Effective Products Categorization with Importance Scores and Morphological Analysis of the Titles

Leonidas Akritidis, Athanasios Fevgas, Panayiotis Bozanis
Data Structuring & Engineering Lab
Department of Electrical and Computer Engineering
University of Thessaly
Volos, Greece
Email: {leoakr,fevgas,pbozanis}@e-ce.uth.gr

***Abstract*—During the past few years, the e-commerce platforms and marketplaces have enriched their services with new features to improve their user experience and increase their profitability. Such features include relevant products suggestion, personalized recommendations, query understanding algorithms and numerous others. To effectively implement all these features, a robust products categorization method is required. Due to its importance, the problem of the automatic products classification into a given taxonomy has attracted the attention of multiple researchers. In the current literature, we encounter a broad variety of solutions, ranging from supervised and deep learning algorithms, as well as convolutional and recurrent neural networks. In this paper we introduce a supervised learning method which performs morphological analysis of the product titles by extracting and processing a combination of words and n-grams. In the sequel, each of these tokens receives an importance score according to several criteria which reflect the strength of the correlation of the token with a category. Based on these importance scores, we also propose a dimensionality reduction technique to reduce the size of the feature space without sacrificing much of the performance of the algorithm. The experimental evaluation of our method was conducted by using a real-world dataset, comprised of approximately 320 thousand product titles, which we acquired by crawling a product comparison Web platform. The results of this evaluation indicate that our approach is highly accurate, since it achieves a remarkable classification accuracy of over 95%.**

I. Introduction

It is common knowledge that e-commerce is one of the fastest growing Web-based enterprises. In the last few years there has been a significant increase in the e-commerce share of total global retail sales. From 7.4% in 2015, the percentage of the online sales has increased to 11.9% in 2018, and it is predicted to reach 17.5% at the end of 2021[1]. Consequently, the research topics related to e-commerce have been rendered particularly important. More specifically, the effective and efficient management, mining, and processing of the products data are presently of top priority for the leading e-commerce platforms.

One of the most fundamental problems in this area is the automatic classification of products into an existing hierarchy of categories. The successful solution of this problem can lead to numerous novel applications, including query expansion and rewriting, retrieval of relevant products, personalized recommendations, etc. On the other hand, the current large-scale commercial systems are now offering tens or even hundreds of millions of products and their warehouses are getting updated constantly at high rates. Therefore, the approach of manual categorization is not scalable and automatic classification emerges even more compelling.

Due to its importance, the problem in question has gained a lot of research attention recently. The methods which we encounter in the relevant literature take into consideration the product titles and/or their textual descriptions to train their classification models. A number of approaches treat the issue of product classification as a standard short-text classification problem and they apply slight modifications of the proposed algorithms. Other methods perform morphological analysis of the titles and the textual descriptions by extracting words and n-grams. Finally, a third family of solutions employ deep learning approaches, which are based on convolutional or recurrent neural networks.

A significant portion of these methods take into consideration additional information –known as meta-data– about a product and extract features from brands, technical specifications, or textual descriptions. Nevertheless, these types of metadata are not always present, whereas, even in case they exist, they are occasionally incomplete or incorrect. In addition, although many models have been effectively applied to text classification, they are difficult to be applied to product categorization due to their lack of scalability, and the sparsity and skewness of the data contained in the commercial product catalogs.

In this paper, we present a supervised algorithm to address this interesting problem. Similarly to some of the existing methods, our approach is based solely on the titles of the products. However, in contrast to the existing methods, our algorithm does not employ standard n-grams of fixed length; instead, it sets a parameter N and extracts all the $1, 2, \ldots, N$ -grams from the titles. All generated n-grams are stored within a dictionary data structure which, for each token, maintains a series of useful statistics. We show that this strategy leads to significant performance benefits.

As might be expected, a portion of the generated n-grams are strongly correlated with a category, whereas others are

[1]https://www.statista.com/statistics/534123/e-commerce-share-of-retail-sales-worldwide/

not. For instance, the unigram *Apple* is not a strong indicator of a single category, since it can be correlated with multiple potential classes such as *Computers*, *Tablets*, *Mobile Phones*, and so on. On the other hand, the bigram *Apple iPhone* is indeed indicative of the category of a product (*Mobile Phones*). Based on this notification, we assign importance scores to each token. The importance score essentially reflects the informational value of a token when it is used to determine the class of a product. During the evaluation of the method, we experimented with multiple scoring functions and we propose one which boosts the classification accuracy of our method.

Finally, the extraction of multiple types of n-grams undoubtedly leads to an increase of the size of the aforementioned dictionary data structure. In this paper we propose a strategy which prunes our dictionary based on the importance scores of the tokens. That is, each token which receives a low importance score is considered of no informational value and it is removed from the dictionary. This method reduces the dimensionality of the feature space and leads to a much more compact model, whereas the degradation of the classification performance is proved infinitesimal.

