# Large-scale Multi-class and Hierarchical Product Categorization for an E-commerce Giant

**Ali Cevahir**
RIT Tokyo, Rakuten Inc.,
Rakuten Crimson House,
1-14-1 Tamagawa, Setagaya-ku, Tokyo
ali.cevahir@rakuten.com

**Koji Murakami**
RIT NY, Rakuten USA,
215 Park Avenue South,
New York NY 10003
koji.murakami@rakuten.com

## Abstract

In order to organize the large number of products listed in e-commerce sites, each product is usually assigned to one of the multi-level categories in the taxonomy tree. It is a time-consuming and difficult task for merchants to select proper categories within thousands of options for the products they sell. In this work, we propose an automatic classification tool to predict the matching category for a given product title and description. We used a combination of two different neural models, i.e., deep belief nets and deep autoencoders, for both titles and descriptions. We implemented a selective reconstruction approach for the input layer during the training of the deep neural networks, in order to scale-out for large-sized sparse feature vectors. GPUs are utilized in order to train neural networks in a reasonable time. We have trained our models for around 150 million products with a taxonomy tree with at most 5 levels that contains 28,338 leaf categories. Tests with millions of products show that our first predictions matches 81% of merchants' assignments, when "others" categories are excluded.

## 1 Introduction

E-commerce has grown rapidly in recent years. Giant e-commerce companies like Amazon, e-Bay, Taobao and Rakuten list millions of products on their sites sold by thousand of merchants. As of May 2016, Japan's largest e-commerce site Rakuten Ichiba[1] hosted 186 million active products sold by 43,363 different merchants. In order to organize products so that customers can navigate and search them easily, products are categorized into multi-level categories. "Women's Fashion > Tops > Sweaters > Long-sleeved knit > Crew neck" is an example for such categories. Rakuten Ichiba contains around 30 thousand categories of up to 5 levels. Merchants need to manually assign each product to one of those categories, which is a tedious task and prone to error. Moreover, merchants may not be accurate while assigning products, the cateogry assignment of the same product listed by different merchants may not be consistent. Automatic category recommendation for given product information helps to solving these problems.

Product classification is a text classification with a large hierarchical product taxonomy, and a lot of research have been conducted with various methodologies (Gupta et al., 2016; Shen et al., 2012; Kozareva, 2015; Qiu et al., 2011). Product classification includes the following challenges: 1) the products sparsely distributed in a large number of categories and data distribution is quite skewed, 2) the length of product titles and descriptions is diverse, 3) even though there are tons of product data, it is not guaranteed that pairs of current product title and assigned category are correct.

In this paper we propose a large-scale classification method for e-commerce products to classify them into thousands of multi-level categories. We use 172 million product title and description data for making predictions on products from Rakuten Ichiba. Please note that our techniques can be applied to other languages as well. Titles and descriptions are preprocessed to extract words, model numbers, sizes and



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



*Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 525–535, Osaka, Japan, December 11-17 2016.

counts. Deep belief nets (DBN) and deep autoencoders (DAE) are used to build models for given data sources. Combining results from different models and different data sources, final predictions are made. Input feature vectors are very high dimensional and sparse for the deep models we train, which makes it impractical to train in usual way. We apply the selective reconstruction idea by Dauphin et al. (2011) for training DBN and DAE. DBN and DAE with selective reconstruction are implemented on GPUs in order to process a large product base within a reasonable amount of time. Conventional methods like multinomial Naive Bayes are not practical to be used in that scale. We compared our results with passive-aggressive learning, and confirmed large accuracy improvement.

We can summarize our contributions as follows:

- We propose a large-scale classification method for e-commerce products to classify them into 28,338 categories organized in 5 level.
- E-commerce-specific features, like product models, sizes, counts are extracted from a corpus which mainly contains Japanese text.
- DBN and DAE with selective input reconstruction are implemented on GPUs.
- We conducted experiments with 172 million product titles.
- Our comparisons with merchants' assignments suggest 81% match, when "others" categories are excluded. We observed that our predictions can sometimes be more accurate than human labeling.

The remainder of this paper is organized as follows. Section 2 introduces related work. Section 3 and Section 4 overview exploited deep models and explain our proposed framework. In Section 5 we explain tokenization and feature extraction from product data. Section 6 presents experimental results of product classification, with varying settings. Section 7 concludes the paper.

