E-Commerce Product Categorization via Machine Translation

LILING TAN, Rakuten Asia Pte Ltd
MAGGIE YUNDI LI and STANLEY KOK, National University of Singapore

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 article, 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. We discuss how the resultant taxonomy directed acyclic graph promotes user-friendly navigation, and how it is more adaptable to new products.

CCS Concepts: • **Information systems → Online shopping**; • **Computing methodologies → Machine translation**; **Classification and regression trees**;

Additional Key Words and Phrases: E-commerce, machine translation, classification

ACM Reference format:
Liling Tan, Maggie Yundi Li, and Stanley Kok. 2020. E-Commerce Product Categorization via Machine Translation. *ACM Trans. Manage. Inf. Syst.* 11, 3, Article 11 (July 2020), 14 pages.
https://doi.org/10.1145/3382189

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 3 to 10 levels deep with thousands of leaf nodes [McAuley et al. 2015; Shen et al. 2012] (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

Authors' addresses: L. Tan, Rakuten Institute of Technology, Rakuten Asia Pte Ltd; email: liling.tan@rakuten.com; M. Y. Li and S. Kok, National University of Singapore, NUS School of Computing, COM1, 13, Computing Dr, Singapore 117417; emails: a0131278@u.nus.edu, skok@comp.nus.edu.sg.

2158-656X/2020/07-ART11 $15.00
https://doi.org/10.1145/3382189

Fig. 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`.

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` → `Flour & Dough` → `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 MT 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, MT systems, through the use of deep learning [Goodfellow et al. 2016], have improved their accuracy by leaps and bounds in recent years [Bahdanau et al. 2015; Kalchbrenner and Blunsom 2013; Sutskever et al. 2014], 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 MT, 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, MT 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 but 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:

—`Food & Beverage` → ⋯ → `Cooking & Baking Supplies` → `Baking Dough & Mixes` → `Pancake & Waffle Mix`
—`Food & Beverage` → ⋯ → `Breakfast` → `Pancake & Waffle Mix`

The contribution of our work lies in empirically demonstrating that MT systems can be applied with great accuracy to the task of e-commerce product categorization. To our knowledge, this repositioning of MT systems for product categorization is not immediately obvious, and to the best of our knowledge, we are the first to demonstrate the viability of doing so. We believe e-commerce firms would reap the preceding operational and technical benefits should they adopt our proposed approach.

Next, we briefly review related work (Section 2). We then describe state-of-the-art MT 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

Most product categorization systems are 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 stepwise, first into higher-level categories, and then into lower-level sub-categories. 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 stepwise approach first performs classification across top-level categories, each of which aggregates products in its lower-level sub-categories to ameliorate the data imbalance problem. After assigning a product to a top-level category, the stepwise 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 stepwise 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 stepwise 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 (KNN), 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 (RNNs) [Hochreiter and Schmidhuber 1997], and the latter uses convolutional neural networks [LeCun et al. 1998].

Several stepwise classifiers have also been used for product categorization. Shen et al. [2012] use simple classifiers (e.g., naive Bayes and KNN) 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 KNN 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 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 MT 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 MT SYSTEMS

In the past, the predominant MT approach was phrase-based machine translation (PBMT), which is grounded in information theory and statistics. Although 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 their 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

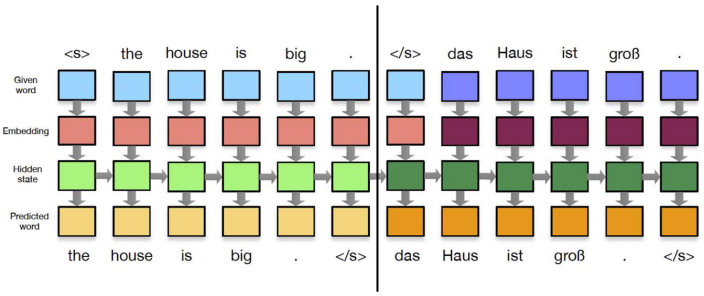

Fig. 2. The encoder-decoder model, also known as the sequence-to-sequence (Seq2Seq) model. The light green boxes to the left of the vertical line make up the encoder RNN. The first dark box to the right of the vertical line encodes the entire source sentence. As we move from lef to right, this source encoding is use to generate words in the target language. (Image from Koehn [2017].)

