Offer Categorization for Price Comparison Websites: Word Embedding Approaches

Rodolpho Rosa[1], Eraldo Fernandes[2], Eduardo Motta[3], Eduardo Akira[3], Rodrigo Guarino[4], and Leandro G. M. Alvim[1]

[1] DCC/IM, Universidade Federal Rural do Rio de Janeiro, Nova Iguaçu - Rio de Janeiro, Brazil
{rodolphorosa,alvim}@ufrrj.br
[2] Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande - Mato Grosso do Sul, Brazil
eraldo@facom.ufms.br
[3] Buscapé Company, São Paulo - SP, Brazil
ed@eduardomotta.com
eakinto@buscapecompany.com
[4] The Hive Brasil Labs
rg@hivebr.com **

Abstract. One key task for price comparison websites is to categorize offers collected from online stores. Classification accuracy impact on searching, recommendation, and website reputation. There are few studies on this topic in the literature, and none of them apply the successful technique of word embedding nor perform a detailed analysis of features. In this work, we compare two different word embedding approaches with the traditional bag-of-words approach, for the task of offer categorization. Firstly, we employ an unsupervised approach in which the embedding is learned from millions of offers using the well known *word2vec* tool. Secondly, we develop a supervised approach in which the embedding and the offer classifier comprise a Convolutional Neural Network (CNN) and both are *jointly* learned. Additionally, we perform a detailed analysis of several features regarding their relevance to offer categorization. We assess our models on a dataset comprising more than 11 million offers collected and manually annotated by the most popular Latin American price comparison website. In our experiments, the CNN model substantially outperforms the other models. We present detailed experimental results that highlight the contribution of different parts of the CNN model. Regarding feature engineering, we notice that all evaluated offer attributes contribute to enhance the classifier performance.

Keywords: Word Embedding, Convolutional Neural Network, Machine learning

** During the production of this paper, Rodrigo Guarino worked at Buscapé Company. Currently, he works at The Hive Brasil Labs.

1 Introduction

A price comparison website (PCW) is a specific search engine used to filter and compare offers from different retailers regarding some criteria such as: price, product characteristics and retailer reputation. PCWs provide services of recommendation, rating and review of products, which have become fundamental services for Internet users. A key aspect of a PCW system is offer categorization, because it impacts on searching, comparison and even recommendation of offers [1, 8, 5, 7, 12]. Given the great volume of offers produced by online retailers nowadays, it is impractical to perform this task manually. Thus, to build accurate offer classifiers is highly valuable.

In this work, we investigate different aspects of the offer categorization task. First, we perform an analysis of the impact of offer attributes on classification performance. PCW systems extract several offer attributes, and these attributes may be useful for categorization. However, attribute values are absent for some offers, and the real impact of each attribute was unclear. Our results indicate that all evaluated attributes are useful.

Another important contribution of this work is a detailed study of different representation methods. The performance of machine learning algorithms is highly dependent on input representation. The most relevant attribute of an offer is its *textual* description and, recently, deep learning approaches have been successfully employed to represent textual data. In this work, we investigate a deep learning network equipped with a word embedding layer followed by a convolution layer. As far as we know, this is the first work to employ word embeddings and convolutional neural networks to offer categorization. We investigate different aspects of both the convolution and the word embedding; comparing variations of our model with a strong bag-of-words baseline.

As stated above, to manually classify all the incoming offers is impractical for a PCW. On the other hand, PCWs still need to constantly measure the classifier performance, since it is impossible to guarantee the accuracy of an automatic classifier on unseen offers. A practical method to perform this assessment consists of manually classifying offers for which the classifier has low confidence on its classification. The confidence threshold must be tuned to balance between human effort (number of offers to be manually classified) and classifier quality (accuracy on high confidence classifications). We present an assessment of our classifier regarding the confidence threshold and its impact on this balance.

