# An Empirical Study of Using An Ensemble Model in E-commerce Taxonomy Classification Challenge

**Yugang Jia**
Winchester, MA
yugang.jia@gmail.com

**Xin Wang**
Newton, MA
wangxin8588@gmail.com

**Hanqing Cao**
Mahwah, NJ
vauus@yahoo.com

**Boshu Ru**
Dept. of Software and Information Systems, UNC Charlotte
Charlotte, NC
boshu.ru@gmail.com

**Tianzhong Yang**
The University of Texas Health Science Center
Houston, TX
tianzhong.yang@gmail.com

## ABSTRACT

In the Rakuten data challenge on taxonomy Classification for eCommerce - scale Product Catalogs, we propose an approach based on deep convolutional neural networks to predict product taxonomies using their descriptions. The classification performance of the proposed system is further improved with oversampling, threshold moving and error correct output coding. The best classification accuracy is obtained through ensembling multiple networks trained differently with multiple inputs comprising of various extracted features.

## CCS CONCEPTS

• **Computing methodologies** → **Machine learning**;

## KEYWORDS

Multi-class classification, Imbalanced classes, Word embedding, Convolutional neural networks, Error correct output coding

## 1 INTRODUCTION

E-commerce sites provide millions of products that are continuously updated by merchants. The correct categorization of each product plays a crucial role in helping customers find the product that meets their need in many aspects, such as product searching, targeted advertising, personalized recommendation and product clustering. However, due to the large scale of the products, it is often not feasible and error prone to manually categorize the products. Therefore, large-scale automatic categorization is in great need.

## 2 RELATED WORK

The challenges in large-scale automatic categorization include: Firstly, the products are sparsely distributed in a large number of categories and the data distribution is far away from uniform distribution. Such imbalanced data could largely deteriorate the categorization performance [5, 21]; Secondly, the commercial product taxonomy is usually categorized in tree structures with thousands of leaf nodes, which adds another layer of difficulty to explore the correlations among large number of taxonomies in a hierarchical structure. Various classification methods, such as flat classification, cascade classification and probabilistic cascading have been deployed in the large-scale taxonomies. [1, 8, 13]. However, it remains to be a challenging problem due to the large data scale, data heterogeneity, and category skewness [24].

In light of the challenges, different methods have been proposed to achieve optimal classification performance. For example, Naive Bayes demonstrates the effectiveness and efficiency for classifying test documents [15, 19], but it has poor performance when some categories are sparse [23]. Support vector machines (SVMs) have been served as a well-established benchmark model for classifying e-commerce product [9]. Chen et. al proposed a multi-class SVM with an extension of using margin re-scaling to optimize average revenue loss [4]. However, SVM has been shown to have longer computing time and it only works well when the number of categories is less than five [5]. As one of the deep learning algorithms, recurrent neural network (RNN) is proposed by Pyo and Ha to deal with the multi-class classification problem with unbalanced data[8], in which the learnt word embedding depends on a recursive representation of the same initial feature space. In addition, convolutional neural network (CNN) achieves remarkable performance in sentence-level classification [10, 11, 27]. Recently, CNN has


*In: J. Degenhardt, G. Di Fabbrizio, S. Kallumadi, M. Kumar, Y.-C. Lin, A. Trotman, H. Zhao (eds.): Proceedings of the SIGIR 2018 eCom workshop, 12 July, 2018, Ann Arbor, Michigan, USA, published at http://ceur-ws.org*

been regarded as a replacement for well-established SVM and logistic regression models [28], which uses pre-trained word vectors as inputs for training the CNN models [28].

Product categorization is a hierarchical multi-class classification problem. Hence, a natural way of classifying product is to use hierarchical classification. However, hierarchical classification suffers from error propagation issue [25]. Kosmopoulos proposed a probabilistic hierarchical classification approach that predicted the leaf categories by estimating the probability of each root-to-leaf path [13]. Cevahir et.al [3] mitigated the error propagation issue in hierarchical classification and achieved better results than flat models by incorporating a large number of features in leaf category prediction.