## 2 Related Work

### 2.1 Text classification in large taxonomies

Xue et al. (2008) worked on deep classification in large-scale text taxonomy. They proposed two stage algorithm consisting of a search stage and classification stage. A language model was trained with over 1 million web documents and 130,000 documents was classified into over 130,000 categories. Qiu et al. (2011) proposed a variant Passive-Aggressive (PA) algorithm with latent concepts and evaluated the method with LSHTC dataset[2], which include over 13,000 categories and over 100,000 documents. Kosmopoulos (2015) proposed an extended hierarchical classification to predict the correct leaf by estimating the probability of each root-to-leaf path. LSHTC dataset has also been used to evaluate the method. Ha-Thuc et al.(2011) exploited classification approach without labeled data. In their algorithm, ontological knowledge was used to define the meaning of categories instead of relying on human-labelled documents. A typology consisting of 1,131 categories was used in the evaluation.

### 2.2 Product classification

There are various works devoted to multi-level category predictions for e-commerce products. Chen and Warren (2013) used multi-class-SVM with cost-sensitive function. They used 1,073 categories from UNSPSC taxonomy [3], which includes over 17,000 categories, and over 1 million products. Gupta et. al (2016) used word clustering and idf values to obtain document vector from product description and showed this document representation worked well in their product classification. Kozareva (2015) has worked on product classification with Yahoo! product data. They compared several classifiers and 5 kinds of features, and showed neural network embedding representation outperformed in product classification with over 300 categories in their category taxonomy.

Shen et. al (2011; 2012) proposed hierarchical classification, which decomposes into a coarse level and a fine level task, and used graph algorithm to discover automatically groups of highly similar classes as product category instead of relying on human-defined hierarchy. The model was trained with 83 million products from eBay.

[2] http://lshtc.iit.demokritos.gr
[3] http://www.unspsc.org/

### 2.3 Text classification with deep learning

In the last couple of years, deep learning algorithms have been exploited to text classification.

Zhang et al. (2015) showed that character-level convolutional network is an effective method in the text classification, and compared various models, including BoW (Bag of Words), word embedding, word-based ConvNet and long-short term memory with several kinds of large-scale dataset. Lai et al. (2015) have also worked on recurrent convolutional neural network (RCNN). A recurrent structure was used to capture contextual information as far as possible when learning word representations. They compared various models with 4 kinds of datasets, including small number of classes and middle size of instances. Kim (2014) reported sentence classification with Convolutional Neural Network. A simple improvement was considered to the convolutional architecture that two input channels are used to allow the employment of task-specific and static word embeddings simultaneously. Evaluation has been conducted with 6 kinds of datasets, including a few classes and small number of instances. Wang et al. (2015) worked on semantic clustering and convolutional neural network for short text classification and used 2 kinds of datasets, which consists of small number of classes and instances.

Ha et. al (2016) used deep learning-based product classification method, which consists of multiple recurrent neural networks (RNNs). They evaluated the method with more than 94 million products with approximately 4,100 leaf categories from NAVER shopping.

## 3 Deep Models Exploited

Recently, neural models gained attention for classification and semantic compression tasks. Deep belief nets are multiple layer neural networks used for classification tasks. They contain all-to-all connections between layers. Top layer of a DBN represents class probabilities of a given input vector. Hinton et al. (2006) proposed greedy layer-wise pre-training for DBN. Each layer (Restricted Boltzmann Machines) is trained by constructive divergence, using 1-step Gibbs sampling. Given class labels for top layer, the network is fine-tuned after greedy layer-wise pre-training is completed.

Deep autoencoders are used for finding compressed representations, i.e., semantic hashes (Salakhutdinov and Hinton, 2009; Hinton and Salakhutdinov, 2011) of given data, so that related input items have closer hash values (Hinton and Salakhutdinov, 2006). Similar to DBN, DAE contains multiple layers of restricted Boltzmann machines (RBM) which are stacked on top of each other. Each RBM is trained one after another in a greedy way and the overall network is fine-tuned after greedy pre-training is completed. Unlike DBN, DAE does not contain any class layer on top; DAE training is fully unsupervised.

Matrix multiplication is the core operation for training deep networks. Weight for each layer corresponds to a matrix of size $v \times h$, where $v$ is the input layer length and $h$ is the output layer length. For large and sparse inputs, sparse operations can be used to construct output layer of an autoencoder. However, input layer should also be reconstructed from dense output vectors for which sparse operations cannot be used in the original algorithm. This is prohibitively time-consuming for large-dimensional input data. In order to train with very high dimensional and sparse inputs, Dauphin et al. (2011) applied reconstruction sampling for stacked denoising autoencoders. In this work, we apply reconstruction sampling for DBN and DAE, both for pre-training of the first layer autoencoder and during fine-tuning by stochastic gradient descent.

