International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 15(2023) pp. 489-499
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Submitted: 3 December, 2022; Accepted: 5 Apr, 2023; Publish: 23 June, 2023

Semantic Data Type Detection via Machine Learning using Semi-synthetic Data and Robust Features

Marc Chevallier[1], **Nicoleta Rogovschi**[1], **Faouzi Boufarès**[1] **and Nistor Grozavu**[1]

[1]LIPN Laboratory, Sorbonne Paris Nord University
99 Av. Jean Baptiste Clément, 93430 Villetaneuse, France
chevallier@lipn.univ-paris13.fr
nicoleta.rogovschi@lipn.univ-paris13.fr
faouzi.boufares@lipn.univ-paris13.fr
nistor.Grozavu@lipn.univ-paris13.fr

Abstract: **Being able to automatically identify the semantic type of tabular data is a useful feature in many areas of the data landscape. This information is especially important for data integration, data science, and data cleaning. Traditional semantic type detection tools based on dictionaries and regular expressions have recently been challenged by methods using machine learning. These new methods are very efficient but require a large amount of data to learn, which limits the use of these methods to semantic types and languages for which a large amount of data is available. To overcome these drawbacks, we introduce a data generation method to produce training data with minimal real data. In addition, we propose several new feature extraction methods that are less dependent on column length, language independent (for a given alphabet) and robust to errors in the data. Experiments conducted on synthetic and real data indicate an accuracy higher than 0.9 which is equivalent to classical methods.**

Keywords: Type detection, Tabular data, Semantic types, Data Profiling, Knowledge discovery, Machine learning

I. Introduction

Detecting the semantic type of tabular data is a crucial task, as this information is widely used in many fields. For example, for data integration, matching schemas requires precise identification of the semantic type in order to find matches between columns of several tables [1]. For automatic data cleaning, the semantic type of the data is used to build rules for data validation and transformation [2, 3]. Finally, in the context of data mining, the semantic type of the data is used to determine the most relevant results to a query.
First of all, we are interested in the semantic types of the columns and not the atomic types. Atomic types are very basic information about the content of the column and are easily identifiable; for example binary, number or string types. This article focuses on semantic types which are complex information such as ”last name”, ”first name” or ”postal code”. The automatic detection of semantic types is typically done using regular expressions and dictionary lookups. Most commercial softwares such as Trifacta[1], Tableau[2], or Microsoft Power Bi[3] search for the semantic type of data in this way. However, they can only identify a limited number of semantic types. The method they use requires that the data do not contain noise and that the semantic type follows a repetitive pattern (e.g., zip codes). For semantic data types that are outside of the most common types, an expert is needed to determine the method to use to detect the type, which is limiting and expensive.

To overcome these drawbacks, new approaches using machine learning have appeared over the last few years [4, 5, 6]. One of the best-known methods is Sherlock [7], which treats the problem of detecting the semantic type of data as a multiclass classification problem. Out of each column, Sherlock extracts 1588 features divided into 4 main types. The ”Global statistics” type features which describe how the values are distributed in the column, the ”Character Distribution” features that describe how the characters are distributed in the columns, and finally, the ”Word embeddings” and ”Paragraph vectors” features that use two word embedding techniques to describe the columns in two different ways. This method is more robust to errors in the data and performs better than conventional methods.
Sherlock’s method has led to refinements, including Sato [8], which considers the problem as a multi-column prediction problem. This means that it predicts the semantic types of all columns in a table at once. This is done in order to improve the predictions by learning the relationships (e.g. a column of first names is often accompanied by a column of last names) that may exist between the columns.

[1]https://www.trifacta.com/fr/
[2]https://www.tableau.com
[3]https://powerbi.microsoft.com

In the following, we present the contribution proposed in section II, first introducing a data generation process and then feature extraction methods. The experimental results obtained are then discussed in section III, before concluding in section IV. The overall process is summarized in the diagram 1.

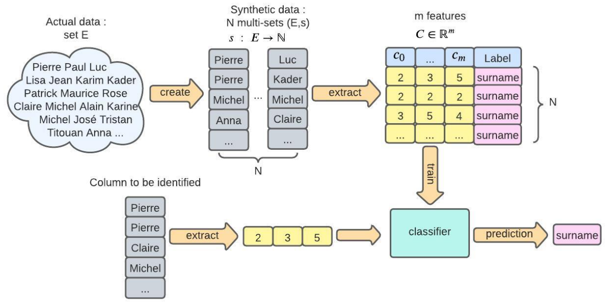

Figure. 1: Diagram summarizing the global functioning of the method

A. *Machine Learning for semantic type detection*

The approach proposed in this work differs from the literature in several points. Indeed, our method is designed to be accessible and reusable in any activity domain, independently of semantic categories. Moreover, it is designed to overcome several shortcomings of the leading algorithm in the field: Sherlock [7]. The first obstacle to the use of Sherlock as well as all other methods using machine learning for this task is the volume of data used [9].

