Electron Commer Res
DOI 10.1007/s10660-017-9268-5

Engineering doc2vec for automatic classification of product descriptions on O2O applications

Hana Lee[1] · **Young Yoon**[1]

Abstract In this paper, we develop an automatic product classifier that can become a vital part of a natural user interface for an integrated online-to-offline (O2O) service platform. We devise a novel feature extraction technique to represent product descriptions that are expressed in full natural language sentences. We specifically adapt doc2vec algorithm that implements the document embedding technique. Doc2vec is a way to predict a vector of salient contexts that are specific to a document. Our classifier is trained to classify a product description based on the doc2vec-based feature that is augmented in various ways. We trained and tested our classifier with up to 53,000 real product descriptions from Groupon, a popular social commerce site that also offers O2O commerce features such as online ordering for in-store pick-up. Compared to the baseline approaches of using bag-of-words modeling and word-level embedding, our classifier showed significant improvement in terms of classification accuracy when our adapted doc2vec-based feature was used.

Keywords O2O application · doc2vec · Online advertisement · Intelligent classification · Paragraph embedding

1 Introduction

Recently, online-to-offline (O2O) services in various areas have gained significant popularity. These services support offline providers to promptly respond to the demands of online consumers by promoting products and services at a low cost.

✉ Young Yoon
young.yoon@hongik.ac.kr

Hana Lee
lhana1127@gmail.com

[1] Department of Computer Engineering, Hongik University, 94 Seocho-gu Wowsan-ro, Seoul, South Korea

Published online: 28 August 2017

However, most O2O services are domain-specific and thus served in separate applications.

As illustrated in Fig. 1, we envision that these various O2O services can rather be served from an integrated platform. Such a platform would be beneficial to the online consumers, since there is no need to keep a myriad of domain-specific O2O appplications. However, for offline sellers, cataloging promotional information can become quite cumbersome and time-consuming. For instance, Fig. 2 shows the product registration page of eBay, a popular multi-national e-commerce site. We can see that the offline providers have to manually classify the product information according to a highly complicated product type hierarchy. Such a complex interface can cause even more inconvenience to the small offline providers who are likely to use mobile business management tools running on hand-held devices. It is not user-friendly to navigate through the complicated classification hierarchy on a small mobile screen. Such inconvenience motivates us to study a natural user interface (NUI) for the integrated O2O services. For example, as shown in Fig. 3, we can have a chatbot in the backend that represents the O2O services. This chatbot accepts requests from both the offline providers and the online buyers. When the offline providers request to register their product descriptions, the chatbot automatically classifies the product information under multi-level categories. On the other hand, upon receipt of the product inquiry from the online buyers, the chatbot automatically retrieves product descriptions under the categories that are related to the inquiry. Note that the product descriptions from the users are all issued in full natural language sentences. And the objective of this paper is to come up with a novel a classifier that classifies these product descriptions more effectively.

Kakao Yellow ID comes close to the integrated O2O service platform we envision [35]. Any offline retailer can register itself to Kakao's mobile chat platform and invite online consumers who want to chat in private. However, human representatives have to be on standby to chat with the consumers. Instead we can have chatbots respond to the requests for product registrations and inquiries as described above.

The product classifier learns to detect related categories based on various representation of the product descriptions. The most popular representation method is based on Bag-of-Words (BOW) modeling. A description can be represented as a vector of words frequencies. Given the vector, the conditional

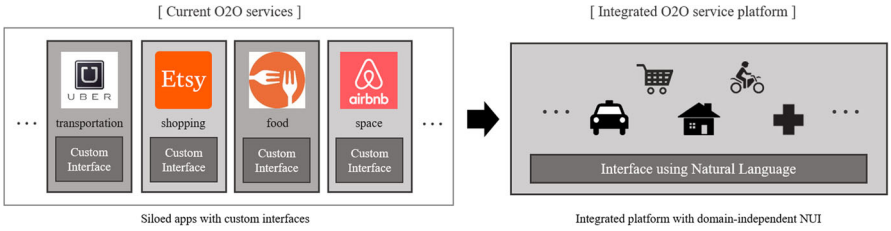

Fig. 1 Current O2O services and integrated O2O platform we propose

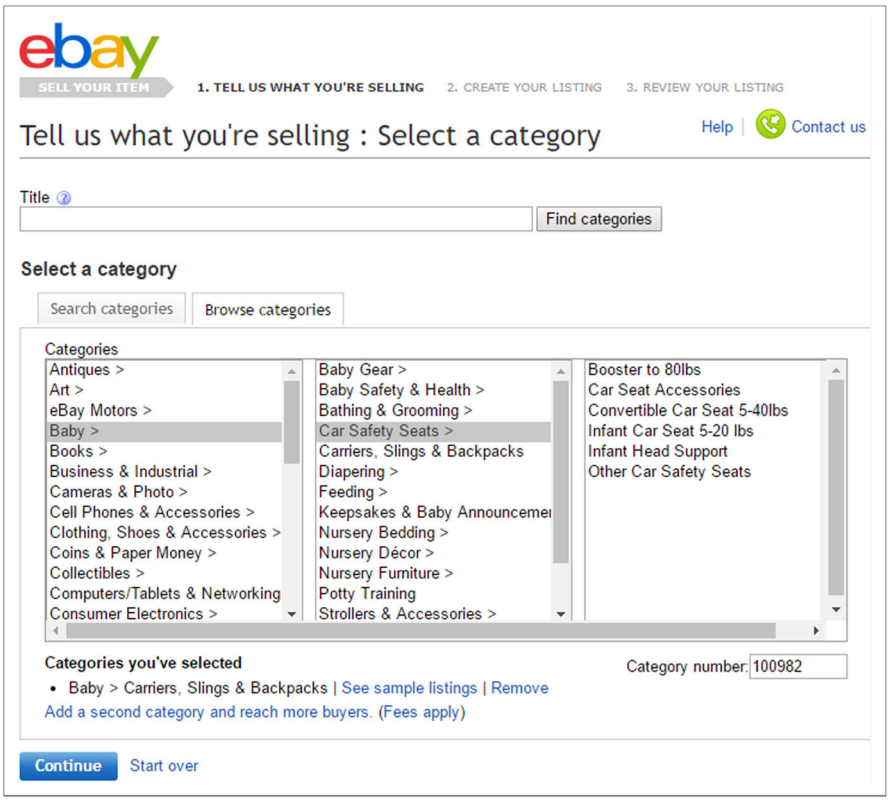

Fig. 2 Example of product registration page of eBay

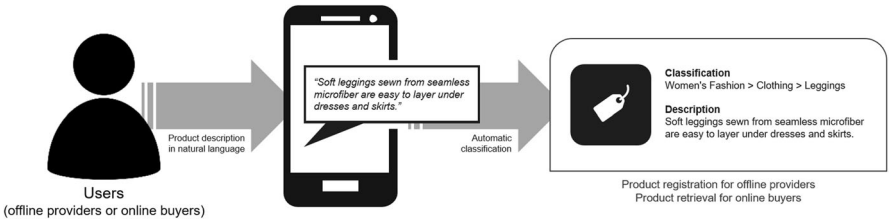

Fig. 3 An example of using natural user interface (NUI)

probability of a description being classified into a category is computed based on Naïve Bayes algorithms. Naïve Bayes classifier is known to work well with larger learning data sizes [14], and it has been successfully adopted in many complex real world situations such as medical data, Chinese text and RNA classification [15, 22, 37].

However, the BOW-based feature extraction does not capture the semantic characteristics of the product descriptions such as word ordering and contexts (e.g.,

surrounding words). Recently, a few research works reported the effects of the feature extraction based on word embedding [4, 16]. Word embedding is a technique to predict surrounding words (word vectors) of a given word. The most notable word embedding algorithm is word2vec that was devised by Mikolov et al. at Google [7, 24]. The word2vec model is effective in word-level representation. However, word2vec may come short in representing the semantics of a specific document.