Let us give an example for reconstruction sampling. Let us consider a network with visible layer of size $v$ and first hidden layer of size $h$. Then, size of the weight matrix $W$ for the first autoencoder becomes $v \times h$. For the forward pass for a minibatch of 3 input vectors, $3 \times v$ matrix $X$ is multiplied with $W$ to generate $3 \times h$ batch output matrix $Y$. After $Y$ goes some non-linear operations, during the backward pass input $\bar{X}$ is reconstructed from $\hat{Y}$ by $\bar{X} = \hat{Y} \times W^T$ (we ignore bias parameters for the sake of simplicity). Say $X[1,a] = X[1,b] = X[2,b] = X[2,c] = X[2,d] = X[3,a] = X[3,e] = 1$ and others are all $0$. Then, for this minibatch, we use only rows $a, b, c, d, e$ of $W$ during training. This means, while reconstructing $\bar{X}$ only $h \times 5$ matrix $W[a,b,c,d,e]^T$ is multiplied with $\hat{Y}$. For each minibatch, only a portion of the weight matrix is trained. After all training batches are processed, training is completed for all rows of $W$.

**Classify**(title, descr, category-ids, trained-models)
// *trained-models:* DBN-{title,descr}-*, DAE-{title,descr}, hashed-training-{titles, descrs}
title-word-vec ← **feature-extractor**(title) // *Regex for product codes etc. and Kuromoji Analyzer*
descr-word-vec ← **feature-extractor**(descr)

// *Nearest points (found by kNN over semantic hashes) are used for both steps*
title-semantic-hash ← **compute-DAE**(DAE-title, title-word-vec)
descr-semantic-hash ← **compute-DAE**(DAE-descr, descr-word-vec)
title-nearest-points ← **nearest-points**(title-word-vec, hashed-training-titles)
descr-nearest-points ← **nearest-points**(descr-word-vec, hashed-training-descrs)

// *Classifier outputs for first step*
title-l1-kNN-probs ← **kNN-classify**(title-nearest-points, level1-category-ids)
descr-l1-kNN-probs ← **kNN-classify**(descr-nearest-points, level1-category-ids)
title-l1-DBN-probs ← **compute-DBN**(DBN-title-l1, title-word-vec)
descr-l1-DBN-probs ← **compute-DBN**(DBN-descr-l1, descr-word-vec)
l1-probs ← **average**(title-l1-kNN-probs, descr-l1-kNN-probs, title-l1-DBN-probs, descr-l1-DBN-probs)
N ← **argmax**(l1-probs) // Predicted level-1 category id

// *Step 2*
title-kNN-probs ← **kNN-classify**(title-nearest-points, leaf-category-ids)
descr-kNN-probs ← **kNN-classify**(descr-nearest-points, leaf-category-ids)
title-N-DBN-probs ← **compute-DBN**(DBN-title-N, title-word-vec)
descr-N-DBN-probs ← **compute-DBN**(DBN-title-N, descr-word-vec)
probs ← **average**(title-kNN-probs, descr-kNN-probs, title-N-DBN-probs, descr-N-DBN-probs)
final-prediction-id ← **argmax**(probs)

Figure 1: 2-step classification for a given product title with trained models.

## 4 Classification Models

Classification is implemented in two steps. In first step, first level categories are predicted. There are 35 first level categories in Rakuten Ichiba. In the second step, leaf categories are predicted.

In each step, two classifiers, DBN and kNN (k-Nearest Neighbors) are used for each of two different data sources (titles and descriptions). Category predictions are made by averaging probability distribution scores of four different classifiers. DBN accepts 0-1 word vectors of high dimensions, and returns class probabilities as output. DAE is used to find semantic hash values for 0-1 word vectors. These semantic hashes are then used for k-nearest-neighbor classification.

For each data source, one DBN model is trained for first level classification and 35 different models are trained for sub-categories under each first level category. In total, we have $2 \times (1 + 35) = 72$ DBN models trained. However, we have only 2 DAE models for semantic hashing of titles and descriptions. In the first step, first level category IDs are used for kNN classification and in the second step, leaf level category IDs are used for kNN classification using the same semantic hashes. Two step classification using trained models is summarized in Figure 1.

kNN classification using high dimensional semantic hash values and hundreds of millions of points (training items) is not practical if all points are traversed during the neighbor search. Therefore, points are first clustered using hierarchical k-means (Böcker et al., 2004). Nearest neighbor search is implemented in the cluster whose center is the closest to the search point.

In the second step for classification, DBN classifiers return results from the same level-1 category tree. On the other hand, kNN classifiers can return any category. It is possible to build different DAE models and different kNN classifiers for each level-1 category, just like we do for DBN classification models.

