Identifying e-Commerce in Enterprises by means of Text Mining and Classification algorithms

Gianpiero Bianchi[1], Renato Bruni[2*], Francesco Scalfati[1]

1) Istat, Direzione centrale per la metodologia e disegno dei processi statistici (DCME), Via Depretis 77, Roma, 00184 Italy
E-mail: {gianbia,scalfati}@istat.it

2) Università di Roma "Sapienza", Dip. di Ing. Informatica, Automatica e Gestionale (DIAG), Via Ariosto 25, Roma, 00185 Italy
E-mail: bruni@dis.uniroma1.it

Abstract

Monitoring specific features of the enterprises, for example the adoption of e-commerce, is an important and basic task for several economic activities. This type of information is usually obtained by means of surveys, which are costly due to the amount of personnel involved in the task. An automatic detection of this information would allow consistent savings. This can actually be performed by relying on computer engineering, since in general this information is publicly available on-line through the corporate websites. This work describes how to convert the detection of e-commerce into a supervised classification problem, where each record is obtained from the automatic analysis of one corporate website, and the class is the presence or the absence of e-commerce facilities. The automatic generation of similar data records requires the use of several Text Mining phases; in particular we compare six strategies based on the selection of best words and best n-grams. After this, we classify the obtained dataset by means of four classification algorithms: Support Vector Machines; Random Forest; Statistical and Logical Analysis of Data; Logistic Classifier. This turns out to be a difficult case of classification problem. However, after a careful design and set-up of the whole procedure, the results on a practical case of Italian enterprises are encouraging.

Keywords: Classification; Big Data; Machine Learning; Data Engineering; Data Mining

1 Introduction

A common basis for several data engineering tasks is the collection of information regarding specific features of enterprises or similar entities. This information is usually obtained by means of surveys, which are costly due to the amount of personnel required for the practical realization of the task, especially personnel from all the respondent organizations. On the other hand, an automatic detection of the features of interest, if possible, would allow consistent savings. One important case is monitoring the adoption of e-commerce in enterprises. This information is necessary, for example, in several economic, social or statistical analysis. An automatic detection of the presence of e-commerce can be done, since in general this information is publicly available on-line through the corporate websites. However, to extract this information with some degree of reliability, several processing steps are required.

Given a set of data records grouped in two or more classes (i.e., labeled), the task of *classification* consists of learning from these labeled data a criterion to assign the class to new unlabeled data. The set of labeled data is called *training set*; another set of labeled data used to evaluate the results of a classification algorithm by comparing the predicted and the actual classes is called *test set*. Classification is a fundamental Data Mining aspect, and many different classification algorithms have been proposed in the literature. See for references, e.g., [16, 19]. The above described detection task can be seen as a classification problem, where each data record refers to the website of a single enterprise, and its class is presence or absence of e-commerce.

Obtaining such data records from the corporate websites is a Text Mining operation. Text Mining is the branch of Data Mining concerning the process of deriving high-quality information from text, see also [13].This area underwent noteworthy improvements in recent years (see, e.g., [22, 14]), with a number of concurrent factors contributing to its progress, first of all the continuous expansion of the Internet and the demand for effective search strategies.

However, the above described problem turns out to be a very difficult case of the classification problem, for several reasons. First, it has a very large dimension, because each record has thousands of fields. Secondly, the data records should be automatically generated from the content of each single website. This is a complex selection process that requires the use of different Text Mining phases. Thirdly, the problem is inherently difficult, since the data produced in this manner are inevitably noisy and not completely homogeneous. Web sites are of course not standardized, part of the information of a website is provided to the human users by means of the graphics rather than the text, etc. Therefore, to obtain a satisfactory accuracy in the classification phase, a quite articulated procedure has to be developed. Previous attempts in solving the same problem are described in [2, 3, 10].

In this work, we describe our strategy for classifying a list of 2,794 websites of Italian enterprises in order to automatically detect whether each website offers e-commerce facilities or not. We proceed as follows. Given the list of the websites of the above enterprises, we extract all the text from those websites by means of an automatic scraping procedure, and we use this text to prepare the data records. Such records should contain only the relevant part of each single website, that is in practice a selection of its words possibly integrated with additional information that could be automatically retrieved. Additional information may be: the presence of credit card or currency logos, the presence of login forms for the users, and in general any non-textual information that would be judged relevant to the analysis. These records are then used to write the *document-term* matrix, that is a matrix reporting, for each record, the frequency of each of the words in the above mentioned selection.

