyze some notable examples of created paths in the English-language RDC dataset.

The product '*Hal Leonard Neil Young-Rust Never Sleeps Guitar Songbook*' has its ground truth root-to-leaf path as `Home & Outdoor > Hobbies > Musical Instruments > Misc Accessories > Sheet Music`. Our Transformer model's predicted path is identical to the ground truth except that it omits *Musical Instruments* from the path. This is intuitively correct because the product is a songbook, which does not belong to the *Musical Instruments* sub-category. This example suggests that our NMT models can prune and restructure the taxonomy tree to more accurately describe products.

Another notable example is the product *Epson WorkForce Pro WP-4023 Inkjet Printer C11CB30 231 Compatible 10ft White*, which has the ground truth category of `Electronics>...>Printers`. Our NMT model predicts the root-to-leaf path as `Office Supplies>...>Printers`, which is intuitively correct because printers constitute general office supplies. This suggests that our NMT models can enrich the representation of products.

Our system-created paths are not constrained by the existing hierarchical ordering of nodes in a taxonomy tree (e.g., it can place a leaf-category node at its start and a top-level-category node at its end). However, we observe that the paths created in our experiments all begin with top-level-category nodes and end with leaf nodes. This is because our machine translation models have successfully learned from their training data the strong bias of top-level-category nodes to appear first and leaf nodes to appear last. Beyond that, the paths conform less to the structure of the taxonomy tree, with some spanning across branches, and moving from lower-level categories to higher-level ones.

6 Conclusion & Future Work

Product categorization is an important problem for e-commerce companies. By changing the framing of the problem from the traditional one of classification to one of machine translation, we show that state-of-the-art machine translation (MT) models surpass previous classification approaches

in categorizing products in two large real-world e-commerce datasets.

Besides enhancing the performance of product categorization, our NMT models also create novel root-to-leaf category paths. These novel paths can help to adapt a product taxonomy to changes in product listings. They also suggest ways to restructure the product taxonomy so that the category paths better accommodate a user's multiple conceptualizations of an product.

Future work includes: crowdsourcing the evaluation of novel root-to-leaf paths, experiments with more MT models and classification models, automatic induction of the product taxonomy from data, etc.

Acknowledgements. This work was done during Maggie Li's internship at the Rakuten Institute of Technology Singapore (RIT-SG). We thank RIT for the collaboration, support and computation resources for our experiments. We also thank Ali Cevahir for his advice on the `CUDeep`-related experiments.

References

Anderson, C. (2006). *The Long Tail*. Hyperion.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Cevahir, A. and Murakami, K. (2016). Large-scale multi-class and hierarchical product categorization for an e-commerce giant. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 525–535.

Chen, J. and Warren, D. (2013). Cost-sensitive learning for large-scale hierarchical classification. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 1351–1360.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Das, P., Xia, Y., Levine, A., Di Fabbrizio, G., and Datta, A. (2016). Large-scale taxonomy categorization for noisy product listings. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 3885–3894. IEEE.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Ha, J.-W., Pyo, H., and Kim, J. (2016). Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 107–115. ACM.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., Liu, S., Liu, T.-Y., Luo, R., Menezes, A., Qin, T., Seide, F., Tan, X., Tian, F., Wu, L., Wu, S., Xia, Y., Zhang, D., Zhang, Z., and Zhou, M. (2018). Achieving human parity on automatic chinese to english news translation. *Computing Research Repository*, abs/1803.05567.

He, R. and McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. International World Wide Web Conferences Steering Committee.

Heit, E. and Rubinstein, J. (1994). Similarity and property effects in inductive reasoning. *Journal of Experimental Psychology. Learning, Memory, and Cognition*, 20 2:411–22.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Koehn, P. (2017). Neural machine translation. In *Statistical Machine Translation*. arXiv:1709.07809.

Kozareva, Z. (2015). Everyone likes shopping! multi-class product categorization for e-commerce. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Kudo, T., Yamamoto, K., and Matsumoto, Y. (2004). Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

LeCun, Y., Bottou, L., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.

Lin, Y.-C., Das, P., and Datta, A. (2018). Overview of the sigir 2018 ecom rakuten data challenge.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

McAuley, J. J., Pandey, R., and Leskovec, J. (2015). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. acm.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119.

Ross, B. H. and Murphy, G. L. (1999). Food for thought: Cross-classification and category organization in a complex real-world domain. *Cognitive Psychology*, 38(4):495–553.

Shafto, P. and Coley, J. D. (2003). Development of categorization and reasoning in the natural world: Novices to experts, naive similarity to ecological knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29 4:641–649.

Shafto, P., Kemp, C., Baraff, E., Coley, J. D., and Tenenbaum, J. B. (2005). Context-sensitive induction. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pages 2003–2008.

Shen, D., Ruvini, J.-D., and Sarwar, B. (2012). Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 595–604, New York, NY, USA. ACM.

Sun, C., Rampalli, N., Yang, F., and Doan, A. (2014). Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proceedings of the VLDB Endowment*, 7:1529–1540.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112,.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Xia, Y., Levine, A., Das, P., Di Fabbrizio, G., Shinzato, K., and Datta, A. (2017). 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: Volume 2, Short Papers*. Association for Computational Linguistics.

Yu, H.-F., Ho, C.-H., Arunachalam, P., Somaiya, M., and Lin, C.-J. (2013). Product title classification versus text classification. Technical report, Department of Computer Science, National Taiwan University, Taipei, Taiwan.

Large-Scale Categorization of Japanese Product Titles Using Neural Attention Models

Yandi Xia, Aaron Levine, Pradipto Das
Giuseppe Di Fabbrizio, **Keiji Shinzato** and **Ankur Datta**
Rakuten Institute of Technology, Boston, MA, 02110 - USA
{ts-yandi.xia, aaron.levine, pradipto.das,
giuseppe.difabbrizio, keiji.shinzato, ankur.datta}@rakuten.com

Abstract

We propose a variant of Convolutional Neural Network (CNN) models, the Attention CNN (ACNN); for large-scale categorization of millions of Japanese items into thirty-five product categories. Compared to a state-of-the-art Gradient Boosted Tree (GBT) classifier, the proposed model reduces training time from three weeks to three days while maintaining more than 96% accuracy. Additionally, our proposed model *characterizes* products by imputing attentive focus on word tokens in a language agnostic way. The attention words have been observed to be semantically highly correlated with the predicted categories and give us a choice of automatic feature extraction for downstream processing.

1 Introduction

E-commerce sites provide product catalogs with millions of items that are continuously updated by thousands of merchants. To list new products in an e-commerce marketplace and expose them to online users, merchants must supply several pieces of meta-data. Rakuten Ichiba[1] is an example of such a large-scale e-commerce platform in Japan, hosting more than 239 million products from over $44,000$ merchants. To improve search relevance and catalog navigation, products must be categorized into a taxonomy tree with thousands of nodes several levels deep (e.g., 6 levels with more than $43,000$ nodes for Rakuten Ichiba).

For such a large taxonomy, manual item categorization is often inaccurate and inconsistent across merchants. Automatic categorization into a full taxonomy tree is feasible, although a layered approach is more practical for scalability and accuracy reasons. For instance, Shen et al. (2012b) uses a two level strategy to combat imbalance. Das et al. (2017) also exploits a similar 2-step cascade categorization.

This work focuses on large-scale categorization of Japanese products for the top-level categories of the Rakuten Ichiba catalog taxonomy. Examples of top-level product categories include *Clothing*, *Electronics*, *Shoes*, and *Books & Media*, as well as less represented categories such as *Travel*, *Communication*, and *Cars & Motorbikes*. We compare Convolutional Neural Network (CNN), Attention CNN (ACNN), and state-of-the-art Gradient Boosted Tree (GBT) classification models trained on more than 18 million catalog items. ACNN model performance is comparable to that of the GBT model with a 7-fold reduction in training time without the need for feature engineering. Additionally, ACNN's attention mechanism selects salient words that are semantically relevant to identifying categories and potentially useful for automatic language-agnostic feature extraction.

2 Related Work

Research on large-scale product categorization has recently come into focus (Shen et al., 2011; Shen et al., 2012b; Shen et al., 2012a; Yu et al., 2013; Chen and Warren, 2013; Sun et al., 2014; Kozareva, 2015). Most contemporary work in this area points out the noise issues that arise in large product datasets and address the problem with a combination of a wide variety of features and standard classifiers. However, the existing methods for noisy product classification have only been applied to English. Their efficacy for *moraic* and *agglutinative* languages such as Japanese remains unknown.

Application of deep learning techniques is gaining grounds for text categorization applications (Kim, 2014; Ma et al., 2015; Yang et al., 2016), however, their application to product data has only been recently reported. Pyo et al. (2016) uses Re-

[1] Ichiba http://www.rakuten.co.jp

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 663–668,
Valencia, Spain, April 3-7, 2017.

current Neural Networks (RNNs) without word embeddings. Furthermore, unlike our proposed model, RNNs cannot impute tokens in title text with attention weights that can be helpful in downstream applications.

Dependency-based deep learning (Ma et al., 2015) has proven useful for sentence classification, but product titles, whether in English or Japanese, are not beholden to the same grammatical rigor. We do not use deeper linguistic techniques such as parsing or Part-of-Speech tagging due to the language-agnostic nature of our categorization techniques. Attention-based deep learning models have been used in the image domain (Xu et al., 2015) and in the generic text classification domain (Yang et al., 2016). However, to the best of our knowledge, this is the first work on simultaneous categorization and attention based salient token selection on Japanese product data.

3 Dataset Characteristics

The data we use is a selection of product listings from Rakuten Ichiba, a large Japanese E-commerce service for thousands of merchants. Each merchant submits their own product data, leading to item names with inconsistent formats and disagreements on genres for the same sets of items. Our training set consists of $18,199,420$ listings and the test set of $2,274,928$ listings, for a 90/10% split. The training data is uniformly sampled before the split. Due to the popularity of certain product types, the balance is unevenly distributed between 35 top-level categories: There are $1,869,471$ in the largest category, but only 925 in the smallest.

Statistics	Training set	Test set
Mean character count/title	62.510	62.506
Standard deviation	31.496	31.492
Mean word count/title	23.187	23.188
Standard deviation	11.945	11.942
Mean character count/word	05.800	05.799

Table 1: Word and character level statistics for our Rakuten Ichiba dataset.

Table 1 shows character and word level statistics per product title in the training and test set. It is evident from the mean word and character counts that, on average, Japanese product titles in our dataset are quite verbose. We thus expect that convolutional neural network based models that rely on full context of the input text, to work better for the categorization task.

4 Modeling Approaches

4.1 Attention Neural Networks

Our ACNN model is related to the work described in Yang et al. (2016), which has been more suitable for well-formed document classification tasks with a limited number of categories.

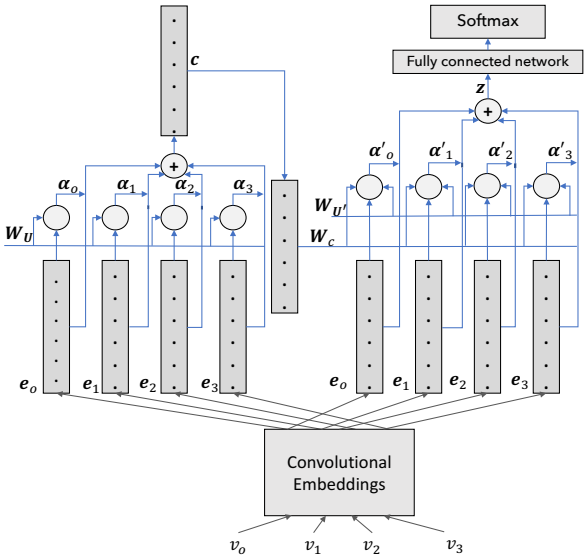

Figure 1: Attentional CNN model architecture

The gating mechanisms shown in the left module of Fig. 1 are akin to the hierarchical model in (Yang et al., 2016). However, their model ends at that module, at which point it is connected to the softmax output layer. In our model, the left module acts as a *context* encoder and the right module acts as an *attention* mechanism that is dependent on the encoded context and input.

Generally speaking, the local convolutional operations are unaware of the existence of preceding or succeeding convolutions over the text sequence. The context module enables propagation of stronger shared parameters through a *context embedding*, leading to the higher weighting of attention over specific parts of the inputs. The propagation strength in the network builds up based on the pattern of context present in the input sequences across several training examples and the training loss incurred for the encoding.

The filters of the convolutional layer (LeCun and Bengio, 1995; Kim, 2014) are convolved with with a window of consecutive observations (characters or words), and produce an encoding of the window. In Fig. 1, $\mathbf{e}_i$ is the encoding of i^{th} window, where, each window is defined over a word or character in the input sequence. For our ACNN model, the input sequence is treated

as a sequence of words and then as a separate sequence of characters with the two distinct sequences being concatenated as a single input sequence, $\{v_0, v_1, ...v_L\}$. The value of L is three in Fig. 1. Each variable, $\mathbf{u}_i$, encodes the non-linearity of the linear manifold on $\mathbf{e}_i$ over a set of shared parameters, $\mathbf{W_U}$, and biases, $\mathbf{b_U}$, as $\mathbf{u}_i = \tanh\left(\mathbf{W}_\mathbf{U}^T\mathbf{e}_i + \mathbf{b_U}\right)$. The variables $\mathbf{e}_i$ actually correspond to *parts of the data* and $\mathbf{u}_i$ help aggregate the values of $\mathbf{e}_i$ projected along the learned directions in the parameter space for $(\mathbf{W_U}, \mathbf{b_U})$. Each value of $\mathbf{u}_i$ is computed independent of $\mathbf{u}_{j\neq i}$. The shared parameter $\mathbf{W_U}$ is a $D \times F$ matrix, where D is a hyper parameter chosen for the attention mechanism and F is the number of convolution filters, both chosen during cross-validation. The variables $\mathbf{e}_i$ and $\mathbf{b_U}$ are F dimensional vectors.

The inputs to the context encoding vector, represented by the variable $\mathbf{c}$, are local softmax functions of the form:

$$\alpha_i = \frac{\exp(\mathbf{u}_i^T\mathbf{w}_u)}{\sum_j \exp(\mathbf{u}_j^T\mathbf{w}_u)} \quad (1)$$

The encoded *context vector* $\mathbf{c}$ is then simply $\mathbf{c} = \sum_i \alpha_i\mathbf{e}_i$. Obtaining the *input encoding* for the attention module is similar to context encoding except that $\mathbf{u}'_i$ depends on a separate set of shared parameters, $\mathbf{W_{U'}}$ as well as $\mathbf{W}_c$, for the context and corresponding bias term $\mathbf{b}$. In this case, we have:

$$\mathbf{u}'_i = \tanh\left(\mathbf{W}_{\mathbf{U}'}^T\mathbf{e}_i + \mathbf{W}_c^T\mathbf{c} + \mathbf{b}\right) \quad (2)$$

The softmax functions α'_i are similarly defined as in Equ. 1, but w.r.t. $\mathbf{u}'_i$ and $\mathbf{w}_{u'}$. The α'_is can be thought as the maximum of the *relevance* of the variables $\mathbf{u}'_i$, according to the context $\mathbf{c}$. The output, $\mathbf{z}$, from the attention module is the weighted arithmetic mean, $\sum_i \alpha'_i\mathbf{e}_i$, where the weight represent the relevance for each input variable v_i, through $\mathbf{e}_i$, according to the context $\mathbf{c}$.

