Lightweight Methods for Large-Scale Product Categorization

Eli Cortez, Mauro Rojas Herrera, Altigran S. da Silva, and Edleno S. de Moura
Department of Computer Science, Federal University of Amazonas, Manaus, AM, Brazil,
E-mail: {eccv,mrh,alti,edleno}@dcc.ufam.edu.br

Marden Neubert
Uol Inc., São Paulo, SP, Brazil, E-mail: marden@uolinc.com

In this article, we present a study about classification methods for large-scale categorization of product offers on e-shopping web sites. We present a study about the performance of previously proposed approaches and deployed a probabilistic approach to model the classification problem. We also studied an alternative way of modeling information about the description of product offers and investigated the usage of price and store of product offers as features adopted in the classification process. Our experiments used two collections of over a million product offers previously categorized by human editors and taxonomies of hundreds of categories from a real e-shopping web site. In these experiments, our method achieved an improvement of up to 9% in the quality of the categorization in comparison with the best baseline we have found.

Introduction

Online shopping sites offer a large diversity of products in hundreds of different categories. Examples of e-shopping sites are: Yahoo! Shopping, Google Products, MSN Shopping, Shopping.com. Such web sites provide several services to their users such as product recommendation, price comparison, and store location. A crucial task to be performed by e-shopping sites to support these services is *product categorization*, the task of categorizing incoming product offers from online stores according to a pre-defined product taxonomy.

In this context, classification systems play a central role in e-shopping systems, being used in the tasks of searching and comparing offered products (Baron, Shaw, & Bailey, 2000; Leukel, Schmitz, & Dorloff, 2002). The result of product classification systems can be seen as a catalog, which is a structure used to store various product offers in a defined taxonomy (Baron et al., 2000).

Several current e-commerce web sites, such as Amazon,[1] eBay,[2] and Shopping UOL,[3] organize their products in catalogs. Such structures support a series of other applications in the e-commerce environment, for instance product search and personalized recommendation (Kwon, Kim, Kim, & Kwak, 2008; Pu, Chen, & Kumar, 2008). Searching for products is often the first stage in a consumer's purchasing process. Thus, the usage of product catalogs to help search systems to find the desired product may improve the user's experience. Well-organized catalogs also facilitate the building of personalized recommendation systems (Cho & Kim, 2004).

The problem addressed here can be better defined by considering how an e-shopping service works. First, features related to each product offer, such as *product description*, *product price*, and *name of the store*, are obtained from online stores. This first step can be performed through a transfer of data from the store to the e-shopping service or by a crawling and wrapping process to collect and extract the required features from the web sites of online stores (Wong, Lam, & Wong, 2008). After acquiring the data, each product offer must then be categorized, that is, associated with a category from the product taxonomy adopted on the e-shopping service.

Owing to the crucial role played by product categorization in their businesses, e-shopping companies usually deploy a team of specialized editors, who examine each incoming offer to categorize them. This approach has two main shortcomings: scalability and consistency.

With respect to scalability, as the product offers are automatically obtained (e.g., by crawlers) from several distinct sources (e.g., online stores), the volume of offers on larger

Received February 8, 2011; revised May 12, 2011; accepted May 13, 2011

© 2011 ASIS&T • Published online 21 June 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/asi.21586

[1]http://www.amazon.com
[2]http://www.ebay.com
[3]http://shoppinguol.com

e-shopping web sites is unbearable for editors to deal with, and categorization can be a bottleneck in the e-shopping businesses. For instance, reports in the literature estimate that manually categorizing a single item takes 5–10λmin (Abels, Hahn, & Oldenburg, 2006; Grabowski, Lossack, & Weißkopf, 2002) and may cost about $0.25 USD (Wolin, 2002).

With respect to consistency, as distinct editors are deployed, subjectivity over ambiguous offers may lead to inconsistent categorization. The high volume of offers being processed can also have an impact on the quality of the manual classification, since there is usually a large number of offers to be processed every day. Such difficulties may lead human editors to produce classification errors which are not desired, but can be accepted in the e-shopping services.

This scenario opens an opportunity for deploying automatic classifiers to help editors with their task (Abels et al., 2006; Wolin, 2002). This, however, represents a challenge by itself for many reasons. First, there is a large number of categories and products to take into account. This restricts the use of the classical machine learning-based classifiers, such as Support Vector Machines (SVM), which are generally avoided when dealing with more than 20 classes (Crammer & Singer, 2002a; Sulzmann, Fürnkranz, & Hüllermeier, 2007). Second, the idiosyncratic way by which products are categorized by human editors makes it hard to model their behavior in automatic classification methods.

In this article, we present a lightweight method for the large-scale categorization of product offers on e-shopping web sites, named here as *TopCat*. Similar to the previous methods in the literature (Wolin, 2002), our method works by looking for the best match between an unseen incoming product offer, represented by its attributes or features, and the categories in a given taxonomy, represented by the attributes/features of the product offers known to belong to those categories. We study the impact of three distinct features in the classification task and study how to combine these features using a probabilistic model. As demonstrated in experiments, these efforts allowed us to obtain more precise categorization results when compared with the previous work.

We tested our method over two datasets of approximately 1.5 million product offers previously categorized by human editors on a taxonomy of hundreds of categories from Shopping UOL. As a baseline, we used AutoCat (Wolin, 2002), *MNB* (Pavlov, Balasubramanyan, Dom, Kapur, & Parikh, 2004), *kNN* (Chakrabarti, 2003), and *CFC* (Guan, Zhou, & Guo, 2009); other categorization methods.

The remainder of this article is organized as follows. The following section overviews the related work. Challenges on product offer classification through the analysis of two real datasets are described in Datasets and Challenges. The details of the techniques we developed for categorizing products are described in TopCat and the subsequent section reports and analyzes the experimental results obtained. The penultimate section discusses the performance issues. Finally, final remarks, conclusions, and suggestions are provided for future work.