In this paper, we develop a novel description representation technique that adapts a document embedding algorithm as presented in [17]. Specifically, we use doc2vec which is a Gensim implementation of the document embedding algorithm[1]. Doc2vec is inspired by the word embedding technique, and it is used to extract surrounding word vectors that are specific to a document. Doc2vec has been proven to be effective mainly in classification of binary sentiments (e.g., positive and negative sentiments) [6, 12, 17]. However, it is not clear whether doc2vec can perform well for a multi-class product classification. Product classification is a more challenging problem as there can be hundreds of product categories to choose from. A blind adoption of doc2vec may not be effective in product classification. Therefore, we consider augmenting the doc2vec-based feature in various ways.

We claim our key contribution as follows. We study the effect of various adaptation of the doc2vec-based feature through the classifier that is trained according to the framework illustrated in Fig. 4. This framework includes the step of collecting thousands of real product descriptions from Groupon, a popular social commerce site that also offers O2O commerce features such as online ordering for in-store pick-up [34].

The rest of this paper is presented in the following structure: (1) In Sect. 2, we first introduce related works in the context of automatic product classification; We also associate our work with a few interesting developments in O2O community; (2) In Sect. 3, we present the learning process to train the automatic product classifier including pre-processing, feature extraction and training data generation; (3) In Sect. 4, we evaluate the performance of our classifier with various feature adaptations. (5) In Sect. 5, we conclude and discuss future research topics.

2 Related work

In this section we introduce related works on automatic product classification and O2O applications.

The work in [19] transformed the taxonomy of the open directory project to a well-organized hierarchical taxonomy of topics. Based on the refined taxonomy, [19] created a new classifier called Merge-Centroid Classifier to classify web pages and advertisement data. The taxonomy of ODF encompasses various topics besides products. Lee et al. [19] used the Centroid Classifier with an emphasis on

[1] https://radimrehurek.com/gensim/.

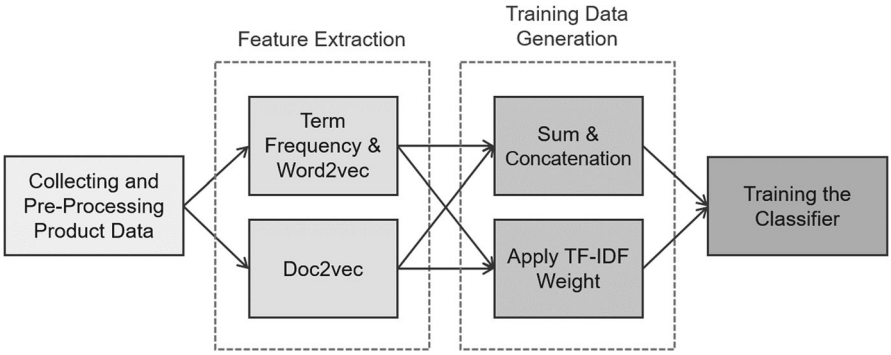

Fig. 4 The framework for feature extraction and traning

hierarchical classifier. Ding et al. [5] classified product data using information retrieval and machine learning. They used classification system of the United Nations Standard Products and Services Code (UNSPSC). They preprocessed the training data such as root extraction, noun phrase extraction and stop words exclusion. They measured the accuracy of the Naïve Bayes classifier algorithm with 41,913 items of product data provided by a French company Hubwoo and showed a maximum of 78% accuracy. Gottipati [8] extracted features with LDA and measured the classification accuracy using the following three algorithms: Naïve Bayes, SVM (Support Vector Machine) and KNN (K-Nearest Neighbor). As a result, Naïve Bayes showed the highest accuracy of 86.2% for top-level classification. A study on the modified Naïve Bayes classifier for E-Catalog classification extends the Naïve Bayes classifier using structural properties of product items [13]. They normalized the weight of each description element such as name, price, and description. They repeatedly assigned random weights to each element until those weights led to the best classification accuracy. With this approach, a maximum of 86% accuracy was achieved [13].

The aforementioned research works primarily used the bag-of-words (BOW) modeling to represent the feature of product descriptions. Recently, a few approaches of using word embedding techniques for product classification have emerged. Das et al. [4] mentioned the usage of word2vec for representation of product descriptions. Given little detail on how word2vec was adapted, this work stated that no improvement was made by using the word2vec-based feature. In contrast, [16] showed the effectiveness of the word2vec-based feature. They averaged the word2vec vectors for all words appearing in a document. They trained a neural network algorithm with product data from Yahoo. They achieved up to 88% classification accuracy given 319 categories with the highest desity at hierarchical levels 3 and 4.

Recent word and document embedding techniques were tested primarily for *binary* sentiment classification [6, 12, 17, 29, 30, 36]. Another binary classification work was studied in the domain of election classification problem by Yang et al. [38]. They were focused on finding optimal context window size and dimensionality for an effective word embedding model. With a training algorithm based on

Convolutional Neural Network (CNN), they are able to achieve a maximum of 76.3% F1-score for classifying document into one of the two labels, 'Election-related' and 'Not Election-related'.

Product classification is a more challenging problem as there can be hundreds of categories. For instance, in [11], word embedding was specificaly used to classify product descriptions into one of 3 categories. This work generated a document vector using LSA and word2vec to represent a product description. They used this feature to train a classifier based on CNN algorithm. However, the highest accuracy rate this work could achieve was only 50.4%.

We have found a few research works on multi-class classification for non-product descriptions word embedding techniques as follows. In [21], Liu et al. proposed topical word embedding (TWE) technique. With TWE, they were able to outperform the approach of using paragraph vectors [17]. Hashimoto et al. [9] proposed to learn contextual information of a document through analyses of relationship between nouns. After training with over 300,000 pairs of nouns, they were able to achieve a maximum of 83.5% F1-score for classifying a document into one of 19 classes. Dai et al. [3] learned various document modeling algorithms such as LDA, BOW and paragraph vector with Wikipedia and arXiv data. They observed that paragraph vector performed significantly better than others. Ma et al. [23] modeled word embedding (word2vec) with datasets from Wikipedia. They tested with short text classification and showed better performance compared to the case where BOW-based feature was used.

Our literature survey tells us that the effectiveness of doc2vec-based features has not been studied for product classification problem. This motivates us to examine whether various adaptation of doc2vec-based features can improve the performance of multi-class product classification.

3 Learning procedure

In this section, we design the overall procedure for constructing a classifier based on various features of product descriptions.

3.1 Collecting and pre-processing product data

Our main source of product descriptions is Groupon which is a world-wide e-commerce marketplace. Groupon also supports an O2O marketplace. Local retailers can sell and promote products for Groupon customers to pick up in-store. As shown in Fig. 5, Groupon used to provide product descriptions in a dedicated section called "Nutshell". Every Nutshell information is written in full sentences. These sentences are mostly concise but sufficiently descriptive. We expect such non-verbose descriptions to be preferred by the users of mobile O2O applications. Every product on Groupon is categorized hierarchically up to three levels. Table 1 shows the number of categories at each level. Table 2 shows the number of product descriptions for each top-most category. We can see that the distribution of the product descriptions is skewed.

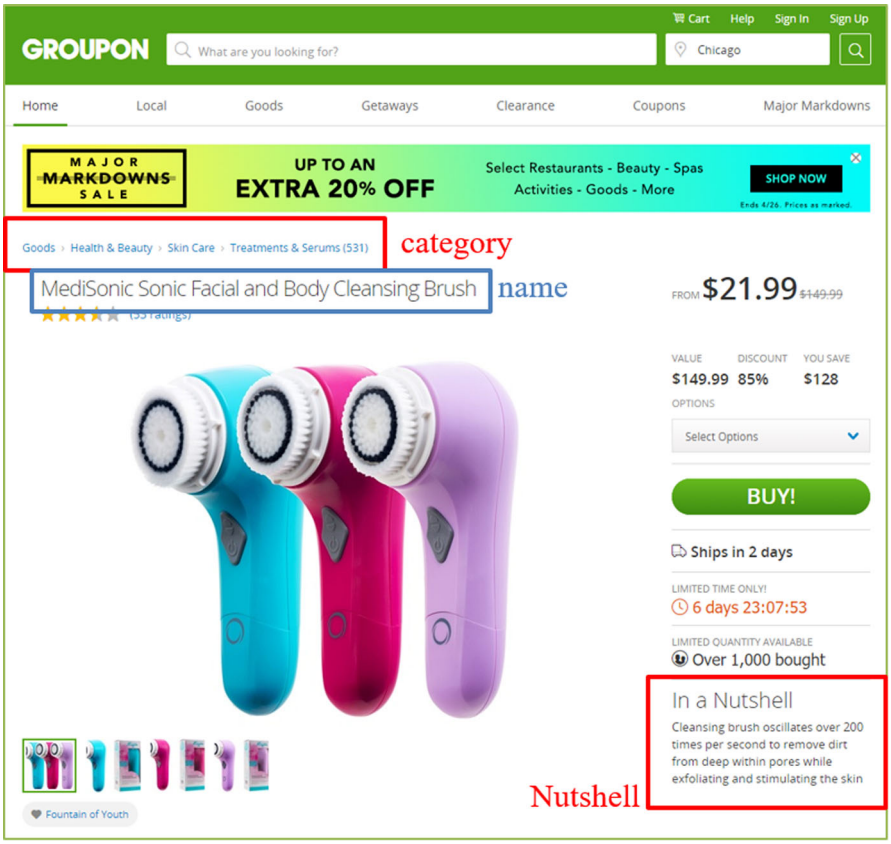

Fig. 5 Example of Groupon's product information page with categories, name and nutshell