We use windows over both word and character observation embeddings of the input text (either tokens or single Japanese characters). We concatenate the word and character encoding vectors and input it to a fully connected layer. A cross-entropy loss is imposed at the output layer.

4.2 Gradient Boosted Trees

GBTs (Friedman, 2000) optimize a loss functional: $\mathcal{L} = E_y[L(y, F(\mathbf{x})|\mathbf{X})]$ where $F(\mathbf{x})$ can be a mathematically difficult to characterize function, e.g., a decision tree $f(\mathbf{x})$. The optimal value of the function is expressed as $F^\star(\mathbf{x}) = \sum_{m=0}^{M} f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$, where $f_0(\mathbf{x}, \mathbf{a}, \mathbf{w})$ is the initial guess and $\{f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})\}_{m=1}^{M}$ are *additive boosts* on $\mathbf{x}$ defined by the optimization method. The parameter $\mathbf{a}_m$ of $f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$ denotes split points of predictor variables and $\mathbf{w}_m$ denotes the boosting weights on the leaf nodes of the decision trees corresponding to the partitioned training set $\mathbf{X}_j$ for region j.

Each boosting round m updates the weights $\mathbf{w}_{m,j}$ on the leaves and creates a new tree. The optimal selection of decision tree parameters is based on optimizing the $f_m(\mathbf{x}, \mathbf{a}, \mathbf{w})$ using a logistic loss.

5 Experimental Setup and Results

5.1 Data Preprocessing

Tokenization of Japanese product titles is done using MeCab[2]. The tokenizer is trained using features that are augmented with in-house product keyword dictionaries. Romaji words written using Latin characters are separated from Kanji and Kana words. All brackets are normalized to square brackets and punctuations from non-numeric tokens are removed. We remove anything outside of standard Japanese UTF-8 character ranges. Finally, canonical normalization changes the code points of the resulting Japanese text into an NFKC normalized[3] form.

For GBT, we use several features – at the tokenized word level, we use counts of word unigrams and word bi-grams. For character features, the product title is first normalized as discussed above. Character 2, 3, and 4-grams are then extracted with their counts, where extractions include single spaces appearing at the end of word boundaries. Feature engineering for GBT uses cross-validation to identify the best set of feature combinations and is thus *time consuming*.

The embedding representation of words and characters for the CNN-based classifiers is performed over the normalized input on which feature extraction for GBT is done. To reduce GPU memory consumption, the CNN-based models are trained on titles from which words and characters that appear in less than 20 titles in the training set are removed. Such rare token removal is not performed on the training data for the GBT models since they are trained on CPU servers.

[2] https://sourceforge.net/projects/mecab/

[3] http://unicode.org/reports/tr15/

5.2 Classifier Comparison

In this section, we compare categorization performance of a baseline CNN model w.r.t. our proposed model and a state-of-the-art GBT classifier. We use 10-fold cross-validation over 90% of the training data to perform parameter tuning.

ACNN model parameter setup - The words and characters are in an embedding vector space of dimension 300. These embeddings are trained on the product title training corpus. We use four different window sizes $1, 3, 4, 5$ for words and another of size 4 for characters. The dimension of the filter encoders $\mathbf{e}_i$ and $\mathbf{e}'_i$ is 250, which is the same as the number of filters. The hidden layer size is the number of window sizes times the number of filters i.e., $1,250$ and we also use a dropout with 0.5 probability on the hidden layer. The CNN models are run for a *maximum of three days* on a server with 8 Nvidia TitanX GPUs and the best model corresponding to the iteration for the lowest validation error is used for test set evaluation.

GBT model parameter setup - For each category, the boosted stumps for the GBT (Chen and Guestrin, 2016) models are allowed to grow up to a maximum depth of 500. The initial learning rate is assigned a value of 0.05 and the number of boosting rounds is set to 50. For leaf node weights, we use L_2 regularization with a regularization constant of 0.5. The GBT models are trained on a 64-core CPU server.

Models	Micro-F1	Training Time
GBT	96.23	3 weeks
CNN	95.90	3 **days**
ACNN-word	96.00	3 **days**
ACNN-word-character	**96.27**	3 **days**

Table 2: Micro-F1 measures for evaluated models. The Micro-precision scores (not shown here) are very similar to the micro-F1 scores with occasional differences in the third and fourth decimal places.

Table 2 shows that our proposed ACNN model – the CNN model augmented with word and character based attention mechanisms, improves over the baseline CNN model by an absolute 0.37%, which translates to more than $8,000$ test titles being correctly classified additionally. Although, the improvement of the proposed model is not significant when using a stringent p-value of 0.0001 (i.e., a typical value used in industrial setting), we emphasize that in practice any increase in accuracy helps (e.g., an additional million items when considering the whole Ichiba catalog).

Both GBT and our proposed ACNN model perform well for top level categorization of Japanese product titles. However, do the models make similar mistakes on the test set?

To this end, we computed the ratio of the sum of the number of listings in the test set per category for which both GBT and ACNN mis-classify but agree on the wrong predicted category, to the total number of mis-classifications from ACNN. The upper bound of this ratio is 1.0, which means that ACNN would make the same mistakes as GBT would. However, from our experiments, the ratio turned out to be 0.37, which means that GBT and ACNN make different mistakes more than 60% of the time. The relatively low value of the ratio indicates that we can gain major benefits for the final top level categorization by using an ensemble of GBT and ACNN models. The ACNN model does worse than GBT on 17 categories with a mean error difference, μ, of 0.78 and standard deviation, σ, of 1.15 and it does better than GBT on the rest of the 18 categories with $\mu = 0.39$ and $\sigma = 0.41$.

Statistics from test set	8000 **titles**	18 **categories**
Mean word count/title	20.930	20.120
Mean character count/word	09.245	05.815
Mean rare word count/title	00.158	00.354

Table 3: Word and character level statistics for: 1) The 8000 titles in the test set, for which ACNN predicts correctly over CNN (**Middle** column); and 2) The 18 categories in the test set for which ACNN performs better than GBT (**Rightmost** column).

Table 3 sheds some insights on why the ACNN model may be doing better over the CNN model, for the 8000 titles in the test set. We compare the average number of characters in the words of the 8000 titles in the test set for which our ACNN model provides correct predictions over the CNN model, to that for the overall test set from Table 1. The count for the former case turns out to be 9.245 that is substantially higher than that for the latter case, which is 5.799. It is thus highly likely that the ACNN model is performing better than CNN by leveraging the longer word and character contexts for these 8000 titles.

On the other hand, removal of the rare tokens (words appearing in less than 20 titles) seem to

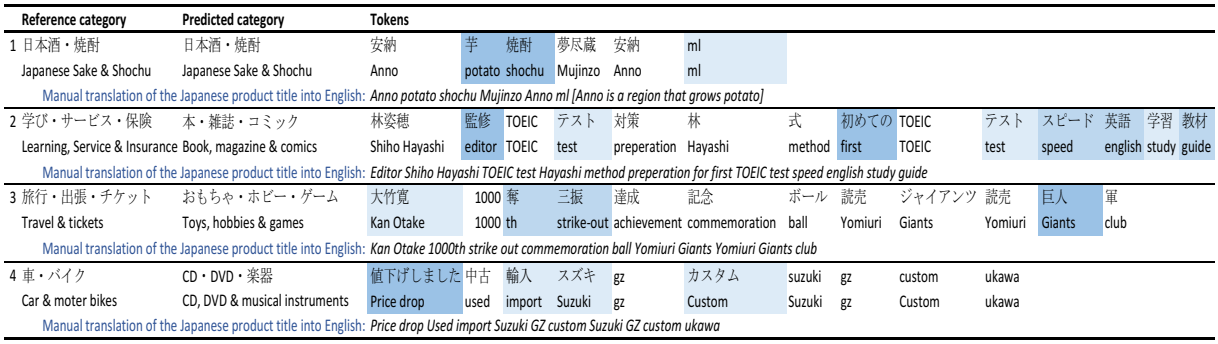

Figure 2: Examples of attention tokens for correct and incorrect classifications with English translations for tokens, product titles, and categories. Gradient colors are coded by attention model weights. Darker shades of blue have higher attention.

have negligible effect on the context of the titles from the subset of 8000 titles. However, the effect is a little more pronounced for the context of the titles from the subset of the 18 categories for which ACNN does better than GBT, but, with a mean error difference of only *half* of that for the other 17 categories on which it does worse.

5.3 Paying Attention Pays Off!

One of the most important aspects of the ACNN model is the ability to highlight words and characters in sequential text tokens automatically through the attention mechanism. Examples of such selected word tokens from test titles can be observed in Fig. 2.

In order to visualize the importance of the words related to the categorization label contribution, we use the attention vectors (e.g., α' scores) generated by the model. The word attention scores accurately localize words that are closely related to the classification labels. For instance, in Figure 2, line 1, the first word highlighted in the product description (higher score) is *potato*, which is one of the main ingredients in the Japanese alcoholic beverage, Shōchū (焼酎), that is referred to in the product title.

For the second example, there is ambiguity between the reference and the predicted category since the product title can be applied to both. In this case, the attention model is highlighting words like *editor*, *English*, and *guide* that may apply to both *Learning services* and *Books*.

The third example in Fig. 2 is an annotation mistake that was correctly captured by the model. Here the attention model is extracting the salient words *Giants*, *strike-out*, and *Kan Otake*, which are related to the predicted category.

Finally, in the fourth example, the attention mechanism assigns high scores to the words *price drop*, *import*, and *Suzuki* where *Suzuki* is a popular car manufacturer and music curriculum in Japan. "Suzuki" is thus inherently ambiguous and our model fails to put attention on context clues like the token "gz", which is a motorbike model.

6 Concluding Remarks

We propose a variant of the popular CNN model, the Attention CNN (ACNN) model, for the task of large-scale categorization of millions of Japanese product titles into thirty-five top level categories. The proposed model can leverage GPUs to **reduce training time** from three weeks for a state-of-the-art GBT classifier to three days while maintaining more than 96% accuracy.

Our language agnostic attention model can **highlight salient tokens**, which are semantically highly correlated to predicted categories. This helps in dimensionality reduction **without the need for feature engineering**.

As future work, we will experiment with ensemble methods to exploit differences in prediction errors from the different models, thereby improving overall classification performance.

References

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794.

Jianfu Chen and David Warren. 2013. Cost-sensitive learning for large-scale hierarchical classification. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 1351–1360.

Pradipto Das, Yandi Xia, Aaron Levine, Giuseppe Di Fabbrizio, and Ankur Datta. 2017. Web-scale language-independent cataloging of noisy product listings for e-commerce. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, April 3-7, 2017, Valencia, Spain*.

Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1329–1333.

Y. LeCun and Y. Bengio. 1995. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of the 53st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 174–179, Beijing, China, July. Association for Computational Linguistics.

Hyuna Pyo, Jung-Woo Ha, and Jeonghee Kim. 2016. Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA. ACM.

Dan Shen, Jean David Ruvini, Manas Somaiya, and Neel Sundaresan. 2011. Item categorization in the e-commerce domain. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1921–1924, New York, NY, USA. ACM.

Dan Shen, Jean-David Ruvini, Rajyashree Mukherjee, and Neel Sundaresan. 2012a. A study of smoothing algorithms for item categorization on e-commerce sites. *Neurocomput.*, 92:54–60, September.

Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012b. Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 595–604, New York, NY, USA. ACM.

Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. 2014. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proc. VLDB Endow.*, 7(13), August.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057. JMLR Workshop and Conference Proceedings.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.

Hsiang-Fu Yu, Chia-Hua Ho, Yu-Chin Juan, and Chih-Jen Lin. 2013. LibShortText: A Library for Short-text Classication and Analysis. Technical report, Department of Computer Science, National Taiwan University, Taipei 106, Taiwan.

Atlas: A Dataset and Benchmark for E-commerce Clothing Product Categorization

Venkatesh Umaashankar[1][0000−0001−5230−1209], Girish Shanmugam S[2][0000−0003−2805−7503], and Aditi Prakash[3][0000−0002−4839−9132]

[1] Ericsson Research, Chennai, India
venkatesh.u@ericsson.com
[2] Ericsson Research, Chennai, India
s.girishshanmugam@gmail.com
[3] University of Colorado, Boulder, Colorado, USA
adpr5166@colorado.edu

Abstract. In E-commerce, it is a common practice to organize the product catalog using product taxonomy. This enables the buyer to easily locate the item they are looking for and also to explore various items available under a category. Product taxonomy is a tree structure with 3 or more levels of depth and several leaf nodes. Product categorization is a large scale classification task that assigns a category path to a particular product. Research in this area is restricted by the unavailability of good real-world datasets and the variations in taxonomy due to the absence of a standard across the different e-commerce stores. In this paper, we introduce a high-quality product taxonomy dataset focusing on clothing products which contain 186,150 images under clothing category with 3 levels and 52 leaf nodes in the taxonomy. We explain the methodology used to collect and label this dataset. Further, we establish the benchmark by comparing image classification and Attention based Sequence models for predicting the category path. Our benchmark model reaches a micro f-score of 0.92 on the test set. The dataset, code and pre-trained models are publicly available at https://github.com/vumaasha/atlas. We invite the community to improve upon these baselines.