However, doing so increases the number of models twice and it requires 4 kNN searches to classify a product, instead of 2. Besides decreasing run-times for training and classification, having a global model for kNN helps decreasing the error propagation ratio. If level-1 category is mispredicted in step 1, there is no way DBN classification can find the correct category. But, mixing with global kNN result may correct mispredictions made in step 1.

## 5 Tokenization and Feature Extraction

We use product titles and descriptions for classification, which are mainly in Japanese. Merchants in Rakuten Ichiba are free to assign their own titles and descriptions for the products they are listing. We assume no dictionary is available for meta-information, like manufacturer name. However, we extract model numbers, counts and sizes from the text. Remaining text is tokenized using Kuromoji[4] Japanese morphological analyzer to extract words.

We first normalize text by converting all Japanese characters to full-width and all non-Japanese characters to lower cases. All HTML tags are cleaned from descriptions. Products' model numbers are predicted using regular expressions, by checking alphabet/number combinations, possibly with spaces and dashes. For example, the words "iPhone 4s" is normalized to "iphone4s". Sizes are estimated by checking whether there exists quantity keywords following numbers. For example, "12.3 cm" is normalized to "12.3cm" and "12cm x 3 cm" is converted to "12cmx3cm". There may be some conflicts with model number extractor and size extractor. In this case, we keep both model numbers predictions and sizes, like "iPhone 4Gb" to be normalized as "iphone4gb 4gb". Japanese have counters for different type of objects. For example, "本" is used to count long and cylindirical objects and "枚" is used to count flat objects. Counters following numbers are not tokenized and kept together, like "3枚".

Japanese words are not split by spaces, so it is not straight-forward to split words. We use Kuromoji in search mode for tokenization of Japanese text and take the base form as word features. Stop words defined in Kuromoji plug-in for Lucene[5] and punctuation marks are excluded from the dictionary.

As a result of above feature extraction process, we have around 26 million words for titles and 47 million words for descriptions. These numbers are much bigger than dictionary sizes of natural languages, because of the nature of e-commerce data. Pre-processing for model number, size and count extraction, as well as misspelled words and failing to properly tokenize Japanese text increase the number of words. It is especially difficult to tokenize Katakana words which are used for writing imported words from other languages and prone to misspelling. We choose a frequency threshold, so that words appearing only a few times in the corpus are not selected as features to be used. Although less-frequent words are more expressive for products, by eliminating them we eliminate most of the noisy information as well. Eliminating less-frequent words also helps making the classifier practical and more robust for classification of new products. Sparse word vectors of the dictionary size after elimination of less-frequent words are accepted as input features for deep network classifiers, which we explain the details in the following sections.

## 6 Experimental Results

In this section, we discuss details of experiments in terms of prediction matching ratios with merchants' assignments and run-time discussions for the methods and implementations we explained in the paper.

### 6.1 Dataset and Model Properties

We used Rakuten product dataset, which is available under Rakuten Data Release program[6]. We processed 280 million (active and inactive) products listed by over 40,000 merchants. Products are assigned to 28,338 active categories. There are many products sharing the same title which we remove beforehand. After deduplication by titles, around 40% of products were eliminated, remaining 172 million titles. 90% of those products were randomly selected as being training data, and remaining 10% as test.

[4] http://www.atilika.org/
[5] http://lucene.apache.org/
[6] http://rit.rakuten.co.jp/opendata.html

We extracted features from training data as explained in Section 5. As a result, 26 million words were extracted for titles and 47 million words were extracted for descriptions. Words appearing more than 50 times in the training data were selected as features for training first-step DBN and DAE networks. As a result, the number of word features (dictionary size) was 968,471 for titles and 1,461,625 for descriptions. We built 4-layer step 1 DBN for titles with layer sizes $\{968471, 1000, 2000, 35\}$, and for descriptions $\{1461625, 650, 2000, 35\}$. 35 is the number of level-1 categories. For DAE, last layer size is 64, meaning we built 64-D semantic hash values for each input word vector.

Input layer is still very large for training deep networks, but the word-vectors are very sparse which makes it possible for deep networks to be trained in reasonable time using the selective reconstruction method explained in Section 3, using latest generation GPUs. Average number of words for a title is 12.9 and average number of words for a description is 98.7. Hence, for a input vector batch of size 100, at largest $1290 \times 1000$ parameters were trained for titles and at most $9870 \times 650$ for descriptions.

DBNs in step 2 for each level-1 category used different feature sets. Dictionary sizes for level-1 categories are much smaller when compared with that of whole corpus. Most frequent 500,000 words were taken as features for second step DBNs. By doing so, if there are more words than 500,000 in a level-1 category, words appearing only once or twice were eliminated. Therefore DBN models for titles and descriptions in step 2 is of size $\{\min(v, 500000), 1800, 2000, n\}$, where $v$ is the dictionary size and $n$ is the number of leaf categories in the corresponding level-1 category.