To combat with imbalanced classification problem, cost sensitive training appears to be an effective solution. Zhihua Zhou et.al [29] show that only over-sampling and threshold moving is effective for training cost-sensitive neural networks by empirical studies. However, it becomes difficult to define costs of misclassification when there are large number of classes. A more recent paper [18] also supports similar claims.

Error-Correcting Output coding (ECOC) is another method that has been used in multi-class text classification to further improve a classifier's accuracy [6, 7]. The idea behind this approach is to encode each class label to a unique binary code with a number of digits, such that redundancy is introduced to the transformed class labels that are then used for training a supervised Machine Learning model. Even if some errors occur in the prediction of a transformed label, we may still be able to recover the correct original label by choosing the one that is the closest to the prediction. This approach helps to reduce the space of model output when there are a large number of class labels. At the meantime it also alleviates the class imbalance problem.

## 3 DATA CHARACTERISTICS

The SIGIR eCom Data Challenge is on large-scale taxonomy classification. The competition requires us to classify each product description to one of the classes accurately in the unlabeled testing data using models developed from the labeled training data. Data is provided as a training data (800 thousands of products) and a testing data (200 thousands of products). The training data has one column having production description and one column having product categories. There are 14 principal product categories represented by numbers, all but one of which have secondary categories of levels ranging from 1 to 7, or 1 to 8 levels in total, which results in a total of 3008 classes. Each of such combination of principal and secondary categories formulates a string of numbers that represents a class, eg "3625>4399>1598>3903", "2296>3597>689".

There are many characteristics and potential challenges in this dataset. At the class level, there exists large variation in the number of samples in each class, resulting in a largely unbalanced dataset (See Table 1). Out of the 3008 classes, 19 classes have only one product, and 1484 (49.3 percent ) classes have less than 25 products, which sums up to be 13,618 products, or 1.70 percent of all products. On the other hand, 12 classes have more than 10,000 products, which sums up to be a total of 256,689 products, or 32.1 percent of all products. The top 3 largest classes include 69915, 30146, and 25481 products respectively (See Table 2). The classes with many samples might embrace both the richness as well as the diversity of data that could result in the inter-class distance being closer than the intra-class distance. The classes with only a handful of samples are expected to be hard to classify accurately in the testing data.

The product description is a mixture of letters, words, numbers and other ASCII characters. At the product description level, there appears to have at least the following challenges. The first challenge is that the difference between data samples in the same class might be larger than that of samples belonging to different classes to the extent of seemingly mislabels. For example, both "Mont Blanc Mb Starwalker Men Eau De Toilette Edt 2.5Oz / 70Ml" (a type of perfume) and "Humminbird Pc11 Power Cord" (an electronic device) are in the category of "3625>4399>1598>3903", while "Creed Green Irish Tweed Eau De Perfum For Men - Small Size 1oz (30ml) ", also a type of perfume, is in category "3625>3005". The second challenge is the categories under the same parent categories (or principal category) are correlated, which would enforce the challenge above. A third challenge is that one abbreviation could have different meanings, for example, "hp" could be "horsepower" or "Hewlett-Packard", "mb" could be "mega byte" or "marble". The fourth challenge is the large variation in the number of words in each product description, ranging from 1 to dozens. While a short description might not provide sufficient information, such as a single word product "Bonjour", a long description might include too much details that the relevant information might be hidden, such as "fosmon 2100mah dual port usb rapid car charger for apple iphone 5c/5s/5/4s/4, samsung galaxy note 3/2, s5/s4/mini/active/s3, lg g3/lg g2/ g2 mini, google nexus 5/4, blackberry z10/z30/q10, htc one (m8), motorola moto e/moto x/moto g, nokia lumia 1020".

Due to the above data characteristics, data preprocessing that filters out noise and keeps relevant information is systematically designed and implemented and various modeling strategies are tested as described in the next sections.

**Table 1: Data Characteristics - Category Size**

| log10(Categorize Size) | Count | Category Size Range |
|---|---|---|
| 0 | 893 | < 10 |
| 1 | 1372 | (10 - 100] |
| 2 | 623 | (100 - 1,000] |
| 3 | 108 | (1,000 - 10,000] |
| 4 | 12 | > 10,000 |

**Table 2: Top 3 Categories**

| Category | Number of Products |
|---|---|
| 2199>4592>12 | 69915 |
| 3292>3581>3145>2201 | 30146 |
| 4015>2337>1458>40 | 25481 |

## 4 PREPROCESSING AND FEATURIZATION

In this section we introduce how the product descriptions are preprocessed and the features extracted to train our model.