Data records and document-term matrix are obtained by applying several Text Mining procedures. In particular, we identified six different effective ways to process the dataset: using only the best words, with or without lemmatization; using only the best n-grams; using both the best n-grams and the best words, with or without the lemmatization; using the best n-grams and then a word embedding procedure. Hence, we obtain six different versions of the dataset, which are finally classified by using four state-of-the-art classification algorithms: Support Vector Machines; Decision Trees; Statistical and Logical Analysis of Data; Logistic classifier. These were selected through preliminary experiments in which such classifiers appeared the most adequate for similar problems.

The main content of this work is therefore the description of innovative mining techniques to convert the problem of the detection of e-commerce into a data classification problem, and a comparison of the practical results of these techniques. The work is organized as follows. Section 2 presents the different Text Mining procedures used. Section 3 describes the selected classification algorithms. Section 4 reports the practical results of the described approach and their analysis. Finally, Section 5 draw conclusions.

2 The Text Mining Phase

We initially extract the text from each single website, by using the web scraping procedure described in Subsection 2.1. This procedure reads the accessible pages of the website and saves to a single text file all the text (and possibly the additional information mentioned in Section 1) that could be retrieved. We receive in input the list of the corporate websites. The list used in our experiments had 2,794 entries, chosen randomly among the Italian enterprises with at least 10 persons employed, hence we obtained 2,794 very large text files. After this, each text file receives the class label, that is whether the corresponding website actually offers e-commerce facilities or not. We denote by D the dataset obtained, each element of D being a text file labeled with the class. To perform the classification task, we initially select within D a training set S composed of 50% of the elements of D. The remaining part of D constitute the test set T. Hence, the elements of T actually have the class label, but that label is kept hidden during the whole process until the classification of T has been performed, and it is only used afterwards, to evaluate the accuracy of the classification produced.

By working on the elements of S, we operate a number of selection steps, in order to obtain records containing, as much as possible, only information that is relevant for the classification, and not very large and unstandardized collections of text. These records are used to write the document-terms matrix. We have identified four basic manners to perform such selection steps, that are listed below. We describe them in detail in the following Subsections 2.2, 2.3, 2.4, 2.5, and we compare their performances in Section 4.

1. Best words with lemmatization.
2. Best words without lemmatization (but with a dictionary).
3. Best n-grams (with or without a dictionary).
4. Best n-grams followed by word embedding.

Some of the above techniques are obviously in mutual exclusion, while others can be combined. In particular, we use also the combination of 1 and 3 and the combination of 2 and 3. These combinations will constitute the fifth and sixth alternatives in Section 4. We observe that, generally speaking, the combinations often produce improvements in the results. However, these combinations also increase the computational burden of the resulting procedure, as it becomes the sum of those given by the two component basic techniques, while the improvements are much more limited. In the following subsections we describe in detail all the operations mentioned here.

2.1 Web Scraping

The web scraping procedure is implemented in java; it works by reading the homepage and all the sub-pages directly accessible from it. It subsequently discards the text of the sub-pages not containing any word from a pre-compiled list I of words of interest for e-commerce (e.g., price, cart, pay, buy, etc.). Indeed, when none of these words appear in a subpage, that page is likely not concerning e-commerce. The scraping procedure also discards all non-alphabetic strings of characters, while it handles all html and javascript code. Note that we do not read the pages not directly linked to the homepage, because they would add a consistent number of words that are however scarcely informative, and thus would mainly constitute additional noise.

On the other hand, for the accepted pages, this procedure performs also Optical Character Recognition (OCR) on all types of images, in order to read also the words provided as images, which are often written with such a special emphasis because they are particularly relevant. Furthermore, this procedure takes the screen-shot of the homepage and performs OCR on that, too. Indeed, many websites try to catch the eye on the key sentences of the homepage by adopting unusual writing techniques. OCR is achieved by using the Tesseract Open Source OCR Engine, initially developed by Hewlett Packard Research before 1995 and subsequently by Google Research since 2006 [26].

Finally, we complement the text extracted so far with the additional information mentioned in Section 1. In our case, we encode with special words the presence in a webpage of the following features: (1) a login form for the users registered to that website; (2) a credit card logo (visa, mastercard, etc.); (3) a currency logo (euro, dollar, etc.). This information may be further enriched by means of systematic interrogations of search engines. An important example is the text provided by a search engine as an automatic summarization of each search result.

The result of the above operations is a very large text file for each website. Now, each of such text files receives the class label, that is positive if the corresponding website actually offers e-commerce facilities, and negative otherwise. The class is obtained from the dataset used in [2], and it may contain a small degree of noise in the class label (misclassification errors). Note also that the obtained dataset D is quite imbalanced: positive records are not more than one fourth of the total. These two aspects contribute to make the classification difficult. However, they have been kept on purpose, since the real cases of the considered problem generally share these features.