For kNN classification, after training DAEs (where we selected k to be 10), semantic hash values were calculated for training data. Those vectors then goes into hierarchical k-means clustering, where the set of hash vectors are clustered into 64 in each level until the number of points within a cluster falls below 10,000.

### 6.2 Hardware and Software Setup

We used a Ubuntu 14.04 Linux server with 4 Nvidia TitanX GPUs for our experiments. Each GPU has 12GB memory and 3072 processing cores. The server has two 12-core Intel CPUs running at 2.4GHz. The system has 96 GB main memory.

Extraction of word features was implemented on CPU, using regular expressions and Kuromoji analyzer. It took around 8 hours to extract features from 280 million titles and descriptions.

DBN and DAE were implemented using CUDA library with C++. Earlier work by Raina et al. (2009) shows that GPU implementation of deep network training can be more than two order of magnitudes faster when compared with single-core CPU implementations. In our implementation, besides the kernels we have written for original operations like selective reconstruction, we exploited CUDAMat (Mnih, 2009), cuBLAS and cuSPARSE libraries (NVIDIA, 2015) for efficient matrix operations. During training iterations, model weights were kept in GPU memory and input word vectors were stored in main memory in sparse format. For even larger models, it is possible to store model weights in main memory and communicate working parameters in each batch with GPUs, and/or stream input features from disk drive. However, these choices considerably slows down GPU training times. During greedy layer-wise pre-training, upper layers were constructed on memory using on-memory sparse input word vectors and trained lower-layer weights. This is a more practical solution than storing each layer output and streaming it for training upper layers, because intermediate layers are not sparse, requiring huge amount of storage for intermediate layers for large number of training samples.

Hierarchical k-means clustering and kNN search using hierarchical k-means tree were also implemented using GPUs, using the guidelines explained in Cevahir and Torii's work (2013). With our settings explained above, it took several days to train deep models and k-means tree using 4 GPUs.

### 6.3 Prediction Recalls

We compared prediction results with merchants' assignments in our test data. Merchants' assignments in Rakuten Ichiba are quite noisy, having the same products by different merchants distributed through different categories and more than 40% of products are in "Others" leaf categories (such as "Women's Fashion > Tops > Sweaters > Long-sleeved knit > Others"). However, it was not easy to eliminate noise

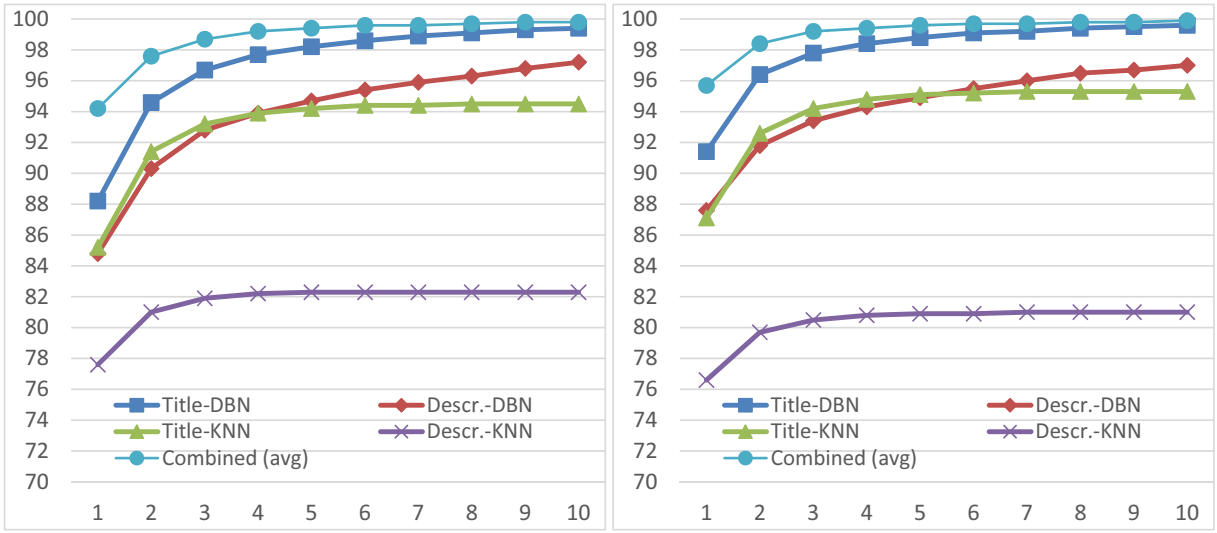


Figure 2: Top level category matching percent recalls (first step matching for 35 level-1 categories). **(a)** All categories including leaf-level categories named "Others", **(b)** excluding leaf-level categories named "Others".

