Extract It! Product Category Extraction by Transfer Learning

Harsh Raj, Aakansha Gupta, and Rahul Katarya

Abstract The categorization of e-commerce products is salient as its quality directly affects search, recommendations, and related personalized services. Putting the products into the best possible category in a hierarchical manner divided by subcategories is challenging due to the vast range of products creating complexity in product information to select suitable categories. Earlier research proves inefficient with a smaller dataset, so we propose a model to extract the most relevant information from the product description and a pre-trained vocabulary to transform it into subcategories for the prediction of the product category tree. A minute inaccuracy, in this case, can hamper customer satisfaction while searching for the products in the desired categories. To tackle these challenges, we merge the areas of machine learning and deep learning with natural language processing (NLP) to propose a multi-level-based product categorization model. We implemented a selective approach starting with the product name being the node of the tree to recursively form a hierarchical tree structure, searching and extracting subcategories from the product description and a pre-trained vocabulary, which eventually made us predict the most relevant categorization of products. Basically, it is a process of extraction of subcategory levels from the product description and a pre-trained vocabulary built by transfer learning. The proposed model was tested on a Flipkart product dataset containing 20,000 products with several features describing the product. The cosine similarity between the predicted and the given product category tree computed was 0.77, which takes the accuracy to 86%.

Keywords Categorization · Word embeddings · Word extraction · Cosine similarity · Classification · E-commerce

H. Raj (✉) · A. Gupta · R. Katarya
Big Data Analytics and Web Intelligence Laboratory, Department of Computer Science and Engineering, Delhi Technological University, New Delhi, India

R. Katarya
e-mail: rahulkatarya@dtu.ac.in

X.-Z. Gao et al. (eds.), *Advances in Computational Intelligence and Communication Technology*, Lecture Notes in Networks and Systems 399,
https://doi.org/10.1007/978-981-16-9756-2_9

1 Introduction

With the rapid growth of e-commerce, giant companies like Amazon, eBay, Flipkart, and Rakuten have listed millions of products on their Web sites which are sold by thousands of merchants. From the perspective of a customer, e-commerce means buying products from the comfort of their home To organize products so that customers navigation becomes easy through the searches, building product taxonomy is equally important. “mobile’s accessories>> mobile accessories>> car accessories>> mobile holders>> adroitz mobile holders>> adroitz premium phone socket holder for htc one” is an example for such category trees. With billions of global digital buyers offering countless numbers of products across a wide range of categories, it is required to categorize the products according to their taxonomy by proposing an efficient algorithm to be customer centric and engage into more businesses through the online market. A product taxonomy puts the products into a systematic structure according to the information about products and their use, making it organized while presenting to the users. This standardized arrangement is also valuable for data presentation and storage. Accurate product categorization helps users quickly find the set of products they are looking for. Product categorization in a hierarchical manner helps users get the right results while browsing a category. It also refines the result when a user is doing a keyword search; the product taxonomy helps clarify or disambiguate a search result set based on the categories. Before the advent of machine learning techniques in NLP, merchants and companies were required to manually assign each product to their respective categories, which is a tedious and time-consuming task with error-prone results. Moreover, manual categorization may not be accurate while assigning products to categories as a product can be listed in different ways by different merchants, making this approach inconsistent. Automatic categorization with machines using a consistent algorithm helps to solve these problems. In general, a product taxonomy makes the data collection process more efficient, ensuring Web sites to support features like search refinement, category browse, and specifications while displaying and comparing products which eventually maximizes conversions and profits.

The next question is: How do you improve automatic product categorization models? This question has troubled researchers for a long time, but the best way out we can get is to use machine learning in analyzing and pertaining to the needs of automation. At this scale, even a small increase in the volume of products can result in large calculations to reclassify everything. The investment into more efficient algorithms for product classification is rising every day with the increasing adoption of e-commerce platforms. Newer solutions are trying to determine categories based on:

- Rough product titles and description
- Images of products
- Parsing metadata.