### Preprocessing

As we described above, the product description contains noise. We mitigate the noise in a trial-and-error way in order to find the balance of signal noise ratio. As a result, we apply the following procedure. We first convert all letters to lower case, and replacd special characters such as parentheses except single hyphen, and repeated characters such as multiple hyphen or dots with spaces. We then unify physical units of "in", "ft", "hp", "ml","oz" and etc that follow a number to "nnnhp", "nnnml", and "nnnoz", respectively. For example, "3.4oz", "3.4 oz" or "3.4-oz" would become "nnnoz". Finally, we remove dash (-), standalone numbers, and extra white spaces. As a result of the preprocessing, distinct words are reduced from 670K to 160K.

### Word-level and character-level embeddings for system input

We use three different sets of word embedding to create vector presentation of words, each of which is as described in more details below. The resulted vectors of each word in each product description are then concatenated (i.e, column-bind) as word embedding features. As shown in Fig. 1, word embeddings are concatenated as the input to train a CNN model. Three sets of word embeddings generated by varying methods are used as three separate inputs.

***Word embeddings pre-trained on Google News.*** The pre-trained word embeddings are trained on part of Google News dataset (about 100 billion words) by using word2vec algorithm [20]. The representation of a word is learnt to be useful for prediction of other words in the sentence. The model contains 300-dimensional vectors for 3 million words and phrases.

***Word embeddings pre-trained on product descriptions.*** We also learn word embeddings from all product descriptions by using word2vec algorithm implemented within Gensim [22], which is an unsupervised learning model. The dimension of word vector is set to be 50 and we train the word2vec model using CBOW algorithm [20] with 5 epochs. In each epoch, we initially set learning rate to be 0.025 and let it linearly decay to 0.0001.

***Word embeddings learnt in training.*** Firstly, we use one-hot coding to represent an individual word in a product description. Then we learn a weight matrix in an embedding layer of proposed model to transform each word into a word vector having 50 dimensions. The word vectors transformed from one-hot coding are used as the input to train the first convolutional layer.

***Character-level embeddings learnt in training***. Besides word embeddings, we also use character level embeddings learnt in training, where the learning approach is similar to the approach of learn word embeddings in training. We use one-hot coding to represent each unique character in the raw texts rather than each word.

### Named entity and part-of-speech tag features

The appearance of named entities might be associated with certain product categories. For example, locations, landmarks, and famous people are often prevalent in categories such as branded perfume, books and movies, while organization names might be seen more commonly in electronic products such as Apple and HP. In addition, individual words associated with the named entities might be rare and therefore filtered by our word frequency requirement, but they could be informative. For example, the word "Beethoven" occurs in only one product description, but it is the name of a famous musician and it carries strong information about what type of product it could be. Therefore, we use Stanford CoreNLP package [17] to extract 23 types of named entities from the product description, namely cause of death, city, country, criminal charge, date, duration, email, ideology, location, misc, money, nationality, number, ordinal, organization, percent, person, religion, set, state or province, time, title, and URL. For each product description, we count the number of words in each entity type and normalize values by the length of sentence, resulting a 23-dimension feature vector representing the distribution of named entity types. Additionally, we generate a 36-dimension feature vector for the distribution of part-of-speech tags, which was also identified by Stanford CoreNLP.