The method presented in Sherlock (for 78 semantic categories) requires 60% of a dataset containing 686765 columns (from VizNet [10]). It is therefore difficult to use the Sherlock approach with other semantic categories than those used in the original article [9]. Indeed, it's hard to get enough examples to use Sherlock on customized semantic categories. Especially since it is shown in Sato [8] that the accuracy rate of Sherlock depends strongly on the number of examples present for the class (i.e. unbalanced classes). Actually, the classes with a low number of examples are the least recognized. To cope with this problem, we introduced a synthetic data generation algorithm. Unlike the classical oversampling in machine learning, in the proposed generation algorithm, the data are created in the same initial feature space (mixed variables). The latter uses a set containing the largest possible number of elements of the targeted semantic domain (the data used were manually collected), to generate the desired number of columns containing these elements using Dirichlet's law (to generate pseudorandom vectors), see (II).

The second limitation is Sherlock's use of two word embedding models during the extraction of column features. This limits the use of the algorithm to one language at a time. Furthermore, not all languages have pre-trained models for these algorithms. To avoid this problem, we did not use the features from word embedding so that our method can be used in multiple languages. The last point to discuss is the number of rows in the columns used. In Sherlock, the number of rows in a column is an important information for the model. This number is used both directly and indirectly in the calculation of several features. This results in a reduction of the model's accuracy when confronted with data not coming from the database used in the article (where the columns do not have the same length). Thus Khurana and Galhotra show that the average accuracy rate observed on 9 datasets is 26.6% for Sherlock and 26% for Sato [11] (which is very low). In order to solve this problem, features that could grow infinitely with the number of lines will not be used.

II. Data generation

In this section, we present the proposed approach for data generation and augmentation. Starting from a data set D describing a semantic type with Card(D) = K, z multisets of cardinal l will be generated (our synthetic columns of length l). This will be done by associating a probability to each element of E.

These probabilities are created from a random vector $[x_1 \dots x_K]$ respecting a Dirichlet distribution of parameters $\alpha_1, \dots, \alpha_K$ [12]. The probability density function of the Dirichlet distribution of order $K \geq 2$ of parameters $\alpha_1, \dots, \alpha_K$ is defined as follows:

$$f(x_1, \dots, x_K; \alpha_1, \dots, \alpha_K) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{1}$$

$$\mathrm{B}(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}, \qquad \alpha = (\alpha_1, \dots, \alpha_K). \tag{2}$$

with

$$\sum_{i=1}^{K} x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i \in \{1, \dots, K\} \tag{3}$$

The vectors respecting the Dirichlet distribution are obtained using a gamma distribution.

$$f(y; \alpha_i, \theta) = \frac{y^{\alpha_i - 1} \mathrm{e}^{-\frac{y}{\theta}}}{\Gamma(\alpha_i) \theta^{\alpha_i}} \tag{4}$$

By drawing K independent variables of following gamma distribution:

$$f(\alpha_i, 1) = \frac{y^{\alpha_i - 1} \mathrm{e}^{-y}}{\Gamma(\alpha_i)} \tag{5}$$

Finally each dimension of the vector is calculated using the following sum.

$$x_i = \frac{y_i}{\sum_{q=1}^{K} y_q} \tag{6}$$

The vectors generated this way have a sum equal to 1. Moreover, the distribution of the values in the vector can be modified by changing the parameters α. In the following, the parameters $\alpha_1, \dots, \alpha_K$ will be referred as α because we use the same value for all dimensions.

To summarize in order to generate a data vector of size l from a real data vector $[d_1 \dots d_K]$ of size K (describing a semantic type), first, we need to generate a vector $[x_1 \dots x_K]$ following a Dirichlet Distribution. Then draw l