1.1 Our Contribution

To the best of our knowledge, this paper deals with the first kind of product categorization using product titles and its description without explicit training, which can be used even with a smaller dataset or with less information. There are two commonly used approaches; one is the single step classification method which considers all the subcategories as one category, and the other is called the stepwise classifier, which treats different subcategories as different levels. The latter involves a recursive approach to get every subcategory, respectively. Separate model calls are initiated to predict every level of taxonomy in a hierarchical manner. Despite a better accuracy given by the model, it increases the number of times model training has to be performed.

In the past, Kozareva [1] proposed a product categorization approach with derived features such as linear discriminant analysis (LDA) [2], N-gram, Bi-gram, etc. In categorizing the products, the authors experimented with two algorithms: one-against-all (OAA) and error correction tournament (ECT). Recent experiments had been evaluated on a large-scale dataset with as many as 18 million product titles and an extensively larger category list [3]. In our experiment, we have used a transfer learning-based feature extractor to get the most similar words from the product description and a pre-trained vocabulary, starting with product title as the lowest level. We have noticed that the product description contains words that are either in the category tree or words similar to it are present. Hence, our proposed approach of extracting similar words from the description or from the pre-trained vocabulary has proved efficient, giving breakthroughs even with relatively smaller data containing less information as compared to past researches. Besides this, there are problems that hamper fair comparisons with other proposed approaches; most of them do not have publicly available implementations with an evaluation done using private datasets.

2 Related Works

Numerous single-step classifiers have been introduced in the past to use them for product categorization. Yu et al. [4] examined a large number of word-level features using several primitive techniques (e.g., n-grams) and did classification through support vector machine (SVM; Cortes and Vapnik [5]). In the paper [6], Sun et al. use primitive classifiers (e.g., Naive Bayes, k-nearest neighbors) and reduced the errors via crowdsourcing manual labor. Ha et al. [7] and Xia et al. [8] used deep learning to produce word vectors from the product attributes (e.g., product title, product image, merchant ID) and utilized their representation in product categorization.

To improve the categorization results, several stepwise classifiers have also been used. Shen et al. [9] used primitive classifiers such as Naive Bayes and k-nearest neighbors and assigned the product to a leaf node via SVM. Similarly, Das et al. [10] used gradient boosted trees [3] with convolutional neural networks (CNN) for it.

Cevahir and Murakami [11] used deep belief networks [12] and k-nearest neighbors (KNNs) in their approach consisting of two steps.

The paper [13] was among the first to use supervised learning to categorize products into known categories. Their approach simply grouped the product into a particular category with similar products given their informational details (e.g., description, name). They analyzed product information considering information distribution to build features to give input to the classifier. Their results showed significant improvement on results obtained through manual labor. Substantially, they were certainly able to pioneer automation in the categorization of products.

Kozareva [14] introduced a categorization technique with a error correction tournament (ECT) and one-against-all (OAA) OAA reduced the multi-classification problem into multiple binary classification tasks by iteratively classifying each product title for a category and comparing it against all other categories. ECT also reduced the problem to binary classification but employed a single-elimination tournament strategy to compare a set of K players and repeated this process for O(log K) rounds to determine the multi-class label.

In this paper [15], the authors proposed a different paradigm via machine translation. In their model, they translated a product's textual description into a sequence of tokens representing product taxonomy in a root-to-leaf path. They did the experiments on two very large real-world datasets and concluded by showing that their approach was better than traditional classification-based models.

In the study [16], researchers at WalmartLabs compared hierarchical models with flat models for categorizing products. The team deployed deep learning to fetch features from each product to build a product impression. They applied a multi-CNN and multi-LSTM-based approach for this stepwise classification task.

Despite various breakthroughs in the field, there has not been much consideration in improving the models to produce state-of-the-art results with considerably reducing computational complexity even with smaller datasets as compared to the existing approaches.