Keywords: Product Categorization · Attention · Seq to Seq models · Image Classification · Computer Vision

1 Introduction

With the Internet revolution, E-commerce has become a major platform for selling products to customers. E-commerce stores host a collection of products ranging from electronics to fashion apparel to grocery. A well-organized E-commerce store lets customers navigate through the website with ease and locate the product they are looking for. Unlike a traditional retail store where you can walk in and seek assistance, online retailers rely on their product catalog or categorization to assist shoppers to find their desired product. Product taxonomy is a tree structure with multiple top and intermediate levels, ending

in leaf nodes. Taxonomy classification is the process of assigning a category path to a particular product in the taxonomy tree. E-commerce sites use hierarchical taxonomies to organize products from generic to specific classes where each level provides more specific details about the product than the previous level. For example, ***Clothing & Accessories > Men > Winterwear > Sweatshirts & Hoodies***. These classification levels are important for an E-commerce store to perform operations such as search, catalog building, recommendation, which thereby hugely influence customer satisfaction and revenue of e-commerce sites. Currently, most of these product classification mechanisms rely on sellers to provide correct details. Each E-commerce store has its own product taxonomy and a seller typically sells in multiple stores. This implies that the seller has to perform such categorizations manually multiple times. Automating this has potential benefits of reduced costs and better catalog quality. Our key contributions in this paper are: **(1)** Developed a clean and rich clothing product taxonomy dataset containing 186,150 images and their corresponding product titles which maps to 52 category paths. **(2)** Proposed a methodology to collect large scale product taxonomy dataset which can be easily extended to categories other than Clothing. **(3)** Trained and compared two benchmark models (Image classification and Attention based Seq to Seq model) that predicts the category path from the product image. Our best model reached an f-score of 0.92 on the test set. **(4)** The dataset, source code and pre-trained models are made publicly available[4] to encourage future research in this area.

The rest of the paper is organized as follows. Related literature is reviewed in Section 2. In Section 3 we explain the methodology that we used to develop the ***Atlas*** dataset. In Section 4 we build our benchmark models and explain our model architecture. In Section 5 we provide our training setup and meta parameters to facilitate reproducible research. In Section 6 we summarize our results. Finally, in Section 7 we conclude and provide details about possible directions for future work.

2 Related Work

A clean and detailed product taxonomy offers several benefits to both the E-commerce store and its customers. However, creating, maintaining or adapting an existing categorization standard is not an easy task. Still, most of E-commerce stores want to have flexibility in the way they organize their catalog and create their product taxonomy.

Initially, techniques from information retrieval and machine learning were applied to solve the problem of product categorization. GoldenBullet [4] is a software environment targeted to automatically classify the products, based on their original descriptions and existent classification standards (such as UNSPSC). It integrates different classification algorithms like Vector space model (VSM), K-nearest neighbor and Naive-Bayes classifier algorithms and some natural language processing techniques to pre-process data. [5] approached product categorization as a hierarchical text classification task. They proposed two differ-

[4] https://github.com/vumaasha/Atlas

ent approaches of building separate classifiers for each level in the hierarchy and a flat classifier that directly predicts the leaf level assignment of a document. They used Support Vector Machine (SVM) classifiers for evaluating both the approaches. [11] presented a simple linear classifier based approach for product categorization using mutual information and LDA based features. In general, the computational complexity involved in some of these traditional machine learning techniques is well beyond linear with respect to the number of training examples, features, or classes. The scale of the E-commerce product categorization requires algorithms capable of processing a huge volume of training data in a reasonable time, capable of handling a large number of classes and also capable of making fast real-time predictions.[18].

The remarkable progress made in the field of deep learning in recent years has provided a better way to approach this problem. [3] has done a detailed study of using Convolutional Neural Networks (CNN) for the product categorization task. They used the Amazon product dataset provided by [17] and text features such as product titles, navigational breadcrumbs, and list price. [6] used multiple Deep Recurrent Neural Network (RNNs) and generated features from the text metadata. In recent times, Sequential model-based approaches have been widely used for product categorization. [8] modeled product categorization as a Sequence to Sequence learning, they used product titles which are a sequence of words as input and predicted the category path as a sequence of category levels in the product taxonomy.

Due to the availability of large high-quality image datasets, the field of Image classification [12] has matured a lot in recent times. Noise and ambiguity is a common problem in textual product titles and description. However, most of the E-commerce products tend to have decent product images, this leads to a natural choice of using images for product categorization. [1], [2], [10] and [16] applied computer vision techniques for fashion apparel categorization based on the product images. The closest to our work is by [13] where they use Seq to Seq model with product titles as input to predict category paths using an LSTM Decoder and beam search for inference. We extend their work in this paper by using product images instead of product titles as input for product categorization. Similar to [13], we learn an Attention based Seq to Seq model.

3 Atlas Dataset

Rakuten made a product classification dataset publicly available in Rakuten Data Challenge [15], However, this dataset contains only the product titles and the levels in the taxonomy are represented using numerical IDs instead of plain text. Real world product taxonomy datasets are not publicly available. Also, there is no widely adopted industry standard for defining product taxonomies. In addition to these, factors like data size, category skewness, and noisy metadata are limiting further research and practical implementation of large scale product categorization. This motivated us to develop a real-world dataset for product categorization.

We developed a new product categorization dataset called ***Atlas***. An E-commerce store typically sells products under several top-level categories such as Electronics, Home & Kitchen, Clothing, etc. In this paper, we focused only on clothing products, . Our dataset contains data corresponding to 52 products and their title, price, image and category path.

3.1 Taxonomy Generation

In the E-commerce world, each store has its own taxonomy. For example the category path for 'Jackets' in Flipkart[5] is ***Clothing >Men's Clothing >Winter & SeasonalWear >Jackets*** and in Amazon[6] it is ***Clothing & Accessories >Men >Jackets***. Treating them as different category paths will lead to noisy taxonomy and training data. We designed our taxonomy based on the similarities in the taxonomy structures across the different e-commerce retailer websites. Our taxonomy is organized to a maximum depth of 3 levels which can assist the consumer to reach their product in not more than 3 clicks. The process of building our taxonomy involved three steps. First, we analyzed and listed the taxonomy structures of popular products, niche, and premium clothing products across different e-commerce stores. Next, we identified the common category paths up to the third level across these websites. Finally, clothing that had the same category paths until the third level were clubbed together irrespective of the dissimilarity in the deeper levels. For example, ***Women >Ethnic Wear >Salwar Kameez >Bollywood*** and ***Women >Ethnic Wear >Salwar Kameez >Anarkali*** are grouped together under the category ***Women >Ethnic Wear >Salwar Kameez*** as they have similar category paths until the third level beyond which it branches into different nodes. Our final taxonomy tree is not an exhaustive list of all clothing categories but cover popular Western Wear and niche Ethnic Wear especially from the Indian Clothing collection. Each of the categories in the taxonomy tree have a maximum depth of 3 levels and totals to 52 category paths. Our taxonomy tree and a few sample products from some of the categories in our dataset can be found here [7].

3.2 Data Collection

We crawled the product listings from popular Indian E-commerce stores. We manually created a mapping of the store's category path that maps to a category path in our taxonomy. We used web scraping tools Scrapy and Selenium. The crawlers extract the information from the HTML content of the product page using CSS selectors. We extracted the *product title, breadcrumb, image and price* corresponding to each product in the product listings. The attributes extracted are stored in JSON format.

3.3 Data Cleaning

It is typical for an E-commerce store to show several images for a single product. Out of these images, not all the images are necessarily a good represen-

[5] https://www.flipkart.com/
[6] https://www.amazon.in/
[7] https://github.com/vumaasha/Atlas/tree/master/dataset#11-taxonomy-generation

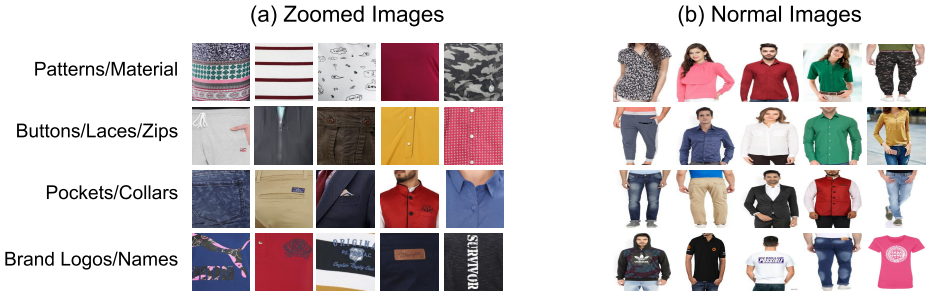

Fig. 1. Examples of (a) Zoomed(dirty) and (b) Normal(clean) images from our ***Atlas*** dataset. The Zoomed images show close-ups of the apparel or cropped versions of the image that make it difficult to recognize the product, whereas the Normal images show figures with the entire product visible.

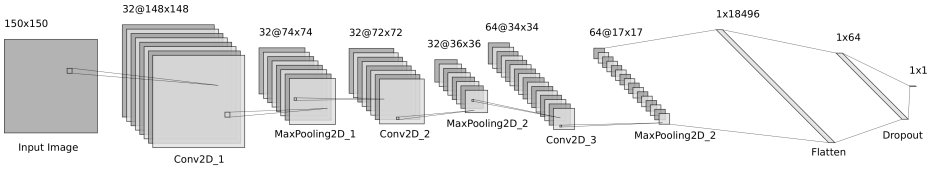

Fig. 2. Architecture of Zoomed Vs Normal Model

tative image of the product. Some images might display packaging, installation instructions, etc. In the case of clothing, we found that many product listings also included zoomed in images that display intrinsic details such as the texture of the fabric, brand labels, button, and pocket styles. Without the context of the product listing, it would be even hard for a human to identify the corresponding product. Including these zoomed in images would drastically affect the quality of the dataset. To find and remove these noisy images manually would take considerable time and effort. We modeled this as a binary classification task (Zoomed Vs Normal Images) and compared Linear SVM with simple 3 layer CNN (Figure. 2) based classification models. We prepared the training data by visual inspection. We segregated noisy and high-quality images into two different folders by looking at the thumbnails of hundreds of product images in a go. Our models were trained on 6005 normal images and 1054 zoomed images and the performance metrics on the test are shown in the Table. 1. We used computer vision based features such as contors and histogram of gradients as input for our LinearSVM. We automated the process of filtering out the noisy images using the CNN model due to its superior performance compared to that of LinearSVM.

4 Benchmark models for Product Categorization

4.1 Resnet34 based Image Classification

We use the cnn_learner available in fast.ai implementation to train our Image classification model. This uses Resnet34 architecture as the backbone of the

Table 1. Metrics for the models used to predict Zoomed Vs Normal images

	CNN			SVM		
	precision	**recall**	**f-score**	**precision**	**recall**	**f-score**
Normal	0.99	0.99	0.99	0.91	0.99	0.95
Zoomed	0.95	0.95	0.95	0.86	0.48	0.62
Average	0.98	0.98	0.98	0.91	0.91	0.90

model, which is followed by AdaptiveConcatPool2d, Flatten and 2 blocks of [nn.BatchNorm1d, nn.Dropout, nn.Linear, nn.ReLU] layers. The first block will have a number of inputs inferred from the backbone arch Resnet34 (512) and the second one will have a number of outputs equal to the number of classes (52) without nn.ReLU activation.

4.2 Attention based Seq to Seq Model

We approach the product categorization problem as a sequence prediction problem by leveraging the dependency between each level in the category path. We use Attention based Encoder-Decoder Neural Network architecture to generate sequences. The Encoder is a 101 layered Residual Network(ResNet) trained on the ImageNet classification task which converts the input image to a fixed size vector. The Decoder is a combination of Long Short-Term Memory(LSTM) along with Attention Network which combines the Encoder output and Attention weights to predict category paths as sequences. Our architecture is similar to works by [21] and [22] used for Neural Machine Translation and image captioning respectively. We extended the source code from pytorch tutorial to image captioning repository by Sagar Vinodababu[8]. Figure 3 shows some of the category paths generated by our model on test images which are not seen during training or validation. From this figure, it can be clearly seen that to generate each category level our model focuses on different parts of the image. To predict the first category level, which is the gender, our model has focused on the face and it has focused on the actual region of the clothing products to predict the next category levels, 'SareeBlouse' and 'Kurta'.

Encoder and Decoder In Encoder, we use Convolutional Neural Network (CNN) to produce fixed size vectors. The images in ***Atlas*** dataset have different dimensions as they were collected from different sources. All these images are resized to have a uniform dimension of 150*150 pixels before being fed as input to the Encoder. The input images are then represented by the 3 color channels of RGB values. The Encoder uses a 101 layered Residual Network pre-trained on the ImageNet classification task which is shown in Figure 4. As we use the Encoder only to encode images and not for classifying them, we remove the last two layers (linear and pooling layers) from the ResNet-101 model proposed by [7]. The images are resized to a fixed size by adding a 2D adaptive average pooling layer which enables the Encoder to accept images of variable sizes. The final

[8] https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning

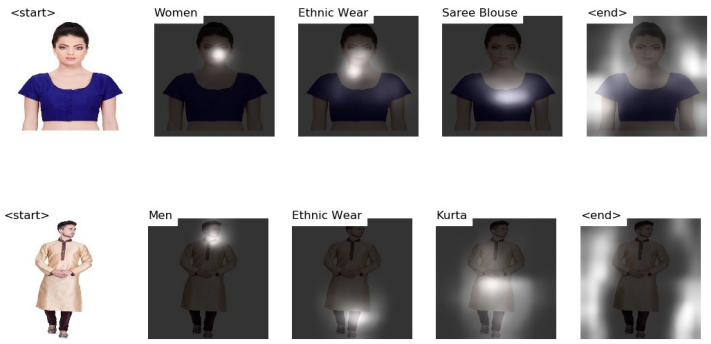

Fig. 3. A sample of category paths predicted on test dataset by our model. We can observe how the Attention focuses on different sections of the image while generating each category level. For example, the face is being focused to predict the first category level - gender.

encoding produced by the Encoder will have the dimensions: batch_size,14,14, 2048.

Recurrent Neural Networks(RNN) are popular for sequential classification task as it considers both the current input and the learnings from the previously received inputs for prediction. Usually, RNN's have short term memory but when combined with Long Short-Term Memory (LSTM) Network they have long term memory as LSTMs contain their information in a memory. We have a stacked LSTM Network along with Attention in our Decoder which is shown in Figure 4.

The Attention Network shown in Figure 4, learns which part of the image has to be focused to predict the next level in the category path while performing the sequence classification task. The Attention Network generates weights by considering the relevance between the encoded image and the previous hidden state or previous output of the Decoder. It consists of linear layers which transform the encoded image and the previous Decoder's output to the same size. These vectors are summed together and passed to another linear layer. This layer calculates the values to be Softmaxed and then passes the values to a ReLU layer. A final softmax layer calculates the weights *alphas* of the pixels which add up to 1. If there are P pixels in our encoded image, then at each time step t,

$$\sum_{p}^{P} \alpha_{p,t} = 1 \tag{1}$$

We use a weighted average across all the pixels instead of a simple average so that the important pixels are assigned greater weights.

The Decoder receives the encoded image from the Encoder using which it initializes the hidden and cell state of the LSTM model through two linear layers.