Related Work

Almost all classification systems currently used in e-shopping web sites take advantage of the pre-existing product catalogs, since, each catalog is composed of a set of keywords, descriptors, and attributes for each class. Well-known product classification catalogs are: UNSPSC, ETIM,[4] and eCl@ss.[5] A comparison between these three catalogs in order to verify the ability to better represent product attributes can be found in Beneventano, Guerra, Magnani, and Vincini (2004). While the previous study can be useful as a guideline for the study presented here, we deal with a different scenario. In our case, each product offer is collected from web pages found on the online store web sites. Once each product offer is extracted from the web site, we classify them into an e-shopping taxonomy. In the scenario considered in Beneventano et al. (2004), the e-shopping web site and the stores establish protocols to exchange product data (Quantz & Wichmann, 2003), this allows the classification system to adopt a pre-defined catalog and then obtain more complex and well-structured data.

A study about the impact of standardization of products in e-commerce was presented in Abels and Hahn (2005). This work investigates the existing approaches to the automatic classification of products and how to deal with the fact of having more than one taxonomy to categorize an incoming product.

Among the several distinct categorization methods found in the literature, at least three alternatives can be considered as the classical methods: SVM (Joachims, Nedellec, & Rouveirol, 1998), *kNN*, and the *Naive Bayes*. Our classification scenario, with hundreds of categories and thousands of product offers, is not suitable for applying the SVM classifier (Joachims et al., 1998), state-of-the-art in categorization tasks, since it suffers from performance problems in scenarios with a large number of categories and training instances (Liu et al., 2005). The other two classical classifiers may be good options to the scenario addressed here and are included in our experiments. A more complete review about categorization methods can be found in Sebastiani (2006), Chakrabarti (2003), and Rish (2001). Below we describe the baselines considered in our experiments, which were composed not only of classical classifiers, but also of specific solutions found in the literature to classify product offers in e-shopping.

A scenario similar to ours was considered in Wolin (2002), where the author proposed a product classification method named as AutoCat. AutoCat uses a variation of the vector space model (Salton & McGill, 1983) that allows typical product attributes to be represented in the model. Example attributes are short and long descriptions, manufacturers' name, and category label. In the AutoCat method, the training phase is accomplished by previously categorized data in which the system takes advantage of different features to represent each product offer. These features are combined using a weight feature optimization method in order to generate

[4]http://www.etim.de

[5]http://www.eclass-online.com

a ranking where the category at the top of the ranking is assigned to a given product offer. As a best result, AutoCat obtained 79.5% of accuracy when using all features studied by the authors. One of the features considered by the authors was the price of the product, which they concluded was not useful to improve the accuracy of their method. In our experiments, we also adopted the price and observed that although it does not improve the *micro-F*1, which is similar to the accuracy, it is useful to improve the quality of results when evaluating them by the metric *macro-F*1.

Pavlov et al. (2004) studied the application of Naive Bayes classifiers as an alternative for classifying product offers in e-shopping. The authors experimented with several heuristic feature transformations, such as IDF and normalization by the length of the documents, to improve the accuracy of Naive Bayes. They applied these transformations to solve two problems: the classification of products on Yahoo! Shopping and clustering the vectors of collocated terms in user queries to Yahoo! Search. We use here the Naive Bayes with the best transformation found by Pavlov et al., the length normalization, as one of the baselines in our experiments. In the classification experiments, with a real collection from Yahoo! Shopping, the Naive Bayes method with length normalization achieved a *micro-F*1 of 82.81% using title of offers and product description as features.

Guan et al. (2009) proposed, in the recent work, a fast Class-Feature-Centroid (CFC) classifier for multi-class, single-label text categorization. Their main motivation was to develop a fast and an effective classifier for web applications, such as e-shopping services. In CFC, a centroid is built from two important class distributions: inter-class term index and inner-class term index. CFC proposes a novel combination of these statistics and employs a denormalized cosine measure to calculate the similarity score between a text vector and a centroid. Experiments on the Reuters-21578 corpus and 20-newsgroup e-mail collection show that CFC consistently outperforms the state-of-the-art SVM classifiers on both *micro-F*1 and *macro-F*1 scores. We have included CFC in our experiments for comparison purposes.

Datasets and Challenges

For our experiments, we adopted two datasets obtained from *Shoppinguol.com*, a popular e-shopping web site. They represent two different snapshots of the database used in *Shoppinguol.com*. Although both datasets share the same general features, differences in the categories and products in each dataset make them suitable for verifying the generality of our study. From now on, we refer to these datasets as UOL8 and UOL9, since they were collected in years 2008 and 2009, respectively. We decided to perform experiments with these two collections because UOL performed a considerably large reorganization of its taxonomy in this period, thus providing two collections that are extracted from the same source, but have different taxonomies.

Each dataset has two components: a pre-defined taxonomy and a set of product offers, where each offer is assigned to a single category on the taxonomy. While UOL uses a hierarchy to present the product offers to users, the offers are classified only in the leafs of the hierarchy, thus resulting in a dataset with a flat taxonomy. Currently, the task of assigning a product offer to a category is performed by human editors at *Shoppinguol.com*, without any automatic method. As both UOL8 and UOL9 have several hundred thousand products, categorizing them required a huge human effort.

TABLE 1. A real product offer.

ID	Description	Price	Store	Category
001	MP3 Player Ipod 1λGB Apple	80.00	Store.com	MP3 Player

TABLE 2. Basic statistics about the datasets used in our study.