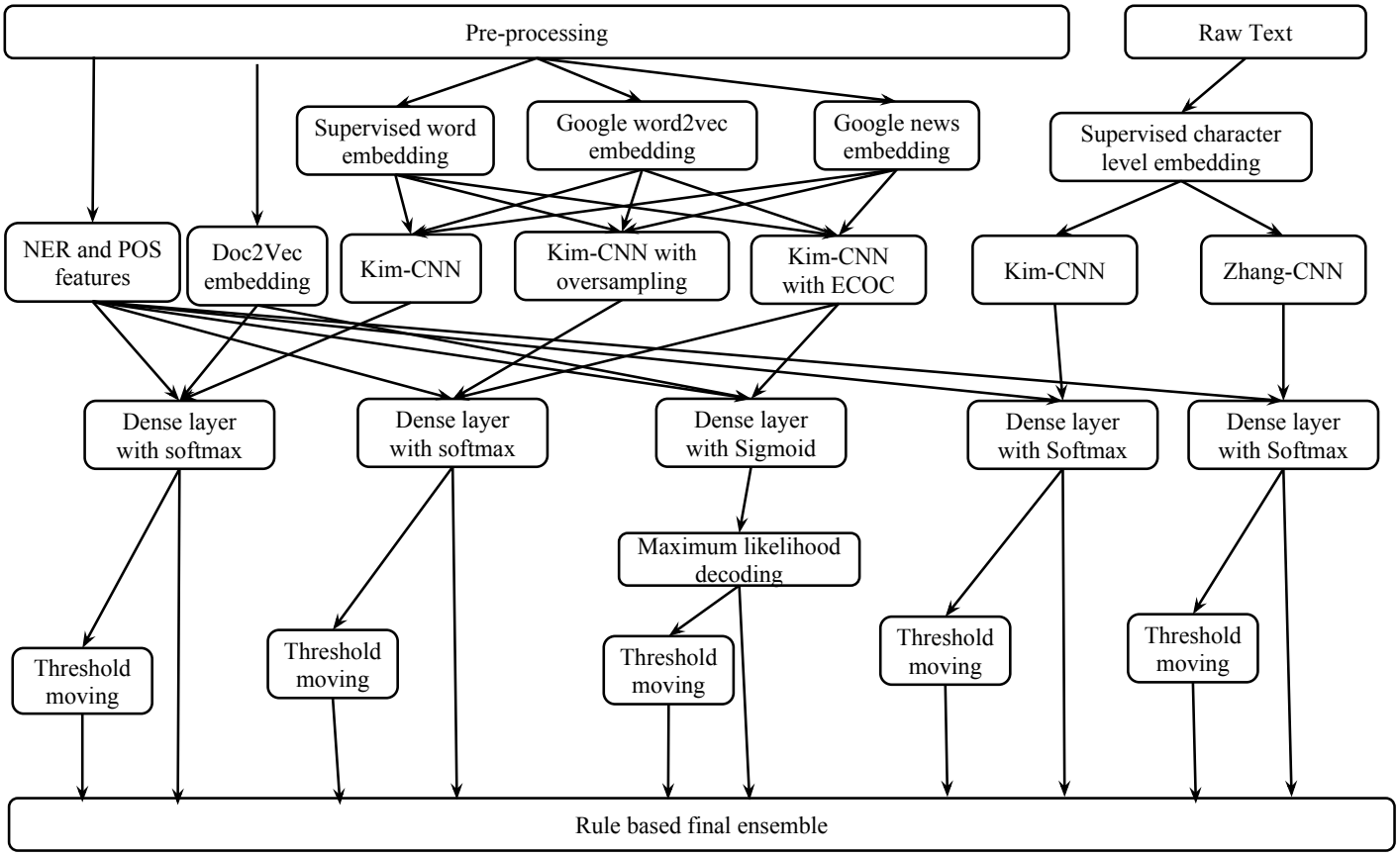
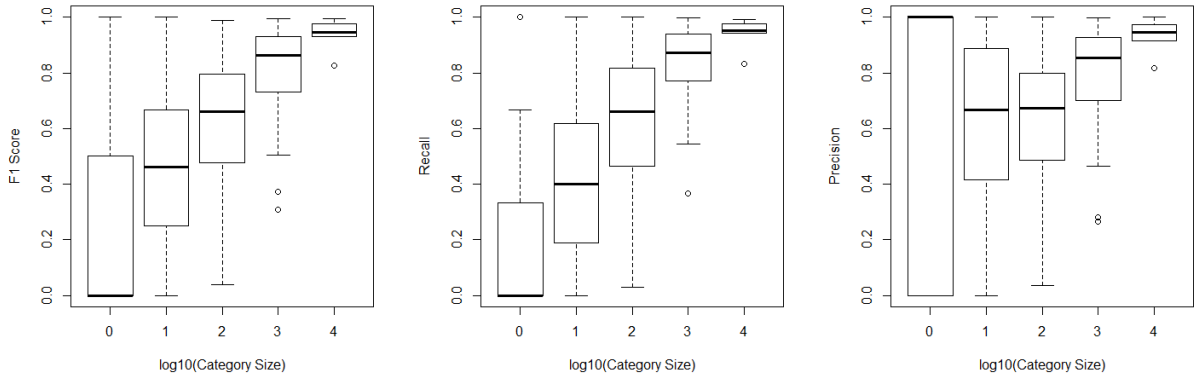