The rest of the paper is organized as follows: Section II, contains an overview of the current state-of-the-art methods in the area. Section III, describes in details the characteristics of the model, the importance scores and the model training and testing algorithms. The experimental evaluation of our proposed method is presented in Section IV, whereas Section V summarizes the conclusions of the paper.

II. Related Work

Due to its importance, the problem of automated product classification has attracted multiple researchers and is being studied systematically, particularly during at the last 5 years. One of the first approaches in this area of research was GoldenBullet, an early system which applied a combination of IR and machine learning techniques (such as Naive Bayes and k-nearest neighbor), to classify the products of an e-commerce platform into a pre-defined taxonomy structure such as UNSPSC[2].

Supervised classification was also used in [1], where the authors introduced a multi-class SVM product classifier in combination with a cost-sensitive function. This classifier was evaluated by employing a part of the UNSPSC taxonomy. The first work which attempted to address the problem by exploiting only the product titles was [2]. This research introduced a binary classifier in combination with the One-Against-All (OAA) and Error Correction Tournament (ECT) strategies. The features for the model were extracted by using lexical information and topic discovery algorithms, such as Latent Dirichlet Allocation [3].

In addition, the work of [4] introduced an algorithm which constructed document bag-of-word vectors from the product descriptions by using traditional approaches based on tf-idf and wordVec. In [5], [6] the classification problem is decomposed into a coarse level task and a fine level task. In the sequel, simple classifier is applied for the coarse level, and a more sophisticated model is used to separate classes at the fine level. Then, a graph algorithm is used to discover groups of highly similar classes. In [7] the authors introduced an algorithm based on the combinations and permutations of the titles tokens to achieve unsupervised products clustering.

During the past few years there is an ongoing line of research which attempts to solve the problem by employing deep learning approaches. More specifically, [8], [9] demonstrated that the character-level convolutional networks are at least competitive against the well-established models such as bag of words, word-based ConvNets and recurrent neural networks. Furthermore, [10] highlighted the effectiveness of the simple convolutional neural networks with little hyperparameter tuning and static vectors. The authors also used task-specific vectors and reported further gains in performance through careful fine-tuning.

Moreover, [11] modeled short texts by using semantic clustering and convolutional neural networks. Particularly, the meaningful multi-scale semantic units are initially detected by a clustering algorithm. In the sequel, they are fed into a convolutional layer, followed by a max-pooling operation. Two other articles, [12] and [13] focused on large-scale deep learning text classification. The former introduced a two stage algorithm which consisted of a search stage and classification stage. On the other hand, the model which was proposed in the latter included multiple recurrent neural networks.

Finally, a comparative study which examines the current machine learning techniques for automatic product categorization is provided in [14].

III. Supervised Products Categorization

In this section, we present the characteristics of the proposed method for products categorization. Initially, we provide some preliminary elements and we establish the theoretical background of the algorithm. In the sequel, we describe the properties of the model and we discuss the training and testing procedures.

A. Theoretical Background

Our analysis begins with the introduction of the taxonomy Y, that is, the hierarchical structure which stores all the categories (also called classes, or labels) in the dataset. This hierarchy is usually represented by a tree data structure which organizes the elements of Y into various levels of subcategories. More specifically, the categories tree contains two types of nodes: i) the parent nodes, which may include any number of subcategories; and ii) the leaf nodes, which do not have any children.

Furthermore, we consider a set $X = \{x_1, x_2, \ldots\}$, which includes all the products in the dataset. There exist various methods to categorize the products of X into the classes of Y. The most common scenario in the current e-commerce platforms dictates that each product x_i is mapped to only one *leaf* category $y_j \in Y$. In other words, a product cannot be

[2]United Nations Standard Products and Services Code, https://www.unspsc.org/

assigned multiple categories, and, additionally, it cannot be mapped to a parent category. The full categorization of a product constitutes a path from the root of the aforementioned tree to the leaf category where the product belongs to. For example, *Computers/Hardware/CPUs* may represent a categorization of three levels for a computer processor.

Each product in the dataset is described by its title τ, which consists of a sequence of several words $(w_1, w_2, \ldots, w_{l_x})$, where l_x denotes the length of the title of x in words. In the real-world commercial environments, a product may also be accompanied by extended descriptions, or other types of explicitly stated meta-data such as brand names, manufacturers etc. However, such information is occasionally not consistent (i.e., it is not provided for all products), incomplete, inaccurate, or noisy. Consequently, in this paper we introduce a supervised algorithm which builds a classification model based on a set of previously classified products, by taking into consideration the provided titles only.

B. Model Properties

In this subsection, we describe in details the characteristics of the product categorization model and we present the feature extraction method, as well as the training and testing algorithms. Most works in the area construct correlations of the title words with the categories of the given taxonomy by applying tf-idf or similar methods. However, a single word may be correlated with multiple categories (in other words, it may be ambiguous), or it may incorrectly indicate a strong relationship with a particular category. As stated earlier, we anticipate that the word *Apple* will be proved ambiguous, since it is strongly correlated with multiple categories such as *Computers*, *Tablets*, *Mobile Phones*, and others. The same phenomenon is expected to occur multiple times, especially for brand names, color and size descriptors, capacities, etc.