3 Dataset

The dataset used is a Flipkart e-commerce sample consisting of 20,000 products with their corresponding features, listing the product description and resulting categorization tree in a comprehensive CSV document separating multiple product features. We took a relatively smaller dataset to prove our model's consistency without extensive training.

The dataset includes several other features, but most of them would not contribute to the accuracy of our prediction but could increase the complexity and inefficiency of the task. We removed duplicate product listings leaving us with Flipkart product titles and descriptions which are related to their multi-level category labels. As inferred from the data of most e-commerce sites, the product description and titles contain most of the words present in the category tree of the respective products.

Subsequently, we took product name and description as the features in our approach containing the information required for the prediction of the taxonomy tree. During training and evaluating our approach, we had split the data into a training set of 16,000 samples and a test set consisting of 4000 samples. We predicted the product taxonomy up to five hierarchical levels.

4 Methodology

We explored various approaches for the tasks and looked more closely at the correlation between the features and the target. We found that the correct extraction of words from the features is the key in predicting the product category.

This section deals with the methodology of our model, starting with preprocessing and selecting the important features; our proposed model first preprocesses the textual features by removing punctuations and numbers and converting words to their lowercase. After conducting experiments with other preprocessing techniques such as stemming and lemmatization, we saw a relative decline in the final accuracy of our model; it might be because the predicted product categories were losing their local context.

4.1 Preprocessing

In our proposed approach, we applied preprocessing steps on product title and description. This is done to extract most of the information about the product and to increase the materiality of the texts describing it in order to predict the taxonomy tree, which may be lost without this step.

Number removal: On noticing the features describing the products, we realized that numbers had very little relevance to a given product category, so we decided to remove them, considering that the discrete distribution of numbers would only adversely affect the given categorization results. We believe that it is not possible to fully capture the information the numbers try to pursue for a given category. For example, 1–9, 3, 6, 0.3, we expect the fact that for certain products, specific numbers may show up more relevance, whose inconsideration will eventually affect the categorization. But to get the vectors of the word, number removal is a necessary step.

Removing punctuation: It was done as a result of manual examination of the data through which we noticed that humans have various options for choosing words and phrases, as a word may have multiple accepted forms. Also, there are punctuations in between the phrases, including commas, parenthesis, question mark, exclamation points, and others. Also, in most cases, concatenated words are used in different senses. Ex: won't, wont, 2D, 2-D, good., good? Removing punctuation resulted in

being one of the most effective feature normalization techniques used to get a clearer context of information through words.

Lowercasing: According to the standards of the English language, sentences should have the first word capitalized. Humans, while writing, often capitalize some words ambiguously, depending on their intent, interpretation, or choice of capitalizing acronyms. Precisely, this cannot always be a useful scaling technique, as it can be contradictory, especially when dealing with product names vs. generic objects at the same time—e.g., Home, home, CD, cd, Windows, windows. As observed, lowercasing does not help much, most likely because of the shift in meaning caused by lowercasing occurrences such as "Schools are good" to "Schools are good" are offset by "Windows XP" to "windows xp", which have very different meanings.

4.2 Feature Selection

As our dataset contains a comprehensive set of features to describe products and their usage, not all of them will be useful to be applied for classification or differentiation between categories. We only considered those features which contain most of the information about the product, so we decided to use feature selection to achieve this. We first looked at the distribution of words of subcategories of the product category tree in every feature using the Jaccard index [17] and prioritized the features which could be considered as important for the task. The value of the Jaccard index was maximum for the following features: description (0.31) and product name (0.42). Therefore, we conducted our experiments using these two features, i.e., description and product name.

Using the product description as the most informative feature increased the accuracy of our model. When we also use the least informative features, the accuracy drastically reduces to get halved. This might be because the mutual information would not be capturing the least informative feature due to the lack of distribution of information of categories in them.