2.2 Best Words with Lemmatization

We process all the elements in S with the software TreeTagger. This tool performs part-of-speech recognition (POS tagging) and lemmatization in several languages. It was initially described in [25] and it was subsequently developed by several contributors. Hence, every word contained in these training files is identified as a particular part of speech (e.g., a noun, a verb, etc.) and it is lemmatized, that is, the inflectional ending of the word is removed in order to return the word to its basic lemma. This allows to group together the different inflected forms of a word (e.g., plurals of nouns, tenses of verbs, etc.) so they can be analyzed as a single item.

This process is applied to any word, at first with the version of TreeTagger developed for the Italian language. If the word is an Italian word, this process returns the lemma corresponding to the word. If the word is not an Italian one, the above process returns that the word is unknown. In this case, the process described above is applied again with the version of TreeTagger developed for the English language. If the word is still unknown, this means that that word belongs to a different language, or that it may have been produced by some error; in any case, that word is discarded. Indeed, many websites of our list are in Italian language only, and we expect that the Italian and the English versions of the multi-language websites are enough for performing our classification task.

Finally, we use this "normalized" training files to extract the *best words*, i.e. the most characterizing words for our detection task. We do this by using a procedure implemented in Python that uses functions available from the Natural Language Toolkit (NLTK) 3.0 (see also [4, 23]). Such a procedure initially removes the stop-words (articles, prepositions, etc.), and then computes, for any remaining word w, a score $s(w) = \chi^2_+(w) + \chi^2_-(w)$, where χ^2_+ is called positive score and χ^2_- negative score. These scores are based on the well known "chi-square test", and the basic idea is to give a measure of the dependence between the presence of w and the class of a file. The positive score is defined as follows:

$$\chi^2_+ = \frac{p(p_{11}p_{22} - p_{12}p_{21})^2}{(p_{11} + p_{12})(p_{21} + p_{22})(p_{11} + p_{21})(p_{12} + p_{22})},$$

where p_{11} is the number of occurrences of w in positive files; p_{12} is the total number of occurrences of w; p_{21} is the number of all distinct words occurring in positive files; p_{22} is the total number of all distinct words; and $p = p_{11} + p_{12} + p_{21} + p_{22}$.

The negative score is defined similarly, except that p_{11} becomes the number of occurrences of w in negative files and p_{21} the number of all distinct words occurring in negative files. Recall that positive files are those generated from websites offering e-commerce facilities; negative files are those generated from websites not offering them. After this, all words are sorted by decreasing score values, and the first ones are called *best words*. In particular, we extract the first 1,000 of them, which constitute the set $W1$ of the 1,000 most characterizing words,

reduced to their underlying lemmas, in Italian or English language, selected on the basis of their conditional frequencies. The value 1,000 was selected as a good compromise between accuracy and speed in our experiments. If we select more than 1,000 words, the procedure becomes slower with very modest improvements in accuracy (allegedly, almost all the words that are really "characterizing" already appear within the first 1,000 positions). On the other hand, if we select less than 1,000 words, the procedure loses an accuracy not sufficiently compensated by the improvements in speed.

2.3 Best Words without Lemmatization

In this case, we perform the same operations described in the above Subsection, but without the lemmatization step. Instead of that, we simply filter all the text by using dictionaries: we accept only the words appearing in a (reasonably complete) list of all Italian and English words, containing also all the possible inflectional forms. These dictionaries were obtained from the mentioned software TreeTagger. In this manner, we extract the set $W2$ of the 1,000 most characterizing words, selected on the basis of their conditional frequencies, in Italian or English language but without detecting the underlying lemmas. Therefore, different inflections of the same lemma will be considered different words. However, this technique has the advantage of being much faster and lighter that the former one, from the computational point of view.

2.4 Best N-grams

N-grams are sequences of n adjacent words which are typically used together. An example is "credit card", which is a bi-gram, i.e., has $n = 2$. To extract the n-grams, we may optionally filter all the words of the text files by using dictionaries. This means accepting only the words appearing in the above mentioned list of all Italian and English words, including also all the possible inflectional forms. Using dictionaries would remove words in languages different from Italian or English, or words erroneously read by the scraping procedure. On the other hand, not using dictionaries would allow to consider also special strings or acronyms that are not real words but are commonly used. From the computational point of view, the first choice leads to a reduced set of candidate n-grams, which consumes less memory, while the second choice consumes more memory but it is faster. After evaluating the advantages of each choice, we adopted the first one for our experiments.

In the case of n-grams, lemmatization is not performed, since substituting words with their basic lemmas may result in losing the identity of many n-grams, which are generally built with specific inflectional forms.

After this, we prepare a list R of the *relevant* words. We define relevant the words that meet the following three criteria: 1) they appears frequently enough (we require that they appear in at least 1% of the text files belonging to one class); 2) they are not stop-words (articles, prepositions, etc.); 3) they respect some conditions on the length (basically we avoid words that are too short or too long, by penalizing words shorter than 3 letters or longer than 12 letters).