Dataset	# Categories	# Product offers
UOL8	783	1,161,186
UOL9	919	830,449

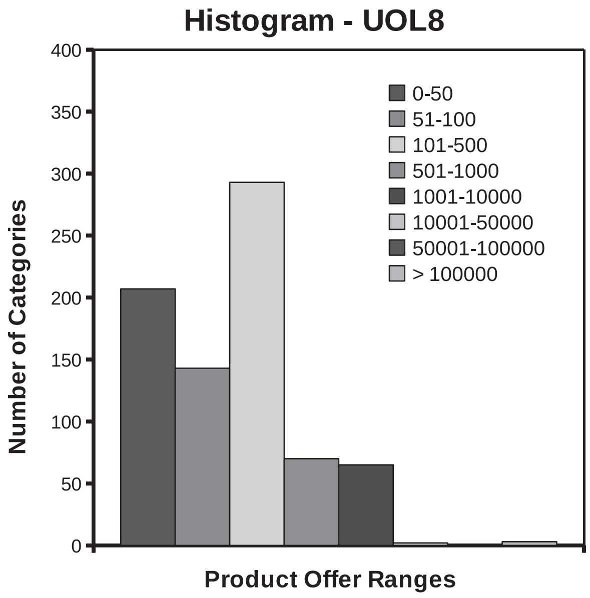
Table 1 presents a typical example of a product offer from the datasets. As can be seen, a product offer has a unique numeric *ID* along with three other fields indicated in Table 1: (1) A small product *Description*, which is composed of a set of words that describe some of the products characteristics. In most cases, the product manufacturer's name is also included in this field; (2) a product *Price*, and (3) the name of the *Store* that is selling this product. The assignment to a category is based on these fields. The offer presented in Table 1 would be assigned to the category *MP3 Players*.

Table 2 presents the features available in UOL8 and UOL9 datasets. To better illustrate the peculiarities of our datasets, Figures 1(a) and (b) present histograms in which categories forming UOL8 and UOL9, respectively, are grouped by ranges of the number of products assigned to them.

As shown in Figure 1(a), 350 (almost 50%) of the categories have no more than 100 product offers. On the other hand, there are three categories with more than 100,000 product offers in each. Indeed, these three categories alone make up more than half of the product offers. A closer look into UOL8 reveals that there are ambiguous cases in the taxonomy in which different categories contain similar product offers. In most cases, products in smaller categories should in fact be assigned to another larger category that is suitable for them. However, human editors decided to leave them as separate categories for subjective reasons that are hard to model. In other cases, new categories are created for specific products only to fulfill commercial requirements.

The taxonomy in dataset UOL9 presents a similar pattern in this regard, as Figure 1(b) shows. Notice that the number of categories containing no more than 100 product offers has increased from 350 in UOL8 to 483 in UOL9.

As it can be noticed from Table 2, from Figures 1(a) and (b), and from the comments above, providing solutions for the automatic categorization of product offers on e-shopping web sites is a challenge for many reasons. First, there is a



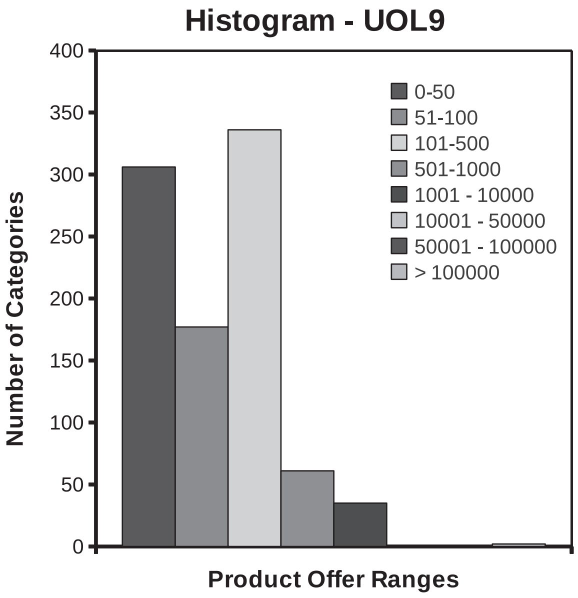


FIG. 1. Categories histogram from UOL8 (a) and UOL9 (b).

large number of categories and products to take into account. Thus, performance and scalability are important issues to be considered. Second, the idiosyncratic way by which products are categorized by human editors makes it hard to model their behavior in automatic classification methods.

These requirements prevent the use of the classical machine learning classifiers, such as SVM (Crammer & Singer, 2002b). Multi-class SVM problems are usually dealt with by using an ensemble of two-class SVMs and in the scenario considered here the number of two-class SVMs required would be prohibitive, since we have several hundreds of classes.

TopCat

TopCat models the problem of classifying product offers using a probabilistic approach. As discussed in the Datasets and Challenges section, each product offer *PO* is represented by three fields: (1) Product Description, (2) Price, and (3) Store. For each of these fields, we compute a probability and then combine these probabilities to estimate the probability of the *PO* belonging to each category. The product offer *PO* is then assigned to the category with the higher probability according to our model.

We model our classification problem through a belief network model similar to those proposed by Ribeiro-Neto and Muntz (1996) for ranking documents in search tasks. The belief network model uses Bayesian networks to provide a graphical formalism for representing the probability model developed. Figure 2 illustrates our network where product offers and categories are represented by their description, store, and product price.

In the network in Figure 2, each node C_j models a category, the node *PO* models a new product offer to be categorized, the nodes from K_1 to K_t model the keywords that appear in descriptions of all the offers found in the collection, the nodes $S_1 - S_q$ represent the stores and nodes $P_1 - P_l$ represent the possible prices that can be found in a new product offer, prices are modeled as discrete values and only prices in the range found in the training examples are modeled in our network. The vectors **k**, **s**, and **p** are used to refer to any of the possible states of the *root* nodes from K_1 to K_t, S_1 to S_q, and P_1 to P_l, respectively.

Each node from CD_1 to CD_N represents a category and is used to model the probability of a category, given the occurrence of a product offer description. Analogously, each node from CS_1 to CS_N models the probability of a category, given the occurrence of a store in a product offer. Each node from CP_1 to CP_N represents the probability of a category, given the occurrence of a price in a product offer. Information available in these nodes are then combined through an *or* operator to compute the final probability of each category, given a product offer, which is represented in nodes from C_1 to C_N. The category with the higher value is then associated with the given product offer.

A *binary* random variable is associated with the node *PO*, which is also denoted by *PO*. A *binary* random variable K_i is also associated with each keyword K_i. A *binary* random variable S_i is associated with each store value S_i and a *binary* random variable P_i is associated with each price. In this notation it should always be clear whether we are referring to the product offer, to the node in the network, or to its associated binary variable. For any variable X in the model, we say that it is 1, denoted by x, to indicate that node X is *on* and X is 0, denoted by $\overline{x}$, to indicate that X is *off*.