For this reason, in our work we also employ n-grams; that is, sequences of n successive words in the title of a product. Compared to the single title words, the n-grams have a lower possibility of being ambiguous. In the previous example, we explained that the word *Apple* may be proved problematic due to its correlation with multiple categories. Nonetheless, the bigrams *Apple iPhone* and *Apple iPad* should be much more specific. We anticipate that the former will be frequently correlated with the category *Mobile Phones*, whereas for the later, the category which we primarily expect to encounter is *Tablets*. In the discussion which follows, we will collectively refer to both n-grams and words by using the term *tokens*.

The previous discussion reveals that the generated tokens may be correlated with one or more categories with a different manner. That is to say, each token t exhibits its own level of *ambiguity* with respect to a given category y. Therefore, it is required that we take into consideration how important a token is during the classification of a product. We quantify this importance by assigning an *importance score* I_t to each token t of a product title according to the following equation:

$$I_t = l_t \log \frac{|Y|}{f_t} \tag{1}$$

where l_t is the length of the token t in words, and f_t is a frequency value which represents the number of the categories which have been correlated with t. This definition embodies the following notations:

- A token which has been correlated with multiple categories is considered of low informational value and should receive a low importance score. The opposite holds for the tokens which frequently appear in combination with only one or two categories.
- The length of a n-gram (in words) is another parameter which contributes to its importance. The previous discussion demonstrated that bigrams are more important than unigrams. More generally, the longer the token, the higher the importance score should be.

Another point, which distinguishes our work from the current approaches, is that here we do not merely employ n-grams of constant length or of constant number. More specifically, we define a parameter N and we construct all the $1, 2, \ldots, N$ -grams. For example, for $N = 3$, the algorithm will generate all the unigrams, bigrams and trigrams of the words of a product title. As demonstrated later in the experimental evaluation, this approach leads to significant performance improvements of the algorithm.

Our proposed model consists of a lexicon data structure which stores all the distinct tokens of the product titles in the training set. Moreover, for each stored entry t, the lexicon maintains:

- its frequency value f_t;
- its importance score I_t; and
- a *relevance description vector (RDV)*, which, for each token-category relationship, includes a pair in the form $(y, f_{t,y})$, where y is the ID of the category and $f_{t,y}$ is another frequency value that reflects how many times t has been correlated with the category y.

This lexicon will assist us in classifying the products of the test set, based on the tokens of the observed products of the training set. Since it includes n-grams of multiple types, we expect that its size will grow quickly, as more samples are examined. For this reason, in Subsection III-E, we propose a dimensionality reduction technique for limiting its size.

C. Model Training

In Algorithm 1 we present a pseudocode of the training algorithm. For each product in the training set, we initially retrieve its respective category ID y, and its title τ. Before the extraction of the tokens, each single word in the title is processed in three ways (step 5): i) *casefolding* (all capital letters are converted to lower case); ii) *punctuation removal* (that is, removal of several unnecessary symbols such as commas, exclamation and question marks, ampersands, etc.); and iii) *stemming* (we compute and store the 'root' of a word instead of the word itself). Regarding stemming, we were particularly interested in studying the effects of eliminating the singular/plural differences in the words of the titles.

The n-grams are typically extracted by employing a window of length n, which captures sequences of n successive words.

Algorithm 1: Model training algorithm

```
1  initialize the lexicon L;
2  for each product x in the training set do
3      retrieve the category y;
4      extract the title τ;
5      perform linguistic processing of τ;
6      for each n ∈ [1, N] do
7          compute all tokens T_n of length n;
8          for each token t ∈ T_n do
9              if L.search(t) == false then
10                 L.insert(t);
11                 set f_t ← 1;
12                 L.insertRDV(t,y);
13                 set f_{t,y} ← 1;
14             else
15                 if L.searchRDV(t,y) == false then
16                     set f_t ← f_t + 1;
17                     L.insertRDV(t,y);
18                     set f_{t,y} ← 1;
19                 else
20                     set f_{t,y} ← f_{t,y} + 1;
21                 end
22             end
23         end
24     end
25 end
26 for each token t ∈ L do
27     set I_t ← l_t log(|Y|/f_t);
28     sort RDV of t in decreasing f_{t,y} order;
29 end
```

Algorithm 2: Categorization algorithm

```
1  for each product x in the test set do
2      initialize candidates list Y';
3      extract the title τ;
4      perform linguistic processing of τ;
5      for each n ∈ [1, N] do
6          compute all tokens T_n of length n;
7          for each token t ∈ T_n do
8              if L.search(t) == true then
9                  for each pair (y, f_{t,y}) in the RDV of t
                       do
10                     if Y'.search(y) == false then
11                         Y'.insert(y);
12                         set S_y ← f(I_{t,y}, Q_{t,y})
13                     else
14                         set S_y ← S_y + I_{t,y}
15                     end
16                 end
17             end
18         end
19     end
20     sort Y' in decreasing S_y order;
21     set y ← Y'[0];
22 end
```

To acquire all the tokens of a product title, this window progressively slides from the left to the right by taking steps of one word at a time. In addition, since we are interested in obtaining all $1, 2, \ldots, N$ -grams, it is required that we employ N such windows, with lengths ranging from 1 to N.