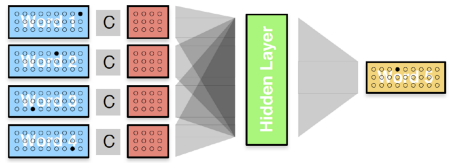

Fig. 3. A language model encoded by a feedforward neural network. The previous 4 words are represented as 1-of-*N* vectors (blue), compressed into continuous vectors (red) using the same matrix C for all words, passed through a hidden layer, and then used to predict the next word as a 1-of-*N* vector (yellow) (Image from Koehn [2017].)

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 and Figure 3). 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 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

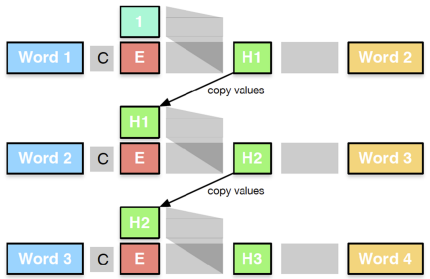

Fig. 4. A language model encoded as an RNN. After predicting Word 2 (yellow), we reuse the hidden layer H1 (green) together with the correct Word 2 (blue) to predict Word 3 (yellow). (Image from Koehn [2017].)

an infinite number of connected hidden layers, one stacked on top of another. Because of this depth of hidden layers, it can potentially learn complex dependencies among words (Figure 4).

Fourth, Cho et al. [2014b] extended RNNs for MT by creating the encoder-decoder model (also known as the sequence-to-sequence (Seq2Seq) model). This model concatenates two RNNs together: one that encodes the source language, and one that decodes the target language. In Figure 2, 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 et al. [2015] extend the attention mechanism by introducing both global and local attention mechanisms. The global attention mechanism functions as described previously 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 MT 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 require a time-consuming, left-to-right, word-by-word traversal of the entire input sentence 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—that is, 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 MT benchmarks, and we use it for our experiments in Section 4.

Table 1. Summary of the RDC and Ichiba Datasets

	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

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]
- ***Rakuten Ichiba***[3]: This dataset from *www.rakuten.co.jp* consists of 280 million products listed by more than 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].

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 of 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). Each MT system in Section 4.2 tokenizes a product title into individual words and then outputs a root-to-leaf path one node category at a time, similar to how a translated sentence is generated one word at a time.

The distribution of products across categories in both datasets is skewed toward the most popular categories as is usually the case in e-commerce domains [He and McAuley 2016; Xia et al. 2017]. Figures 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.

[1] https://sigir-ecom.github.io/data-task.html.
[2] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl.
[3] https://rit.rakuten.co.jp/data_release/.

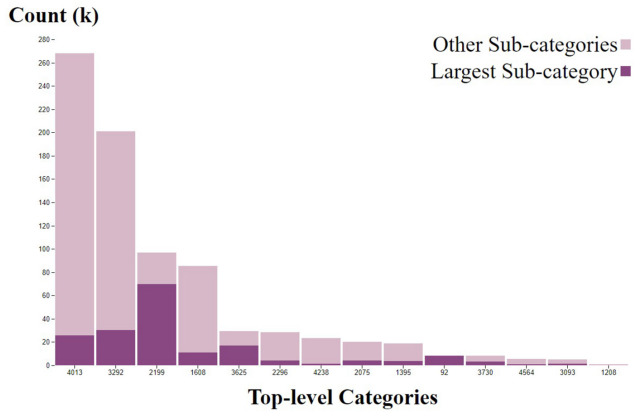

Fig. 5. Skewed product distribution in RDC. The categories on the *x*-axis from left to right are Jacket, Electronics, Automotive & Parts, Clothing, Shoes & Accessories, Beauty & Personal Care, Media, Office Supplies, Sports & Fitness, Toys, Toddlers & Baby, Everything Else, Health, Pet Supplies, Bags & Luggage, and Food & Beverage.