Two virtual category levels **<start>** and **<end>** which denote the beginning and end of the sequence are added to the category path. The Decoder LSTM uses teacher forcing proposed by [20] for training. The Decoder uses a **<start>** marker which is considered to be the zeroth category level. The **<start>** marker along with the encoded image is used to generate the first-top level of the category path. Subsequently, all other levels are predicted using the sequence generated so far along with the Attention weights. An **<end>** marker is used to mark the end of a category path. The Decoder stops decoding the sequence further as soon it generates the **<end>** marker. At each time step, the Decoder computes the weights and Attention weighted encoding from the Attention Network using its previous hidden state. Another linear layer is added to create a sigmoid-activated gate and the Attention weighted encodings are passed through it and concatenated with the embedding of the previously generated category path and fed into the LSTM Decoder to generate the new hidden state which is also the next predicted level. The next level is predicted using a final softmax layer from the hidden state of the Decoder. The softmax layer transforms the hidden state into scores which are stored for further utilization in beam search for selecting 'k' best levels.

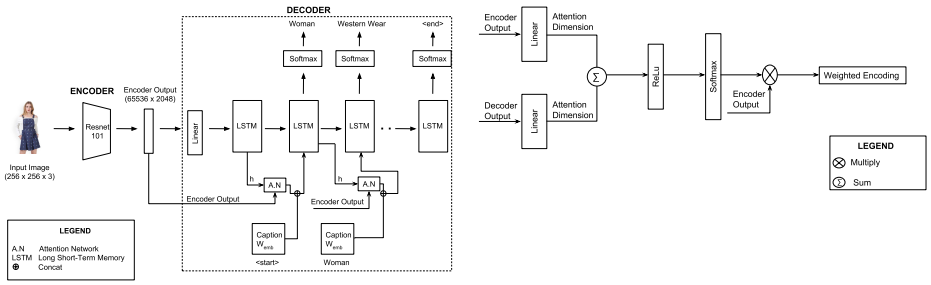

Fig. 4. Encoder - Decoder with Attention Network

5 Training

5.1 Model hyperparameters

Zoomed Vs Normal LinearSVM Model We trained LinearSVM available in Scikit-learn with C set to 0.0001, class weight set to 'balanced' using hinge loss. The optimal C value was identified using a grid search.

Zoomed Vs Normal CNN Model We trained for 10 epochs using Binary CrossEntropy as loss function, RMSProp Optimizer with a learning rate set to 0.001, rho set to 0.9 and decay set to 0.0.

Resnet34 based Image Classification We trained for 17 epochs using Categorical CrossEntropy as loss function and Leslie Smith's one cycle policy [19] for choosing the learning rate. We used early stopping to terminate the training

process when the decrease in validation loss is less than 0.001 for 3 consecutive epochs.

Attention based Seq to Seq Model We trained our model in GPU for 3 epochs with a batch size of 128 and dropout rate as 0.5 after which the validation accuracy stop improving. We used Adam optimizers with a learning rate of 1e-4 and 4e-4 for Encoder and Decoder respectively. We picked the beam width as 5 based on our experiments. Regularization parameter for doubly stochastic Attention was set to 1 and gradient clipping was set to an absolute value of 5. The pre-trained model can be downloaded from here[9].

5.2 Hardware

(1) Nvidia GPU GEFORCE GTX 1080 Ti 11GB RAM **(2)** Intel® Xeon® Processor E5-2650 v4 30M Cache, 2.20 GHz, 12 Cores, 24 Threads **(3)** 250 GB RAM **(4)** CentOS 7

6 Results

We evaluated the proposed model on our dataset having 186,150 clothing images and their category paths. We split our dataset into train, validation and test sets similar to the splits used in the work by [9]. Stratified random sampling was carried out on our dataset with training set having 65% of data(119,155 images), 5% in the validation set(11,147 images) and 30% in the test set(55,848 images). The Resnet34 classification model and the Seq to Seq model trained on our Atlas dataset achieved an overall micro f-score of 92% and 90% respectively. A comparison of the f-scores of both the benchmark models over support size of leaf categories is shown in Figure 5. Though we observe that the classification model's performance is better than Seq to Seq model, we believe the reason is that we have only 52 categories at the moment. As the number of categories increases, the structure in the taxonomy can be leveraged better using Seq to Seq model. In addition to Seq to Seq models predicting the category paths, it also explains the reason behind the predictions which is shown in Figure 3.

[14] claim that using Seq to Seq model for product categorization helps to identify new category paths in the taxonomy. However, in our experiments, we have observed that all the new category paths that are generated by the Seq to Seq model are not always valid. In our case our model generated 5 new category paths which are shown in Table 2 out of which we found only 2 to be valid. Therefore, a manual inspection of newly created category paths is needed to filter out the category paths which could be used to enrich the taxonomy.

Table 2. Valid and invalid category paths created by Seq to Seq model

Valid Category paths	**Invalid Category paths**
Women>Western Wear>Blazers&Suits	Men>Western Wear>Dresses
Women>Western Wear>Jackets	Men>Western Wear>Tanktops & Camisoles
	Women>Inner Wear>Shorts

[9] https://goo.gl/forms/C1824kjmbuVo7H6H3

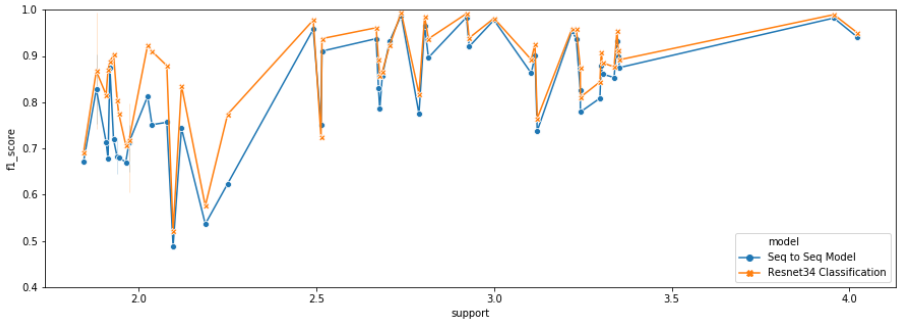

Fig. 5. F-scores of our benchmark models over leaf level categories ordered by their sample size. Note that the sample size in x axis is in log scale

7 Conclusion and Future Works

This paper introduces ***Atlas***, a fashion apparel dataset with 186,150 apparel images along with their corresponding product titles. We have open-sourced the code base and the procedure to build the dataset of images and their taxonomy. We have proposed two benchmark models using classification and Attention based Sequence approaches to predict product taxonomy.

In the future, we plan to extend our ***Atlas*** dataset by adding more categories and products thereby increasing the total number of category paths. We would avoid the generation of invalid category paths in our Seq to Seq model by considering the taxonomy structure while decoding and explore Transformer Networks instead of Recurrent Neural Networks (RNN).

8 Acknowledgements

The First Author Venkatesh Umaashankar worked extensively in the problem of Product Categorization using text attributes during his tenure at Indix. He thanks Krishna Sangeeth, Sriram Ramachandrasekaran, Anirudh Venkataraman, Manoj Mahalingam, Rajesh Muppalla and Sridhar Venkatesh for their help and support.

Bibliography

[1] Bossard, L., Dantone, M., Leistner, C., Wengert, C., Quack, T., Van Gool, L.: Apparel classification with style. In: Asian conference on computer vision. pp. 321–335. Springer (2012)

[2] Chen, H., Gallagher, A., Girod, B.: Describing clothing by semantic attributes. In: European conference on computer vision. pp. 609–623. Springer (2012)

[3] Das, P., Xia, Y., Levine, A., Di Fabbrizio, G., Datta, A.: Web-scale language-independent cataloging of noisy product listings for e-commerce. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. vol. 1, pp. 969–979 (2017)

[4] Ding, Y., Korotkiy, M., Omelayenko, B., Kartseva, V., Zykov, V., Klein, M., Schulten, E., Fensel, D.: Goldenbullet: Automated classification of product data in e-commerce. In: Proceedings of the 5th international conference on business information systems (2002)

[5] Dumais, S., Chen, H.: Hierarchical classification of web content. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 256–263. ACM (2000)

[6] Ha, J.W., Pyo, H., Kim, J.: Large-scale item categorization in e-commerce using multiple recurrent neural networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 107–115. ACM (2016)

[7] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[8] Hiramatsu, M., Wakabayashi, K.: Encoder-decoder neural networks for taxonomy classification. In: eCOM@SIGIR. CEUR Workshop Proceedings, vol. 2319. CEUR-WS.org (2018)

[9] Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3128–3137 (2015)

[10] Kiapour, M.H., Yamaguchi, K., Berg, A.C., Berg, T.L.: Hipster wars: Discovering elements of fashion styles. In: European conference on computer vision. pp. 472–488. Springer (2014)

[11] Kozareva, Z.: Everyone likes shopping! multi-class product categorization for e-commerce. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1329–1333 (2015)

[12] 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)

[13] Li, M.Y., Kok, S., Kok, S.: Unconstrained product categorization with sequence-to-sequence models. In: eCOM@SIGIR. CEUR Workshop Proceedings, vol. 2319. CEUR-WS.org (2018)
[14] Li, M.Y., Kok, S., Tan, L.: Don't classify, translate: Multi-level e-commerce product categorization via machine translation. arXiv preprint arXiv:1812.05774 (2018)
[15] Lin, Y., Das, P., Datta, A.: Overview of the SIGIR 2018 ecom rakuten data challenge. In: eCOM@SIGIR. CEUR Workshop Proceedings, vol. 2319. CEUR-WS.org (2018)
[16] Liu, Z., Luo, P., Qiu, S., Wang, X., Tang, X.: Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1096–1104 (2016)
[17] 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 on Research and Development in Information Retrieval. pp. 43–52. ACM (2015)
[18] Shen, D., Ruvini, J.D., Sarwar, B.: Large-scale item categorization for e-commerce. In: Proceedings of the 21st ACM international conference on Information and knowledge management. pp. 595–604. ACM (2012)
[19] Smith, L.N.: Cyclical learning rates for training neural networks. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 464–472. IEEE (2017)
[20] Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural computation **1**(2), 270–280 (1989)
[21] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
[22] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: International conference on machine learning. pp. 2048–2057 (2015)

Large Scale Product Categorization using Structured and Unstructured Attributes

Abhinandan Krishnan
akrishnan@walmartlabs.com
WalmartLabs
USA

Abilash Amarthaluri
aamarth@walmartlabs.com
WalmartLabs
USA

ABSTRACT

Product categorization using text data for eCommerce is a very challenging extreme classification problem with several thousands of classes and several millions of products to classify. Even though multi-class text classification is a well studied problem both in academia and industry, most approaches either deal with treating product content as a single pile of text, or only consider a few product attributes for modelling purposes. Given the variety of products sold on popular eCommerce platforms, it is hard to consider all available product attributes as part of the modeling exercise, considering that products possess their own unique set of attributes based on category. In this paper, we compare hierarchical models to flat models and show that in specific cases, flat models perform better. We explore two Deep Learning based models that extract features from individual pieces of unstructured data from each product and then combine them to create a product signature. We also propose a novel idea of using structured attributes and their values together in an unstructured fashion along with convolutional filters such that the ordering of the attributes and the differing attributes by product categories no longer becomes a modelling challenge. This approach is also more robust to the presence of faulty product attribute names and values and can elegantly generalize to use both closed list and open list attributes.

KEYWORDS

neural networks, text classification, extreme classification, eCommerce, structured datasets

ACM Reference Format:
Abhinandan Krishnan and Abilash Amarthaluri. 2019. Large Scale Product Categorization using Structured and Unstructured Attributes. In *KDD '19: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, August 04–08, 2019, Anchorage, Alaska.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Categorizing products into a hierarchical taxonomy has become a central part of the organizational efforts of eCommerce companies. It is a critical step that acts as a precursor to multiple downstream systems like search, facets etc. In the eCommerce context, it is posed as an extreme multi-class classification problem and is relatively well studied in both academia and industry.

There are several challenges that are unique to product categorization in the eCommerce domain that are not observed in traditional extreme classification challenges like ImageNet. Product catalogs are in constant flux with old products being retired and new products being added on a daily basis. Due to the dynamic nature of the product catalog, the hierarchical taxonomy against which they are categorized is also in constant flux although not at the same rate as the items in the catalog. Hence, we need to classify products with acceptable performance against a non-stationary dataset where both the sample space and the label space change at the same time. This also means that both the training and validation sets need to keep changing to reflect the latest snapshot of the distribution of the catalog. Acquiring labeled data for this changing catalog is also an extremely expensive process and hence intelligent sampling strategies need to be employed to reuse as many previously labeled examples as possible. With marketplace platforms such as Amazon, Walmart, eBay etc., there are new sellers and vendors being added everyday which results in a wide distribution of data quality levels for the products being setup. While most products have some common attributes like title, description, image etc., every product also has a unique set of structured attributes describing it depending on the category the product belongs to. The total set of unique attributes in the catalog is in the tens of thousands (N) while an individual product might only possess a few attributes (k«N) that are relevant to it. In addition, the quality of product attributes widely varies by seller. Each product attribute also has its own value space that presents a modelling challenge. All these added complexities make product categorization an extremely challenging problem to tackle.

Next, in section 2, we briefly review related work in extreme multi-class classification and product categorization in particular. Section 3 outlines our preliminary approach to classification using hierarchical multi-class models, our move to a flat classification scheme using two different flavors of Deep Learning models, a baseline architecture for structured attributes using word averaging and finally an innovative way to use all available structured attributes in a product in an unstructured format along with convolutional filters. In section 4, the experimental setup is described, specifically the details about the dataset, preprocessing, training schedule, dictionaries, embeddings, models and their deployment. Finally in section 5, we provide some results comparing the different approaches we have experimented with.

KDD '19, August 04–08, 2019, Anchorage, Alaska

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Many websites, especially in eCommerce, organize their product catalog into concept hierarchies or taxonomies. Some common examples are the Wordnet hierarchy, Google's Product Taxonomy, the Open Directory Project (ODP) etc. Extreme multi-class classification against such taxonomies has been worked on for a relatively long time and several approaches have been explored over the years. The two most common approaches that have been adopted are flat single-step classifiers and hierarchical multi-step classifiers.

Yu et. al. [2] explored several word level features in conjunction with linear classifiers like SVMs for classification. Kozareva [7] used several word features (n-grams, LDA, Word2Vec embeddings etc.) followed by linear classifiers. Ha et. al. [5] proposed multiple LSTM blocks, one for each unstructured or structured attribute followed by fully-connected layers for classification. Xia et. al. [12] proposed a variation of CNNs called Attention CNNs applied on Japanese product titles for classification.