For each token t captured by the aforementioned window, we perform a search in the lexicon L (step 9). In case it is not present, we insert it and we set its frequency f_t equal to 1. We also initialize its RDV and we insert the pair $(y, f_{t,y})$ with $f_{t,y} = 1$ (steps 9–12). In the opposite case, where t already resides in the lexicon, we traverse its RDV by looking up for y. In other words, we initially search if t has been previously correlated with y. If it has not (i.e. y does not exist in the RDV of t), we initially increase f_t by one (step 16). In the sequel, we insert the the pair $(y, f_{t,y})$ with $f_{t,y} = 1$ in the RDV of t. Finally, in case y has been previously correlated with t, we merely increase the corresponding f_{t_y} value by one.

At this point, all tokens in the titles of the products have been inserted into the lexicon L. In the last three steps (28–31), we traverse L and, for each token $t \in L$, we perform two operations: i) we compute its importance score by applying equation 1; and ii) we sort its respective RDV in decreasing $f_{t,y}$ order. In this way, the strongest (t, y) correlations are moved towards the beginning of the list. This arrangement will assist us in the next phase (Subsection III-E) where we introduce a pruning strategy for reducing the size of the model.

D. Model Testing

Now we are ready to apply our trained model to categorize the unlabeled products of the test set. Algorithm 2 shows the basic steps of the categorization phase. At first, for each product x in the test set, a list of candidate categories Y' is initialized. In the sequel, the title τ is processed by following the same three steps of the training phase; i.e., casefolding, punctuation removal, and stemming.

We continue in a similar spirit to the training algorithm, and we extract all the tokens from the title of the product by applying the same sliding windows of length $[1, N]$. For each extracted token t, a search in the lexicon L is performed. In case it is successful, we retrieve the frequency f_t, the importance score I_t, and the entries of the respective RDV.

In the sequel, we traverse the RDV of t and for each category y we look-up the candidates list Y'. In case the look-up fails, we insert y in Y', and we compute a score $S_{t,y}$ based on the importance score I_t of each involved token and another quantity $Q_{t,y}$ (we will shortly present the details of the calculation of $S_{t,y}$). In the opposite case, we merely update S_y. After all the tokens have been examined, we sort the candidates list Y' in decreasing score order and we categorize the product x in the highest scoring category $Y'[0]$.

Now let us elaborate on the computation of the $S_{t,y}$ scores. Here we adopt a strategy similar to the ranking in Web

search engines and we define $S_{t,y}$ as a linear combination of the 'query independent' importance scores I_t and a 'query dependent' term $Q_{t,y}$ as follows:

$$S_{t,y} = k_1 I_t + k_2 Q_{t,y} \tag{2}$$

where k_1 and k_2 are two constant quantities. Regarding the $Q_{t,y}$ term, we experimented with dozens of functions and we found that the best accuracy is achieved by employing a variation of the well-established td-idf scoring scheme which has been proved quite effective in many classification tasks. The variation we used is given by the following equation:

$$Q_{t,y} = \log f_{t,y} \log \frac{|Y|}{f_t} \tag{3}$$

where $f_{t,y}$ is the aforementioned frequency value which stores the number of correlations of a token t with a category y.

E. Dimensionality Reduction & Model Pruning

The existence of multiple types of tokens in the aforementioned lexicon structure is translated to an increase in the overall number of features extracted by the model. This in turn leads to a high-dimensional feature space, which must be reduced to ensure the scalability of our solution in case large datasets are utilized. In this subsection, we present a method which effectively decreases the dimensionality of the involved feature space, and, also, it has a minimal impact on the classification effectiveness of the algorithm.

More specifically, our proposed dimensionality reduction approach takes into consideration the importance scores $I_{t,y}$ which we discussed earlier and preserves only the tokens whose importance score is greater than a pre-defined constant *cut-off threshold* C. The rest of the tokens are considered of low informational value (i.e. they are considered ambiguous) and are evicted from the model. The cut-off threshold is defined as a percentage T of the greatest importance score in the lexicon according to the following equation:

$$C = T \max_{t,y} I_{t,y} \tag{4}$$

We experimented with various values of T and in our evaluation we present detailed diagrams of how T affects the overall size of the model, and the fluctuation of the accuracy of our algorithm against the adversary classification approaches.