The product name feature refers to the name of the products, which is crucial in determining the taxonomy tree as it will fill the lowest level in the product taxonomy tree. Product name will be used in the extraction of other subcategories as it will initiate the recursive extraction process from the product description and a pre-trained vocabulary.

The target feature for our dataset is the product category tree. It is divided into hierarchical sublevels (subcategories) with the use of " >> ", punctuation which got removed in the preprocessing step. The uniqueness of our proposed model is that it predicts the taxonomy tree just by modifying the extraction of words from the description and a pre-trained vocabulary with finer accuracy even without explicit training.

4.3 Predicting the Taxonomy Tree

This section will introduce our proposed method for extracting the taxonomy tree to get a hierarchical level-based categorization of products, starting with the product name as the first level(subcategory) and recursively searching its most similar(suitable) words from the product description and a pre-trained vocabulary.

4.3.1 Feature Representation

There is a need to get the vectors of words to be used in any machine learning model; in our paper, we used Word2Vec [18], GloVe [19], FastText [20], Paragram [21] word embeddings for the purpose. The method used transfer learning where the pre-trained embedding model was used to get the vectors of texts without explicitly training it for the task.

Word2Vec: Word2Vec [18] being a very popular pre-trained word embedding model was developed by Google. They have also been applied to various tasks such as recommendation systems, knowledge extraction and discovery, and different problems related to text analysis. Word2Vec model architecture contains a feed-forward neural network with one hidden layer. Therefore, it is also known to have a shallow neural network architecture. For the purpose of vector conversion of words, we pre-trained the Word2Vec [18] model, producing a sequence of 300-dimensional embeddings for various words and phrases.

GloVe: GloVe [19] is an unsupervised learning algorithm used for word vectorization. While training, a co-occurrence matrix is created which represents the linear substructures of the word in its vector space w.r.t. the occurrence of other words. The objective for training is to minimize the difference in dot products of word vectors and the logarithm of the probability of co-occurrence of those words. Subsequently, the ratio in a logarithm is same as the difference of the logarithms, so this objective is ultimately the ratios of co-occurrence probabilities with vector differences in a space of vectors. Because of this, the word vectors perform extraordinarily on tasks related to word analogy and correlation. For the purpose of word vectorization, we pre-trained the GloVe [19] model producing a sequence of 300-dimensional embeddings for words and phrases.

FastText: Abovementioned methods ignore the morphology of words, while creating vectors for each word in the vocabulary. The main limitation comes with syntactically rich languages as these kinds of languages have larger vocabularies and a relatively complex syntax, yielding lower-quality representations. Bojanowski et al. [20] in 2016 introduced a model on this principle of Skip-gram called FastText which perceives each word as a bag of n-gram characters. It learns the embeddings for each word by taking the sum of its corresponding n-gram embeddings. For text vectorization, pre-trained FastText [20] model was used which produced a sequence of 300-dimensional embeddings for each word and phrase.

Paragram: Proposed by Wieting et al. [21] is a model for word embedding for learning with pairs of paraphrase from PPDB [22] database. Precisely, this method encodes phrases to a vector space by minimizing the difference in the cosine similarity in the space with the scores of pairs of paraphrase phrases. For text vectorization, pre-trained Paragram [21] model was used which produced a sequence of 300-dimensional embeddings for each word and phrase.

The pre-trained model used in word vectorization is able to produce a large vocabulary, which is used in refining our model's accuracy in predicting the product category tree.

4.3.2 Weighted Word Embeddings

This section deals with the approach devised to find text embeddings for product names. Most of the product names are not in the form of words but are represented as texts containing words, and to treat the product name as the first level in the product category tree, we have to represent it as a single vector.

To calculate the text embeddings, our model takes the term frequency-inverse document frequency (TF-IDF) [23] mean of word embedding. For each product name, we calculated a weighted mean of the word vector obtained using the Word2Vec [18], GloVe [19], FastText [20], and Paragram [21], where the weights are the TF-IDF value of the word.