We perform our empirical studies on a large dataset comprising more than 11 million offers collected and manually categorized by the Buscapé Company (the most popular PCW in Latin America). Buscapé uses a hierarchical categorization composed by almost four thousand categories. However, we restrict ourselves to classify offers among the 25 top categories from the original hierarchy. We think this is enough for the kind of study we are performing here. In the future, we plan to consider the whole hierarchy, but this involves considering more complex metrics that take into account this kind of structure. Our best model (a convolutional neural network) achieves 96.82% on accuracy and 93.77% on macro F-score.

This work is organized as in the following. In Section 2, we discuss some previous works related to ours. In Section 3, we describe the proposed machine learning models along with the feature engineering process. Next, in Section 4, we present the empirical setup and the most important experimental findings. Finally, in Section 5, we present our final considerations and some promising research directions.

2 Related Work

In [14], the authors propose the AutoCat system for product classification. This model uses a variation of the vector spatial model so that attributes such as description, store name and category can be well represented. These features are combined through a weight optimization method in order to generate a ranking where the highest weight category is assigned a product. The system operates on a sparse array containing references of terms indexed by a list of categories in which they were observed. In this way, products are grouped into category vectors. The best result was 79.5% accuracy using all studied features. Price was the attribute that contributed least for results.

In [11], the authors solve the problem of offer categorization on Yahoo! Shopping products using the *Naive Bayes* algorithm. However, due to the structure of product representation and the high number of categories, this problem is still a challenge for batch classifiers such as *SVMs*. The authors overcome the limitations of this methodology by studying the efficiency of transformations in the data using a classifier *Naive Bayes*. A series of experiments with different transformations was performed on real data sets containing just under 100,000 products and 58 categories. The results indicated that some of them were able to leverage significantly the accuracy of the classifier.

In [6], the authors present a study on the categorization of offers using a probabilistic model. The approach used looks for the best combination of an unseen supply, represented by its attributes or characteristics, and the categories in a given taxonomy, represented by the attributes / characteristics of the known offers that belong to those categories. The offers are composed by the fields: description, price and name of the store. Thus, a probabilistic calculation is made on each of these fields. These probabilities are then combined to estimate the probability of belonging to each category. In this way, an offer is assigned to the category with the highest probability. Experiments were performed using two sets of real-world data. The authors concluded that the price and store name are useful in the sorting process but provide little improvement when compared to the description of the offer.

3 Machine Learning Models

The most important data of an offer is its textual description. Thus, in order to apply machine learning algorithms to solve offer categorization, one needs to decide how to represent offer descriptions. In this section, we describe the machine learning models developed in this work. We organize our models into three categories: *BOW* – bag of words, *UWE* – unsupervised word embedding, and *CNN* – convolutional neural network. The main difference among these models is the representation of offer descriptions. In this section, we also describe the additional features used with our models.

3.1 Bag of Words

The BOW model is probably the most popular approach used to represent textual data to feed in machine learning algorithms. In this technique, an input text is represented as a binary vector on a N-dimensional space, where N is the number of words occurring in the data (which is usually in the order of hundreds of thousands). Each word in the vocabulary is associated with an index in the input vector. Then, given an offer description, its representation comprises a sparse vector where the indexes of every word within the description has value equal to one, and the remaining indexes have value equal to zero. The main drawback of this technique is sparsity, since most offer descriptions comprise less than 15 words, while the vector size is in the order of hundreds of thousands.

3.2 Unsupervised Word Embedding

Word embedding approaches have been successfully applied to several problems that involve textual data [4, 2]. The basic idea is to *embed* the original BOW space into a much lower dimensional space. In this space, each word is represented by a dense real-valued vector. For most tasks, the word embedding size (the size of the word vectors) ranges from some dozens to some hundreds; which is much lower than the original BOW space. This idea is a type of dimensionality reduction, since the embedding can be computed from a large corpus of unlabeled text (using BOW representation, for instance). The *word2vec* tool[5] [9, 10] implements a very efficient algorithm to *unsupervisedly* learn a word embedding from a large corpus. Basically, given a large corpus (millions of words), *word2vec* learns word vectors that are good to discriminate real contexts (sequence of words present in the corpus) from noisy contexts (contexts obtained by replacing some word within the real sequence by a random word). By employing a word embedding technique, we avoid the sparsity issue present in BOW models. However, the use of such embeddings is not straightforward, because the description length varies from offer to offer, and machine learning algorithms require a fixed length representation. To tackle this issue, we concatenate the vectors representing the first k words in an offer description. Then, we feed this representation to a SVM algorithm. We call this approach *unsupervised word embedding* (UWE).