Following the network described in Figure 2, we take the information provided by the description field to compute the probability $P(cd_j|po)$, i.e., the probability that the variable CD_j is *on* given that the variable *PO* is *on*. According

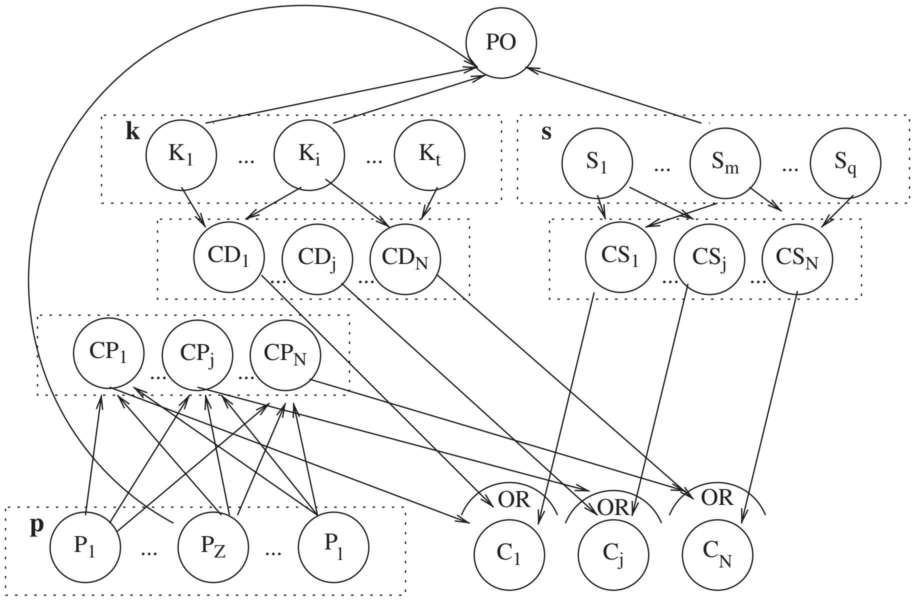


FIG. 2. Belief network for combining information related to a product offer PO that belongs to a store represented by node S_m, whose associated price is represented by node P_z, and whose description contains keywords represented by nodes K_1 and K_i.

to the model presented in Ribeiro-Neto and Muntz (1996), we can write

$$P(cd_j|po) = P(cd_j|\mathbf{k})P(po|\mathbf{k})P(\mathbf{k}) \tag{1}$$

where vector **k** represents a state where only nodes associated with the keywords that occur in the product offer PO are active. Further, we model the prior probabilities $P(\mathbf{k})$ and $P(po|\mathbf{k})$ setting their values to 1, and setting the probability of each other possible states of the set of nodes from K_1 to K_t as 0.

We now need to define how to compute the value of $P(cd_j|\mathbf{k})$, which should be proportional to the likelihood of each term of PO occurs in the category C_j represented by node CD_j. We then model such probability as

$$P(cd_j|\mathbf{k}) = \eta \sum\nolimits_{\kappa \in terms(\mathbf{k})} \frac{f(\kappa, C_J)}{\sum_{c' \in C} f(\kappa, c')} \tag{2}$$

where the function *terms*(**k**) returns the set of terms related to the active nodes in **k**, i.e., the terms that occur in the description of product offer associated with node PO. The value η is a normalizing constant (Pearl, 1988), set to $1/|PO|$, where $|PO|$ is the number of words in the product offer associated with node PO. C_j is the category represented by node CD_j. The set C is the set of all categories that is composed of the taxonomy and function $f(\kappa, c)$ gives the frequency of a term κ on the examples of product offer descriptions that belong to a category c in the training dataset.

Following similar ideas, we also compute the value of $P(cs_j|po)$ as

$$P(cs_j|po) = P(cs_j|\mathbf{s})P(po|\mathbf{s})P(\mathbf{s}) \tag{3}$$

where **s** is a state where only the node related to the store of the product offer PO is active. We also model the values of $P(po|\mathbf{s})$ and $P(\mathbf{s})$ as 1. And compute $P(cs_j|s)$ as

$$P(cs_j|\mathbf{s}) = \frac{fs\ (store(\mathbf{s}), C_j)}{\sum_{c' \in C} fs(store(\mathbf{s}), c')} \tag{4}$$

where the function *store*(**s**) returns the set store related to the active node in **s**, i.e., the store of the product offer PO. C_j is the category represented by node CS_j. The set C is the set of all categories that is composed of the taxonomy and *fs*(*store*(**s**), *c*) is a function that gives the frequency of a store *store*(**s**) on the examples of product offers that belong to the given category c in the training dataset.

Finally, we also compute the value of $P(cp_j|po)$ as

$$P(cp_j|po) = P(cp_j|\mathbf{p})\ P(po|\mathbf{p})\ P(\mathbf{p}) \tag{5}$$

where **p** is a state where the only node active is the one associated with the price of the product offer represented by node PO. First, we also define the prior probabilities $P(\mathbf{p})$ and $P(po|\mathbf{p})$ as 1.

Given a price p of a product offer PO and a category C_J represented by node CP_j, we apply a simple yet an effective approach proposed in Agrawal, Chaudhuri, Das, and Gionis (2003) to estimate the probability $P(cp_j|\mathbf{p})$ based on prices of known products in the category C_j. Assuming that the price values in each product offer follow a Gaussian distribution, $P(cp_j|\mathbf{p})$ is estimated as the mean value of a probability density function. We normalize this function by the maximum probability density, which is reached when a given value is equal to the mean. Further, we add a restriction to make $P(cp_j|\mathbf{p})$ be zero for categories that do not share description keywords with the product offer. This restriction could be easily added to the model presented in

Figure 2, but it would make the network less didactic and would require further space for explaining it. We define the $P(cp_j|\mathbf{p})$ as

$$P(cp_j|\mathbf{p}) = \begin{cases} e^{-\frac{price(\mathbf{p})-\mu(C_j)}{2\sigma(C_j)^2}} & \text{if } P(cd_j|po) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $price(\mathbf{p})$ is a function that returns the price value associated with the only active node of vector $\mathbf{p}$; $\sigma(C_j)$ and $\mu(C_j)$ are standard deviation and mean, respectively, of price values from known products in the category C_j represented by node CP_j. The intuition behind $P(cp_j|\mathbf{p})$ is that the closest the price of the new product offer is from the mean price value of a category, the higher the $P(cp_j|\mathbf{p})$ is.