Weigend et. al. [1] used a hierarchical classification scheme with one meta-classifier to determine the broad topic followed by individual topic level classifiers to distinguish nuances within each topic. They also observed that using neural networks in place of standard logistic regression resulted in improved performance. Shen et. al. [9] reformulated this into a two level classification problem ignoring the prior hierarchy and distributing the leaf nodes fairly evenly across top level categories. They used a kNN classifier at the top level followed by individual SVM classifiers for each second level node.

Yundi Li et. al. [13] used a Machine Translation approach to generate a root-to-leaf path in a product taxonomy. Gupta et. al. [4] trained path-wise, node-wise and depth-wise models (with respect to the taxonomy tree) and trained an ensemble model based on the outputs of these models. Zahavy et. al. [14] use decision level fusion networks for multi-modal classification using text and image inputs.

Flat classification models have more parameters per model but perform inference only once per product and hence have a lower latency. We can also batch products together to improve throughput. Hierarchical models on the other hand need to deal with products individually since each product may trace a different path in the taxonomy tree.

All of the above approaches use unstructured product attributes like title, description etc. to perform classification. Ha et. al. [5] used a limited set of structured attributes but used an independent LSTM block for each structured attribute which does not scale when each category of products contains different sets of attributes and there are thousands of product attributes overall that need to be considered.

3 OUR APPROACH

At Walmart, we experimented with multiple approaches to tackle this problem. Before delving deeper into all our experiments, we will present an overview of the information that a Walmart product contains. These are the attributes that will feed into the Machine Learning models.

3.1 What does a product contain?

A product is any commodity that may be sold by a seller. Within our catalog, every product contains a mix of unstructured and structured attributes which describe the product. Examples of unstructured attributes include ***product name***, ***product short description***, ***product long description***, ***shelf description***, ***synopsis*** etc. Examples of structured attributes include ***screen size***, ***color***, ***gender***, ***fabric material***, ***hard drive capacity*** etc. Structured attributes may have a closed list or an open list of values. For example, ***gender*** has a closed list of values while ***brand*** is an open list. Every product may also have multiple product images associated with it. These structured and unstructured attributes along with product images can all potentially be used to categorize the product.

3.2 Hierarchical Classification Approach

We first experimented with a hierarchical classification scheme similar to Weigend et. al. [1] using a bag-of-words hash feature on titles and descriptions followed by an entropy maximization based multinomial logistic regression classifier at each node in the taxonomy tree. The training data would vary based on the node the classifier was being trained for but the extracted features and classification scheme remained the same. This approach had several advantages and disadvantages.

- Advantages:
  - This architecture lent itself to parallelization at training time since each of the individual models could be trained in parallel.
  - Considering the rate at which the product catalog changes, we could focus specifically on retraining the parts of the model that needed attention rather than retrain the entire model at every iteration.
  - We could also target specific models for improvement without adversely affecting the performance of the rest of the models in the hierarchy.
- Disadvantages:
  - Top-1 predictions were very fast but top-k predictions were extremely slow since multiple paths (potentially all paths) along the hierarchy needed to be explored before returning the top-k predictions.
  - Large fraction of errors were made at the root level and the performance of the overall model was bounded above by the performance of the model at the root.

This hierarchical model is easy to deploy and could run efficiently without requiring GPUs. However, the taxonomy hierarchy itself had a significant impact on the performance of the model at the root. For example, product types ***Athletic Shoes*** and ***Dance Shoes*** appear under the category ***Sports & Outdoor*** while product types ***Casual & Dress Shoes*** and ***Safety Shoes & Boots*** appear under the category ***Clothing, Shoes & Accessories*** even though all 4 product types may be close to each other with respect to product content. Such confusions make it harder for the root node classifier to distinguish between categories at a high level for several types of products. After conducting an analysis, we found that this indeed contributes the largest fraction to the drop in accuracy.

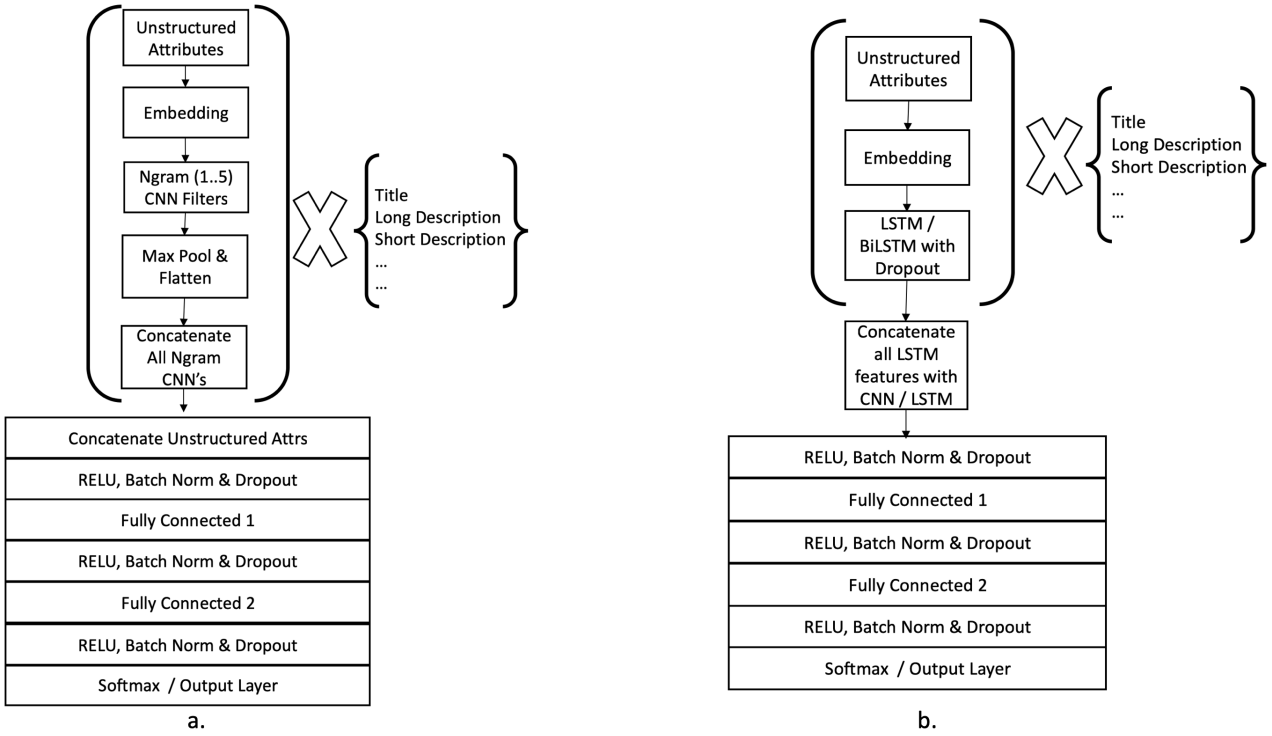

Figure 1: Model architectures used for unstructured attributes. Figure a. shows the Multi-CNN architecture and Figure b. shows the Multi-LSTM architecture.

3.3 Flat Classification Approach

Recently, we explored a single-step flat categorization approach using Deep Learning based models. These models seemed to benefit from the large amount of labeled data we had acquired and also demonstrated robustness to random label noise compared to the hierarchical model. In this paper, we describe a Multi-LSTM and a Multi-CNN based approach to perform product categorization. We also present a novel way to use structured attributes that can scale to any number of product attributes and any cardinality in the value space for a particular attribute. This method of using the structured attributes can be added on to any baseline model architecture and should provide a significant boost in classification performance. We observed that it achieves very similar improvements over both the Multi-LSTM and Multi-CNN architectures.

3.3.1 *Generating Word Dictionaries*.

For each unstructured attribute, like ***product name*** or ***product short description***, we go through the entire catalog and gather data for the attribute from every available product to build a word dictionary. We then trim this dictionary to retain a subset of words. The number of words we retain for each attribute depends on the original size of the dictionary and the variety of words found for that particular attribute. We use word embeddings with 200 dimensions for each attribute which are initialized randomly and updated over the course of training. We also experimented with Word2Vec and Glove embeddings with and without updates during the course of training. However, using these embeddings degraded performance marginally and we also observed that the dictionaries associated with these pretrained embeddings did not capture a lot of very important tokens in our catalog. We also experimented with embeddings of size as low as 50 and found that these lower dimensional embeddings had comparable performance to embeddings with 200 dimensions.

3.3.2 *Multi-CNN Model*.

In this approach, we use an independent set of convolutional filters of multiple lengths (1,2,3,4,5) on the word embeddings for each unstructured attribute, as shown in Figure 1a. 128 filters of each length are used and each filter is of size $n*200$ where n represents the filter size. This approach is similar in motivation to Kim [6]. The multiple length convolutional filters in essence capture n-gram features from pieces of text. The activations generated by each set of convolutional filters are passed through a max pooling layer along the last dimension to retain one activation per filter. These activations for each attribute are then concatenated and passed through multiple fully connected layers followed by a softmax layer at the end for classification. The performance of this approach was very similar to that of the Bidirectional Multi-LSTM model and led to an improvement of almost 20% over the hierarchical model.

3.3.3 *Bidirectional Multi-LSTM Model*.

In this approach, we use one Bidirectional LSTM layer on the embeddings for each unstructured attribute, as shown in Figure 1b. The

Table 1: Example product

Attribute Name	Attribute Value
product_name	Rails Womens Plaid Spread Collar Button Down Top
product_short_description	This Rails Button Down Top is guaranteed authentic. It's crafted with 100% Rayon.
assembled_product_weight	0.5 Pounds
color	White
clothing_size_type	Regular
clothing_size_group	Women
maternity	N
age_demographic	Women
brand	Rails
fabric_material	Jersey
clothing_size	S
style_sleeve	Long Sleeves
actual_color	White Navy Sky
country_of_origin_assembly	CN
personalizable	N

activations generated by each LSTM layer can either be concatenated directly or put through another bidirectional LSTM following which we add multiple fully-connected layers and softmax at the end for classification. This approach is similar to the one used by Ha et. al. [5], however, we found that using a second level LSTM that takes in the activations of the first level of LSTMs as time steps performs slightly better than just concatenating the activations and using fully connected layers on top. Using this approach led to an improvement of almost 20% over the hierarchical model.

3.4 Using Structured Attributes

Every product has a small set of structured attributes which varies based on the category. Some attributes are common across categories while others are specific to a particular category. For example, ***assembled_product_width*** is an attribute that is relevant to both ***Cell Phones*** and ***End Tables*** even though these product types appear under completely different categories. In this case, the value of the attribute may have some useful information regarding the category or product type to which the product could potentially belong. On the other hand, an attribute like ***diaper_size*** is only relevant to diaper related product types. In such cases, just the presence of the attribute is a strong indicator about the product type.

3.4.1 *Traditional Use of Structured Attributes.*

Guo and Berkhahn [3] proposed Entity Embeddings for Categorical Variables for the Rossmann Store Sales Kaggle competition. This approach helps avoid feature sparsity and captures semantic relationships between the entities in a euclidean space. This is the latest state of the art method using structured attributes. However, one of the limitations of this method is that separate entity embeddings are needed for each categorical variable which quickly becomes unmanageable when the number of attributes grows to a few thousand. In addition, lots of attributes whose value spaces are open list are not necessarily categorical and hence may need other representations.

3.4.2 *Building Attribute Word Dictionaries.*

We build a common word dictionary across all attribute names and values unlike our approach with unstructured attributes where we build a separate dictionary for each attribute. This common dictionary is then trimmed down based on the total number of words in it. It is recommended to explicitly ensure that all the tokens present in the complete set of product attribute names are present in the dictionary after it has been trimmed down. This is accomplished by building a separate dictionary for tokens in the attribute names alone and then merging it with the joint dictionary for attribute names and values. This enables the model to extract richer features from these attribute name value pairs.

3.4.3 *Combining Structured Attributes.*

Once the dictionary is built, we combine all the structured attributes associated with the product in an unstructured fashion. We use both attribute names and values to achieve the best performance. We observed that using just the attribute names improves performance marginally but using both names and values together provides the best results. Attribute names and values are broken down into natural language by stripping out underscores, dashes and other such special delimiters and then combined in the order below.

<attr_name> <attr_value> <separator> <attr_name> <attr_value> <separator> ... <attr_name> <attr_value>

This is very similar in format to other unstructured attributes like ***product_name*** or ***product_long_description***. The custom separator token is added so that the convolutional filters have a marker depicting the end of one attribute and the beginning of the next attribute. In our experiments, we found that using the separator token improves classification accuracy over not using it. An example product is shown in Table 1. The combined structured attribute set for this product is shown below:

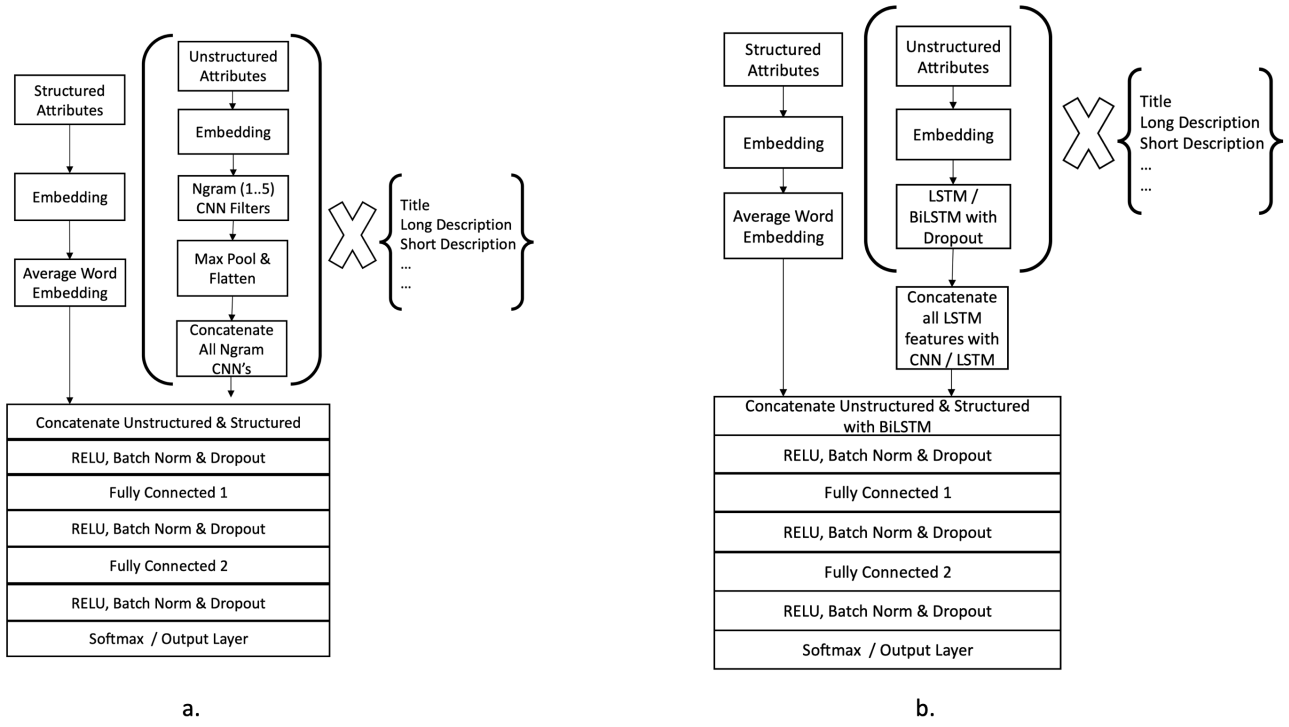

Figure 2: Model architectures used for unstructured and structured attributes using averaged word embeddings . Figure a. shows the Multi-CNN architecture and Figure b. shows the Multi-LSTM architecture.

assembled product weight 0.5 Pounds __sep__ color White __sep__ clothing size type Regular __sep__ maternity N __sep__ clothing size group Women __sep__ age demographic Women __sep__ brand Rails __sep__ fabric material Jersey __sep__ clothing size S __sep__ style sleeve Long Sleeves __sep__ actual color White Navy Sky __sep__ country of origin assembly CN __sep__ personalizable N.