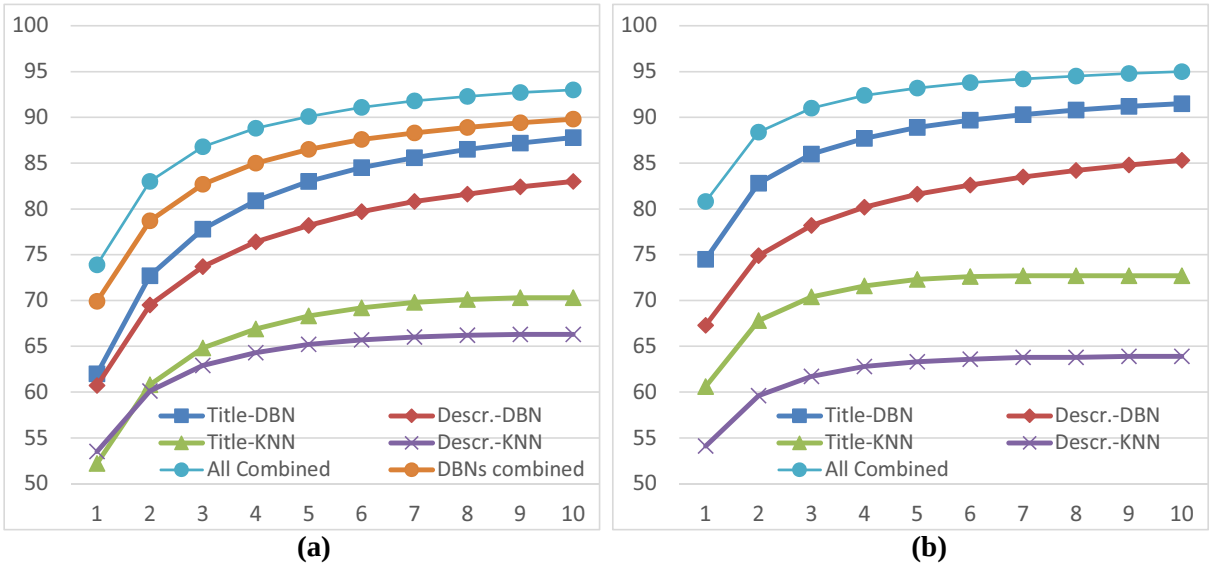


Figure 3: 2-step category matching percent recalls. **(a)** All 28,338 categories including leaf-level categories named "Others", **(b)** excluding leaf-level categories named "Others".

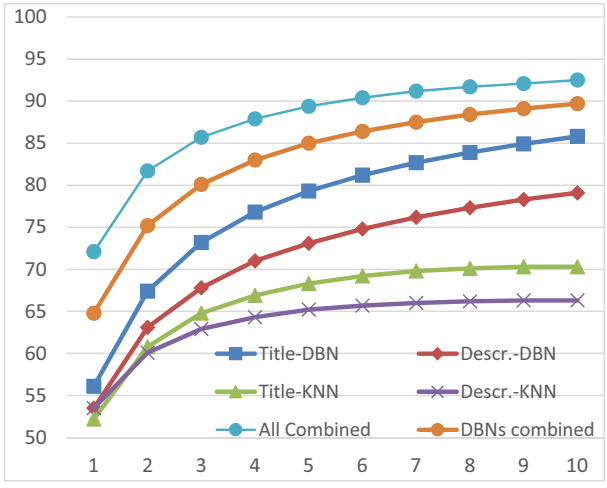


Figure 4: 1-step category matching percent recalls for all 28,338 categories including "Others".