Table 1 Number of categories by classification levels

Level-1	11
Level-2	79
Level-3	507

The product descriptions are pre-processed as follows. Stop words such as conjunctions and articles are removed. We conducted stemming operation to extract the root of a word [10]. For example, "run", "runs", and "running" are all counted as the root word "run". Interestingly, we found out that quite a few descriptions differ only by some numerals. For example, Apple smartphones iPhone5 and iPhone6 are oftentimes described with an identical set of tokens, and they appear under the same category. It is better to treat these descriptions to be identical. Thus, we prune out the numerals such as model numbers so that the redundancy in our data set can be increased further. Also, we exclude any description that has at most two words as it would not sufficiently carry any meaning. After the pre-processing, we obtained a corpus of 36,138 unique words.

Table 2 The number of product descriptions for each level-1 category

Category	# Of descriptions
Electronics	10,618
Women's fashion	12,720
Entertainment	4423
Jewelry & watches	1616
Sports & outdoors	1561
Auto & home improvement	4669
Health & beauty	14,669
Grocery, household & pets	3446
Men's fashion	7379
For the home	14,999
Baby, kids & toys	3900
Total	80000

3.2 Feature extraction

For each cleansed product description, we extract features. We first introduce the baseline feature extraction methods. Then we explain the feature extraction based on document embedding.

3.2.1 The baseline features

The first baseline approach is to use the classic bag-of-words model. With the bag-of-words model, we simply represent each product description as a vector of term frequencies. This modeling approach can be effective in cases such as spam filtering [33]. However, the bag-of-words model does not preserve the semantic representation, such as ordering of words.

The second baseline approach employs word2vec[2]. Word2vec is a word embedding technique that can predict surrounding words of a given word. The surrounding words are predicted by maximizing the log probability as defined as follows.

$$\sum_{t=1}^{T} logP(w_t|w_{t-c}, \cdots, w_{t+c}) \tag{1}$$

T is the total number of all words in the corpus. w_t is a word to predict. c is the number of surrounding words to predict. The prediction task is defined in Eqs. (2) and (3). Maximization of the objective function is done by training the neural network as shown in Fig. 6.

$$logP(w_t|w_{t-c}, \cdots, w_{t+c}) = \frac{exp(u_{w_t} \cdot v)}{\sum_w exp(u_w \cdot v)} \tag{2}$$

[2] A comprehensive explanation is available in [32]. A visual explanation is available at https://ronxin.github.io/wevi/.

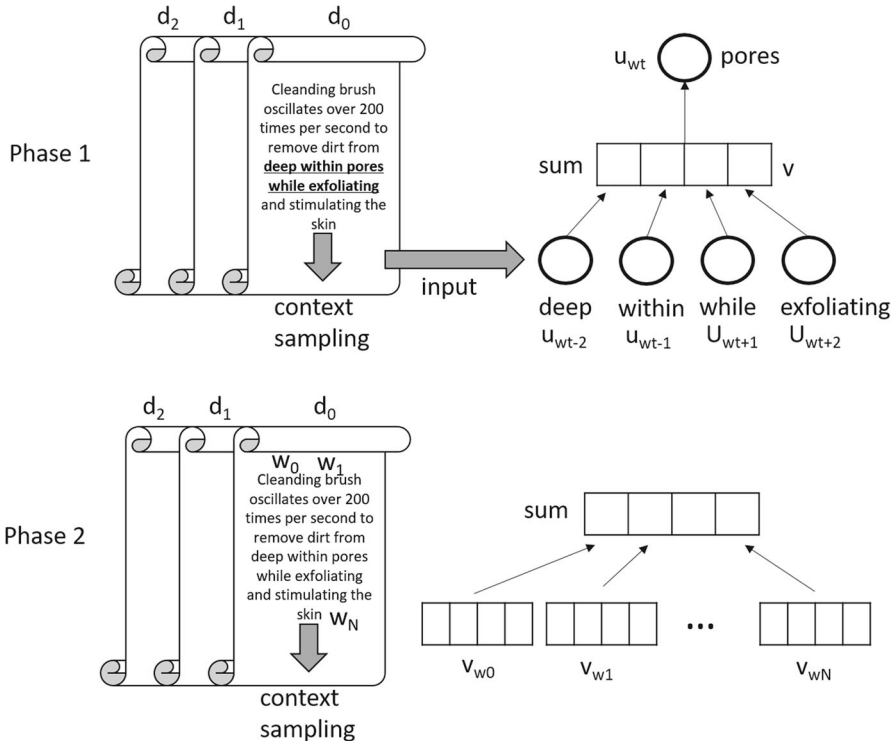

Fig. 6 Learning the next word using fixed-length sample context from every product description

$$v = \sum_{t' \neq t, -c \leq t' \leq c} v_{w_{t'}} \tag{3}$$

u_w is a output vector and v is a sum of words in the input vector.

After the training operation completes, a vector W is computed for every product description as follows.

$$W = \sum_{t=1}^{N} v_{w_t} \tag{4}$$

N is the number of words in a document. v_{w_t} is the predicted vector of surrounding words. We take W as the feature of every product description. v_{w_t} can be useful for inferring semantic relation between words. However, it is questionable whether the sum of inferred v_{w_t} can sufficiently convey the semantics of a given product description. We do not train the neural network to infer v_{w_t} of a word in the context of specific product descriptions. Therefore, the inferred v_{w_t} may not be relevant to a given description. That is, the product description may not even contain the words in v_{v_t}.

3.2.2 Feature extraction using document embedding

We introduce a novel approach of extracting feature using document embedding techniques. Specifically we adapt doc2vec that implements the methods of distributed representation of paragraphs as presented in [17]. With this approach, we try to capture the semantics of words within the context of a product description.

As shown in the following equation, we add an ID of a product description (d_t) to the object function in Eq. (5)

$$\sum_{t=1,i=1}^{T,N} logP(w_t|w_{t-c},\cdots,w_{t+c},d_i) \tag{5}$$

N is the total number of documents. The prediction task we defined in Eqs. (2) and (3) is adjusted accordingly as follows.

$$logP(w_t|w_{t-c},\cdots,w_{t+c},d_t) = \frac{exp(u_{w_t}\cdot v)}{\sum_w exp(u_w\cdot v)} \tag{6}$$

$$v = \sum_{t'\neq t,-c\leq t'\leq c} v_{w_{t'}} + d_i \tag{7}$$

Maximization of the objective function (Eq. 5) is done by training the neural network as shown in Fig. 7. After the training completes, v immediately becomes the feature of product description (d_i). As explained in [17], the ID d_i can be thought of as another word. The context $v_{w_{t'}}$ is fixed-length and randomly sampled from a sliding window over the product description.

3.2.3 Training data generation and weight adjustments

Every product description is converted into a tuple of features and categories. Features can be either a sum or a concatenation of vectors we obtain through the process we explained above. We generate a set of these tuples for each classification level.