Subsequently, we apply the *ngrams* procedure, again from the Natural Language Toolkit (NLTK) 3.0, to identify the n-grams that contain at least one of the words of R. After this, we use another procedure to compute, for any such n-gram z, a score $s(z) = \chi^2_+(z) + \chi^2_-(z)$, where χ^2_+ is called positive score and χ^2_- negative score. These scores are again based on the well known "chi-square test", and the basic idea is to give some measure of the dependence between the presence of the words constituting z and the class of a file. Assuming that z is a bi-gram, the positive score is defined as follows:

$$\chi^2_+ = \frac{q(q_{11}q_{22} - q_{12}q_{21})^2}{(q_{11} + q_{12})(q_{21} + q_{22})(q_{11} + q_{21})(q_{12} + q_{22})},$$

where q_{11} is the number of positive files containing all the words constituting z; q_{12} is the number of positive files containing only the first word of z; q_{21} is the number of positive files

containing only the second word of z; q_{22} is the number of positive files not containing any of the words constituting z; and $q = q_{11} + q_{12} + q_{21} + q_{22}$. The negative score is defined similarly, except that all the above values are computed for negative files. In case z has $n \geq 3$, the above formula is consequently expanded. We extract n-grams using all the values of n from 2 to 5. In case an n-gram with n words is fully contained in a larger n-gram with $(n+1), \ldots, 5$ words, we remove the larger n-gram. We do this because we assume that the presence of the shorter n-gram is more significant than that of the larger one. We observe that, in our experiment, this practically leads to using almost only bi-grams, with consequent computational advantages.

All n-grams are sorted by decreasing values of the score s, and the first ones are called *best n-grams*. In particular, we take the first 1,000 of them, thus obtaining the set Z of the 1,000 most characterizing n-grams containing at least one relevant word of the set R. Again, the value 1,000 was selected as a good compromise between accuracy and speed in our experiments. If we select more than 1,000 n-grams, the procedure becomes slower with almost no improvements in accuracy (allegedly, almost all the n-grams that are really "characterizing" already appear within the first 1,000 positions). If we select less than 1,000 n-grams, the procedure loses an accuracy not sufficiently compensated by the improvements in speed.

2.5 Best N-grams followed by Word Embedding

In addition to the operations described in the above subsection, after the identification of the best n-grams, we apply a word embedding procedure. To this aim, we use the open source procedure *doc2vec*, available from the Python library GenSim [27].This procedure implements a strategy for the unsupervised learning of continuous representations for large blocks of text, such as sentences, paragraphs or entire documents. We initially reduce each single text file to its intersection with the words of the set Z, obtaining the sets of records S_Z, T_Z. Note that they actually constitute an automatic summarization of the texts.

We then use *doc2vec* to convert each record $r \in S_Z, T_Z$ into a vector v of k integer numbers that should represent r at a higher abstraction level. This is obtained by representing each word of Z as a point (i.e., a tuple of numerical coordinates) in a vector space, by evaluating the distances among these points, and by subsequently clustering these points into $k =\sim |Z|/5$ clusters by a k-means algorithm (see, e.g., [16]). The distance between two words is computed as the probability of finding them in the same context (that is, a sequence of words of predetermined length). Then, the vector v associated with the generic record r is composed of the cardinalities of the k intersections between: the set of all the words in r and the set of the words of each of the k clusters. Eventually, all these vector representations $\{v_h\}$ of the records constitute a set V. These vectors $\{v_h\}$ can be used as records themselves, and can undergo the classification step described in the next section.

2.6 The Data Records Produced

So far, we have described the selection of the sets $W1, W2, Z, V$. We also define the sets $A1 = W1 \cup Z$ and $A2 = W2 \cup Z$, composed of the single words and n-grams that should be more relevant for the considered classification task. Now, given a generic record r, either in the training set S or in the test set T, we define r_{W1} as the intersection between r and $W1$, that is, we keep in r_{W1} only the words belonging also to set $W1$. We similarly define r_{W2}, r_Z, r_{A1}, r_{A2}. Moreover, if we use the word embedding procedure and we substitute records with their vector representations V, we denote the generic vector by r_V. Hence, we have six different and alternative versions of each record: $r_{W1}, r_{W2}, r_Z, r_V, r_{A1}, r_{A2}$.

We denote by $S_{W1}, S_{W2}, S_Z, S_V, S_{A1}, S_{A2}$ the result of the above text mining operations on all the elements of the training set S, and by $T_{W1}, T_{W2}, T_Z, T_V, T_{A1}, T_{A2}$ the result of the same operations on all the elements of the test set T.