IV. Experiments

Now let us we demonstrate the effectiveness of the proposed method in terms of both classification accuracy and trained model size.

All the experiments were performed on a workstation equipped with a single Intel CoreI7 7700@3.6GHz CPU (single-core mode) and 32GB of RAM, running Ubuntu Linux 16.04 LTS.

The dataset we used was acquired by deploying a crawler on a popular product comparison platform, shopmania.com. This service enlists tens of millions of products organized in a three-level hierarchy comprised of 230 categories. The two higher levels of this hierarchy include 39 categories, whereas the third lower level accommodates the rest 191 leaf categories. Each product is categorized into this tree structure by being mapped to only one leaf category. Some of these 191 leaf categories contain millions of products. However, shopmania.com allows only the first 10,000 products to be retrieved per category. Under this restriction, our crawler managed to collect 313,706 products.

In the following analysis of the experimental results, we will refer to the proposed algorithm by using the abbreviation *SPC (Supervised Products Classifier)*. The accuracy of SPC is compared against two well-established supervised classification algorithms: Logistic Regression (LogReg), and Random Forests (RanFor).

A. Classification Accuracy

In this subsection we examine the effectiveness of our approach in terms of products classification accuracy. Figure 1 illustrates the performance of SPC and its two adversary algorithms for the four following feature spaces:

- $N = 1$: this feature space includes only the words of the product titles, that is, no n-grams are constructed (top-left diagram).
- $N = 2$: we take into consideration both the words and the bigrams extracted from product titles (top-right diagram).
- $N = 3$: the feature space consists of all the unigrams, bigrams and trigrams (bottom-left diagram).
- $N = 4$: the feature space consists of all the $1, 2, 3, 4$-grams (bottom-right diagram).

For each feature space, we measure the classification accuracy of all methods for various pruning levels. In particular, we modify the value of the parameter T of equation 4 in the range [0.0,1.0] by increasing its value by 0.1 each time. For instance, the pair $N = 2, T = 0.5$ is translated into a feature space which includes the words and the bigrams of the product titles whose importance score is greater than half of the highest computed importance score. Consequently, for the requirements of this experiment we constructed 44 different datasets, that is, 11 datasets for each value of N.

The performance evaluation was conduced by using 60% of the records in the dataset for training, whereas the rest 40% are left for testing. To ensure the correctness of our experiments and to present unbiased results, all the adversary approaches were given exactly the same set of features with the same importance scores as weights.

For the first feature space ($N = 1$, top-left diagram of Figure 1) and with all words present ($T = 0$), our method achieved a remarkable product classification accuracy of 88%. This accuracy is essentially left intact even if we remove all the words whose importance score is lower than the 20% of the maximum. At $T = 0.3$ and $T = 0.4$ the accuracy is slightly decreased at 86.1% and 84.8% respectively. After these values, the performance degrades quickly. SPC outperformed both the opponent classification methods in all cases. Regarding Logistic Regression, the accuracy started from roughly 82% (at $T = 0$) and it decreased at similar rates as SPC. On the other