3.4.4 ***Structured Attribute Features using Word Averaging.*** Wieting et. al. [11] showed that a simple word averaging method performs well on sentiment classification. We use the same method as a feature extractor and create a joint embedding for all structured attribute names and values in a given product. The joint embedding is simply the average of the word embeddings of the tokens found in the structured attribute names and values. This feature is then concatenated with the features extracted by the initial layers of the Multi-LSTM or the Multi-CNN models from the unstructured attributes. This is considered the baseline approach to incorporate structured attributes in our classification model. We observe that even this simple averaging of embeddings is an extremely useful feature and results in a lift in classification accuracy. The model architecture for this approach is shown in Figure 2.

3.4.5 ***Structured Attribute Features using Convolutional Filters.***
We use the same model architecture as the Multi-CNN model mentioned above to extract features from the combined structured attribute string. These features are concatenated with the features extracted by the Multi-LSTM or the Multi-CNN model. We use convolutional filters of multiple lengths (1,2,3,4,5) that capture features from the pairs of attribute names and values. There are 128 filters for each filter length. We originally experimented with one channel of convolutional filters for attribute names alone and another channel of convolutional filters for both attribute names and values. However, this did not improve performance over just using one channel for both attribute names and values. This is likely because the filters with smaller lengths typically capture features from the attribute names while the filters with larger lengths capture features from the attribute names and values together. Thus both types of features are being captured by the same channel itself. Figure 3 shows the updated model architectures with the block for structured attributes added.

3.4.6 ***Advantages of Using Structured Attributes in an Unstructured Format.***
There are several advantages to using structured attributes in an unstructured format along with convolutional filters as described above.

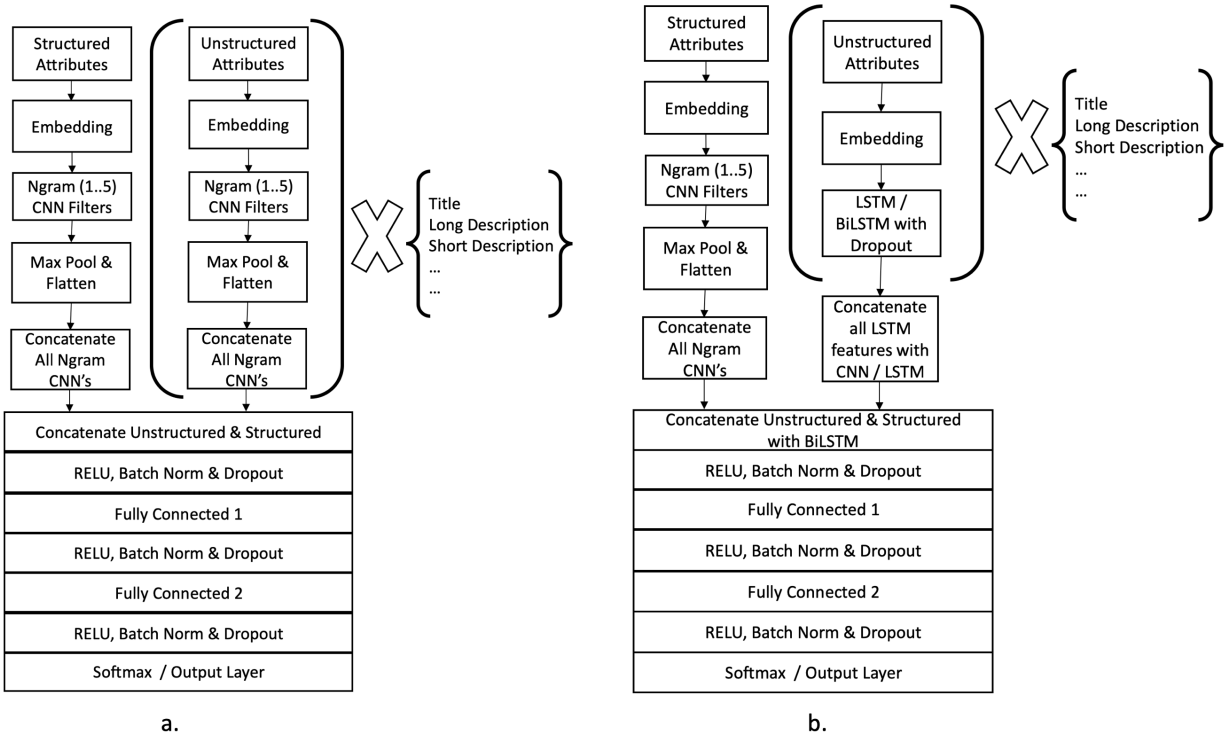

Figure 3: Model architectures used for unstructured and structured attributes. Figure a. shows the Multi-CNN architecture and Figure b. shows the Multi-LSTM architecture.

- Typically structured attributes are used as hand engineered features that are directly concatenated to the final fully connected layers. Our approach removes the need to hand engineer features for each individual attribute since the convolutional filters are able to capture interesting features relevant to the classification problem.
- With the current state of the art approaches for using structured attributes as explained in the previous section, there is an inherent sparsity in the feature set. Also, the number of parameters required for the proposed method increases linearly with the number of attributes and the representation format of each additional attribute.
- Since we break down the attribute names and values into natural language tokens, this approach should generalize well to new attributes and new values added to existing closed-list or open-list attributes.
- By adding the features extracted from structured attributes, we can get better representations for products in general and these representations can be used for a variety of other tasks.
- Using word embeddings to represent the tokens present in these attributes also helps capture semantic relationships between multiple attributes or between attributes and their values. Common representations can also be learned for entities identified in these attribute values which can be shared with the other channels for unstructured attributes and potentially for other tasks.

However, if the number of structured attributes available is small and limited, traditional approaches may outperform this method.

4 EXPERIMENTAL SETUP

4.1 Data

Over the years, internal and external crowd sourcing has enabled us to collect large quantities of labeled data for product type categorization. Currently our dataset contains approximately 25 million labels, which is split into 3 subsets (80-10-10) to generate train, validation and test sets. These products represent approximately 6000 leaf product types in our taxonomy. We adopted a bootstrapping approach to collect the external crowdsourced data where we trained a model with the existing data and sent out suggestions to the crowd, who would provide us with feedback. While this approach has helped us collect massive quantities of data, it also has an inherent label bias since the crowd does not have an intimate knowledge of the Walmart taxonomy and is likely to pick a close-enough label if the right label is not presented in the suggestions provided. Our hierarchical model is especially sensitive to this kind of label noise since it uses traditional logistic regression based classifiers. The dataset is also very unevenly distributed for each product type. An example of this uneven distribution is shown in Figure 4. This is typical in an eCommerce catalog, where some product types, like

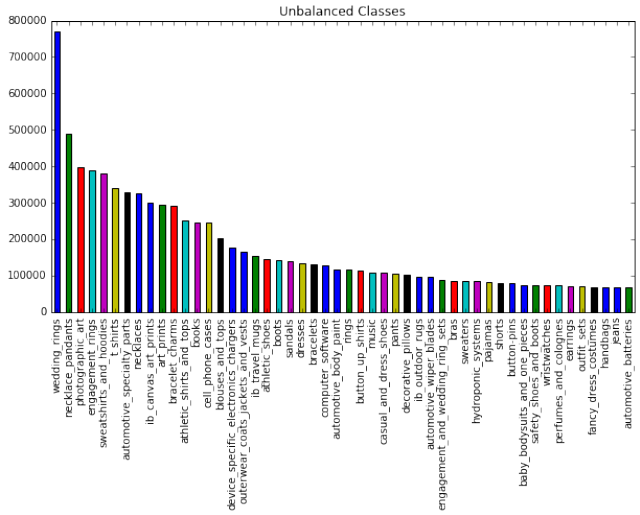

Figure 4: Unbalanced class distribution in the dataset

T-Shirts have several tens of millions of products while others like ***Zithers*** have less than ten products. We stratify low support classes by repeating samples until each class has at least 200 samples. We also ensure that we perform a stratified split into train, validation and test sets such that the sample distributions across these sets are similar.

4.2 Preprocessing

Standard whitespace tokenization is used to tokenize both unstructured and structured attributes. Tokenization and dictionary lookups are not precomputed but performed on the fly during training. Preprocessing is an expensive step however and does slow down training when done on the fly. In our case, since the product information keeps changing frequently, we typically don't store preprocessing results ahead of time before training.

4.3 Training Schedule

For both the Multi-LSTM and the Multi-CNN model, we adopt similar training procedures. We use SGD with restarts to train these models as described in Smith [10]. We start with one epoch and double the cosine annealing period with each cycle. The models with the lowest validation loss values are usually seen after around 5-6 epochs of training. Even though we also update the randomly initialized word embeddings at training time, the model converges to a loss value within 10% of the final loss value within one epoch after which it slowly improves and reaches its peak.

4.4 Dictionary and Embedding Details

As mentioned above, word embeddings with 200 dimensions were used for each unstructured attribute. The same embedding size was used for structured attributes too. The dictionary used for ***product_name*** had 500K tokens, for ***product_description*** had 1 million tokens and for ***structured attributes*** had 100K tokens.

4.5 Model Details and Training Time

The hierarchical model has approximately 1.5 billion parameters since it uses one model at each node in the taxonomy tree. The Multi-LSTM model has approximately 180 million trainable parameters while the Multi-CNN model has approximately 330 million trainable parameters. However, we have achieved similar results where both the models were compressed to around 65 million trainable parameters.

The hierarchical model is embarrassingly parallel, but uses CPUs to train and hence takes almost a day to train. The Multi-CNN model with structured attributes takes around 2.5 hours to complete one epoch while the Multi-LSTM model with structured attributes takes around 7 hours. Considering the training time, inference time and overall model performance on the test set, the Multi-CNN model wins over the other two model architectures.

4.6 Hardware and Software Details

- P100 GPUs used to train the CNN and LSTM models
- LSTM models were trained on Keras while the CNN models were trained on PyTorch
- Hierarchical Model trained using scikit-learn and parallelized using Spark

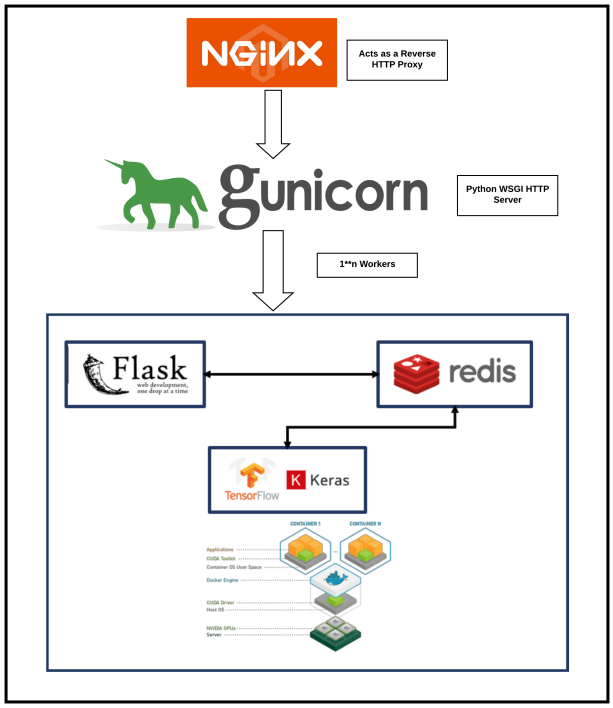

Figure 5: Deployment Architecture

4.7 Deployment

Our Machine Learning models are typically deployed as Micro Services to which other clients can send requests individually over HTTP. We have carefully designed our system with ideas borrowed

Table 2: Model evaluation results

Model architecture	Top-1 acc.	Top-2 acc.	Top-3 acc.
Hierarchical	70%	-	-
Multi-LSTM	89.9%	94.79%	96.42%
Multi-LSTM + Word Embedding Avg.	91.4%	95.95%	97.18%
Multi-LSTM + Struct. Attr.	92.28%	96.29%	97.38%
Multi-CNN	89.45%	94.93%	96.5%
Multi-CNN + Word Embedding Avg.	89.97%	95.31%	96.77%
Multi-CNN + Struct. Attr. w/o Separator	91.08%	95.87%	97.18%
Multi-CNN + Struct. Attr.	92.15%	96.36%	97.51%

from Rosebrock [8] to maximize overall throughput and minimize latency as shown in Figure 5. Since GPUs are suited for batch work loads, we microbatch multiple requests from different clients by adding a Redis buffering layer. These queued requests are sent as a minibatch to utilize the GPU effectively. A configurable poll interval (currently 0.3s) and a batch size of 1024 were chosen based on latency vs throughput trade-offs. In our load tests, we observed a throughtput of 750 rps on a single P100 GPU with 6 CPU Cores.

Table 3: Top-5 product types that benefit from using structured attributes with support >= 100 products

Product Type	Δ f1 Score
Tank Tops	0.365
Chemistry Experiment Kits	0.317
Office Boxes	0.257
Outdoor Flags & Banners	0.249
Shape Sorting Toys	0.241

5 RESULTS

We have observed that attaching the structured attributes block (word embeddings followed by convolutional filter activations) to either model architecture improves overall accuracy by approximately 2.7% on the evaluation set as shown in Table 2. Considering that we have 6000 classes to classify against, this is a very significant increase in accuracy. This also shows that regardless of the model architecture, using this block for structured attributes improves overall accuracy. The model architectures we use, Multi-LSTM or Multi-CNN also lend themselves to variable sized inputs and hence can handle arbitrarily large unstructured attribute values. Upon performing an analysis of the top product types (with a support of over 100 products) in the evaluation set that benefit from using structured attributes, we found that ***Tank Tops***, had the highest lift in f1 score. This is due to the presence of multiple attributes like ***sleeve_style*** and ***clothing_top_style*** whose values indicate that the product is a ***Tank Top***. Similarly, another product type that sees a significant lift in f1 score is ***Office Boxes***. This is again because of the presence of the attribute ***office_box_type*** whose presence itself indicates that the product is an ***Office Box***. From the above examples, we see that there are two useful features extracted from the structured attributes.