3 The Classification Procedures

Many different classification approaches have been proposed in the literature, based on different paradigms and data models. There is not a single approach capable to outperform all the others on every instance of the problem. However, given a specific category of problems, it is generally possible to identify which approaches provide the best performances for that category. We applied different classifiers by using scikit-learn [24], that is a very good machine learning package currently included into scientific Phyton distributions. In our case, the best results in the preliminary tests have been obtained with: Support Vector Machines; Decision Trees; Logistic classifiers. We also obtained comparable results with another technique called Statistical and Logical Analysis of Data, currently not included in scikit-learn. Therefore, we selected these four classifiers for our full experiments. They are briefly described below. The fields of each record corresponding to the words/n-grams constitute the set of input or independent variables; the class constitutes the target or dependent variable. For each of these classifiers, we search for the parameters providing the best accuracy performances by using a simple grid search approach. Note that a more careful parameter selection could lead to further modest improvements in these performances; however this is not the focus of this work.

Support Vector Machines (SVMs) are supervised learning models that build a deterministic binary linear classifier. This technique is based on finding a separating hyperplane that maximizes the margin between the extreme training data of opposite classes. New examples are then mapped into that same space and predicted to belong to a class, on the basis of which side of the hyperplane they fall on. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, by implicitly mapping their inputs to a higher dimensional space, see also [11, 28]. We use the Python implementation of this classifier that is available in the scikit-learn package by means of the function SVC().

Decision trees are a supervised learning model that maps observations about the input variables to conclusions about the target variable. The goal is to create a decision tree that predicts the value of the target variable based on combinations of the values of the input variables. Each internal node is associated with a decision concerning the value of an input variable that best splits the training set. Different algorithms can be used to determine the input variables associated with the internal nodes, see also [20]. Several decision trees can be combined as an ensemble classifier, obtained the so-called Random Forest approach. Random forests are generally more robust and can achieve better performances than the single decision trees. For this reason, we use such a version of the decision tree methodology in our experiments. We use the Python implementation of this classifier that is available in the scikit-learn package by means of the function RandomForestClassifier().

Statistical and Logical Analysis of Data (SLAD) is a classification methodology based on Boolean logic and discrete optimization [7]. It constitutes a recent version of the classical approach of Logical Analysis of Data (LAD) [5] incorporating scoring criteria proposed in [6]. To apply this approach, all values must be converted into binary form by means of a discretization process called binarization. The domain of each field is partitioned in a finite number of subdomains that are encoded using binary attributes. Since the number of obtained binary attributes is often very large, a selection step is performed. After this, the selected binary attributes are used to build the patterns. A pattern is a conjunction of binary attributes, also called conditions, characterizing one class. Each pattern receives a weight, that is a measure of its importance for the classification. Finally, each unlabeled record is classified on the basis of the weighted sum of the patterns covering that record. We use the C++ implementation of this classifier that was used in the work [7].

Logistic regression is a regression model where the target variable is categorical; hence, it can be used to perform classification. This approach measures the relationship between the target variable and one or more independent variables by estimating the probabilities using a logistic function, which is the cumulative logistic distribution, see also [15]. Logistic regression can be seen as a special case of the generalized linear model and thus analogous to linear regression. We use the Python implementation of this classifier that is available in the scikit-learn package

by means of the function LogisticRegression().

4 Experimental Results

We apply the described procedure to a list of 2,794 corporate websites. Each record has a class value, although this information may be noisy. A record is positive if the corresponding website offers e-commerce facilities, and it is negative otherwise. The dataset is very imbalanced, since roughly 20% of the entries are positive. The resulting classification problem is very difficult. Indeed, it is very easy to reach an 80% of accuracy by simply predicting all records as negative. However, this result would be completely useless from the practical point of view. Obtaining the correct identification of the positive records constitutes the main target in this type of applications, and this identification is particularly challenging.

To perform the classification task, in each dataset we select a training set S of 1,397 records, that is 50% of the total dataset. To tackle the issue of imbalance in the training set (see also [17, 18]), we operate as follows. First, we perform a partial resampling by randomly undersampling the majority class and oversampling the minority class (by replication), until obtaining a dataset of the same size but containing roughly 40% positive entries. Then, we adjust the misclassification costs (computed during the training phase) by using weights inversely proportional to the class frequencies in the resampled training set. Finally, we focus on performance measures possessing low sensitivity to data imbalance.

We perform experiments using the six versions of the dataset described at the end of Section 2 and the four classifiers described in Section 3. In the following tables, SVM denotes Support Vector Machines; RF denotes Random Forest; SLAD denotes Statistical and Logical Analysis of Data; LC denotes Logistic Classifier.