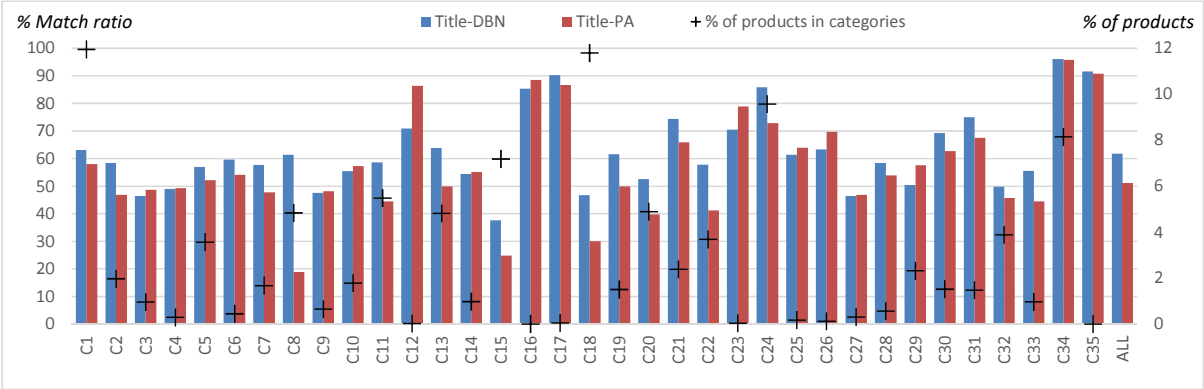


Figure 5: Comparison of category matching results grouped by first-level categories for title-DBN with a Passive-Aggressive algorithm with kernel slicing (Yoshinaga and Kitsuregawa, 2010).

for millions of test data in 28,338 categories by hand, hence we present prediction recalls considering the merchants' assignments as the basis.

Figure 2**a** depicts level-1 categorization matching percent ratios up to top-10 predictions in first step and Figure 3**a** depicts final categorization matching ratios after step 2. We excluded items from "Others" categories in evaluation results depicted in Figure 2**b** and Figure 3**b** as many products in those categories were miscategorized by merchants. Titles usually better define products, so title-based approaches yield better matching ratios than description-based approaches. DBN matching ratios were better than kNN classification, but the overall combination of methods by score averaging yielded much better matching ratios than each individual method.

Both using different data sources and different models affected the increase in matching ratios. The *DBNs combined* line in Figure 3**a** shows the matching ratios when only DBN models for titles and descriptions are combined. First prediction matches by 70% with the merchant assignments, which is better than individual model results, but combining kNN models as well increases the overall matching ratio for the first prediction 4% more.

The effect of making categorization in 2-steps, instead of using big 1-step models, can be observed by comparing results from Figure 3**a** and Figure 4. In Figure 4, categorization results are presented by using direct classification in one step with features used in the first step of 2-step approach. Although 2-step approach suffered from error propagation, number of features used in the second step is much larger, hence the overall matching ratios are better. Matching ratios of individual approaches had large gaps, but the gap reduces to 1.8% if combined models were compared for first predictions.

In order to confirm the effectiveness of deep network training for classification, we compared our results with a passive-aggressive (PA) learning algorithm (Crammer et al., 2006) with kernel slicing, using OPAL tool (Yoshinaga and Kitsuregawa, 2010). Figure 5 depicts the comparison between second step DBN classification and PA classification matching results using title data only for products grouped by level-1 categories. Please note that, we also tried one-step direct classification to 28,338 categories with PA, but it was not able to be trained on single server with 96 GB main memory, because of the memory overflow problem. It can be confirmed from the figure that DBN works better for 23 categories out of 35. Categories that PA works better were usually very small. The overall performance difference was between title-DBN and title-PA is more than 10%.

Although overall performance was worse for the passive-aggressive approach, one may think that it can still be used to increase accuracy by combining with the models which we have explained above. However, the scores given by the algorithm was not suitable for combining by score averaging.

### 6.4 Sample Results

Although we provide results for matching between our predictions and merchant categories, the actual correct prediction ratios are different. There are four possibilities in case of a mismatch between a prediction and the corresponding merchant: merchant correct / prediction incorrect, merchant incorrect /

| | |
|---|---|
|  | **Product title:** Sweet Mother - Isaac Andrews<br>**Merchant Cat.:** Books, Magazines & Comics > Western Books > Books For Kids<br>**Predicted Cat.:** Books, Magazines & Comics > Western Books > Fiction & Literature |
|  | **Product title:** トヨトミ ［KS-67H］ 電子火式流型石油ストブ KS67H<br>**Merchant Cat.:** Flowers, Garden & DIY/DIY & Tools > Others<br>**Predicted Cat.:** Consumer electronics > Seasonal home appliances > Heating appliance > Oilstove > 14+ tatami (wooden) , 19+ tatami (rebar) |
|  | **Product title:** レンタル【RG87】はかまフルセット/大学生/小学生/高校生/中学生<br>**Merchant Cat.:** Women's Fashion > Japanese style > Kimono > Hakama<br>**Predicted Cat.:** Women's Fashion > Kimono > Rental |
|  | **Product title:** 夜咄用具 スキヤろうそく 大<br>**Merchant Cat.:** Kitchenware, tableware & cookware > Japanese tableware > Tea utensils > Other<br>**Predicted Cat.:** Kitchenware, tableware & cookware > Japanese tableware > Small bowl |

Table 1: Sample results. Omitted description information. Images are for reference, not used for classification.

prediction correct, both correct, both incorrect. See Table 1 for sample results for those 4 cases.