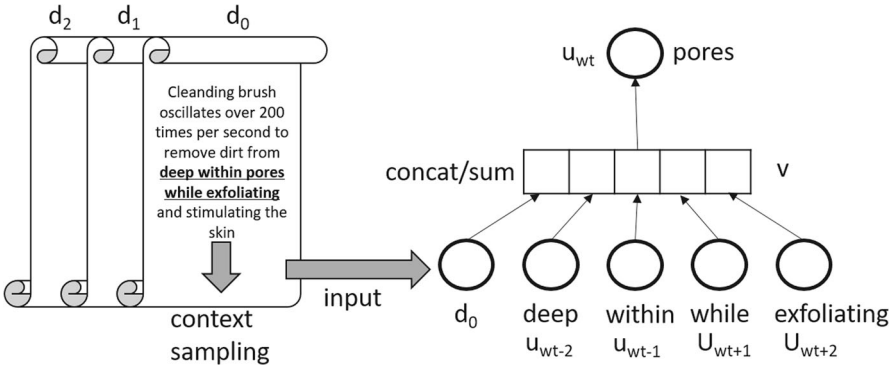

Fig. 7 Learning the next word using fixed-length sample context from every product description

Also we devised a method for assigning different weights to every word according its significance. A word is significant if it could strongly represent a category. Common words that appear often in multiple categories are likely to confuse the learning algorithm. Therefore, they cannot be considered to be significant.

To determine the significance, we first considered the employment of the Latent Dirichlet Allocation (LDA) method [2]. LDA is an unsupervised learning method that groups documents into a configurable number of clusters. Each cluster is represented with a term vector. We considered giving a higher weight to a word in a product description, if the word is listed in one of the term vectors. However, in our training data, there are categories that have similar subcategories, such as "Women's Fashion" and "Men's Fashion" which are under the parent category, "Fashion". The problem was that LDA was not effective in distinguishing the difference between similar subcategories.

Instead of LDA, we adapt term frequency-inverse document frequency (TF-IDF) [31] to assess how significant a word is in determining a category for a given description. The TF-IDF value is obtained by multiplying the ratio of the word in a document and the reciprocal of the ratio used in the entire descriptions. The higher the TF-IDF value of a word is, the more often the word is used in one description but not the other.

The algorithm that calculates the weights of representative words for each category using TF-IDF is specified in Algorithm 1. In order to obtain the representative word and the TF-IDF value for each category, the total sum of words is obtained for each category. Then, we compute the TF-IDF value for each word used in the product descriptions under a category. 'TF' is the number of words in the category, and 'IDF' is the number of items in which the word appears in the whole category. We multiply these two values to obtain the TF-IDF value of the word. We can multiply each word vector by this TF-IDF value as a way to assign weights.

Input: Input file F_{input} from user
Output: TF-IDF value list $tfidf$
$AllCat$ = make list of all category from F_{input}
for $category \in AllCat$ **do**
  $Word_{cat}$ = Make the list of words that appear under $category$
  sum = total word count of the $Word_{cat}$
  for $word$ *in* $Word_{cat}$ **do**
    $tf[word][category] \leftarrow (word$ frequency in $category)$ / sum
    $|C| \leftarrow$ # of categories under which $word$ appears
    $idf[word][category] \leftarrow log(AllCat + \frac{1}{|C|})$
    $tfidf[word][category] \leftarrow tf[word][category] * idf[word][category]$
  end
end

Algorithm 1: Algorithm to calculate TF-IDF value for each word under every product category

4 Evaluation

In this section, we discuss the effectiveness of the various features we designed in Sect. 3.2.

4.1 Experiment environment

All of our experiments were conducted on a 64-bit CentOS (version 6.6) computer with a 24-core Intel Xeon CPU @ 2.10 GHz and a 64GB RAM. Our classifier is trained based on the logistic regression algorithm that is implemented with Python Scikit-learn library [27]. We scrapped over 50,000 product descriptions from Groupon. Every product description is pre-labeled with multi-level categories (level-1, level-2, level-3). The set of product descriptions is empirically divided into training set and test set at 7:3 ratio. We used Gensim library (version 0.13.2) to extract the features based on word embedding and document embedding, as explained earlier in Sects. 3.2.1 and 3.2.2.

4.2 The effect of various feature adjustment

We trained our classifier with different types of features that represent a product description as follows:

- a vector representing the frequency of every word (denoted as *frequency*)
- an average of all word vectors that were learned through the word2vec-based word embedding (denoted as *word2vec*)
- a concatenation of frequency vectors and word2vec (denoted as *word2vec* + *frequency*)
- a concatenation of product description token and word vectors learned through the doc2vec-based document embedding (denoted as *doc2vec*)
- a concatenation of the frequency and doc2vec features (denoted as *doc2vec* + *frequency*)

Tables 3, 4, 5 show the experimental results of various feature adjustments. We measured accuracies (Acc), macro-averaged precisions (Pre), Recalls (Rec), F-measures (F), Rank (R), and Variation (Var) at each classification level. Var value indicates increase and decrease when TF-IDF weight was applied to each feature. The detailed reasoning behind the Var values is explained in Sect. 4.4.

The results are also shown as graphs in Figs. 8, 9, 10. They show the F-measures at each level. On level-1, as shown in Fig. 8, we trained with 500 to 3500 training data per category. For the level-1 classification, our classifier performed the worst when only doc2vec or word2vec was used as a feature. The F-measure ranged from 27.0 to 63.8%. However, when the doc2vec feature was combined with the frequency feature, the F-measure was improved significantly, ranging from 83.5 to 90.2%. The F-measure ranged from 80.2 to 86.6% when the frequency vector was the only feature used. Augmenting the frequency vector with word2vec did not improve the accuracy.

Table 3 F-measure of level-1 classification

Feature	Level-1					
	Acc	Pre	Rec	F	R	Var
(a) *frequency*	0.868	0.871	0.864	0.866	4	–
(b) *frequency+w*	0.863	0.863	0.861	0.861	6	−0.021
(c) *word2vec*	0.585	0.574	0.567	0.561	7	–
(d) *word2vec+frequency*	0.869	0.872	0.865	0.868	3	+0.399
(e) *word2vec+frequency+w*	0.865	0.866	0.862	0.863	5	+0.390
(f) *doc2vec*	0.428	0.429	0.434	0.427	8	–
(g) *doc2vec+frequency*	**0.903**	**0.903**	**0.901**	**0.902**	1	+0.311
(h) *doc2vec+frequency+w*	0.888	0.885	0.887	0.885	2	+0.294

Table 4 F-measure of level-2 classification

Feature	Level-2					
	Acc	Pre	Rec	F	R	Var
(a) *frequency*	0.828	0.824	0.746	0.772	3	–
(b) *frequency+w*	0.819	0.801	0.729	0.751	6	−0.021
(c) *word2vec*	0.518	0.429	0.372	0.372	8	–
(d) *word2vec+frequency*	0.830	0.818	0.749	0.771	4	+0.399
(e) *word2vec+frequency+w*	0.820	0.809	0.740	0.762	5	+0.390
(f) *doc2vec*	0.621	0.639	0.499	0.525	7	–
(g) *doc2vec+frequency*	**0.898**	**0.876**	**0.815**	**0.836**	1	+0.311
(h) *doc2vec+frequency+w*	0.883	0.858	0.799	0.819	2	+0.294

Table 5 F-measure of level-3 classification

Feature	Level-3					
	Acc	Pre	Rec	F	R	Var
(a) *frequency*	0.659	0.627	0.593	0.596	4	–
(b) *frequency+w*	0.639	0.615	0.577	0.579	6	−0.017
(c) *word2vec*	0.178	0.114	0.136	0.106	8	–
(d) *word2vec+frequency*	0.664	0.632	0.594	0.597	3	+0.491
(e) *word2vec+frequency+w*	0.645	0.620	0.578	0.583	5	+0.477
(f) *doc2vec*	0.541	0.458	0.434	0.424	7	–
(g) *doc2vec+frequency*	**0.703**	**0.666**	**0.628**	**0.633**	1	+0.209
(h) *doc2vec+frequency+w*	0.683	0.652	0.613	0.615	2	+0.191