3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) have been around for some decades now and have been applied for many image problems. A CNN can also be applied

[5] https://code.google.com/archive/p/word2vec/

to sequential data like texts [3]. In such contexts, these networks are also called Time-Delay Neural Networks. The key aspect of CNNs is space invariance. In the case of texts, this means a CNN can *learn* features that are invariant to word position within an input text. For instance, a CNN is able to learn that the term *Tênis* is highly correlated to the category *Fashion and Adornments*, regardless of its position within the offer description. This is a crucial advantage of CNN in comparison to UWE.

In this work, we employ a CNN-based model to offer categorization. We depict such model in Figure 1. This model is a deep learning model that includes

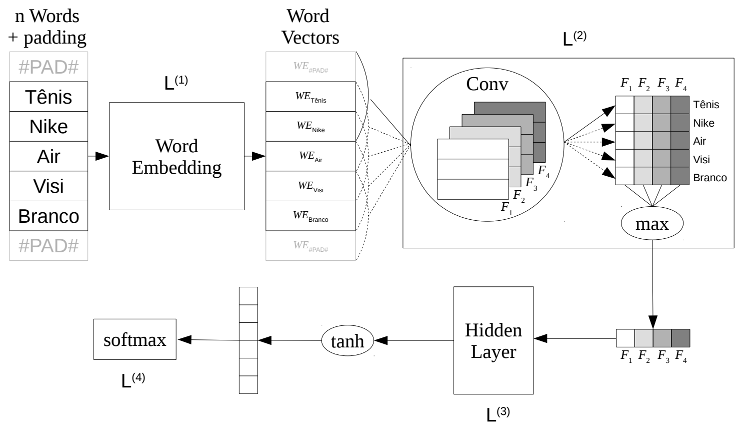

Fig. 1: CNN model applied to the text *Tênis Nike Air Visi Branco*.

a convolutional layer. The network input is an offer description comprised by n words augmented with $t-1$ *padding* words in the beginning and in the end of the text. A word embedding layer ($\mathrm{L}^{(1)}$) takes this sequence as input and outputs an $(n+t-1) \times d$ matrix comprised by the concatenated vectors of all $n+t-1$ words in the input sequence. Each word vector lies in R^d. Then, the convolutional layer $\mathrm{L}^{(2)}$ scans the word sequence and applies a set of r filters $(F_1, F_2, \ldots, F_r)$ for each *real* word. In fact, the input for each filter is a window of t consecutive words (in the figure, $t = 3$), comprising a $t \times d$ matrix. Each filter is also a $t \times d$ matrix that is multiplied to each word window in an element-wise fashion, and the multiplied values are all summed up to result in a unique value for each combination of word window and filter. Thus, after applying r filters along n word windows, we end up with an $n \times r$ matrix. Since n varies from offer to offer, we apply a max pooling operation over this matrix. This operation selects, for each filter, the maximum value along all word windows. Therefore, the output of the max pooling operation comprises r values, one for each filter. This vector is then fed to an ordinary hidden layer with a hyperbolic-tangent activation function. Finally, a softmax layer outputs the network prediction: a probability distribution over the categories.

The parameters of all these layers can be jointly trained by backpropagating a loss value by means of stochastic gradient descent (or any other learning algorithm). The learned parameters may include even the word embedding. This is another key advantage of CNN over UWE. While in the latter approach the word embedding is learned in an unsupervised fashion and kept fixed along the supervised training phase; in the former, the word embedding can be updated using supervised feedback, which is very beneficial.