After training the classifiers on the training sets $S_{W1}, S_{W2}, S_Z, S_V, S_{A1}, S_{A2}$, performing 5-fold cross-validation, we obtain the sets of the predictive models $M_{W1}, M_{W2}, M_Z, M_V, M_{A1}, M_{A2}$. Then, we use them to predict the class for the records in the corresponding test sets T_{W1}, T_{W2}, T_Z, T_V, T_{A1}, T_{A2}. The number of terms actually included in each predictive model is clearly quite variable, depending on the type of dataset, the classification algorithm, etc. However, we could estimate for this number a range going approximatively from 10% to 50% of the total in the majority of the cases. Finally, by knowing the real class of the records in the above test sets, we compute the confusion matrix and we use its elements (true positives TP, false negatives FN, true negatives TN, false positives FP) to evaluate the following performance measures:

- Accuracy a, defined as the percentage of correct predictions over all predictions:
$$a = \frac{100(TP + TN)}{TP + FN + TN + FP}.$$
- Precision p, also called the positive predictive value, defined as the percentage of true positive records in all positive predictions: $p = \frac{100\ TP}{TP + FP}$.
- Sensitivity s, also called the true positive rate, defined as the percentage of correct positive predictions in all real positive records: $s = \frac{100\ TP}{TP + FN}$.
- F1-score, which is the harmonic mean of precision and sensitivity:
$$F_1 = \frac{200\ TP}{2TP + FP + FN}.$$

Note that, for the detection of e-commerce, F1-score appears to be the most relevant performance measure, since it fully evaluates the correct identification of the positive records, that is the most important and difficult task, and because it has low sensitivity to data imbalance. Due to its broad usage, we also report the classification accuracy, tough we should observe that this measure typically does not provide enough insight and it is sensible to data imbalance.

Table 1 reports the results of the four described classifiers on the records obtained using only the best words with lemmatization (S_{W1}, T_{W1}); Table 2 reports the same results on the records obtained by using only the best words without lemmatization (S_{W2}, T_{W2}); Table 3 reports the same results on the records obtained by using only the best n-grams (S_Z, T_Z); Table 4 reports the same results on the records obtained by using the vector representations given by the word embedding procedure (S_V, T_V); Table 5 reports the same results on the records obtained by using the best words with lemmatization and the best n-grams (S_{A1}, T_{A1}); finally, Table 6 reports the same results on the records obtained using the best words without lemmatization and the best n-grams (S_{A2}, T_{A2}).

Table 1: Results on S_{W1}, T_{W1}: best words with lemmatization.

Algorithm	Accuracy	Precision	Sensitivity	F1-score
SVM	84.97 %	68.34 %	58.03 %	62.77 %
RF	84.82 %	67.29 %	59.34 %	63.07 %
SLAD	84.70 %	64.51 %	60.24 %	62.30 %
LC	83.54 %	64.04 %	56.07 %	59.79 %

Table 2: Results on S_{W2}, T_{W2}: best words without lemmatization.

Algorithm	Accuracy	Precision	Sensitivity	F1-score
SVM	85.11 %	65.33 %	63.59 %	64.45 %
RF	84.47 %	68.75 %	61.31 %	63.82 %
SLAD	84.45 %	64.15 %	59.03 %	62.54 %
LC	83.54 %	64.04 %	56.07 %	59.79 %

Table 3: Results on S_Z, T_Z: best n-grams.

Algorithm	Accuracy	Precision	Sensitivity	F1-score
SVM	85.11 %	77.09 %	45.24 %	57.02 %
RF	85.18 %	69.74 %	62.02 %	65.43 %
SLAD	85.47 %	64.67 %	61.00 %	62.68 %
LC	83.32 %	65.38 %	50.16 %	56.77 %

Table 4: Results on S_V, T_V: best n-grams followed by word embedding.

Algorithm	Accuracy	Precision	Sensitivity	F1-score
SVM	82.89 %	60.78 %	60.98 %	60.88 %
RF	84.75 %	66.43 %	60.98 %	63.59 %
SLAD	84.72 %	66.13 %	60.58 %	63.39 %
LC	82.61 %	61.65 %	53.77 %	58.44 %

By analyzing the above results, we observe what follows.

1. The Random Forest classifier (RF) provides the best performances in our experiments. However, we notice that Support Vector Machines (SVM) and Statistical and Logical

Table 5: Results on S_{A1}, T_{A1}: best words with lemmatization and best n-grams.

Algorithm	Accuracy	Precision	Sensitivity	F1-score
SVM	84.75 %	65.97 %	62.30 %	64.08 %
RF	85.61 %	69.26 %	62.31 %	66.04 %
SLAD	85.97 %	66.67 %	62.18 %	64.31 %
LC	83.60 %	64.29 %	56.77 %	60.89 %

Table 6: Results on S_{A2}, T_{A2}: best words without lemmatization and best n-grams.