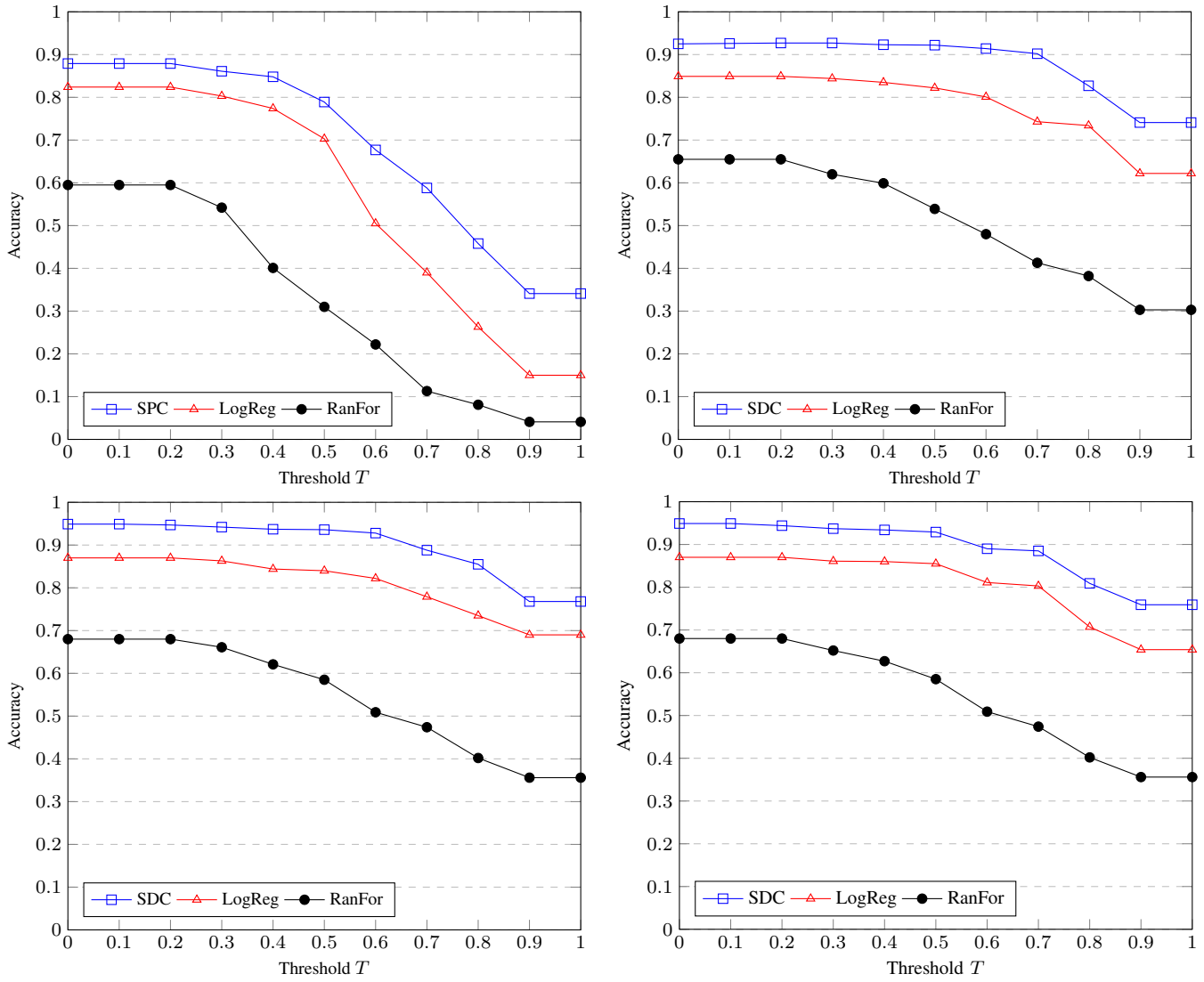

Fig. 1. Comparison of the classification performance of SPC against Logistic Regression (LogReg) and Random Forests (RanFor). Accuracy fluctuation with respect to the value of the cut-off threshold T for: i) $N = 1$ (top left), ii) $N = 2$ (top right), iii) $N = 3$ (bottom left), and iv) $N = 4$ (bottom right).

hand, the performance of Random Forests was significantly lower; for $T = 0$ it was measured at 59.5% and it dropped in higher rates compared to the other two algorithms.

Similar notifications can be made for the rest of the examined feature spaces. In the second feature space ($N = 2$, top-right diagram of Figure 1), the accuracy of SPC increases to 92.5% and begins to decrease for values of $T > 0.7$. On the other hand, the accuracy achieved by Logistic Regression and Random Forests for all words and bigrams ($T = 0$) was 85% and 65.5%, respectively. The diagram shows that the performance of Logistic Regression degrades at a rate which is similar to the one of SPC, whereas for Random Forests, the degradation occurs at a remarkably higher rate.

Our algorithm achieved its highest performance for $N = 3$ and $T \in [0.0, 0.2]$; its accuracy in these cases reached 95.1%. Similarly to the previous case, the accuracy falls below 90% for a threshold value of $T > 0.7$, whereas for $T = 0.6$ it is measured at 93%, only 2.1% lower than the maximum. As we will see shortly in the following subsection, the model size for $T = 0.6$ is smaller by 47% in terms of number of tokens in the lexicon, and 54% in terms of stored RDV entries. In other words, *by pruning nearly half of the model data, we lose only about 2% of its effectiveness*. Regarding the two other methods, the accuracy for Logistic Regression in this experiment was measured at 87%, significantly lower than that of our SPC. The performance of Random Forests was even worse, at 68%.

Notice that the same values of maximum accuracy was also measured in the largest feature space ($N = 4$), for all the three examined methods. This leads to the conclusion that there is no practical need to compute tokens which are longer than 3 words. In this case, the decrease in the performance is slightly faster for SPC, since the accuracy falls below 90% for $T > 0.6$. The performance degradation pattern is similar for both