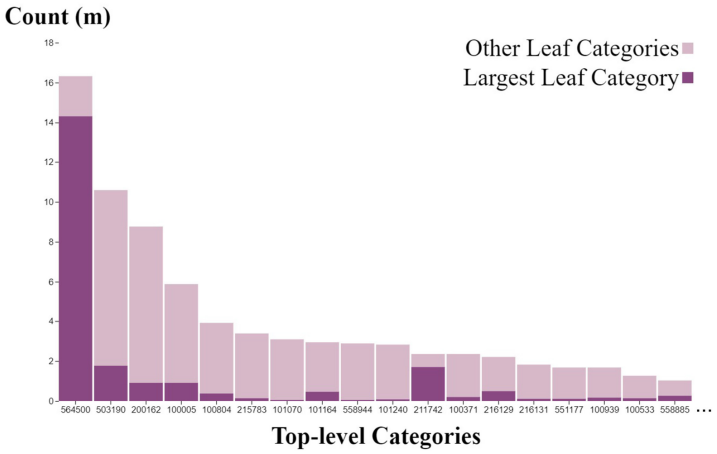

Fig. 6. Skewed product distribution in Ichiba. The categories on the *x*-axis from left to right are (translated from Japanese) Beauty, Cosmetics & Fragrances, Car Accessories & Bike Accessories, Books, Magazines & Comics, Flowers, Garden & DIY, Home Decor, Bedding & Shelves, Daily Necessities, Stationery & Handicrafts, Sports & Outdoor, Toys, Hobbies & Games, Kitchenware, Tableware & Cookware, CD, DVD & Musical Instruments, TV, Audio & Camera, Women's Fashion, Jewelry & Accessories, Bags, Accessories & Designer Items Men's Fashion, Beauty, Cosmetics & Fragrances, Kids, Baby & Maternity, and Shoes.

4.2 Models

For our 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 hyper-parameters of our models are given in Table 2. Note that the RNN hidden layer size is specific to the attentional Seq2Seq model, whereas the feedforward network (FFN) hidden layer and attention heads hyper-parameters are only used in

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

Table 2. Hyper-Parameters of Attentional Seq2Seq and Transformer Models

	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

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.

We compare our MT 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 DBN 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 DBNs, CUDeep also uses 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 rerank 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 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 F-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 $TP_c, P_c, R_c, F_c, F_c^{weighted}$ respectively are 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. (Note that for weighted variants of precision, recall, and F-score, the F-score may not lie between precision and

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

Table 3. Results of Our NMT Systems vs CUDeep Classification Systems on the RDC Dataset

		RDC		
		P	**R**	**F**
CUDeep	**DBN**	72.19	74.72	72.86
	KNN	71.14	72.10	70.94
	DBN+KNN	73.46	75.57	73.85
Our NMT Models	**Attentional Seq2Seq**	74.03	73.43	72.50
	Transformer	74.44	75.25	73.83
	Seq2Seq+Transformer	**75.22**	**75.65**	**74.19**

Table 4. Results of Our NMT Systems vs CUDeep Classification Systems on the Ichiba Dataset

		Ichiba		
		P	**R**	**F**
CUDeep	**DBN**	78.09	78.29	77.52
	KNN	79.24	78.69	78.66
	DBN+KNN	82.65	82.27	82.05
Our NMT Models	**Attentional Seq2Seq**	82.65	82.27	82.05
	Transformer	83.79	83.59	**84.74**
	Seq2Seq+Transformer	**85.08**	**84.31**	84.26

recall.) Since the precision and recall are components of the F-score, we reported them as well in our empirical results lest a reader wishes to see how they contribute to the F-score.