4.3.3 Evaluation Metric

We have used cosine similarity to measure the similarity to recursively produce the product taxonomy tree by extracting the most similar words. Cosine similarity [24] is chosen as the measure as it gives accuracy based on the contextual as well as semantic relations between the input vectors. It measures the similarity between two vectors by taking the cosine of the angle to determine the degree of closeness. For NLP tasks, it is mainly used to measure similarity in textual data. The similarity score ranges from -1 to 1, with the latter being a perfect score.

To select a word (from the description or the pre-trained vocabulary) as the next subcategory in the taxonomy tree, the similarity score between the vectors of the previously predicted subcategory and the product description should be greater than a predetermined threshold value. We also marked the words so that no subcategory gets repeated. Searching from the pre-trained vocabulary over the product description is introduced to tackle the unavailability of similar words.

If our algorithm is not able to search for any word in the text (description) whose similarity score tends to be greater than the predetermined threshold value, then we took the most similar word from the vocabulary, which is not equal to any of the preceding subcategories as the next level in the taxonomy tree.

On exploring, we found that most of the product category trees were having five subcategories on average. So we repeated the above steps 5 times to get the product

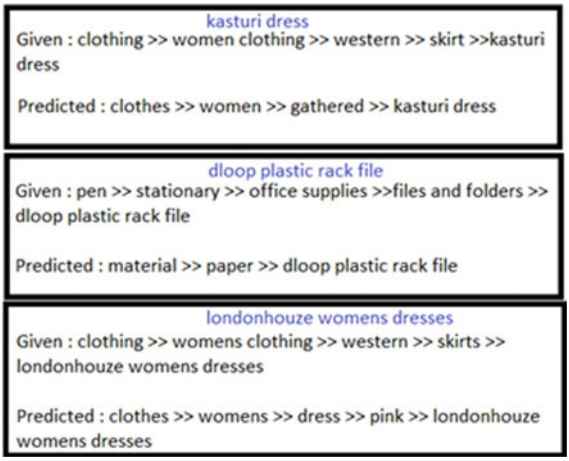

Fig. 1 Comparison of predicted against given product category trees

taxonomy consisting of five hierarchical levels, which predicted efficient results with relatively fast execution and low complexity (Fig. 1).

5 Results

This section describes the results for the sets of experiments conducted. As the product taxonomy tree should be evaluated both on semantic and syntactic similarity, the final evaluation was done on both cosine similarity [24] and F1-score. The model training is done to predict taxonomy tree which shall minimize the loss of product names, so a combination of both the scores gives a better scheme for evaluating the method.

The comparison in accuracy for different feature representations techniques (Word2Vec [18], GloVe [19], FastText [20], Paragram [21]) was done based on the cosine similarity [24] and the F1-score, and the results are shown in Table 1.

While evaluating, we noticed that the precision is always higher than recall which means that the result does not contain out of context words.

Table 1 Results comparing paragram, Word2Vec, GloVe, and FastText embedding techniques

Feature representation	Cosine similarity	Precision	Recall	F1-Score
Paragram	0.78	0.77	0.71	0.74
Word2Vec	0.83	0.79	0.71	0.75
GloVe	0.86	0.85	0.74	0.79
FastText	0.86	0.88	0.79	0.83

6 Conclusion

In the paper, we have presented a product categorization model on the basis of extraction of sub categorical levels from the product description and a pre-trained vocabulary of about 3 million words and phrases with their vectors, which significantly improved the accuracy correlating the predicted and given category tree. We classified products into a five-level taxonomy. Our experiments showed that GloVe [19] and FastText [20] embeddings-based extraction of product taxonomy lead to the best performance reaching 0.86 and 0.82 in cosine similarity [24] and F1-score, respectively. We also manually examined the produced categorization outputs and found that often our predicted results are more specific and fine-grained in comparison with those provided manually.