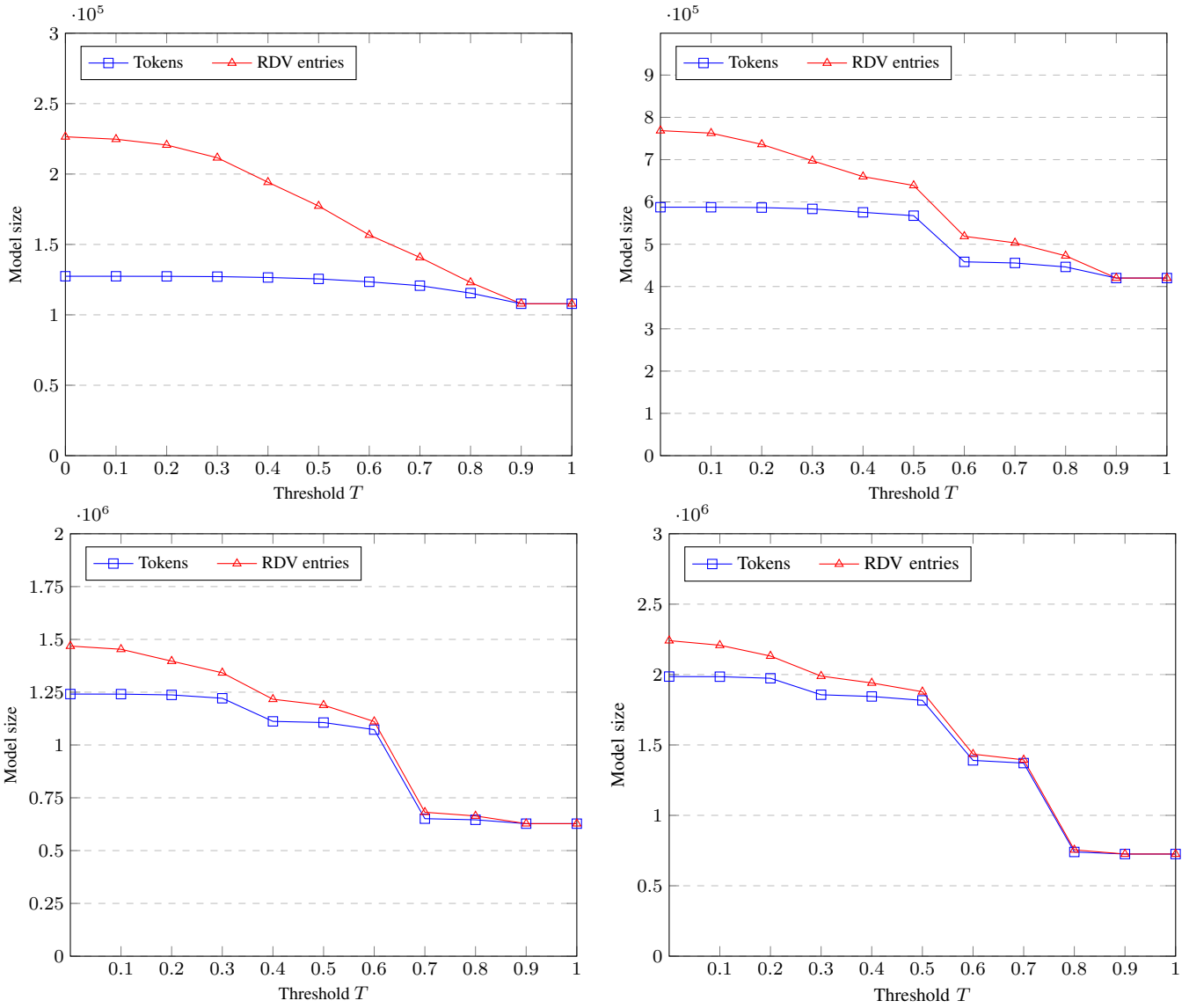

Fig. 2. Model size (overall number of tokens and RDV entries) reduction with respect to the value of the cut-off threshold T for: i) $N = 1$ (top left), ii) $N = 2$ (top right), iii) $N = 3$ (bottom left), and iv) $N = 4$ (bottom right).

Logistic Regression and Random Forests.

B. Model Size

Here we study the size of the model generated by SPC and we evaluate the effectiveness of our dimensionality reduction method.

In Table IV-B we report the increase in the size of our model with respect to the value of n. In case $n = 1$ (i.e. only single words are included and no n-grams are constructed), the model consists of 127,407 words and their RDVs include 226,450 relationships with various categories. This means that each word is correlated on average with roughly 1.8 categories. In the next row we show the model size when only the bigrams are inserted into the lexicon. There are about 460 thousands of such bigrams in the product titles of our dataset. Since their RDVs store approximately 542 thousands items, each bigram is associated on average with only 1.17 categories. On a similar spirit, the trigrams and the 4-grams are correlated on average with 1.07 and 1.04 categories. These numbers verify one of our basic assumptions which states that the longer a token is, the less ambiguity it carries. Consequently, higher importance scores should be assigned to longer tokens.

In the sequel, we examine how our proposed dimensionality reduction strategy affects the size of the model. The setup of this experiment is the same as in the evaluation of the classification performance; that is, we have the aforementioned four feature spaces and we modify the value of the cut-off threshold T in the range [0.0,1.0]. The results are depicted in Figure 2 and illustrate the reduction in the number of tokens and the number of RDV entries stored in the lexicon, as T increases. Notice that for $T = 0$, the model sizes are identical to the ones that we report in Table IV-B.