4.4 Results

Table 3 and Table 4 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 in the following.

From Table 4, the bolded numbers show 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). 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.

Table 5. Confidence Interval of 1,000 Iterations of Bootstrap Resampling on the Best-Performing Models on the RDC Dataset

	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 6. Confidence Interval of 1,000 Iterations of Bootstrap Resampling on the Best-Performing Models on the Ichiba Dataset

	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 7. Effects of Data Size with Respect to Systems' Weighted F-score

	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

To establish the statistical significance of our results, we conducted 1,000 iterations of bootstrap resampling on the best-performing model in each system to find out the 95% confidence interval of their performance scores. Tables 5 and 6 contain the results.

We investigated the effect of training data size on the performance of the systems on the RDC data. Table 7 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 7, we see that our MT-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.

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.

Table 8. Count of Full-Path Categories Created

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

We present the count of novel categorization paths created by each of our models in Table 8. 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 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 MT 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 AND 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 MT, we show that state-of-the-art 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 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, evaluating the impact of the new root-to-leaf paths on user interaction, experiments with more MT models and classification models, and automatic induction of the product taxonomy from data, among others.

This research was partly funded by MOE AcRF Tier 1 grant (R-253-000-146-133) to Stanley Kok.

REFERENCES

Chris Anderson. 2006. *The Long Tail*. Hyperion.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3 (Jan. 2003), 993–1022.

Ali Cevahir and Koji Murakami. 2016. Large-scale multi-class and hierarchical product categorization for an e-commerce giant. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (COLING'16)*. 525–535.

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

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. arXiv:1409.1259.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27.

Pradipto Das, Yandi Xia, Aaron Levine, Giuseppe Di Fabbrizio, and Ankur Datta. 2016. Large-scale taxonomy categorization for noisy product listings. In *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data'16)*. IEEE, Los Alamitos, CA, 3885–3894.

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

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

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*. ACM, New York, NY, 107–115.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, et al. 2018. Achieving human parity on automatic Chinese to English news translation. arXiv:1803.05567.

Ruining He and Julian McAuley. 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*.

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

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18, 7 (2006), 1527–1554.

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

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing*. 1700–1709.

Philipp Koehn. 2017. Neural machine translation. arXiv:1709.07809.

Zornitsa Kozareva. 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 (NAACL HLT'15)*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 230–237.

Yann LeCun, Léon Bottou, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.

Yiu-Chang Lin, Pradipto Das, and Ankur Datta. 2018. Overview of the SIGIR 2018 eCom Rakuten Data Challenge. In *Proceedings of the 2018 SIGIR Workshop on eCommerce (SIGIR eCom'18)*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

Julian J. McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 785–794.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. 3111–3119.

Brian H. Ross and Gregory L. Murphy. 1999. Food for thought: Cross-classification and category organization in a complex real-world domain. *Cognitive Psychology* 38, 4 (1999), 495–553. DOI: https://doi.org/10.1006/cogp.1998.0712

Patrick Shafto and John D. Coley. 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 (2003), 641–649.

Patrick Shafto, Charles Kemp, Elizabeth Baraff, John D. Coley, and Joshua B. Tenenbaum. 2005. Context-sensitive induction. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*. 2003–2008.

Dan Shen, Jean-David Ruvini, and Badrul Sarwar. 2012. Large-scale item categorization for e-commerce. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12)*. ACM, New York, NY, 595–604. DOI: https://doi.org/10.1145/2396761.2396838

Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. 2014. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *Proceedings of the VLDB Endowment* 7 (2014), 1529–1540.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. 3104–3112,.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Red Hook, NY, 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

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*.

Hsiang-Fu Yu, Chia-Hua Ho, Prakash Arunachalam, Manas Somaiya, and Chih-Jen Lin. 2013. *Product Title Classification Versus Text Classification*. Technical Report. Department of Computer Science, National Taiwan University, Taipei, Taiwan.

Received May 2019; revised October 2019; accepted February 2020