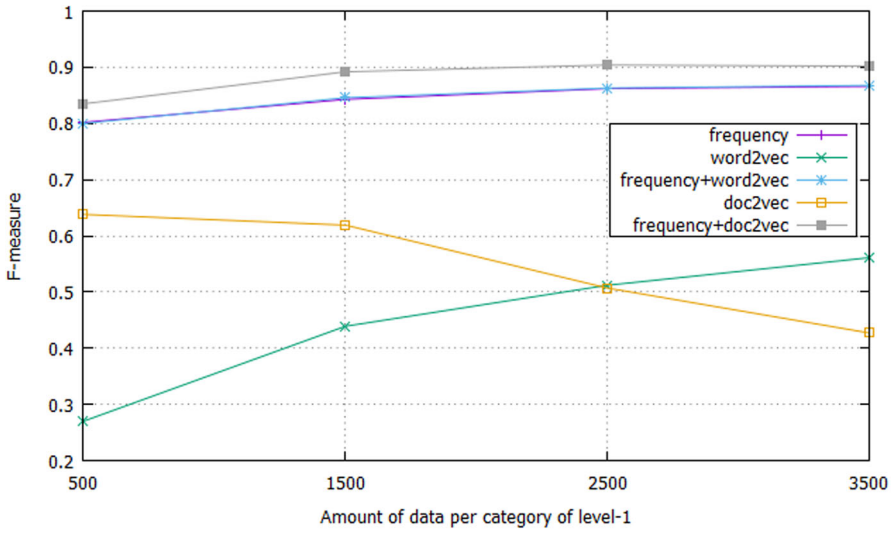

Fig. 8 F-measure of the level-1 classifications with various features

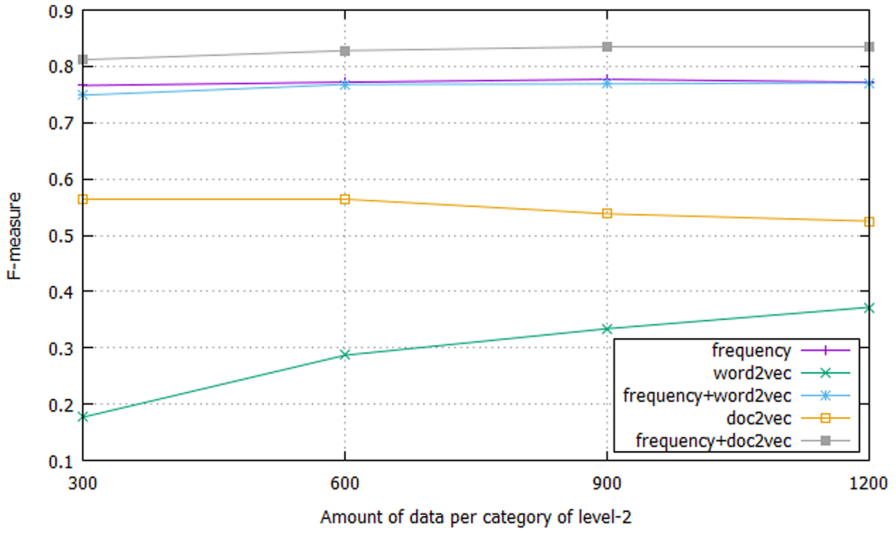

Fig. 9 F-measure of the level-2 classifications with various features

We observed a similar trend on the level-2 classification. In this case, we trained with 300 to 1200 training data per category. As shown in Fig. 9, when using word2vec, our classifier exhibited the lowest F-measure between 17.7 and 37.2%. Similar result was exhibited when doc2vec was the only feature used. However, the F-measure range jumped to 81.2–83.6% when doc2vec was combined with a frequency vector. When we used frequency only, the accuracy ranged from 80.2 to 86.6%.

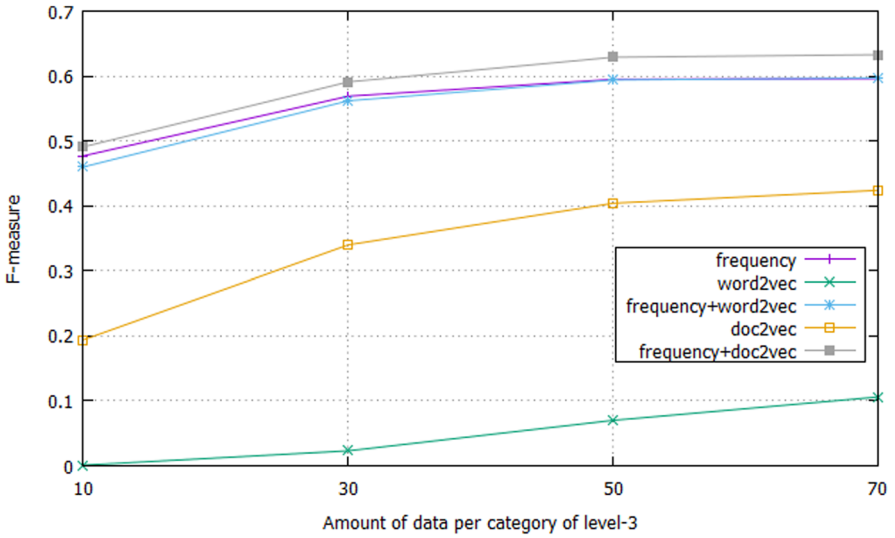

Fig. 10 F-measure of the level-1 classifications with various features

For the level-3 classification, we trained with 10 to 70 training data per category. Compared to using word2vec vectors, a noticeably higher F-measure was observed when frequency was used. However, the overall level-3 classification results are consistent with the results of the upper-level classifications.

We can empirically conclude that our classifier performed the best when the combination of doc2vec and frequency was used. The high accuracy was obtained by capturing not only the frequently-appearing words but also their surrounding words (context) that are specific to a product description. The context extracted by word2vec may not necessarily be relevant to the product description. This is why the classifier that was trained with word2vec consistently yielded a lower F-measure than the classifier that was trained with the composite feature of doc2vec and frequency.

4.3 The effect of various doc2vec parameter adjustment

We observed the effect of various ways of adjusting the doc2vec-based feature. We first turned on the random context sampling rate by configuring the *sampling* option of Gensim library to 1e-5. By default, the sampling was turned off, which was the setting we used for the experiments in the previous section. Also, for the experiments in the previous section, we concatenated the product description token and the word vectors for the doc2vec-based document embedding. We compared the concatenation approach against the method of taking the sum of product description token and the word vectors.

Tables 6, 7, 8a, b show the experimental results of doc2vec parameter adjustments. The result is also shown as a graph in Fig. 11. We can see that our

Table 6 F-measure of the classifier with adjusting the doc2vec parameter and data generation method in level-1 classification

Parameter	Level-1				
	Acc	Pre	Rec	F	Var
(a-1) *Sampling*	0.872	0.875	0.869	0.872	−0.030
(b-1) Concatenation	0.869	0.871	0.864	0.867	−0.035
(c-1) Random	0.876	0.873	0.830	0.849	−0.053
(a-2) No *sampling*	**0.903**	**0.903**	**0.901**	**0.902**	–
(b-2) Sum/average					
(c-2) Per category					

Table 7 F-measure of the classifier with adjusting the doc2vec parameter and data generation method in level-2 classification

Parameter	Level-2				
	Acc	Pre	Rec	F	Var
(a-1) *Sampling*	0.837	0.831	0.758	0.783	−0.053
(b-1) Concatenation	0.826	0.825	0.745	0.770	−0.066
(c-1) Random	0.835	0.780	0.678	0.712	−0.124
(a-2) No *sampling*	**0.898**	**0.876**	**0.815**	**0.836**	–
(b-2) Sum/average					
(c-2) Per category					

Table 8 F-measure of the classifier with adjusting the doc2vec parameter and data generation method in level-3 classification

Parameter	Level-3				
	Acc	Pre	Rec	F	Var
(a-1) *Sampling*	0.666	0.629	0.598	0.600	−0.033
(b-1) Concatenation	0.648	0.615	0.578	0.581	−0.052
(c-1) Random	0.679	0.610	0.540	0.547	−0.086
(a-2) No *sampling*	**0.703**	**0.666**	**0.628**	**0.633**	-
(b-2) Sum/average					
(c-2) Per category					

classifier performed better when the sampling was turned off. When the sampling is turned on, some important context can not be learned during the training of our classifier. However, when sampling is turned off, both the overhead of training and the model size increase. The F-measure increased by 9% when the sum was used. We can infer that the concatenation does not capture important semantic information. It is more advantageous to use a dense vector by taking the sum.