3.4 Feature Engineering

An offer consists of a set of attributes that describe a product. In our dataset, for instance, an offer with the description *Tênis Nike Air Visi Branco* includes a category (*Fashion and Adornments*), a price (*199.99*), a store name (*Dafiti*) and a store category (*Shoes*). In this work, we investigate how different features based on these attributes influence on the accuracy of the classification. More specifically, we use the features: (i) *offer description*, (ii) *first word in the description*, (iii) *bigrams of the description*, (iv) *store category*, (v) *category whose average price is the closest to the offer price*, and (vi) *store name.*

4 Experiments

In this section, we present the used dataset, describe the empirical environment, and discuss our main findings.

4.1 Dataset

We use a large dataset provided by The *Buscapé Company*, a real price comparison website. It comprises more than eleven million offers distributed among 25 different categories. In Figure 2, we present the distribution of these offers along these categories. We notice a great unbalance among these categories. For instance, the four most frequent ones add up to more than 50% of the offers; while the five least frequent ones[6] (which we summarize in *Others*) add up to less than 1%. Regarding the descriptions of the offers, their number of words vary between 1 and 61, with an average of 8 words; the standard deviation is about 4 words, while the mode is equal to 6 words.

Since this dataset is very large, in order to tune model parameters and assess our models, we perform a holdout validation. The original dataset is split into two sets: (i) *Train*, comprising 90% of the offers which are used for training the final models (after model selection) and also for performing model selection; and (ii) *Test*, comprising the remaining 10% of the offers which are used for evaluating the final models. In order to perform model selection, we further divide the *Train* split into two sets: (i) *Train2*, with 80% of the *Train* split for

[6] Vinyl (0.59%), Religious Articles (0.14%), Art and Antiquity (0.10%), Tobacco (0.04%) and Digital Musical ($< 0.01\%$).

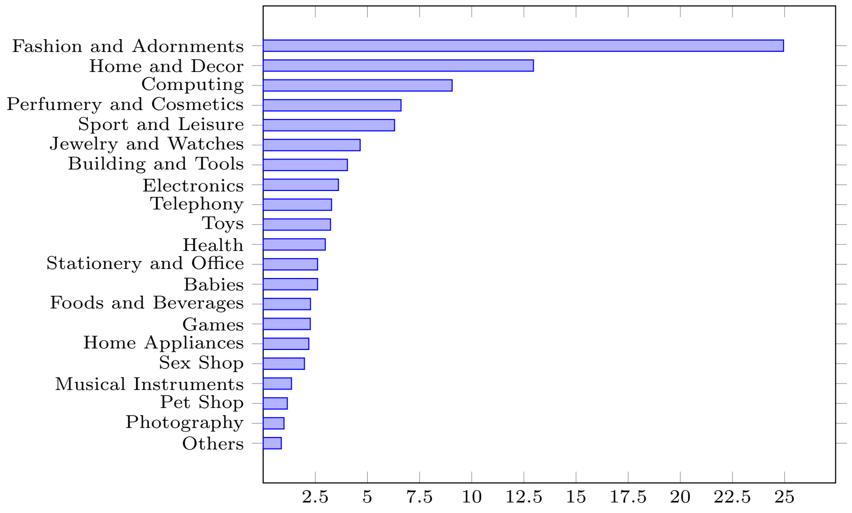

Fig. 2: Percentage of offers per category along the dataset.

training during the model selection phase; and (ii) *Validation*, with 20% of the *Train* split for validation during the model selection phase. In the following, all models are trained on *Train2* and evaluated on *Validation*.

4.2 Training

For the experiments with BOW and UWE representation models, we opted for SGDClassifier (*Stochastic Gradient Descent Classifier*), available in *scikit-learn*. *Modified Huber loss* was used as cost function. For the CNN classifier, we maximize the log-likelihood given by the softmax layer and update the parameters using ordinary backpropagation by means of the Theano framework [13]. Regarding the CNN classifier, we opted for 2 epochs and learning rate of 0.05%. We adopted windows of 5 words and 100 convolution filters. Offers' description words are represented as 100-dimensional word embeddings, while each additional feature is included as a 50-dimensional one-hot vector.

4.3 Feature Assessment

The most important feature for offer categorization is definitely the offer description. However, as seen in Section 3, other features may be relevant to predict the categorize an offer.