The final conditional probability $P(c_j|\mathbf{po})$ can now be computed using a disjunctive operator *or*. This is accomplished as follows:

$$\begin{aligned} P(c_j|\mathbf{po}) = 1 - (1 - P(cd_j|\mathbf{po})) \\ \times (1 - P(cp_j|\mathbf{po})) \\ \times (1 - P(cs_j|\mathbf{po})) \end{aligned} \quad (7)$$

The computation of the probability $P(c_j|\mathbf{po})$ is then used in TopCat to associate a class with each new product offer to be classified. Given a new product offer represented by a node PO, we assign it to the class C_j with higher value of $P(c_j|po)$.

Using Segments of Description

We performed several experiments using our classification model. After studying the results obtained in the initial model, we realized that the position of occurrence of terms in the description may play an important role in the classification process. For instance, it is common to have the product identified in the initial words of a product offer description. As another example, several CD offers contain the word CD at the end of the offer description. Based on this observation, we tested a model where we break the description into segments of text and then use each segment as an isolated feature for classification.

In our experiments, we always broke the description field into three segments, with the first two containing a maximum of α words, and the last segment containing the remaining words of the description. We have varied the value of α in our experiments and shown how to find the best α for new collections.

For the sake of space, we did not draw our belief network containing the segment information. This network would contain new vectors of nodes for each of the three segment features and the final *or* operation would include the probabilities that come from each segment: $p(cseg1_j|po)$, $p(cseg2_j|po)$, and $p(cseg3_j|po)$. The remaining nodes of the network, including the nodes representing the complete description as a feature would not change. It is important to notice that the segments probabilities are computed analogously as $P(cd_j|po)$ (Eq. 1). Eq. (7) is now substituted by the more general equation below:

$$\begin{aligned} P(c_j|\mathbf{po}) = 1 - (1 - P(cd_j|\mathbf{po})) \times (1 - P(cp_j|\mathbf{po})) \\ \times (1 - P(cs_j|\mathbf{po})) \times (1 - P(cseg1_j|\mathbf{po})) \\ \times (1 - P(cseg2_j|\mathbf{po})) \times (1 - P(cseg3_j|\mathbf{po})) \end{aligned} \quad (8)$$

Experiments

We carried out experiments using two distinct datasets from a real e-shopping web site: *UOL8* and *UOL9*. First, we evaluate the impact of varying the segment sizes that are used in *TopCat*. Second, we present a study of the impact of each feature that is used by our proposed method. After that we performed error analyzes with humans in order to manually verify cases where TopCat did not provide a product offer the same category that was assigned by humans in our dataset. Finally, we compare *TopCat* with four distinct baselines. These experiments are intended to show how each classification method behaves in a real scenario.

In all experiments we report in this section we used a five-fold cross-validation method (Mitchell, 1997). Thus, each value presented here represents the average of the values obtained in each of the five folds. Average results obtained in the five folds are presented and the variation between the results of each fold is smaller than 0.001 for all values presented in the experiments. Thus, we only show average results in all cases. Further, it is important to mention that all the differences between methods presented in the article are statistically significant.

Baselines

In our comparative experiment we have used four distinct classification methods as baselines. We present the best parameters setting in all reported results for each baseline.

One of the baselines is *AutoCat*, proposed in Wolin (2002). This method proposes the usage of different features to represent each product. Each feature returns a similarity score. According to our datasets, we were able to use four of those features. They are: Category Label, Category Label Phrases, Short Description, and Short Description Phrases. To combine each feature score and generate the final similarity, we have performed the feature weight optimization method that is proposed in Wolin (2002). To provide a fair comparison between *AutoCat* and *TopCat*, we performed experiments with *AutoCat* using the same sources of information adopted in *TopCat*: the description, the store, and the price. However, as it was reported previously by the authors, the results when using price and store are slightly worse. Thus, we decided to report only the *AutoCat* results obtained with the four features mentioned above, which is also the best combination we found for it.

The second baseline is the well-known *kNN* classifier (Chakrabarti, 2003). In kNN, each product offer is represented in a vector space and the similarity between an input offer and the previously known offers is computed using cosine measure (Chakrabarti, 2003). The assignment decision is made by considering the category to which belong the clear majority of the k known instance with the highest similarity with the input offer. In all *kNN* results reported here, we have used $k = 4$, which was the best result we obtained in the experiments. Thus, the results presented by *kNN* can be considered as an upper bound.

The third baseline is the method *CFC* (Guan et al., 2009), which uses a centroid-based approach and finally we use the modified Naive Bayes proposed for classifying e-shopping products by Pavlov et al. (2004), which will be referred to as MNB (modified Naive Bayes). These last three baselines use only textual information and thus we converted the price and store information into tokens that were added to the description, using the resulting text to represent each product offer. As we show in the experiments, the results obtained by these three baselines are worse than those presented by *NoStoreNoPrice* combination of features when using *TopCat*. The *NoStoreNoPrice* combination represents *TopCat* using only the textual information available in the description of product offers, thus using the same information available for these three baselines.

Metrics for the experimental evaluation. To evaluate each classification approach, we used the traditional classification metrics of *micro*-averaged $F1$ (*micro-F*1) and *macro*-averaged $F1$ (*macro-F*1) (Witten & Frank, 2005). Let P be the set of product offers used for testing. Let P' be the set of product offers correctly classified by the classifier.

The metric *micro-F*1 is defined as the percentage of correctly classified candidates within the set of all classified candidates:

$$micro\text{-}F1 = \frac{|P \cap P'|}{|P|} \tag{9}$$

To obtain the *macro-F*1, we need to measure the precision and recall for each category in the taxonomy. The precision Pr_c of a given category c is the percentage of product offers correctly classified as being of category c, over all product offers classified as being of category.