As the word vectors created were pre-trained on general text corpus extracted from the web, so in some cases they were not able to grasp the context of the words in the product names specific to the e-commerce industry. Pre-training these word embeddings on task-specific text corpus might help in improving the vectorizarion of given texts and thus the model performance.

References

1. Campbell JC, Hindle A, Stroulia E (2015) Latent dirichlet allocation: extracting topics from software engineering data. Art Sci Anal Softw Data 3:139–159. https://doi.org/10.1016/B978-0-12-411519-4.00006-9
2. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022
3. Friedman JH (2000) Greedy function approximation: a gradient boosting machine. Annals of Stat 29:1189–1232
4. Yu H, Ho C, Arunachalam P, Somaiya M, Lin C (2012) Product title classification versus text classification. Csie.Ntu.Edu.Tw, pp 1–25. [Online] Available http://www.csie.ntu.edu.tw/~cjlin/papers/title.pdf
5. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
6. Sun C, Rampalli N, Yang F, Doan AH (2014) Chimera: large-scale classification using machine learning, rules, and crowdsourcing. Proc VLDB Endow 7(13):1529–1540. https://doi.org/10.14778/2733004.2733024
7. Ha J-W, Pyo H, Kim J (2016) Large-scale item categorization in e-commerce using multiple recurrent neural networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 107–115
8. Xia Y, Levine A, Das P, Di Fabbrizio G, Shinzato K, Datta A (2017) Large-scale categorization of japanese product titles using neural attention models. In: Proceedings of the 15th conference of the european chapter of the association for computational linguistics: Volume 2, Short Papers. Association for Computational Linguistics
9. Shen D, Ruvini J-D, Sarwar B (2012) Large-scale item categorization for e-commerce. In: Proceedings of the 21st ACM international conference on information and knowledge management, CIKM '12, New York, NY, USA. ACM, pp 595–604
10. Das P, Xia Y, Levine A, Di Fabbrizio G, Datta A (2016) Large-scale taxonomy categorization for noisy product listings. In: Big Data (Big Data), 2016 IEEE international conference on, IEEE, pp 3885–3894

11. Cevahir A, Murakami K (2016) Large-scale multi-class and hierarchical product categorization for an e-commerce giant. In: Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers, pp 525–535
12. Hinton GE, Osindero S, Teh Y-W (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
13. Chen J, Warren D (2013) Cost-sensitive learning for large-scale hierarchic al classification. Int Conf Inf Knowl Manag Proc 1351–1360. https://doi.org/10.1145/2505515.2505582
14. Kozareva Z (2015) Everyone likes shopping! multi-class product categorization for e-Commerce. In: NAACL HLT 2015–2015 conference of the north american chapter of the association for computational linguistics: human language technologies proceedings conference, pp 1329–1333. https://doi.org/10.3115/v1/n15-1147
15. Li MY, Kok S, Tan L (2018) Don't classify, translate: multi-level e-commerce product categorization via machine translation. arXiv, pp 1–15
16. Krishnan A, Amarthaluri A (2019) Large scale product categorization using structured and unstructured attributes. arXiv, 2019
17. https://en.wikipedia.org/wiki/Jaccard_index
18. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. Adv Neural Inf Process Syst 1–9
19. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation Proc. EMNLP 12:1532–1543
20. Bojanowski P, Grave E, Joulin A, Mikolov T (2016) Enriching word vectors with subword information. arXiv:1607.04606.
21. Wieting J, Bansal M, Gimpel K, Livescu K, Roth D (2015) From paraphrase database to compositional paraphrase model and back trans. ACL 3:345–358
22. Ganitkevitch J, Van Durme B, Callison-Burch C (2013) PPDB: the paraphrase database. In: Proceedings of HLT-NAACL, pp 758–764
23. Arora S, Liang Y, Ma T (2016) A simple but though baseline for sentence embeddings. Iclr 15:416–424
24. Faisal R, Kitasuka T, Aritsugi M (2012) Semantic cosine similarity. Semantic Scholar 2(4):4–5