In order to evaluate these features, we provide each one of them for the classifier separately. Our baseline is an classifier that uses the offers' description alone. BOW is used to represent the offers' description, while every additional feature is included in the classifier using one-hot representation. In Figure 3, we

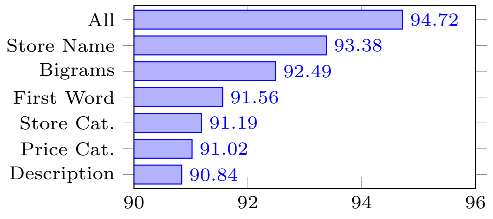

Fig. 3: Impact of different features on the BOW model accuracy.

present the impact of each feature in the accuracy of the BOW classifier. We also present the impact of using all features together.

As one can notice, the *store name* is the most relevant feature, increasing accuracy by 2.54%. The use of *bigrams* increases accuracy by 1.65%. This shows that word context is relevant; even a very limited context like this one. The *first word* feature increases accuracy by 0.72%. The *store category* feature increases accuracy by 0.35%. The *price category* feature is marginally relevant, improving accuracy by only 0.18%. Finally, when considering all features in the same model, we achieve a significant improvements of 6.5%. From these experiments, we observe that all features are useful; but the *store name* and *bigrams* features are much important than the others.

4.4 Unsupervised Word Embedding

As described in Section 3, unsupervised word embedding approaches have been successfully applied to many tasks. In this work, we employ the *word2vec* tool in order to unsupervisedly learn a word embedding from the *Train2* split of the *Buscapé* dataset.

In this section, we first analyse the impact of the word vector length on the UWE model performance. In Figure 4, we report the accuracies of the UWE model using different sizes for the word vectors. As we can see, the increase in performance is remarkable as the number of dimensions increases. Accuracy faces steady growth between 40 and 120 dimensions. From that point on, the increase slows down. Hence, for UWE models, vector size is directly related to model accuracy. However, the impact becomes less noticeable as the vector size increases.

Since the UWE model uses only the k first words of an offer description, we additionally assess the impact of different values for k. As a baseline, we apply the BOW model using only the k first words in the offer description. In Figure 5, we present the accuracies of both models, UWE and BOW, for $k = 3, 6, 9$. Moreover, the difference between the two models increases as k increases. We speculate that this behaviour is due to the fact that most of the offers comprise up to six words. Since the representation of short descriptions is extended with artificial words, large values of k can hurt UWE performance.

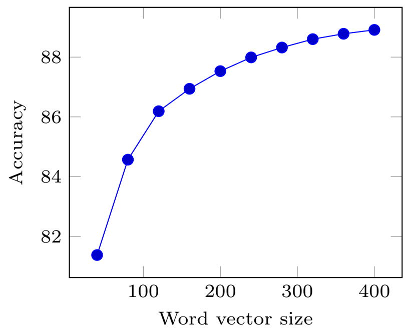

Fig. 4: Impact of word vector size on the UWE model accuracy.

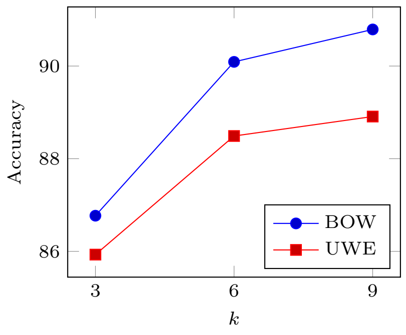

Fig. 5: Accuracies of UWE and BOW models using the k first words in an offer description.

4.5 Models Comparison

Now, we report on the experiments with six different classifiers. The BOW, UWE, CNN and CNN/UWE classifiers are trained with the offers' description alone. While the CNN classifier uses supervisedly learned embeddings as representation model, the CNN/UWE uses the embeddings generated by *word2vec*. Afterwards, we provided BOW and CNN with additional features. We trained the BOW+ftrs with all features presented in section 3.4; whereas CNN/UWE uses *store name* and *store category* as additional features. In Figure 6, we present the accuracy of these classifiers. We can see again that the BOW classifier substantially outperforms the UWE one by almost 2% on accuracy when only offer description is used.