The recall R_c of each category c is defined as the percentage of product offers correctly classified as begin of category c, over all product offers that actually belong to category c.

Thus, let T be the set of all categories in the taxonomy. The *macro*-averaged precision is defined as the average precision for all categories, $Pr = \sum_{cinT} Pr_c \big/ |T|$ and the *macro*-averaged recall is also an average of all recall values for each category, $Rr = \sum_{cinT} R_c \big/ |T|$.

Hence, *macro-F*1 is the harmonic mean between *macro*-averaged recall and *macro*-averaged precision:

$$macro\text{-}F1 = \frac{2\ Pr\ R_c}{Pr + R_c} \tag{10}$$

Varying the Segment Sizes

One important aim in our method is to determine the maximum segment size (α) that provides the best classification

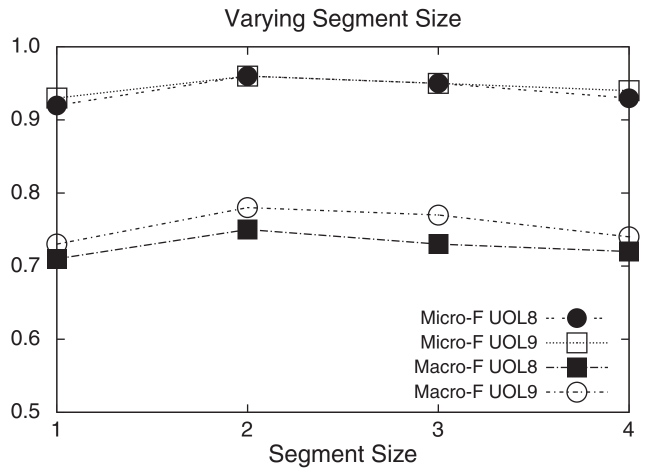


FIG. 3. Results obtained when varying the segment size in our model.

results. In order to study these parameters, we have run a five-fold cross-validation of TopCat varying the parameter α from 1 to 4 in both collections. The results are depicted in Figure 3, where we can see that the best values are achieved when using size 2 in both collections. When looking at the results, we observed that it is common to have a match between category names and the first terms found in product offers. This match can partially explain the results presented in Figure 3, since they are more frequent in the first three positions of the offer descriptions and become less frequent after the third position. For the remaining experiments, we set the value 2 for α.

Studying the Impact of Features

Another aspect of our experiments was to assert the impact of each feature in the classification results. This information is particularly important for features like store and segments of descriptions, which are proposed by us, and for the feature price, which was proposed by Wolin (2002), but is revisited by us here. To assert the individual impact of each feature in our experiments, we performed experiments with *TopCat* with all features and compared it with its performance when removing individual features. We also included a version removing all the segment information to assert the impact of the segmentation strategy in the method. Thus, we have the following versions of *TopCat* in these experiments: with all features, referred to as *TopCat*; without price, referred to as *NoPrice*; without store, referred to as *NoStore*; without description, referred to as *NoDescription*; without first segment of description, referred to as *No1stSeg*; without second segment of description, referred to as *No2ndSeg*; without third segment of description, referred to as *No3rdSeg*; without any segment of description, referred to as *NoSeg*.

We also add a combination using only information from the description in our model, using the complete description and the segments of the description as features and removing price and store information, referred to as *NoStoreNoPrice*. This latter combination is useful to verify

TABLE 3. Results of TopCat using nine combinations of the features studied in our experiments.

	UOL8		UOL9	
Variation	*Micro-F*1	*Macro-F*1	*Micro-F*1	*Macro-F*1
TopCat	0.96	0.75	0.96	0.78
NoPrice	0.95	0.73	0.95	0.77
NoStore	0.94	0.72	0.94	0.74
NoDescription	0.95	0.75	0.94	0.76
No1stSeg	0.91	0.67	0.90	0.67
No2ndSeg	0.94	0.73	0.94	0.77
No3rdSeg	0.95	0.74	0.94	0.76
NoSeg	0.91	0.66	0.89	0.64
NoStoreNoPrice	0.93	0.73	0.94	0.77

TABLE 4. Number of agreements between users and our automatic classification (TopCat) and users and classifications assigned in the reference collection (Reference) when analyzing a sample of randomly selected offers where our automatic method diverges from the reference set in collection UOL9.

	User 1	User 2	User 3	User 4	User 5
TopCat	75	77	75	57	71
Reference	24	35	37	26	27

whether our model is still better than the baselines even when using no extra feature information, since *CFC* and *MNB* are textual classifiers.

Table 3 presents the results obtained when applying *TopCat* with all versions described above. The most important conclusion from this table is that the partition of product offer descriptions into segments has the higher impact in the classification results in both collections and both metrics. For instance, when removing all segment features, the results in UOL8 vary from 0.96 to 0.91 in *micro-F*1 and vary from 0.75 to 0.66 in terms of *macro-F*1 (see line *NoSeg* in the table). On the other hand, the usage of the whole description is still useful, although with a smaller impact, even when combined with its segments. For instance, in UOL9 TopCat would lose 0.02 points both in *macro-F*1 and *micro-F*1 if description was not used (see *NoDescription* in the table).

Another interesting conclusion we can take from Table 3 is that price information has a low positive impact in the classification results. Considering that any improvement in the quality of results is important in this classification task, price can still be considered as an alternative feature. Regarding the feature store, we can say that it had an important contribution in *TopCat* results, being able to improve both *micro-* and *macro-F*1 in both collections. The store feature was particularly useful in the metric *macro-F*1. *Macro-F*1 results would be 0.03 smaller in UOL8 and 0.04 smaller in UOL9 if we had not included this feature in TopCat (see line *NoStore* in the table).

Finally, another important conclusion is that the first segment of the description is by far the most important segment, which indicates that in a scenario where efficiency is essential for the system it may be combined to store and produce competitive results. We experimented with this combination and results were 0.94 in *micro-F*1 and 0.70 in *macro-F*1 for collection UOL9. Notice however that this option would cause significant loss in the *macro-F*1 results.