**Figure 1: Illustration of analysis system for product title categorization**




(a) (b) (c)

**Figure 2: Classifier performance (F1-score, recall and precision) as a function of Category Size**

### Creating sentence level representation

We use doc2vec algorithm proposed in "Distributed Representations of Sentences and Documents" by Quoc Le et al. [14], to create another set of features for product descriptions. The algorithm modifies word2vec to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. We represent each product description by a 50 dimensional feature vector and we train the doc2vec model with 5 epochs.

## 5 PROPOSED MODEL FOR PRODUCT TAXONOMY CLASSIFICATION

An overall architecture of our system is shown in Fig. 1. We introduce the key components in the proposed model in the following sections. We train five models separately with each one using multiple data inputs and varying setups. For making the final predictions, we ensemble the predicting results from these models. All the models are trained using Adam algorithm [12] with a learning rate 0.001.

### Kim-CNN architecture

Several recent studies have examined CNN models for text classification tasks and reported CNN based models achieved outstanding performance [2, 11, 16]. We adopt the design of Kim's CNN model to extract informative patterns from the word embedding representation of input data [11]. Kim's model contains paralleled convolution filters of three different kernel sizes to protrude informative patterns in a sub area of the input data. The kernel size ($k$) of filter determines the magnitude of the sub area in the case of text, that is the number of continuous words in a sentence. By mixing convolution filters of three continuous sizes ($k-1$, $k$, and $k+1$), the network can learn patterns in the sentence at three different scales. This design is similar to combining n-gram features of different scales (e.g., unigram, bigram, and trigram). The max pooling filter scans through outputs of convolution filters and preserves only the max value in each area. This operation washes out information that is less relevant to the classification task and reduces the dimensionality of features extracted by convolution filters. The output is further processed by one layer of fully-connected neurons to condense output matrix.

Based on the Kim-CNN architecture proposed in [11] that used kernels in the sizes of 3, 4 and 5, we insert one more kernel with size being 1 to capture the unigram features and we apply a batch normalization layer followed by a dropout layer after the fully-connected layer of a standard Kim-CNN, then the output is further processed by a second fully-connected layer to yield final outputs.

### Oversampling and threshold moving

We adopt an oversampling strategy to improve the model's performance on imbalanced data. In training one of the five networks as shown in Fig. 1, we initially draw 256 examples from training data to form one batch. We add 768 more examples to the batch of data by duplicating those ones whose classes having less than 5000 examples in the entire training set. Within the 768 duplicated examples, we further define three categories of classes according to the class sizes in training set: classes with 1000 - 5000 examples, 100 - 1000 examples and less than 100 examples. The proportion of three categories of classes should follow 1:2:4 to form the total 768 oversampled examples. In the end we have 1024 examples in each batch.

We also apply a threshold moving strategy to adjust model predictions to alleviate data imbalance problem. The original output of our models, which are the probabilities for each class, are divided by the class sizes before we yield final class labels. This strategy will reduce the probabilities yielded on large classes but increase those on smaller classes. Our experimental results show that this strategy is able to further improve model performance.