Algorithm	Accuracy	Precision	Sensitivity	F1-score
SVM	84.32 %	64.14 %	63.93 %	64.04 %
RF	85.76 %	69.06 %	62.95 %	65.87 %
SLAD	84.60 %	64.11 %	60.24 %	61.30 %
LC	83.82 %	64.79 %	56.72 %	60.49 %

Analysis of Data (SLAD) are not much worse in the same experiments. In particular, SVM techniques are generally able to give more stable results, since they are less sensible to overfitting issues, and they also exhibit a better flexibility with respect to the other classifiers (and for these reasons SVM are often used in practical applications, see, e.g., [1, 21, 12]). SLAD, on the other hand, offers the advantages of providing intelligible logic rules for the classifications, which could be used for gaining more insight on the analyzed phenomenon (see, e.g., [8]).

2. The use of the best n-grams alone is able to provide results that, for this particularly difficult classification problem, are very good (Table 3). However, the joint use of best n-grams and best words (Tables 5 and 6) allows to attain a slightly better performance. Moreover, this last option is considerably more robust, as shown by the results of all the four classifiers and not only those of the best one.

3. When focusing on the results of the best words (Tables 1 and 2), we can analyze the differences between the use of lemmatization and the use of a dictionary. In our case, using only a dictionary, without returning each word to its basic lemma, appears slightly more convenient. This means that the presence of specific inflectional forms has a precise significance with respect to the presence of e-commerce. Indeed, in the vast majority of the cases of e-commerce, very specific inflectional forms of the key words are used (e.g., the present of the verbs buy, pay, book, etc.) On the other hand, when using also the n-grams (Tables 5 and 6), the results become almost comparable. Apparently, many specific inflectional forms are captured also by the n-grams, so the possible advantages of not using lemmatization cease. In any case, lemmatization has the advantage of saving some memory, because different words may correspond to the same lemma.

4. The use of the word embedding technique can provide slightly more robustness with respect to the use of best n-grams alone. The improvements are not fully apparent in our experiments, because the results of this technique usually improve when the number of n-grams used in it becomes consistently larger than 1,000. However, this was not possible in our experiments, due to the very large memory requirements of this technique. Testing the use of larger sets of n-grams with this technique will be part of our future work.

The computational times and the memory usage of the procedures are quite reasonable, considering the large size of the problem. In particular, using a PC with i7 processor and 16GB RAM, the fastest technique is the one producing S_Z, T_Z (only the best n-grams), which takes about

2 hours. The use of the dictionary and the set of relevant words R avoids the generation of several useless n-grams, saving also memory. The generation of S_V, T_V requires a very moderate additional time, but a considerable additional memory. The generation of S_{W2}, T_{W2} requires slightly more time, about 3 hours, tough the memory usage is inferior. On the other hand, the generation of S_{W1}, T_{W1} by using lemmatization is a much slower process, and requires about 6 hours, tough the memory usage is still inferior because different words may correspond to the same lemma. The combinations of best words and best n-grams clearly require the times and the memory needed by its two components. Hence, the generation of S_{A2}, T_{A2} requires about 5 hours, while the generation of S_{A1}, T_{A1} needs about 8 hours. The small enhancement in performances is therefore paid with a considerable increase in complexity.

5 Conclusions

In several large-scaled surveys, the automatic individuation of some feature of interest can be seen as a classification problem. Determining whether an enterprise website offers e-commerce facilities or not is a particularly interesting case. However, to solve the problem by means of a classification algorithm, it is necessary to convert each website into a record describing that website in a compact and tractable manner. Such records should contain only the relevant portion of the information of the websites, and we found that a careful selection of the information inserted in the records is a key element for obtaining satisfactory performances in the classification. We have presented several alternative techniques for doing this, and we have observed that introducing too much information may be not beneficial because of the amount of noise introduced at the same time. Another issue faced in the classification of similar data is data imbalance. This can be addressed by recurring to resampling, weights' adjustments in the learning phase, and the use of performance measures with low sensitivity to data distributions.

The results of our automatic procedure may be used either for replacing surveyed data for statistical purposes, thus saving the cost of actual surveys (especially for the respondents), or to validate and enrich already surveyed data, for example those contained in the Business Register of a Statistical Office, thus improving their quality and reliability. The present analysis should support in designing a procedure to solve real-world problems of the described type, possibly integrated by tacking into account the specific requirements of the analyzed practical case. Future work may include the extension of the described procedure to other languages different from Italian and English; the evaluation of other classification algorithms in this task; an analysis of the robustness of the proposed approach [9]; the application of the detection via web to other features different from the presence or the absence of e-commerce.

Conflict of Interest. The authors declare that there is no conflict of interest regarding the publication of this article.