Here we are primarily interested in two model sizes: i) the

TABLE I
MODEL SIZES FOR VARIOUS VALUES OF n.

n	Tokens	RDV entries
1 (words)	127,407	226,450
2 (bigrams)	460,144	542,180
3 (trigrams)	653,525	699,639
4 (4-grams)	744,699	772,578
$1, 2$	587,551	768,630
$1, 2, 3$	1,241,076	1,468,269
$1, 2, 3, 4$	1,985,775	2,240,847

initial, and ii) the size of the model at the point where accuracy starts to decrease significantly. For this reason, it is required that we combine the data from both Figures 1 and 2.

For $N = 1$, the top-left diagram of Figure 1 shows that at $T = 0.4$ the accuracy starts to decrease at high rates. At this point, Figure 2 tells us that the model includes 126,528 tokens and 194,184 RDV entries. Compared to the initial model size (127,407 tokens and 226,450 RDV entries), we have a decrease of less than 1% in the number of tokens and about 15% in the number of RDV entries. This means that for $N = 1$, our method prunes only a few words which have long RDVs (i.e. they are correlated with a large number of categories, consequently, they are ambiguous). Although the reduction in the model size is not very large, the situation improves significantly for the other larger feature spaces (which also yield higher accuracy values).

Indeed, in the feature space which includes both words and bigrams ($N = 2$), the accuracy starts to decrease at $T = 0.7$ (top-right diagram of Figure 1). At that point, the model contains 455,635 tokens and 503,292 RDV entries. Compared to the initial 587,551 tokens and 768,630 RDV entries, the reduction we achieve is approximately 23% in the number of tokens and 35% in the RDV entries.

Finally, for $N = 3$ and $N = 4$, the reduction in the model sizes is even greater: the number of tokens is decreased by 47% and 63%, respectively, whereas the number of RDV entries is reduced by 54% and 66%, respectively.

V. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a new supervised learning method for automatic product categorization. The problem is particularly important for all e-commerce platforms as a robust solution can lead to numerous novel applications such as similar product suggestions and personalized recommendations.

Our proposed model is based on the extraction of various types of n-grams which, in combination with the single words and other types of tokens, improves the effectiveness of the categorization process. According to the proposed method, all tokens are inserted into a lexicon and they are assigned importance scores based on various parameters which reflect their ambiguity. The ambiguity is a notion which represents how significant a token is when it is used to determine the class of an unlabeled product.

The existence of multiple types of tokens may enlarge the model significantly and, therefore, affect its scalability in a negative manner. For this reason, we also proposed a dimensionality reduction method based on the importance scores of the tokens. According to this method, the non important tokens are removed from the lexicon of the model, since they are considered of low informational value.

The evaluation of our method demonstrated its effectiveness, since for a feature space which includes the words, the bigrams and the trigrams of the product titles, the classification accuracy exceeds 95%. Furthermore, we experimentally proved that in this case, the pruning of nearly half of the model data decreases accuracy by an insignificant margin of only 2%.

Finally, we intend to further study and develop this model with the aim of testing it in real-world large-scale and high-dimensional data from actual product catalogs. A detailed investigation of the efficiency of the model is also within our intentions.

REFERENCES

[1] J. Chen and D. Warren, "Cost-sensitive Learning for Large-scale Hierarchical Classification," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, 2013, pp. 1351–1360.

[2] Z. Kozareva, "Everyone Likes Shopping! Multi-class Product Categorization for E-commerce," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015, pp. 1329–1333.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[4] V. Gupta, H. Karnick, A. Bansal, and P. Jhala, "Product Classification in E-commerce using Distributional Semantics," *arXiv preprint arXiv:1606.06083*, 2016.

[5] D. Shen, J. D. Ruvini, M. Somaiya, and N. Sundaresan, "Item Categorization in the E-commerce Domain," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2011, pp. 1921–1924.

[6] D. Shen, J.-D. Ruvini, and B. Sarwar, "Large-scale Item Categorization for E-commerce," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, 2012, pp. 595–604.

[7] L. Akritidis and P. Bozanis, "Effective Unsupervised Matching of Product Titles with k-Combinations and Permutations," in *Proceedings of the 14th IEEE International Conference on Innovations in Intelligent Systems and Applications*, 2018.

[8] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[9] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, vol. 333, 2015, pp. 2267–2273.

[10] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1746–1751.

[11] P. Wang, J. Xu, B. Xu, C. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic Clustering and Convolutional Neural Network for Short Text Categorization," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 2, 2015, pp. 352–357.

[12] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu, "Deep Classification in Large-scale Text Hierarchies," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2008, pp. 619–626.

[13] J.-W. Ha, H. Pyo, and J. Kim, "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*, 2016, pp. 107–115.

[14] C. Chavaltada, K. Pasupa, and D. R. Hardoon, "A Comparative Study of Machine Learning Techniques for Automatic Product Categorisation," in *International Symposium on Neural Networks*, 2017, pp. 10–17.