- The presence or absence of a particular attribute
- The value of a particular attribute

The convolutional filters are able to extract both these features efficiently from the structured text. Table 3 shows the top 5 product types ordered by the lift in f1 score using this approach.

6 CONCLUSIONS

Several insights, observations and conclusions were derived from this large-scale experiment some of which are mentioned below:

- **Training size**: For such extreme classification problems, we observe that having a large dataset with a good variety of samples to train significantly improves overall accuracy. Stratifying the dataset to improve representation for low support classes has also been shown to improve performance.
- **Choice of model**: In the presence of such large amounts of data, Deep Learning models significantly outperform traditional Machine Learning models. In this case, both Multi-LSTMs and Multi-CNNs perform equally well. However, the Multi-CNNs have the advantage of being more parallelizable and hence will be faster both at training and inference time.
- **Taxonomy structure**: Having the right taxonomy structure significantly improves the performance of classification models. Logical and mutually exclusive divisions of product types into categories helps improve classification performance significantly, especially using hierarchical models. Having more specific product types than very general ones is also recommended to aid better classification.
- **Word embeddings**: Using separate embedding spaces for each unstructured attribute seemed to perform better than using a common embedding space for all attributes. This is likely because the semantic relationships between words present in each attribute may vary.
- **Word embedding size**: Most of our experiments were performed with 200 dimensional word embeddings. However, embeddings as small as 50 dimensions also yielded similar results. Also, trimming the word dictionaries for each attribute seemed to act as an implicit regularizer and helped performance in some cases.
- **Using a separator**: While concatenating the structured attribute names and values together, it is beneficial to use a separator token between two attributes. This helps the model not to relate the values of one attribute with a different attribute name.

REFERENCES

[1] Erik D. Wiener Andreas S. Weigend and Jan O. Pederson. 1999. Exploiting Hierarchy in Text Categorization. *Information Retrieval* 1, 3 (October 1999), 193–216. https://doi.org/10.1023/A:1009983522080

[2] Hsiang fu Yu, Chia hua Ho, Prakash Arunachalam, Manas Somaiya, and Chih jen Lin. 2012. *Product title classification versus text classification.* Technical Report.

[3] Cheng Guo and Felix Berkhahn. 2016. Entity Embeddings of Categorical Variables. *arXiv e-prints*, Article arXiv:1604.06737 (April 2016), arXiv:1604.06737 pages. arXiv:cs.LG/1604.06737

[4] Vivek Gupta, Harish Karnick, Ashendra Bansal, and Pradhuman Jhala. 2016. Product Classification in E-Commerce using Distributional Semantics. *CoRR* abs/1606.06083 (2016). arXiv:1606.06083 http://arxiv.org/abs/1606.06083

[5] Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. 2016. Large-Scale Item Categorization in e-Commerce Using Multiple Recurrent Neural Networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 107–115. https://doi.org/10.1145/2939672.2939678

[6] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1746–1751. https://doi.org/10.3115/v1/D14-1181

[7] Zornitsa Kozareva. 2015. Everyone Likes Shopping! Multi-class Product Categorization for e-Commerce. In *HLT-NAACL*.

[8] Adrian Rosebrock. 2018. Deep learning in production with Keras, Redis, Flask, and Apache. https://www.pyimagesearch.com/2018/02/05/deep-learning-production-keras-redis-flask-apache/.

[9] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012. Large-scale item categorization for e-commerce. *ACM International Conference Proceeding Series*, 595–604. https://doi.org/10.1145/2396761.2396838

[10] Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017), 464–472.

[11] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards Universal Paraphrastic Sentence Embeddings. *arXiv e-prints*, Article arXiv:1511.08198 (Nov. 2015), arXiv:1511.08198 pages. arXiv:cs.CL/1511.08198

[12] Yandi Xia, Aaron Levine, Pradipto Das, Giuseppe Di Fabbrizio, Keiji Shinzato, and Ankur Datta. 2017. 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: Volume 2, Short Papers*. Association for Computational Linguistics, 663–668. http://aclweb.org/anthology/E17-2105

[13] Maggie Yundi Li, Stanley Kok, and Liling Tan. 2018. Don't Classify, Translate: Multi-Level E-Commerce Product Categorization Via Machine Translation. *arXiv e-prints*, Article arXiv:1812.05774 (Dec. 2018), arXiv:1812.05774 pages. arXiv:cs.CL/1812.05774

[14] Tom Zahavy, Alessandro Magnani, Abhinandan Krishnan, and Shie Mannor. 2016. Is a picture worth a thousand words? A Deep Multi-Modal Fusion Architecture for Product Classification in e-commerce. *CoRR* abs/1611.09534 (2016). arXiv:1611.09534 http://arxiv.org/abs/1611.09534

Multi-Label Product Categorization Using Multi-Modal Fusion Models

Pasawee Wirojwatanakul[1] **and Artit Wangperawong**[2]
[1]New York University, pw1103@nyu.edu
[2]U.S. Bank, artit.wangperawong@usbank.com

Abstract

In this study, we investigated multi-modal approaches using images, descriptions, and titles to categorize e-commerce products on Amazon. Specifically, we examined late fusion models, where the modalities are fused at the decision level. Products were each assigned multiple labels, and the hierarchy in the labels were flattened and filtered. For our individual baseline models, we modified a CNN architecture to classify the description and title, and then modified Keras' ResNet-50 to classify the images, achieving F_1 scores of 77.0%, 82.7%, and 61.0%, respectively. In comparison, our tri-modal late fusion model can classify products more effectively than single modal models can, improving the F_1 score to 88.2%. Each modality complemented the shortcomings of the other modalities, demonstrating that increasing the number of modalities can be an effective method for improving the performance of multi-label classification problems.

Introduction

Background

To sell products on many e-commerce systems, sellers are tasked with providing categories for their products. Automating product classification can reduce manual labor time and cost, giving sellers a better experience when uploading new products. Such auto-labeling can also benefit the buyers, as sellers manually tagging their own products may be inaccurate or sub-optimal. An performant classifier is important, as mislabelled products may lead to missed sales opportunities due to buyers not being able to effectively locate the things they want to buy.

Prior studies have approached product categorization as a text-classification task (Kozareva 2015; Yu et al. 2012; Vandic, Frasincar, and Kaymak 2018). However, ideally multiple types of inputs can be considered, including title, description, image, audio, video, item-to-item relationships, and other metadata. Although a few recent studies have explored product categorization using both text and images (Åberg 2018; Zahavy et al. 2016), here we report on a strategy for combining an arbitrary number of inputs and modes. We specifically demonstrate a multi-modal model based on images, titles, and descriptions. Our task is different from a regular multi-class classification problem, as a product may be labeled with more than one class. Most products will appear in many classes and sub-classes. Classes can be nested in another class and potentially nested in another sub-class. Given the large amount of products uploaded and the numerous possible labels applicable, machine learning can be used to automatically classify the products in a more efficient manner.

As images, titles and descriptions are different modalities of data that can each capture unique aspects of a product, we explored fusing individual models trained for each modality. Note that although fusing and ensembling both involve combining multiple models, for the purposes of our discussion each of the models utilize different modalities of data in fusion, whereas the same data passes through all of the models in an ensemble. There are two common ways to fuse different modal networks: late fusion and early fusion (Fig. 1). Late fusion refers to combining the predictions (outcome probabilities) of multiple networks using a certain policy. Such a policy can be using the maximum or minimum of the outcomes. In contrast, in early fusion vector representations of each modality can be extracted at an early level and fused with one another through concatenation or addition to produce a multi-modal representation vector. The model then performs classification on the resulting multi-modal representation vector.

Related Works

Convolutional neural network (CNN) architectures (Kim 2014) have been used to classify the title of products (Zahavy et al. 2016). In such a model, the first layer uses a random word embedding. The same study also used a VGG network for image classification (Simonyan and Zisserman 2014). While they experimented with both early and late fusion, only the late fusion resulted in an improvement in performance. The image and text classifiers were trained separately to achieve maximal performance individually before being combined by a policy network. The policy network which achieved the highest performance was a neural network with 2 fully-connected layers and took in the top-3 class probabilities from the image and text CNNs as input. Their dataset contained 1.2 million images and 2,890 possible shelves. On average, each product falls in 3 shelves.

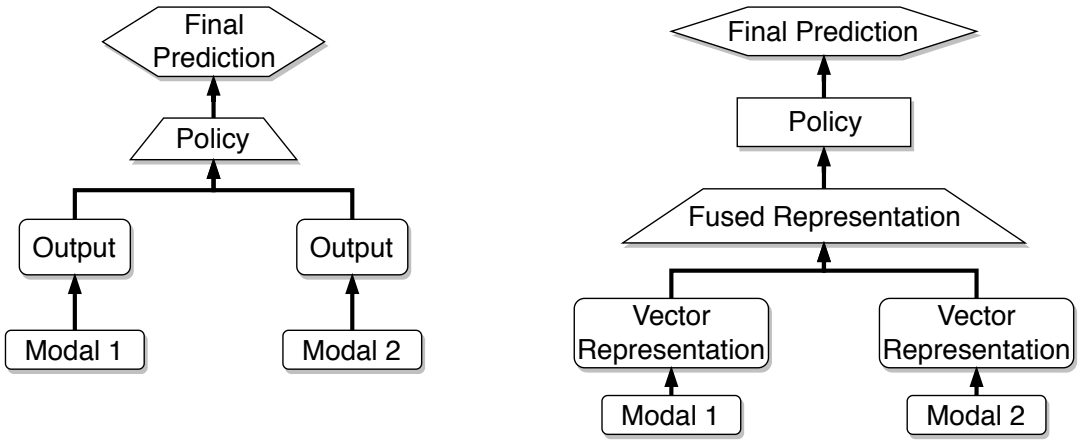

Figure 1: The diagram on the left represents late fusion and the diagram on the right represents early fusion.

Their model is considered effective when the network correctly outputs one of the three shelves.

Existing studies have also used the image, title, and description of an ad/product to classify products into single categories (Åberg 2018). Åberg concatenated the title and description, and used fastText (Joulin et al. 2016) as the baseline model for text classification, while using the Inception V3 for image classification. Åberg also explored a similar implementation of Kim's CNN architecture (Kim 2014) but could not achieve the level of performance of fastText.

The Åberg study used a dataset containing 96,806 products belonging to 193 different classes. Since each product was assigned exclusively to one class, multi-label categorization was not addressed. Hence a softmax function could be applied in the final layer before outputting the class probabilities. Similar to work of Zahavy et al. described above, both late and early fusion were explored, and late fusion yielded better results. Both heuristic policies and network policies were explored. Heuristic policies refer to some static rule; as an example, the mean of the probabilities from different modals. Network policies refer to training a neural network that takes the output probabilities from different networks and produces a new probability vector.

Dataset

The dataset used in our study comprises of 9.4 million Amazon products (McAuley et al. 2015). The class hierarchical information was not available, as the classes and subclasses were pre-flattened as given. We randomly sampled 119,073 products from this dataset, in which the first 90,000 products are kept for the training set. After pre-processing, there are 122 possible classes in which a product can belong to. Unlike many previous studies, here each product can be assigned multiple labels. Each product in the dataset contains the image, description, title, price, and co-purchasing network.

Product categorization systems can be challenging to build due to the trade-off between the number of classes and efficacy. As an example, adding more classes and subclasses to a product might make it easier to discover, but more classes would also increase the likelihood of an incorrect class being applied. To address this issue, some studies reduced the number of sub-classes (Zahavy et al. 2016; Lyu, Lee, and Li 2017). One method is to create a shelf and categorize the products based on the shelves they are in. A shelf is a group of products presented together on the same e-commerce webpage, which usually contains products under the same categories (Zahavy et al. 2016). Since our dataset does not contain the webpage information necessary to form shelves, our method was to remove the classes containing less than 400 products.

On average, each product belongs to 3 categories after pre-processing. The maximum number of products in a category is 37,102 and the minimum number of products in a category is 558. On average, there are 2,919 products per category. In addition, we can see from Fig. 2 that the number of products per categories is not evenly distributed, which could introduce bias into the model.

Baseline Models

In order to understand how much we benefit from fusing the different modal classifiers, we report the baseline results for each modal below. We evaluate our results using the F_1 score (micro-averaged), which is an accepted metric for multi-label classification and imbalanced datasets (Yang and Liu 1999). During training, for all classifiers, we used Adam (Kingma and Ba 2014) as our optimizer and categorical cross-entropy as our loss function. To accommodate multi-labeling, the final activations for each classifier utilizes the sigmoid function. Although both titles and descriptions are textual data, we leverage their different use-cases by treating them as different modalities, allowing us to perform different pre-processing steps as described below.

Description Classifier

The description was pre-processed to remove stop words, excessive whitespace, digits, punctuations, and words longer than 30 characters. In addition, sentences were truncated to 300 words. To classify the pre-processed descriptions,

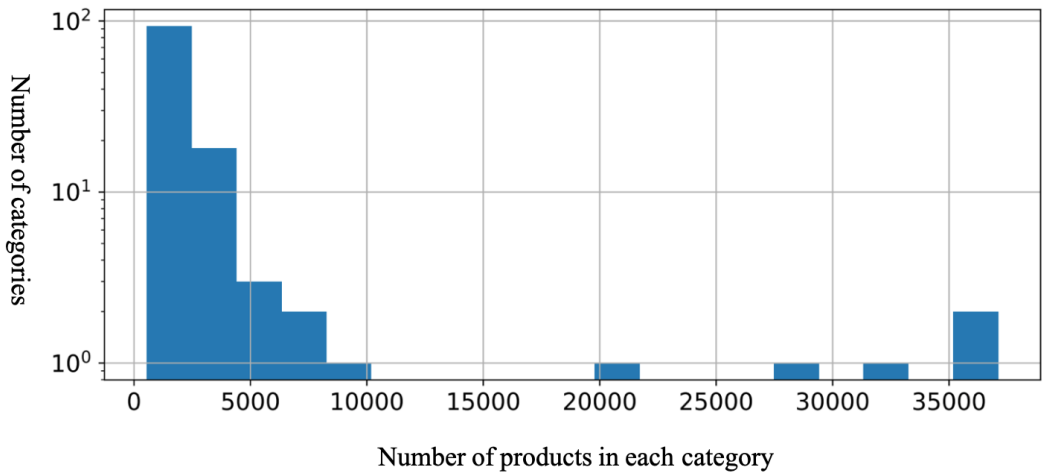

Figure 2: The x-axis represents the number of products in a category, whereas the y-axis represents the number of categories with that number of products.

we slightly modified Kim's CNN architecture for sentence classification. Kim's architecture is a CNN with one layer of convolution on top of word vectors initialized using Word2Vec (Kim 2014; Mikolov et al. 2013). Max-pooling over time is then applied (Collobert et al. 2011), which serves to capture the most important features. Finally, dropout is employed in the penultimate layer.