References

[1] L.M. Rabelo Baccarini, V.V. Rocha e Silva, B. Rodrigues de Menezes, W. Matos Caminhas, SVM practical industrial application for mechanical faults diagnostic, Expert Systems with Applications, 38(6), 6980-6984, 2011.

[2] G. Barcaroli, G. Bianchi, R. Bruni, A. Nurra, S. Salamone, M. Scarnò, Machine learning and statistical inference: the case of Istat survey on ICT, Proceeding of 48th scientific meeting of the Italian Statistical Society SIS 2016, Salerno, Italy, 2016. Editors: M. Pratesi and C. Pena ISBN: 9788861970618

[3] G. Barcaroli, A. Nurra, S. Salamone, M. Scannapieco, M. Scarnò, D. Summa, Internet as Data Source in the Istat Survey on ICT in Enterprises, Austrian Journal of Statistics 44(2), 31–43 (2015).

[4] S. Bird, E. Klein, E. Loper, Natural Language Processing with Python. OReilly Media, 2009.

[5] E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik, An Implementation of Logical Analysis of Data, IEEE Transactions on Knowledge and Data Engineering 12(2), 292-306 (2000).

[6] R. Bruni, Reformulation of the Support Set Selection Problem in the Logical Analysis of Data, Annals of Operations Research Vol. 150(1), pag. 79-92 (2007).

[7] R. Bruni, G. Bianchi, Effective Classification using a small training set based on Discretization and Statistical Analysis, IEEE Transactions on Knowledge and Data Engineering 27(9), 2349-2361 (2015).

[8] R. Bruni, G. Bianchi, C. Dolente, C. Leporelli, Logical Analysis of Data as a Tool for the Analysis of Probabilistic Discrete Choice Behavior, Computers & Operations Research DOI: 10.1016/j.cor.2018.04.014, 2018.

[9] R. Bruni, G. Bianchi, Robustness Analysis of a Website Categorization Procedure based on Machine Learning, Technical report DIAG, Sapienza University of Rome, forthcoming.

[10] D. Blazquez, J. Domenech, J. Gil, A. Pont, Automatic detection of e-commerce availability from web data, Proceedings of First International Conference on Advanced Research Methods and Analytics (CARMA2016), València, Spain, 2016.

[11] C.-C. Chang and C.-J. Lin, Training ν-support vector classifiers: Theory and algorithms, Neural Computation, 13(9), 2119-2147 (2001).

[12] M. De Santis, F. Rinaldi, E. Falcone, S. Lucidi, G. Piaggio, A. Gurtner, L. Farina, Combining optimization and machine learning techniques for genome-wide prediction of human cell cycle-regulated genes, Bioinformatics, 30(2), 228-233, 2013.

[13] R. Feldman, J. Sanger, The Text Mining Handbook. Cambridge University Press, 2006.

[14] W.W.M. Fleuren, W. Alkema, Application of text mining in the biomedical domain, Methods, 74, 97-106 (2015).

[15] D.A. Freedman, Statistical Models: Theory and Practice. Cambridge University Press, 2009.

[16] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2nd ed., Springer, 2009.

[17] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering 21(9), 1263-1284, 2009

[18] H. He, Y. Ma (eds), Imbalanced Learning: Foundations, Algorithms, and Applications IEEE Press, 2013. ISBN: 978-1118074626

[19] W. Klosgen, J.M. Zytkow (eds), Handbook of Data Mining and Knowledge Discovery, Oxford University Press, 2002.

[20] W.-Y. Loh Fifty years of classification and regression trees, International Statistical Review, 82(3), 329-348, 2014.

[21] Y. Ma, G. Guo (eds), Support Vector Machines Applications, Springer, 2014. ISBN 978-3-319-02299-4

[22] A.K. Nassirtoussi, S. Aghabozorgi, T.Y. Wah, D.C.L. Ngo, Text mining for market prediction: A systematic review, Expert Systems with Applications, 41(16), 7653-7670, 2014.

[23] NLTK 3.0 Documentation. http://www.nltk.org/book/

[24] F. Pedregosa et al., Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011).

[25] H. Schmid, Improvements in Part-of-Speech Tagging with an Application to German. Proceedings of the ACL SIGDAT-Workshop. Dublin, Ireland, 1995.

[26] R. Smith, An Overview of the Tesseract OCR Engine. in Proc. of the Ninth International Conference on Document Analysis and Recognition, 629-633, 2007 ISBN:0-7695-2822-8 IEEE Computer Society, Washington, USA, 2007.

[27] R. Rehurek, P. Sojka, Software Framework for Topic Modelling with Large Corpora, Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Malta, 2010.

[28] V. Vapnik, The Nature of Statistical Learning Theory. Springer, second edition, 1995.