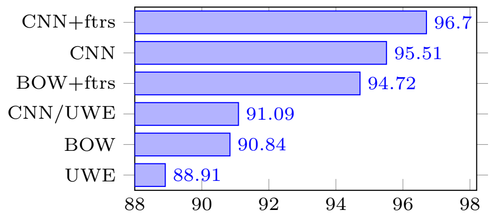

Fig. 6: Accuracy comparison of different classifiers.

One of the most restrictive aspects of the UWE model regards its locality dependence. This model input comprises the concatenated vectors of the first k words in the offer description. This means that when the same word occurs in different positions of the description, the classifier understands it as completely different inputs. In order to understand the implications of this aspect, we modify our CNN model by removing the hidden layer and also avoiding to update the word embedding during the supervised learning. This model is quite similar to the UWE model. However, instead of using k fixed words as input, the model applies the convolution and the pooling operations over the whole offer description, learning r filters that are location independent. We denote this model CNN/UWE. We can see that this model improves more than 2% on accuracy over the UWE model and even outperforms the BOW model. This result confirms that the locality independence given by the convolution is very important.

The three topmost results in Figure 6 are related to BOW and CNN models. The CNN model is our complete convolutional model but using only the offer description as input. The BOW+ftrs model corresponds to the ordinary BOW model using the offer description and the additional features. Finally, the CNN+ftrs model includes the additional features as input for the CNN. We can see that the CNN model outperforms the BOW+ftrs model even not using additional features. When we include the additional features, the CNN+ftrs model obtains a substantially higher accuracy than the BOW+ftrs model. These results demonstrate how superior the CNN model is compared to the traditional BOW model. Moreover, by comparing the CNN model with the CNN/UWE model, we can see how important is the supervised updates to the word embedding. Besides updating the word embedding using the supervised signal, the CNN model also includes a hidden layer. However, additional experiments (not reported here) show that the impact of the hidden layer on the CNN performance is quite small compared to the impact of the supervised updates.

4.6 Model Utility

The overall accuracy is an important metric because it gives *one* value that allows us to directly compare different models. However, one important aspect of a practical model is its individual performance on every category. A useful model should not present poor performance on any category. In Figure 7, we present the F1 metric achieved by the CNN+ftrs model for each category (sorted by F1 values). By comparing these results with the category frequencies from Figure 2, we can observe that the performance is usually lower for less frequent categories, which is expected. More importantly, the only category with F1 below 80% is *Digital Music*.

In Figure 7, we include two vertical lines indicating the model accuracy (blue line) and macro F1 (red line). The achieved macro F1 value is more than 3 points lower than the accuracy value. This is mainly due to the F1 value achieved on the *Digital Music* category. We even crop the figure because this category F1 value is only 23.52%. This value is much lower than the rest and is not an acceptable performance for a practical system. However, this is a degenerated category, since its frequency on the whole dataset is approximately 0.0006%. This is a too scarce concept to learn from and to evaluate a model on. For instance, if we remove this category from the dataset, we achieve a macro F1 value of 96.70%, which is much closer to the model accuracy.

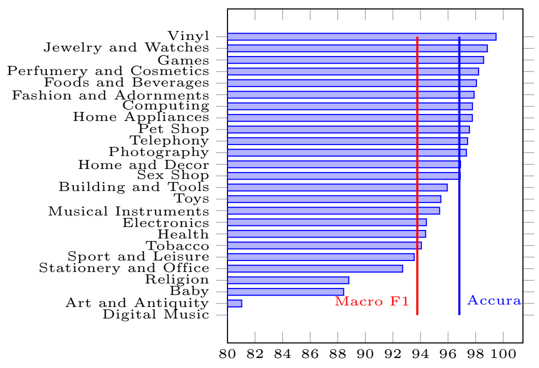

Fig. 7: F1 per category for the CNN+ftrs model.