Error Analysis

In this section we investigate the cases where *TopCat* did not provide the same category assigned by humans, which we consider as errors in our experiments. It is important to stress that humans may assign wrong categories to some product offers and also, while in our dataset each offer is associated with a single category, some offers may fit in more than one category in the taxonomy. Thus, it is important to study the error cases in order to check whether they are really classification errors or not.

To perform this study, we randomly selected a set of 100 product offers where *TopCat* disagreed with the human classification provided in the dataset. For each of these offers, we presented the two alternatives to five users, asking them to mark which category is correct for the considered product offer. In cases where the user thinks that both alternatives are correct he can mark the two options and the user can also mark none if he thinks that neither of the options are suitable for the current product offer.

For the error analysis we performed experiments only with collection UOL9 due to the high cost of manual effort required in the experiments. The results are presented in Table 4. As it can be noticed, the results show that the five users agreed that the category that was provided by *TopCat* was in fact correct in more than 71%. Further, there is a surprising conclusion that in fact users have a low percentage of agreement with the reference set in these cases. These results indicate that TopCat was able to identify a better categorization in these cases.

Given the high error rate in the cases where humans do not agree with *TopCat*, we decided to also perform experiments to check whether a set of users would agree with classification in cases where both TopCat and the reference set have assigned the same category to product offers. For this experiment, we selected 100 product offers and asked users to say whether the assigned category is correct or not for each offer. Results are depicted in Table 5, from where we can see that, in this sample and according to these five users, the overall quality of the classification when TopCat agrees with the reference set was above 96%. Although this indicates that the reference set is not free of errors, it also shows that it can be used as a quite safe reference to assert the quality and compare automatic classification methods.

Comparison to Baselines

In this section, we report experiments performed comparing *TopCat*, *AutoCat*, *kNN*, *CFC* and *MNB* in the task

TABLE 5. Number of agreements between users and our automatic classification (TopCat) when analyzing a sample of randomly selected offers where our automatic method assigns the same category assigned in the reference set in collection UOL9.

	User 1	User 2	User 3	User 4	User 5
TopCat	96	96	97	98	99

TABLE 6. Table 6 Comparative results.

Metric	TopCat	AutoCat	MNB	kNN	CFC
UOL8					
*Micro-F*1	0.96	0.87	0.84	0.88	0.73
*Macro-F*1	0.75	0.56	0.62	0.72	0.38
UOL9					
*Micro-F*1	0.96	0.87	0.83	0.88	0.79
*Macro-F*1	0.78	0.56	0.63	0.75	0.39

of automatic categorized product offers in e-shopping. *AutoCat* results present the best combination of features achieved when using *AutoCat*, including description.

Table 6 presents the categorization results obtained by the experiment methods when applied to collections UOL8 and UOL9 using metrics *macro-F*1 and *micro-F*1. In spite of the large difference in the number of offers and categories found in the two collections, results and conclusions obtained are the same in the experiments with them.

TopCat was the best option in both collections and both metrics adopted. The best baseline was *kNN* in both collections. TopCat was 9.1% superior in *micro-F*1 and 2.7% in *macro-F*1 in experiments with UOL8 when compared with *kNN*. It is important to notice that, as discussed in the next section, the computational costs of *kNN* are higher than all the other methods. Gains over the other baselines, which have the same time complexity as *TopCat*, were all considerably high in both metrics and collections. For instance, when compared with *AutoCat*, the second best baseline, gain was 9.1% in *micro-F*1 and 34.0% in *macro-F*1 when performing experiments with UOL8.

A final interesting comment is that the results presented by the baselines are all inferior to the results presented by *TopCat* when using only attribute description, presented in Table 3 as *NoStoreNoPrice*. Further, *TopCat* is superior to the best *AutoCat* result even when using no segment information, *NoSeg* presented in Table 3, thus being superior when using exactly the same attributes available to *AutoCat* and exactly the same features.

Time Performance Issues

Besides the importance of obtaining high-quality results, product offer classification methods should be able to scale when the number of categories and product offers grow. For instance, SVM (Joachims et al., 1998), a state-of-the-art classification method is not viable in this problem. We tried to run SVM light (Joachims, 1999), which is an optimized implementation of SVM and we were not able to execute the program even when running with a 4-GB computer and using only 10% of the training samples. The time complexity costs of the experimented methods are the same for methods *TopCat*, *AutoCat*, *AutoCatSeg*, *MNB*, and *CFC*, which are $O(P \cdot C)$, where P is the number of product offers and C is the number of categories in the collection. If we consider that C is constant, the methods could be considered as linear. Each of these methods needs to compute a probability, or similarity value, between each product offer and a model that represents a category in the database. The final time to run all these methods depended on implementation details, but all of them run in a few minutes in UOL9.

The method *kNN* is $O(P^2)$, computes the similarity between each product offer and all product offers in the training set. This procedure is implemented by using the traditional implementation of vector space model with inverted indexes. To give an idea about the impact of this change in the costs, our implementation of *kNN* took about 20λh in UOL9.

Conclusions and Future Work

In this research work, we proposed a new product offer classification approach that combines a set of features to produce high-quality classification results in e-shopping systems. Regarding the study of features, we have concluded that information about the price and the store of the offer is useful for the classification process, but results in not much improvement when compared with the product offer description, with gains of 3.2% at UOL8 and 1% at UOL9 in *micro-F*1.

One of the main contributions of this study is to show that the segmentation of description of product offers, a feature that is commonly available on e-shopping systems, results in a significant improvement in the quality of classification results. For instance, segment information was able to improve the *macro-F*1 results by 13.6% at UOL8 and 20.3% at UOL9 when compared with the best result obtained by us without segment information. Further, gain was also high when using metric *micro-F*1.

The new way of modeling the problem and the study of features presented in this article may contribute to produce better practical results in e-shopping classification systems and provide an interesting insight for future work in this area. As future work, we are interested in investigating the usage of our classification approach as a tool for finding classification errors in e-shopping services where humans perform a manual classification. We considered this possibility after verifying that in our human classified dataset there is a higher concentration of classification errors for offers, where the results of TopCat diverge from the category assigned by humans.