We also varied the size of the doc2vec feature from 100 to 500. The default size we used for the experiments in the previous section was 300. As shown in Table 9, the F-measure did not show any sharp increase beyond the feature size of 200. This is another indication that a dense vector with the size of around 300 is sufficient.

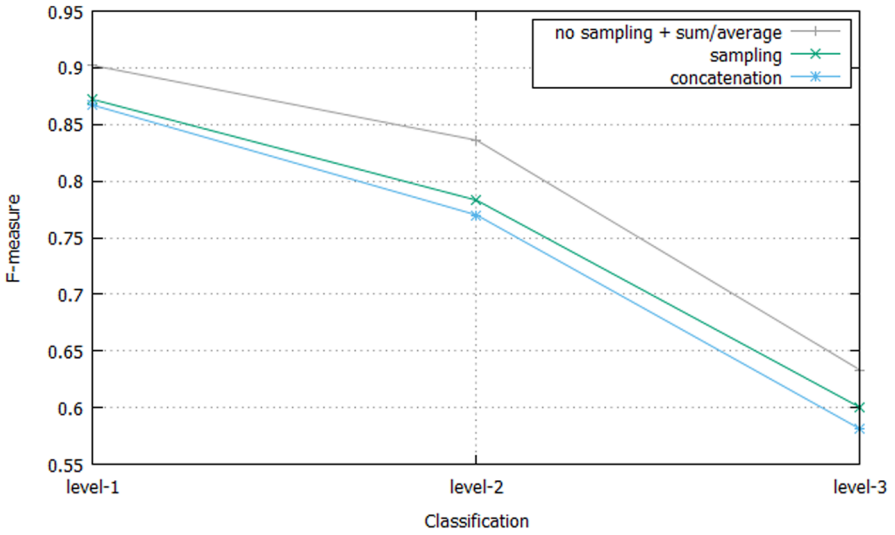

Fig. 11 F-measure of the classifier when adjusting the sampling and concatenation parameter

Table 9 F-measure of the classifier with adjusting the feature size parameter

Classification	Size of the doc2vec feature				
	100	200	300	400	500
Level-1	0.901	**0.902**	**0.902**	0.900	0.900
Level-2	0.817	0.832	0.836	**0.837**	**0.837**
Level-3	0.622	**0.633**	**0.633**	**0.633**	**0.633**

4.4 The effect of data generation methods and weight adjustments

We used two methods for sampling product descriptions into a training set. One is to extract the fixed N number of product descriptions from every category. The other is to extract product descriptions randomly. Tables 6, 7, 8c show the experimental results of adjusting doc2vec parameters. This result is shown as a graph in Fig. 12. We can see that, our classifier performed better when we used the fixed samples per category. Note that the distribution of the number of product descriptions over categories are skewed. When the product descriptions are randomly sampled, the learning is biased towards the category with more training data. This caused an overfitting problem and led to the decreased of the accuracy.

As we have observed in our previous work, applying the TF-IDF weight to the frequency vector can greatly improve the accuracy of the classification based on a Naïve Bayes classifier [18]. The TF-IDF value reflects the importance of a word in distinguishing a topic for a given document. We multiplied every word vector V_w in the doc2vec-based feature by the TF-IDF value of w. V_w is the word vector of w. We trained the classifier with this weighted feature. Contrary to our initial conjecture, using the weighted feature led to the decrease of F-measure, as shown in Tables 3, 4, 5b, e, h. This result is also shown as a graph in Figs. 13, 14, 15. We attribute this

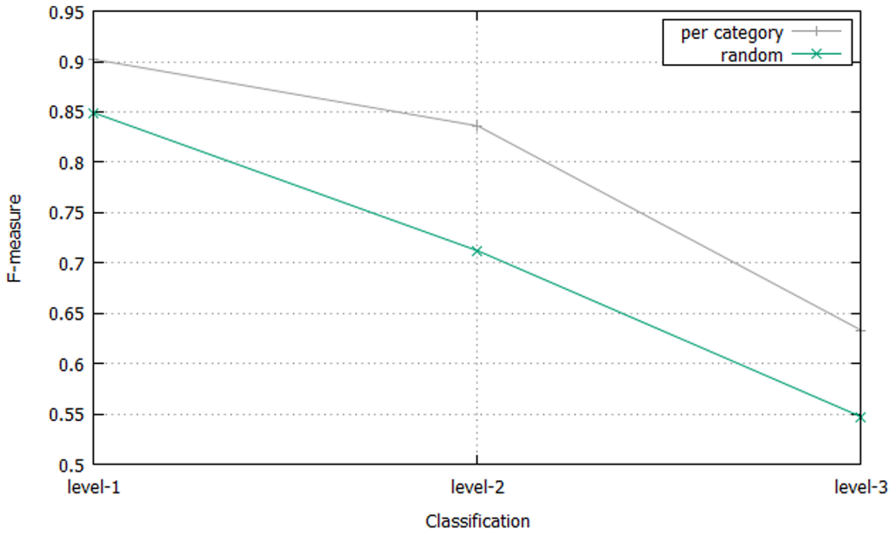

Fig. 12 F-measure of the classifier with adjusting the data selection method

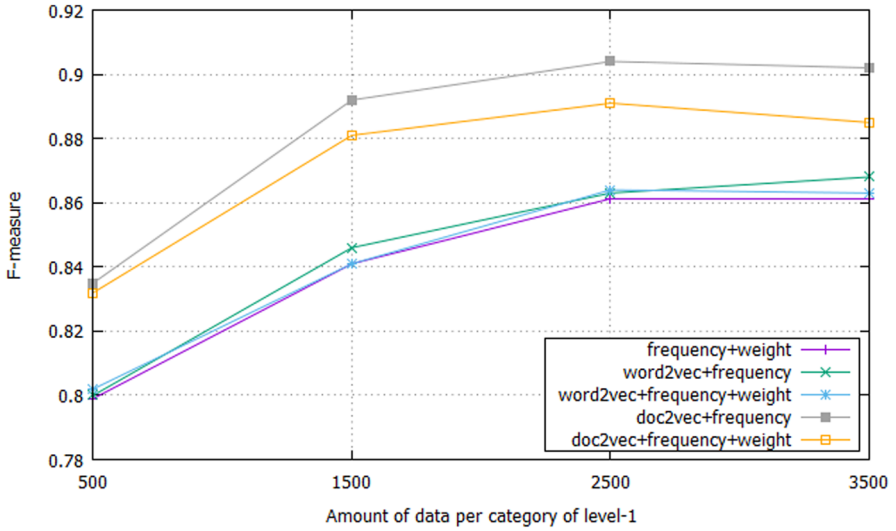

Fig. 13 F-measure when TF-IDF weight was applied for level-1 classification

decrease to the nature of logistic regression that is different from the Naïve Bayes classifier. With the logistic regression algorithm, our classifier learns to infer the importance of each word vector in recognizing a topic. In case of the Naïve Bayes classifier, the topic is predicted based on the Bayesian inference given the pre-weighted frequency vector.

Lastly, we observed that removing the stop words had no effect on the classification accuracy.