Unlike the models used in prior studies mentioned above (Kim 2014), we used GloVe as our embedding. Words not covered by GloVe were initialized randomly. For our dataset, GloVe covers only 61.0% of the vocabulary from the description. Our first convolution layer uses a kernel of size 5 with 200 filters. We then performed global max pooling, followed by a fully connected layer of 170 units with ReLU activations. Our final layer is another densely connected layer of 122 units with sigmoid activation. This model achieves 77.0% on the test set.

Title Classifier

Although an identical classifier to the description classifier was used for the title, the title data was pre-processed differently. For the title, we did not remove the stop words and limited or padded the text to 57 words. We again chose GloVe for the embedding, in which words not covered were initialized randomly. GloVe covers 77.0% of the vocabulary from the title. This model achieves 82.7% on the test set.

Image Classifier

We modified the ResNet-50 architecture (He et al. 2015) from Keras by removing the final densely connected layer and adding a densely connected layer with 122 units to match the number of labels we have. In addition, we changed the final activation to be sigmoidal. ResNet-50 is based on the architecture that achieves competitive results compared to other state-of-the-art models for image classification. We also used the pre-trained imagenet weights, which has been trained on the imagenet dataset (Deng et al. 2009), containing more than 14 million images. We kept the weights of the earlier layers frozen and trained only the deeper/later layers (Yosinski et al. 2014). We experimented with the number of trainable layers, in which our top model was trained on the last 40 layers, achieving a F_1 score 61% on the test set.

Summary

The results summarized in Table 1 underscore that the classifiers differ in discriminative powers as the title and description classifiers significantly outperform the image classifier. This result is consistent with previous similar studies reporting a significant difference between the image and title classifiers (Zahavy et al. 2016). Moreover, we have shown that the description classifier also significantly outperforms the image classifier. Such results suggests that text can provide more relevant information regarding a product's categories.

Table 1: F_1 scores for each individual classifier.

Modal	$\mathbf{F_1}$ (%)
Image	61.0
Title	82.7
Description	77.0

Error Analysis

Table 2 exhibits that the top misclassified categories for each classifier generally reflect their inadequate representation in the dataset. Recall that the average number of products per category is 2,919. The Accessories category contains the most products (924) out of all the misclassified categories, but it is still far below the average. In addition, we can see that the top misclassified categories for each classifier seldom overlap between the modal classifiers. For the categories that the image classifier performed poorly, the description and title classifiers perform better and vice versa.

Table 2: The top 15 most misclassified categories/classes for each classifier. The fraction represents the number of products which should be predicted as class c_i, but is not, over the total number of products that is in c_i.

Image	**Description**	**Title**
Martial Arts (146/154)	Women (107/134)	Horses (167/265)
Ballpoint Pens (192/203)	Accessories (663/924)	Novelty, Costumes & More (176/303)
Reptiles & Amphibians (136/144)	Clothing, Shoes & Jewelry (481/686)	Women (77/134)
Small Animals (327/349)	Boating (222/327)	Accessories (521/924)
Chew Toys (123/132)	Novelty, Costumes & More (194/303)	Feeding (137/244)
Squeak Toys (202/223)	Parts & Components (135/212)	Clothing, Shoes & Jewelry (384/686)
Cards & Card Stock (222/246)	Men (183/291)	Hunting & Tactical Knives (94/175)
Filter Accessories (108/120)	Chew Toys (83/132)	Balls (99/185)
Other Sports (188/210)	Balls (116/185)	Hunting Knives (80/152)
Bedding (154/174)	Boating & Water Sports (368/591)	Boating (171/327)
Tape, Adhesives & Fasteners (231/262)	Tape, Adhesives & Fasteners (161/262)	Small Animals (176/349)
Birds (432/490)	Office Furniture & Lighting (340/556)	Men (146/291)
Pumps & Filters (270/308)	Forms, Recordkeeping & Money Handling (196/323)	Chew Toys (65/132)
Cages & Accessories (125/144)	Hunting Knives (89/152)	Boating & Water Sports (275/591)
Horses (229/265)	Team Sports (228/399)	Carriers & Travel Products (65/142)

This suggests that we should be able to combine the classifiers to effectively complement each other's shortcomings for a more effective overall result.

Multi-Modality

As prior studies found that late fusion models were more effective than early fusion models (Åberg 2018; Zahavy et al. 2016), here we focus our studies on improving late fusion.

Predefined Policies

First we evaluated the efficacy of our models using predefined rules in order to compare with other non-static policies. We experimented with max policy and mean policy of the output from each of the classifiers. The max policy selects the highest output for each class prediction from among the image, label, and title classifiers. This can be represented as

$$o_{\text{max}} = \max(o_{\text{image}}, o_{\text{title}}, o_{\text{description}}), \tag{1}$$

where $o_{\text{image}}, o_{\text{title}}, o_{\text{description}} \in R^{122}$ represent the output from each classifier, and the mean policy can be represented as

$$o_{\text{mean}} = \frac{o_{\text{image}} + o_{\text{title}} + o_{\text{description}}}{3}. \tag{2}$$

Both mean and max policy resulted in lower F_1 scores when compared to the top classifier, which is the title classifier. The mean policy yielded 81.7%, while the max yielded 78.8% F_1 scores. Intuitively, each classifier contributes equally to the mean policy. Therefore, we would expect that the average performance is less than that of the best performer. For the max policy, the erroneous maximal outputs from the low performing classifiers detriment the ultimate predictions.

Linear Regression

We trained a simple ridge linear regression model to fuse the individual classifiers into a single classifier. The model achieves 83.0% on the test set. The model can be considered as minimizing the mean-squared error loss function according to

$$\min_{\boldsymbol{w}} ||\boldsymbol{wX} - \boldsymbol{y}||_2^2 + \alpha||\boldsymbol{w}||_2^2, \tag{3}$$

where $\boldsymbol{y}$ is the true label, $\boldsymbol{wX}$ is the predicted label, and α is the L_2 regularization strength. Nevertheless, the simple non-static policy can outperform static policies above.

Bi-Modal Fusion

Prior studies involved two neural networks, one for classifying images and another for classifying titles, using late fusion (Zahavy et al. 2016). For comparison purposes, we examined models developed from fusing two of the three modal networks in this study. The first fused network includes the image classifier's output and the title classifier's output in a similar fashion with prior studies (Zahavy et al. 2016). We then fused the title classifier's output and description classifier's output for the second fused network and fused the image classifier's output and description classifier's output for the third network. All three networks were fused the same way, using a three layer neural network to concatenate the outputs from each of the classifiers. The first, second, and third layers contained 200, 150 and 122 units, respectively. All the activations were sigmoidal. The image-description, image-title, and description-title fused networks yielded F_1 scores of 82.0%, 85.0%, and 87.0%, respectively (see Table 3).

Tri-Modal Fusion

Finally, we developed a tri-modal model to include the titles, images, and descriptions (see Fig. 3). To our knowledge, we are the first to fuse three classifiers/neural networks to categorize products. We fused the three classifiers as in the baseline models using a policy network, which is an additional neural network that takes in the output of each of the classifiers. We varied the number of layers, activation functions, and units of the neural networks. Through hyperparameter optimization, we found that the top policy network consists of three layers. It uses the sigmoidal activation on the first

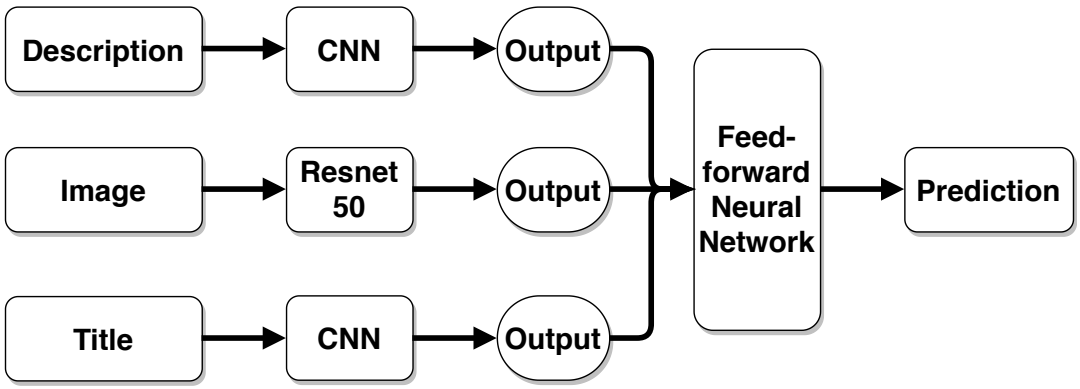

Figure 3: The proposed tri-modal fusion model architecture for product categorization using title, description and image.

and last layers and hyperbolic tangent activation on the middle layer. This fused model achieves an F_1 score of 88.2% surpassing all of the previous methods.

Table 3: F_1 scores for fused classifiers.

Model	$\mathbf{F_1}$ (%)
Max	78.8
Mean	81.7
Linear Regression	83.0
Image-Description Fused	82.0
Image-Title Fused	85.0
Title-Description Fused	87.0
Image-Description-Title Fused	88.2

Discussion

Compared to Table 2, the proportion of misclassified products has reduced significantly in Table 4. In examining Accessories, Horses, Clothing, and Shoes & Jewelry, we can see that the proposed method outperforms the individual classifiers by a considerable margin. However, the proposed method fails to significantly reduce the number of misclassified products on certain categories, such as Chew Toys. According to Table 2, each of the individual classifiers performed poorly predicting products as Chew Toys. This suggests that there remains categories that are underserved across all classifiers. To address this shortcoming, more data or other modes could be considered in future work. On the other hand, the result also suggests that as long as one classifier performs well on some of the tasks, it is sufficient for the overall model. For example, the number of misclassified products in Clothing, Shoes & Jewelry dropped from 384 to 256. Overall, this method improves over the top individual classifier and the top two-modals fused network by 5.5% and 1.2%, respectively (see Table 3).

Conclusion

We have shown that the title classifier can outperform the description classifier, and that the description classifier can outperform the image classifier. Moreover, a tri-modal fused network comprising of all three modalities outperformed any of the bi-modal fused networks. The performance improvements can be attributed to each of the classifiers addressing at least complementary portions of the tasks to account for the shortcomings of each individual classifier.

Table 4: The top 15 most misclassified categories using our tri-modal fusion method.

	Fused Image-Title-Description
1	Chew Toys (64/132)
2	Accessories (417/924)
3	Women (58/134)
4	Novelty, Costumes & More (127/303)
5	Snacks (143/363)
6	Hunting Knives (59/152)
7	Men (112/291)
8	Clothing, Shoes & Jewelry (256/686)
9	Hunting & Tactical Knives (65/175)
10	Shampoos (58/158)
11	Balls (67/185)
12	Squeak Toys (79/223)
13	Horses (92/265)
14	Boating (113/327)
15	Airsoft (68/200)

While this study focused on late fusion, an early fusion approach can be explored in the future. In addition, more products, including products that may not fall under the predefined categories, can be added to reduce overfitting. A better text classifier can be built with pre-trained bi-directional contextualized language models (Devlin et al. 2018). Transformers can be considered to replace CNNs and RNNs for both text and images (Vaswani et al. 2017; Wangperawong 2018; Niki Parmar 2018).

Finally, one possible extension to our work could be to build a vector representation of the products. Just as how word embeddings enabled us to more effectively classify text, a product embedding can be useful for capturing the relationship between products. Such a product embedding could help discover products that are "similar" for recom-

mendation purposes and be used as input to a model to predict categories.

References

[Åberg 2018] Åberg, L. 2018. *Multimodal Classification of Second-Hand E-Commerce Ads*. Ph.D. Dissertation, KTH Royal Institute of Technology.

[Collobert et al. 2011] Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. P. 2011. Natural language processing (almost) from scratch. *CoRR* abs/1103.0398.

[Deng et al. 2009] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[Devlin et al. 2018] Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[He et al. 2015] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep residual learning for image recognition. *CoRR* abs/1512.03385.

[Joulin et al. 2016] Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of tricks for efficient text classification. *CoRR* abs/1607.01759.

[Kim 2014] Kim, Y. 2014. Convolutional neural networks for sentence classification. *CoRR* abs/1408.5882.

[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kozareva 2015] Kozareva, Z. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1329–1333.

[Lyu, Lee, and Li 2017] Lyu, F.; Lee, J.; and Li, Y. 2017. Category classification for amazon items using hidden state node embeddings of large graphs.

[McAuley et al. 2015] McAuley, J. J.; Targett, C.; Shi, Q.; and van den Hengel, A. 2015. Image-based recommendations on styles and substitutes. *CoRR* abs/1506.04757.

[Mikolov et al. 2013] Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space.

[Niki Parmar 2018] Niki Parmar, Ashish Vaswani, J. U. L. K. N. S. A. K. D. T. 2018. Image transformer. *arXiv* abs/1802.05751.

[Simonyan and Zisserman 2014] Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition.

[Vandic, Frasincar, and Kaymak 2018] Vandic, D.; Frasincar, F.; and Kaymak, U. 2018. A framework for product description classification in e-commerce. *J. Web Eng.* 17(1&2):1–27.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *CoRR* abs/1706.03762.

[Wangperawong 2018] Wangperawong, A. 2018. Attending to mathematical language with transformers. *CoRR* abs/1812.02825.

[Yang and Liu 1999] Yang, Y., and Liu, X. 1999. A re-examination of text categorization methods. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, 42–49. New York, NY, USA: ACM.

[Yosinski et al. 2014] Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? *CoRR* abs/1411.1792.

[Yu et al. 2012] Yu, H.-F.; Ho, C.-H.; Arunachalam, P.; Somaiya, M.; and Lin, C.-J. 2012. Product title classification versus text classification. In *Csie. Ntu. Edu. Tw*.

[Zahavy et al. 2016] Zahavy, T.; Magnani, A.; Krishnan, A.; and Mannor, S. 2016. Is a picture worth a thousand words? A deep multi-modal fusion architecture for product classification in e-commerce. *CoRR* abs/1611.09534.