### Error correcting output coding

To leverage the hierarchy of class labels and explore label correlation, we also used error correcting output coding [6, 7]. For our cases with large amount of classes, we create a unique binary coding for each taxonomy. The coding for each taxonomy level is different, for example, the first level taxonomy has 3 digits while the second level taxonomy has 4 digits. Then we concatenate all codes corresponding to multiple taxonomies to form the new output coding. The original taxonomy prediction problem is transformed into a multi-label learning problem using the encoded labels. The longest sequence of taxonomies contains 8 taxonomy codes. We pad those encoded labels of taxonomy sequences less than 8 with zeros.

### Model ensembling

The models that are trained for ensembling are shown in Fig. 1 and descriptions of these models are as the following:

- Model 1 : The last fully connected layer uses the input including: (1) the output of Kim-CNN networks; (2) NER features; (3) doc2vec features.
- Model 2 : The last fully connected layer uses the input including: (1) the output of one Kim-CNN model trained with upsampling; (2) the output of the other Kim-CNN trained with ECOC; (3) NER features.
- Model 3 : The last fully connected layer uses the input including: (1) the output of Kim-CNN networks trained with ECOC; (2) NER features and (3) doc2vec

features. The output of Model 3 is decoded from the yielded multi-label predictions in a way of calculating the likelihood of predicting each individual class label firstly and then we assign the original class label with the highest computed likelihood to the example.

- Model 4 : We train a Kim-CNN model using character level embeddings of raw texts. The last fully connected layer uses the input including: (1) the output of Kim-CNN; (2) NER features and (3) doc2vec features.
- Model 5 : Different from Kim-CNN that concatenates the feature maps generated from different window sizes in parallel, we train a new CNN model with six layers sequentially, which is referred as Zhang-CNN [26]. The last fully connected layer uses the input including: (1) the output of Zhang-CNN; (2) NER features and (3) doc2vec features. We use the similar hyperparameters that have been proposed in their paper with fine tuning.

The output of each model is also adjusted by adopting threshold moving such that each model provides 2 versions of predictions that are in terms of probabilities, with or without threshold moving. We derive an ensembling procedure to combine multiple model predictions. Firstly, we select the predictions from Model 1, 4 and 5 to form a initial set of candidate predictions, containing 6 versions of predictions. We repeat training Model 2 for 3 times with random data shuffling and oversampling to provide 6 more versions of predictions and add those to the set. We adopt 3 different ways of creating output codings and repeatedly train Model 3, to obtain 6 more versions of predictions. We observe that using even more predictions beyond 6 versions from Model 2 or 3 will not further improve the overall performance. In the end we implement a majority voting over all versions of predicted labels within the candidate set to predict the final labels. When there is a tie, we choose the label with the highest averaged probability.

## 6 PARAMETER TUNING AND ERROR ANALYSIS

For training the proposed model, hyper-parameters such as number of filters among {256, 512, 1024} for each convolutional layer, dropout rate among {0.1, 0.2, 0.3, 0.4, 0.5} and regularization parameter for convolutional and fully connected layers among {$1\times10^{-6}$, $3\times10^{-6}$, $1\times10^{-3}$, $3\times10^{-3}$, $1\times10^{-1}$} are tuned using a hold-out set in the training data. We adopt the same combination of window sizes 3, 4, 5 as what was used in Kim-CNN [11], otherwise we add a new window spanning only one word which was helpful to predict the taxonomy correctly with selected a few single words. We select an architecture with the best performance. The adopted hyper-parameters for Kim-CNN model are as the following :

**Table 3: Testing Results**

| Metric | Testing-Stage1 | Testing-Stage2 |
|---|---|---|
| Precision | 0.8545 | 0.8528 |
| Recall | 0.8172 | 0.8172 |
| F1 | 0.8278 | 0.8295 |

- Number of filters : 1024
- Window size of Kim-CNN: {1, 3, 4, 5}
- Dropout rate : 0.5
- Regularization parameter : $3\times10^{-6}$

In the tuning process, firstly we fix the number of filters to be 256 and we tune other parameters. Then we increase the number of filters to 512 and 1024 and the latter one shows a better performance. We also notice that a higher dropout rate 0.5 would help to get the best performance. Besides, our model is quite sensitive to the regularization parameter in the sense of converging speed. Once we increase it from $3\times10^{-6}$ to $1\times10^{-3}$, the model converges much slower but with no increase in performance.