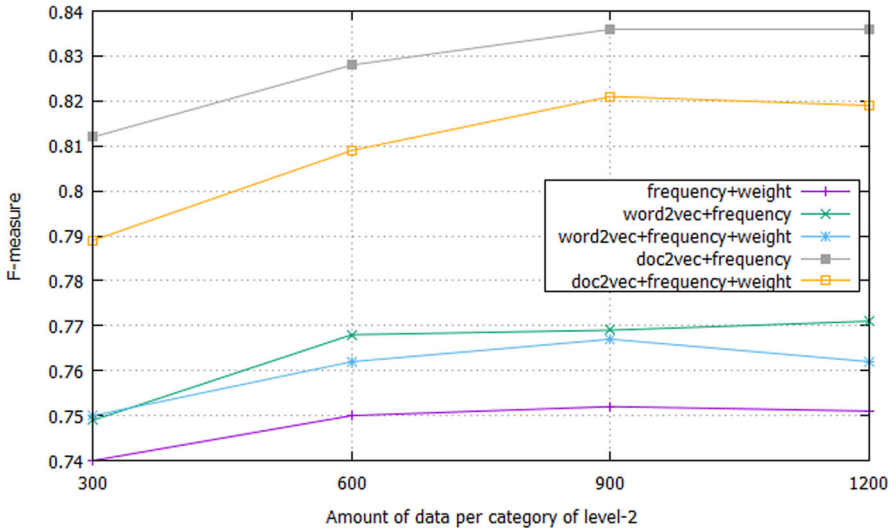

Fig. 14 F-measure when TF-IDF weight was applied for level-2 classification

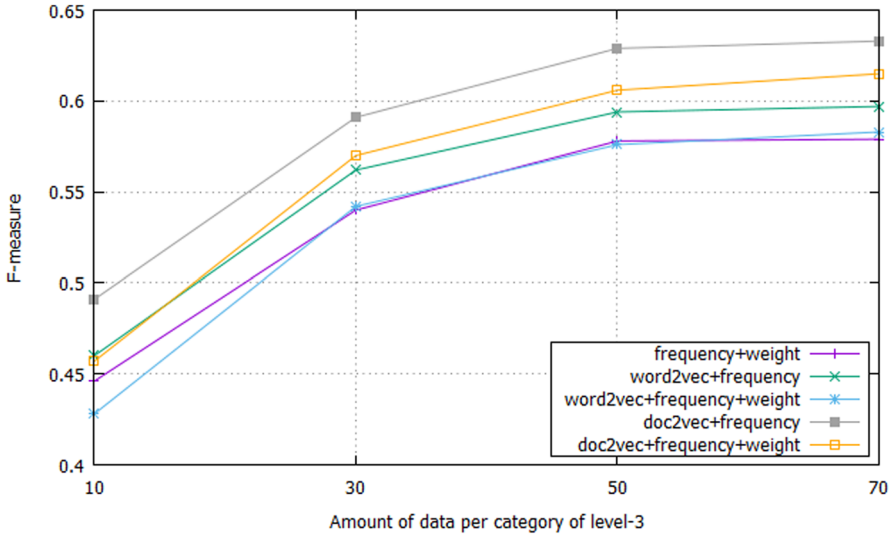

Fig. 15 F-measure when TF-IDF weight was applied for level-3 classification

4.5 Analyzing the ambiguity between topics

We computed *misclassification rate* between every pair of level-1 topics, $MR(C_{ij})$, as defined in Eq. (8).

$$MR(C_{ij}) = \frac{|\{tc \in Tset_i \bigwedge tc \Rightarrow C_j\}|}{|Tset_i|} \tag{8}$$

C_{ij} is a pair of categories, C_i and C_j. $MR(C_{ij})$ measures how many test cases (tc) are misclassified under category C_j ($tc \Rightarrow C_j$) while they were originally classified under category C_i.

The misclassification rates for all C_{ij} pairs are visualized on the confusion matrix in Fig. 16. A larger circle size indicates a higher misclassification rate. The confusion matrix is compared against the one we obtained in the previous work that used word-wise weighted frequency vectors [18]. We can see that the misclassification rates decreased for most of the C_{ij} pairs when our classification method was used (Fig. 16a). However, the misclassification rate between $C1$ (Electronics) and $C6$ (Auto&Home Improvement) was still high. The test cases under these categories contained common words such as "light", "bulb", "fan", and "flashlight" that could be related to both automobile, home and electronic products. We can see that the inherent ambiguity between categories can confuse not only the automatic classifier but also the humans.

Given this observation, we can branch out into some new research directions. The structure of product classification hierarchies can be designed in such a way to minimize the semantic ambiguity between topics. We can also address the fundamental limitation of the statistical learning approaches. For instance, many gender-neutral words commonly appear in product descriptions under both Men's Fashion and Women's Fashion categories. The statistical classifiers may not be able to capture rarely-used gender-specific words that are more effective in distinguishing between the two categories. We can study novel methods for automatically annotating product descriptions with semantically important keywords. We can retrieve these keywords from a product ontology such as the one devised in [26] that defines the list of key attributes for every product type. However, it should be technically challenging to identify the hidden and implicit attributes in the product descriptions automatically.

In terms of document search systems, we can consider placing redundant copies of product descriptions under multiple categories that are semantically similar to each other. With this approach, the availability of product descriptions can be improved. At the same time, the search speed can be enhanced if the product

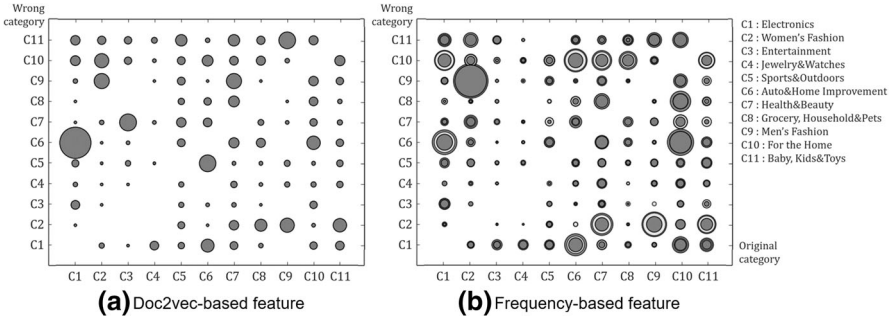

Fig. 16 Misclassifications of test cases at the top-most level (level-1). The x and y axes in represent 11 level-1 categories. The x-axis is the original category for test cases, and the y-axis is the category under which the test cases were miclassified

descriptions under the semantically similar categories are searched in parallel for given queries. With this approach, the recall of documents can be improved. However, there is a challenging tradeoff, i.e., the search precision can be compromised.

4.6 Comparison with related works

In this section, we compare our results with other related studies that we introduced in Sect. 2. The comparision table is given in Table 10. We show the number of classes (Class) used in each study, classification accuracy (Acc) and macro-averaging F-measure (F).

Rows from (a) to (e) show the performance results of product classification techniques. In case of (a), [19] devised a classifier called Merge-Centroid Classifier to categorize web pages and advertisement data. They used the taxonomy of the open directory project (ODP) and achieved 46.0% macro-averaged F-measure to classify web pages and advertisements into one of 3,883 classes. The taxonomy of ODP includes items from all domains, whereas we used the product data from Groupon as it is more suitable for training the product description classifier. Lee et al. [19] used the Centroid Classifier with an emphasis on a hierarchical classifier. In contrast, we used the logistic regression algorithm with the doc2vec-based feature that is augmented with term frequency vectors. With this approach, the accuracy was improved up to 90.3% for the top-level classification (11 classes).

In case of (b), [5] classified product data using information retrieval and machine learning. They tested with 32 UNSPSC categories and showed 78.0% classification accuracy. In case of (c), [8] tested classification with LDA-based fearues. They classified WITH 45 categories and showed 86.2% classification accuracy. We showed higher accuracy given more level-2 categories. In case of (d), [13] extends the Naïve Bayes classifier using structural properties of product items. In contrast to [13], we used the logistic regression algorithm and confirmed the negative effect of word-wise weighting method. Note that [13] trained the classifier on *structured*

Table 10 Accuracy comparison with related studies

Study	Class	Acc	F
(a) Lee et al. [19]	3883	–	0.460
(b) Ding et al. [5]	32	0.780	–
(c) Gottipati [8]	45	0.862	–
(d) Kim et al. [13]	256	0.860	–
(e) Kozareva [16]	319	–	0.880
(f) Our work	11	**0.903**	**0.902**
	79	**0.898**	**0.836**
	510	**0.703**	**0.633**
(g) Ju et al. [11]	3	0.504	–
(h) Liu et al. [21]	20	0.815	0.806
(i) Hashimoto et al. [9]	19	0.799	0.835
(j) Ma et al. [23]	8	0.855	–

product record, whereas we trained on product descriptions in full natural language sentences. In case of (e), Kozareva et al. [16] used word2vec-based feature that averaged the word2vec vectors for all words appearing in a document. They trained a neural network algorithm with product data from Yahoo. However the word2vec-based feature turned out to be less effective for our case of classifying product descriptions from Groupon. Higher F-measure was achieved when the combination of doc2vec and frequency vectors was used in our experiments with Groupon data.