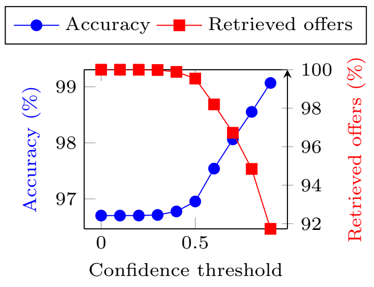

Fig. 8: Accuracy (left axis) and percentage of offers retrieved (right axis) by the CNN+ftrs model for different values of the confidence threshold.

Another practical aspect of a categorization model is that the PCW needs to constantly assess its performance. New offers are constantly arriving, and it is impossible to completely guarantee the model performance on unseen offers. One common approach is to manually categorize low confidence classifications. In Figure 8, we vary the confidence threshold of the CNN+ftrs model and plot two values: the model accuracy for high confident offers (left y-axis) and the percentage of such offers over all offers (right y-axis). We can see that the confidence threshold has almost no effect below 0.3.

5 Conclusions

In this work, we investigate the offer categorization problem in the context of price comparison websites. We firstly explore the characteristics of the dataset and analyse the usefulness of several characteristics of an offer. The store name turns out to be a very strong feature. Although the other features are less effective, all of them impact positively on the classifier performance, and a model using all features together achieves a significant improvement.

We also investigate the application of word embedding methods and compare them with the traditional bag-of-words model. We evaluate two word embedding approaches. The first one learns the word embedding in an unsupervised way from millions of offer descriptions. The second one supervisedly learns the word embedding by means of a convolutional neural network. The CNN model substantially outperforms our strongest BOW model. We experimentally show that the locality independence given by the convolution layer and the supervised updates of the word embedding are key aspects of the CNN model.

In this work, we ignore the hierarchical structure of the categories and use only the 25 top categories in the dataset. We argue that this decision does not invalidate our findings. Nonetheless, we recognize that categorizing offers within a hierarchical structure is indeed relevant to PCWs. In the near future, we plan to extend our work to this scenario. Another interesting research direction is to include offer images in the classification task. Usually, offers include product images that carry relevant information.

References

1. Baron, J.P., Shaw, M.J., Bailey Jr, A.D.: Web-based e-catalog systems in b2b procurement. Communications of the ACM 43(5), 93–100 (2000)
2. Bengio, S., Heigold, G.: Word embeddings for speech recognition. In: Proceedings of the 15th Conference of the International Speech Communication Association, Interspeech (2014)
3. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning. pp. 160–167. ICML '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1390156.1390177
4. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. 12, 2493–2537 (Nov 2011), http://dl.acm.org/citation.cfm?id=1953048.2078186
5. Cortez, E., Herrera, M.R., da Silva, A.S., de Moura, E.S., Neubert, M.S.: Lightweight methods for large-scale product categorization. JASIST 62(9), 1839–1848 (2011), http://dx.doi.org/10.1002/asi.21586
6. Cortez, E., Rojas Herrera, M., da Silva, A.S., de Moura, E.S., Neubert, M.: Lightweight methods for large-scale product categorization. Journal of the American Society for Information Science and Technology 62(9), 1839–1848 (2011)
7. Kwon, I.H., Kim, C.O., Kim, K.P., Kwak, C.: Recommendation of e-commerce sites by matching category-based buyer query and product e-catalogs. Computers in Industry 59(4), 380–394 (2008)
8. Leukel, J., Schmitz, V., Dorloff, F.D.: Modeling and exchange of product classification systems using xml. In: Advanced Issues of E-Commerce and Web-Based Information Systems, 2002.(WECWIS 2002). Proceedings. Fourth IEEE International Workshop on. pp. 242–244. IEEE (2002)
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
11. Pavlov, D., Balasubramanyan, R., Dom, B., Kapur, S., Parikh, J.: Document preprocessing for naive bayes classification and clustering with mixture of multinomials. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 829–834. ACM (2004)
12. Pu, P., Chen, L., Kumar, P.: Evaluating product search and recommender systems for e-commerce environments. Electronic Commerce Research 8(1-2), 1–27 (2008)
13. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688 (May 2016), http://arxiv.org/abs/1605.02688
14. Wolin, B.: Automatic classification in product catalogs. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 351–352. ACM (2002)