Our error analysis shows that the classifier works well for categories whose sample sizes are big, for example, more than 1000 cases, and not so well for smaller categories. Figure 2 shows the performance on a random validation dataset (20 percent of training data) using ensemble of models trained with over-sampling, threshold moving and error correcting output coding techniques. The result inspires us to adopt various sampling strategies aiming to increase the sample size for those small categories.

## 7 RESULTS AND DISCUSSION

The models are firstly trained on 80% of the given 800K training samples and validated on the rest 20% of data to tune the hyperparameters. Then the models are retrained with fixed hyperparameters following an early stopping strategy in training using all given 800k samples and tested on 200K samples with unknown labels (Table 3). We achieve a good performance, with a F1 score of 0.8295.

We observe that feature engineering is particularly important in further improving performance using CNN. We went through a path that as we added more relevant but somehow different features, for example the NER features, the performance was improved accordingly. We also observe that there is considerable performance difference among different models. The performance of models based on character level embedding are not as good as the others. However, it helps to improve the overall performance of ensemble model.

The threshold moving method is very helpful to increase the precision of individual model, which is critical in final ensembling. The oversampling and ECOC algorithms can

add additional randomness and improve the performance of ensembling model to certain extent.

## ACKNOWLEDGMENTS

The author would like to thank the organizer of SIGIR 2018 eCom Data Challenge (Rakuten Institute of Technology Boston (RIT-Boston)) for their support.

## REFERENCES

[1] Rohit Babbar, Ioannis Partalas, Eric Gaussier, and Massih R Amini. 2013. On flat versus hierarchical classification in large-scale taxonomies. In *Advances in Neural Information Processing Systems*. 1824–1832.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[3] Ali Cevahir and Koji Murakami. 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*. 525–535.

[4] Jianfu Chen and David Warren. 2013. Cost-sensitive learning for large-scale hierarchical classification. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 1351–1360.

[5] Pradipto Das, Yandi Xia, Aaron Levine, Giuseppe Di Fabbrizio, and Ankur Datta. 2016. Large-scale taxonomy categorization for noisy product listings. In *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 3885–3894.

[6] Thomas G. Dietterich and Ghulum Bakiri. 1995. Solving Multiclass Learning Problems via Error-correcting Output Codes. *J. Artif. Int. Res.* 2, 1 (Jan. 1995), 263–286.

[7] Rayid Ghani. 2000. Using Error-Correcting Codes for Text Classification. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 303–310.

[8] 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, 107–115.

[9] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer, 137–142.

[10] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

[11] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980

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

[14] Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*. JMLR.org, II–1188–II–1196.

[15] David D Lewis and Marc Ringuette. 1994. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, Vol. 33. 81–93.

[16] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep Learning for Extreme Multi-label Text Classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 115–124.

[17] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.

[18] Maciej A. Mazurowski Mateusz Buda, Atsuto Maki. 2017. A systematic study of the class imbalance problem in convolutional neural networks. *arXiv:1710.05381* (2017).

[19] Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, Vol. 752. Citeseer, 41–48.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg 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 - Volume 2 (NIPS'13)*. Curran Associates Inc., USA, 3111–3119.

[21] Michael Pazzani, Christopher Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. 1994. Reducing misclassification costs. In *Machine Learning Proceedings 1994*. Elsevier, 217–225.

[22] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50.

[23] Dan Shen, Jean-David Ruvini, Rajyashree Mukherjee, and Neel Sundaresan. 2012. A study of smoothing algorithms for item categorization on e-commerce sites. *Neurocomputing* 92 (2012), 54–60.

[24] 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*. ACM, 1921–1924.

[25] Carlos N Silla and Alex A Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 22, 1-2 (2011), 31–72.

[26] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15)*. MIT Press, Cambridge, MA, USA, 649–657.

[27] Ye Zhang, Stephen Roller, and Byron Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. *arXiv preprint arXiv:1603.00968* (2016).

[28] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* (2015).

[29] Zhi-Hua Zhou and Xu-Ying Liu. 2006. Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem. *IEEE Trans. on Knowl. and Data Eng.* 18, 1 (Jan. 2006), 63–77.