Rows from (g) to (j) show the performance results of multi-class non-product classification based on word embedding techniques. In case of (g), word embedding was used to classify documents into one of 3 categories. The highest accuracy rate this work could achieve was only 50.4%. In contrast, we were able to achieve 70.3% accuracy for classifying products into more than 500 bottom-level categories. In case of (h), Liu et al. proposed topical word embeddings (TWE) and evaluated the performance of multi-class text classification. They achieved 81.5% classification accuracy and 80.6% macro-averaging F-measure given 20 classes. In case of (i), Hashimoto et al. [9] tested word embedding technique for text classification given 19 relation categories. They achieved 79.9% classification accuracy and 83.5% macro-averaging F-measure. We outperformed these works by achieving 89.8% accuracy and 83.6% F-measure with 79 level-2 classes. In case of (j), Ma et al. [23] used word2vec for classifiying Wikipedia articles. They achieved a maximum of 85.5% F-measure with 8 level-1 classes. Our classifier showed 90.2% F-measure given 11 level-1 classes.

Relatively higher classification performance we achieved based on our successful engineering of doc2vec-based features motivates us to look further into other document embedding techniques. For instance, [25] proposed recurrent neural networks (RNN) with long short-term memory cells. With this approach, richer semantic information such as word ordering patterns can be captured. This method was effective in retrieving relevant web documents. As a future work, we can test the effectiveness of long short-term memory cells for product classification.

As another line of future work, we can consider training our classifier with product descriptions from Etsy. Etsy is an user-friendly online platform for the buying and selling of handcrafted objects [1]. Similar to Groupon, Etsy supports an O2O service by launching mobile card readers for in-person offline payment [28]. Etsy is another interesting source of training data. The description of the handcrafted objects is typically more verbose and expressive than the terse product descriptions on Groupon. We can test the effectiveness on such verbose product descriptions. However, small offline retailers may prefer to enter concise product description through a mobile commerce interface for a quick and easy product promotion. Studying the effect of our classifier on the design of mobile commerce interfaces for the O2O services is an interesting research topic [20].

5 Conclusion and future work

In this paper, we used a doc2vec-based document embedding technique to extract features from product information described in full natural language sentences. We augmented the doc2vec-based feature in various ways and tested its effectiveness in

automatically classifying product descriptions. We trained our classifier with up to 53,000 real product descriptions collected from Groupon. Compared to the state-of-the-art approach of using word embedding technique (word2vec), the classification F-measure improved by up to 9% when the classifier was trained with the combination of doc2vec-based feature and frequency vectors. However, when we applied TF-IDF weights to the feature vectors, we observed a decrease in classification F-measure. Our evaluation result shows that representing the semantics of product descriptions with document embedding is highly effective for a challenging product classification problem. As a future work, we plan to integrate our classifier into a novel interactive product registration and search interface and study its effect on the user experience.

Acknowledgements This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03931324) and 2017 Hongik University Research Fund.

References

1. Abrahams, S. L. (2008). *Handmade online: The crafting of commerce, aesthetics and community on Etsy com*. Chapel Hill: The University of North Carolina.
2. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.
3. Dai, A. M., Olah, C., & Le, Q. V. (2015). Document embedding with paragraph vectors. arXiv preprint arXiv:1507.07998.
4. Das, P., Xia, Y., Levine, A., Di Fabbrizio, G., & Datta, A. (2016). Large-scale taxonomy categorization for noisy product listings. In *2016 IEEE international conference on big data (big data)* (pp. 3885–3894). IEEE.
5. Ding, Y., Korotkiy, M., Omelayenko, B., Kartseva, V., Zykov, V., Klein, M., et al. (2002). Goldenbullet: Automated classification of product data in e-commerce. In *Proceedings of the 5th international conference on business information systems*.
6. Dos Santos, C. N., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *COLING* (pp. 69–78).
7. Goldberg, Y., & Levy, O. (2014). word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722.
8. Gottipati, S. (2012). E-commerce product categorization srinivasu gottipati and mumtaz vauhkonen. Stanford C229 Final Projects.
9. Hashimoto, K., Stenetorp, P., Miwa, M., & Tsuruoka, Y. (2015). Task-oriented learning of word embeddings for semantic relation classification. arXiv preprint arXiv:1503.00095.
10. Hull, D. A., et al. (1996). Stemming algorithms: A case study for detailed evaluation. *JASIS*, *47*(1), 70–84.
11. Ju, R., Zhou, P., Li, C.H., & Liu, L. (2015). An efficient method for document categorization based on word2vec and latent semantic analysis. In *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing (CIT/IUCC/DASC/PICOM)* (pp. 2276–2283). IEEE.
12. Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
13. Kim, Y. G., Lee, T., Chun, J., & Lee, S. G. (2006). Modified naïve bayes classifier for e-catalog classification. In *Data engineering issues in e-commerce and services* (pp. 246–257). Springer.
14. Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD* (vol. 96, pp. 202–207). Citeseer.
15. Kononenko, I. (1993). Inductive and bayesian learning in medical diagnosis. *Applied Artificial Intelligence an International Journal*, *7*(4), 317–337.

16. Kozareva, Z. (2015). Everyone likes shopping! multi-class product categorization for e-commerce. In *HLT-NAACL* (pp. 1329–1333).
17. Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *ICML* (vol. 14, pp. 1188–1196).
18. Lee, H., Lim, E., Cho, Y., & Yoon, Y. (2016). Automatic classification of product data for natural general-purpose o2o application user interface. In *The 2016 fall conference of the KIPS* (pp. 382–385).
19. Lee, J. H., Ha, J., Jung, J. Y., & Lee, S. (2013). Semantic contextual advertising based on the open directory project. *ACM Transactions on the Web (TWEB)*, *7*(4), 24.
20. Lee, Y. E., & Benbasat, I. (2003). Interface design for mobile commerce. *Communications of the ACM*, *46*(12), 48–52.
21. Liu, Y., Liu, Z., Chua, T. S., & Sun, M. (2015). Topical word embeddings. In *AAAI* (pp. 2418–2424).
22. Lu, S. H., Chiang, D. A., Keh, H. C., & Huang, H. H. (2010). Chinese text classification by the naïve bayes classifier and the associative classifier with multiple confidence threshold values. *Knowledge-Based Systems*, *23*(6), 598–604.
23. Ma, C., Xu, W., Li, P., & Yan, Y. (2015). Distributional representations of words for short text classification. In *VS@ HLT-NAACL* (pp. 33–38).
24. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
25. Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., et al. (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, *24*(4), 694–707.
26. Panetto, H., Dassisti, M., & Tursi, A. (2012). Onto-pdm: Product-driven ontology for product data management interoperability within manufacturing process environment. *Advanced Engineering Informatics*, *26*(2), 334–348.
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.
28. Perez, S. (2014). Etsy moves further into the offline world with launch of card reader for in-person payments. https://techcrunch.com/2014/10/23/etsy-moves-further-into-the-offline-world-with-launch-of-card-reader-for-in-person-payments/.
29. Ren, Y., Wang, R., & Ji, D. (2016). A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences*, *369*, 188–198.
30. Ren, Y., Zhang, Y., Zhang, M., & Ji, D. (2016). Context-sensitive twitter sentiment classification using neural network. In *AAAI* (pp. 215–221).
31. Robertson, S. (2004). Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, *60*(5), 503–520.
32. Rong, X. (2014). word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.
33. Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *Learning for text categorization: Papers from the 1998 workshop* (vol. 62, pp. 98–105).
34. Scholl, N. B., Crawford, J., & Puckett, J. (2013). Online ordering for in-shop service (2013). US Patent App. 13/839,414.
35. Staykova, K. S., & Damsgaard, J. (2016). Platform expansion design as strategic choice: The case of wechat and kakaotalk. http://aisel.aisnet.org/ecis2016_rp/78.
36. Tang, D. (2015). Sentiment-specific representation learning for document-level sentiment analysis. In *Proceedings of the eighth ACM international conference on web search and data mining* (pp. 447–452). ACM.
37. Wang, Q., Garrity, G. M., Tiedje, J. M., & Cole, J. R. (2007). Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Applied and Environmental Microbiology*, *73*(16), 5261–5267.
38. Yang, X., Macdonald, C., & Ounis, I. (2016). Using word embeddings in twitter election classification. arXiv preprint arXiv:1606.07006.