In case merchant is correct and prediction is incorrect, confidence scores for predictions are lower. In the first example in Table 1, although it is predicted as a Western book, the type of the book is mispredicted. Second example is a typical mismatch case where merchant is incorrect. Products in "Others" categories, which account for 40% of all products, have high probability of being misplaced. It is possible to predict detailed correct categories for such products. In the third example, both merchant and our prediction can be considered as correct, as the product is a hakama style kimono, but it is rental. The candle in the last example is used for tea ceramonies, but it is not a tea utensil. It is predicted as small bowl because of the explanations about candle stand in the product description, but the prediction confidence score was quite low, 0.045.

## 7 Conclusion and Future Work

In this work, we have presented a categorization system for large scale e-commerce data using product titles and descriptions. We have trained our system with a dataset having hundreds of millions of products and tens of thousands of categories. Our tests confirm high matching ratios of predictions with merchant-assigned categories. We used different data sources and different algorithms for classification, where the final scores are calculated by averaging of scores of different models' results. Exploration and evaluation of different combination techniques of results are left as future work.

Although we have utilized all textual content about products in this work, we ignored image content as it takes a lot of time to process images. Our initial evaluations utilizing image data, which we have not discussed in this work, suggest that it is possible to increase the system performance by several percent. We leave full evaluation of the prediction system with image data as a future work.

## References

A. Böcker, S. Derksen, E. Schmidt, and G. Schneider, 2004. *Hierarchical K-means Clustering*. Modlab, Universit at Frankfurt AM MAIN.

Ali Cevahir and Junji Torii. 2013. High performance online image search with gpus on large image databases. *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, 4(3):24–41.

Jianfu Chen and David Warren. 2013. Cost-sensitive learning for large-scale hierarchical classification of commercial products. In *Proc. of 22nd Conference on Information and Knowledge Management (CIKM 2012)*, pages 1351–1360.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7(March):551–585.

Yann N. Dauphin, Xavier Glorot, and Yoshua Bengio. 2011. Large-scale learning of embeddings with reconstruction sampling. In *Proc. of 28th International Conference on Machine Learning*, pages 945–952.

Vivek Gupta, Harish Karnick, Ashendra Bansal, and Pradhuman Jhala. 2016. Product classification in e-commerce using distributional semantics. *arXiv preprint arXiv:1606.06083*.

Jung-Woo Ha, Hyuna Pyo, and Jeonghee Kim. 2016. Large-scale item categorization in e-commerce using multiple recurrent neural networks. In *Proc. of 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Viet Ha-Thuc and Jean-Michel Renders. 2011. Large-scale hierarchical text classification without labelled data. In *Proc. of the fourth ACM international conference on Web search and data mining (WSDM 2011)*, pages 685–694.

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

Geoffrey Hinton and Ruslan Salakhutdinov. 2011. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, 3(1):74–91.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751.

Aris Kosmopoulos, Georgios Paliouras, and Ion Androutsopoulos. 2015. Probabilistic cascading for large scale hierarchical classification. *arXiv preprint arXiv:1505.02251*.

Zornitsa Kozareva. 2015. Everyone likes shopping! multi-class product categorization for e-commerce. In *Proc. of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1329–1333.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2267–2273.

Volodymyr Mnih. 2009. Cudamat: a cuda-based matrix class for python. Technical report, Tech. Rep. UTML TR 2009–004, Department of Computer Science, University of Toronto.

NVIDIA. 2015. Nvidia cusparse and cublas libraries. `http://docs.nvidia.com/cuda/`.

Xipeng Qiu, Xuanjing Huang, Zhao Liu, and Jinlong Zhou. 2011. Hierarchical text classification with latent concepts. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 598–602.

Rajat Raina, Anand Madhavan, and Andrew Y. Ng. 2009. Large-scale deep unsupervised learning using graphics processors. In *Proc. of the 26th Annual International Conference on Machine Learning (ICML 2009)*, pages 873–880.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

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

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

Peng Wang, Jiaming Xu, Bo Xu, Cheng-Lin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. 2015. Semantic clustering and convolutional neural network for short text categorization. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing 2015*, pages 26–31.

Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. 2008. Deep classification in large-scale text hierarchies. In *In Proc. of the 31th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2008)*, pages 619–626.

Naoki Yoshinaga and Masaru Kitsuregawa. 2010. Kernel slicing: Scalable online training with conjunctive features. In *Proc. of the 23th International Conference on Computational Linguistics (COLING 2010)*, pages 1245–1253.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *In Proc. of Advances in Neural Information Processing System 28 (NIPS)*, pages 649–657.