As another future direction, we are also interested in investigating the performance of *TopCat* as a method to identify when a new product does not fit in any of the categories in the current taxonomy. Further, we are also interested in cases where the new product offer belongs to multiple categories, thus performing a multi-label classification.

Acknowledgments

This work is partially supported by projects INWeb (MCT/CNPq grant 57.3871/2008-6), MinGroup (CNPq grant 575553/2008-1) and SABACO (CNPq grant575541/2008-3); by individual CNPq fellowship grants to Edleno S. de Moura (307861/2010-4) and Altigran S. Silva (308380/2010-0) and by CAPES scholarships to Eli Cortez and Mauro Rojas Herrera.

References

Abels, S., & Hahn, A. (2005). Automatic classification and re-classification of product data in e-business. In Proceedings of the 2005 Symposium on Applications and the Internet Workshops (pp. 350–353). Washington, DC: IEEE Computer Society.

Abels, S., Hahn, A., & Oldenburg, G. (2006). Reclassification of electronic product catalogs: The "Apricot" approach and its evaluation results. Informing Science, 9, 31.

Agrawal, S., Chaudhuri, S., Das, G., & Gionis, A. (2003). Automated ranking of database query results. In Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR 2003) (pp. 888–899). Retrieved from http://www.cidrdb.org/

Baron, J., Shaw, M., & Bailey Jr., A. (2000). Web-based e-catalog systems in B2B procurement. Communications of the ACM, 43(5), 93–100.

Beneventano, D., Guerra, F., Magnani, S., & Vincini, M. (2004). A web service based framework for the semantic mapping amongst product classification schemas. Journal of Electronic Commerce Research, 5(2), 114–127.

Chakrabarti, S. (2003). Mining the web. Los Altos, CA: Morgan Kaufmann Publishers.

Cho, Y., & Kim, J. (2004). Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce. Expert Systems with Applications, 26(2), 233–246.

Crammer, K., & Singer, Y. (2002a). On the algorithmic implementation of multiclass kernel-based vector machines. The Journal of Machine Learning Research, 2, 265–292.

Crammer, K., & Singer, Y. (2002b). On the learnability and design of output codes for multiclass problems. Machine Learning, 47(2), 201–233.

Grabowski, H., Lossack, R., & Weißkopf, J. (2002). Datenmanagement, 2002: Datenmanagement in der Produktentwicklung [Data management: Data management in product development]. München und Wien.

Guan, H., Zhou, J., & Guo, M. (2009). A class-feature-centroid classifier for text categorization. In Proceedings of the World Wide Web Conference (pp. 201–210). New York: ACM Press.

Joachims, T. (1999). SVMLight: support vector machine. SVM-Light Support Vector Machine, University of Dortmund. Retrieved from http://svmlight. joachims.org/.

Joachims, T., Nedellec, C., & Rouveirol, C. (1998). Text categorization with support vector machines: Learning with many relevant. In Proceedings of the 10th European Conference on Machine Learning (ECML-98) (pp. 137–142), Chemnitz, Germany. Berlin: Springer.

Kwon, I., Kim, C., Kim, K., & Kwak, C. (2008). Recommendation of e-commerce sites by matching category-based buyer query and product e-catalogs. Computers in Industry, 59(4), 380–394.

Leukel, J., Schmitz, V., & Dorloff, F. (2002). Modeling and exchange of product classification systems using XML. In Proceedings of the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (pp. 242–244). Washington, DC: IEEE Press.

Liu, T., Yang, Y., Wan, H., Zeng, H., Chen, Z., & Ma, W. (2005). Support vector machines classification with a very large-scale taxonomy. ACM SIGKDD Explorations Newsletter, 7(1), 36–43.

Mitchell, T. (1997). Machine learning. New York: McGraw-Hill.

Pavlov, D., Balasubramanyan, R., Dom, B., Kapur, S., & Parikh, J. (2004). Document preprocessing for naive bayes classification and clustering with mixture of multinomials. In Proceedings of the ACM Special Interest Group on Knowledge Discovery and Data Mining Conference (SIGKDD) (pp. 829–834). New York: ACM Press.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: Networks of plausible inference, 2nd edition. Los Altos, CA: Morgan Kaufmann Publishers.

Pu, P., Chen, L., & Kumar, P. (2008). Evaluating product search and recommender systems for E-commerce environments. Electronic Commerce Research, 8(1), 1–27.

Quantz, J., & Wichmann, T. (2003). E-business-standards in Germany-Research project commissioned by the German Federal Ministry of Economics. Final report (Short version).

Ribeiro-Neto, B., & Muntz, R. (1996). A belief network model for IR. In Proceedings of ACM Special Interest Group on Information Retrieval (SIGIR) (pp. 253–260). New York: ACM Press.

Rish, I. (2001). An empirical study of the naive Bayes classifier. In Proceedings of the International Joint Conferences on Artificial Intelligence Workshop on Empirical Methods in Artificial Intelligence (IJCAI 2001) (pp. 41–46). San Francisco: Morgan Kaufmann.

Salton, G., & McGill, M.J. (1983). Introduction to modern information retrieval, 1st edition. New York: McGraw-Hill.

Sebastiani, F. (2006). Classification of text, automatic. The encyclopedia of language and linguistics (pp. 457–462), Vol. 14.

Sulzmann, J.-N., Fürnkranz, J., and Hüllermeier, E. (2007). On pairwise naive bayes classifiers. In J.N. Kok, J. Koronacki, R.L. de Mántaras, S. Matwin, D. Mladenic, & A. Skowron (Eds.), European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD). Lecture Notes in Computer Science, 4701, 371–381.

Witten, I., & Frank, E. (2005). Data mining: practical machine learning tools and techniques. Amsterdam: Elsevier.

Wolin, B. (2002). Automatic classification in product catalogs. In Proceedings of ACM Special Interest Group on Information Retrieval (SIGIR) (pp. 351–352). New York: ACM Press.

Wong, T., Lam, W., & Wong, T. (2008). An unsupervised framework for extracting and normalizing product attributes from multiple web sites. Proceedings of ACM Special Interest Group on Information Retrieval (SIGIR) (pp. 35–42). New York: ACM Press.