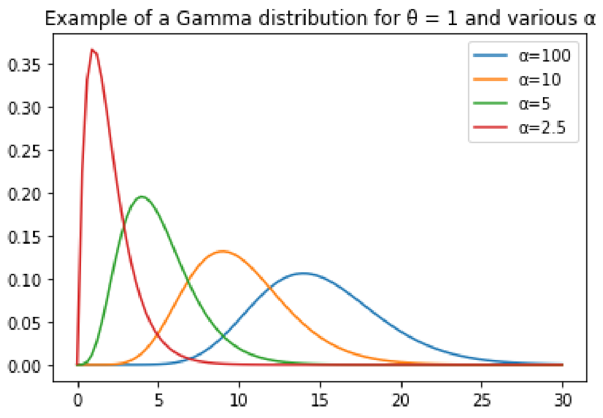

Figure. 2: Example of a Gamma distribution for various α and θ=1

elements in $[d_1 \ldots d_K]$ by using as probability of drawing for each d_i the value x_i.
The smaller the value of α, the more likely it is that the distribution of values in $[x_1 \ldots x_K]$ is unbalanced (this will change the distribution of the gamma distribution that is used to generate each dimension of the vector).
The algorithm 1 presents a method to generate nbr_elem examples (columns) from a vector of real data D for a given α value.
This method allows to generate data of all types (numerical, alphanumerical). We have chosen to use the Dirichlet distribution because it allows to generate "customized" examples by modifying the α parameter vector; we can thus generate a larger variety of examples, as the real distribution of the data is unknown.

```
Algorithm 1: Data_generation
  input : D:list, nbr_elem:int, $\alpha$:int, $max\_col\_size$:int,
    $min\_col\_size$:int
  initialization : $x \leftarrow []$,
  $prb \leftarrow nbr\_elem$ samples of dimension length(D)
    from a Dirichlet distribution of parameter $\alpha*$ "vector
    of ones" of size length(D)
  for $i \leftarrow 0$ to $nbr\_elem$ - 1 do
      $n\_elem \leftarrow$ random value between $min\_col\_size$
        and $max\_col\_size$
      $x[i] \leftarrow n\_elem$ elements of $D$ chosen according
        to the distribution $prb[i]$
  end
  return $x$
```

A. Feature extraction process

The transformation of columns of different lengths into vectors of fixed size is an essential pre-requisite for the use of most machine learning models. The method described in this article is inspired by the one presented by Sherlock [7] however only a subset of the features presented in Sherlock is used [4].

[4]https://github.com/mitmedialab/sherlock-project

First, features that depend directly on the language of the words in the column are not used, so features such as "Word embeddings" and "Paragraph vectors" (which use classical text vectorization methods) are not reused. Additionally, the characters within the columns are transformed to lowercase.
The 512 features extracted from the columns are divided into two main types. The so-called general features are composed of two subcategories. The first one consists of general statistics (22 features) on the column: entropy, presence or absence of null value, percentage of each type of values (numeric, textual, null). This grouping also contains the mean and standard deviation of the number of cells containing characters of the following types: alphabetic, numeric and special or the mean and standard deviation of the number of words in each cell. The second subcategory of features is created by using statistics (Mean, Min, Max, Var, Median) on the length of the strings in the column.
The second type of features is centered on the characters, it consists of statistics calculated on the number of occurrences of each character in each cell. We have chosen to treat the following characters: {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'é', 'è',' ! ', '"', '#', '$', '%', '&', '"', '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '¡', '=', '¿', ' ? ', '@', '[', ']', '_', '`', '{', '|', '}', '~', ´, '\x0c'}.
We use the characters of the Latin alphabet because we target columns using this alphabet, but this approach can be customized for other alphabets by modifying the characters employed. The computed statistics are: Min, Max, Mean, Median, Variance, presence or absence of the character and presence or absence of the character in all rows of the column. This gives a total of 490 features. The whole feature extraction process is summarized in figure 3.

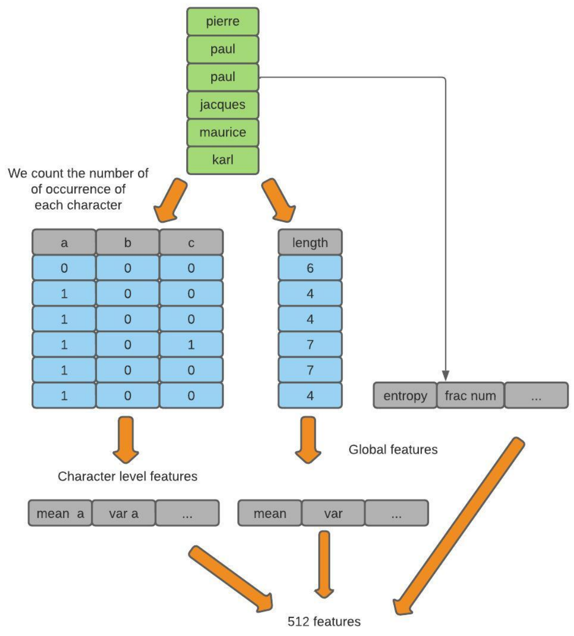

Figure. 3: Process for creating the feature vector

B. *Robust features and Bigram*

Real data are usually noisy, the subject of noise in labels has already been studied using a method reusing Sherlock features [13]. However, while these features are effective they are not robust to some kind of error in the data. For example the feature based on the presence of a letter in all the elements of the column or those based on the total absence of a letter in the column are not robust [14]. Indeed, changing a single element of the column, regardless of its length, can change these characteristics from true to false.
In addition, most of the statistics used are not robust, so the mean, minimum, maximum and variance are very sensitive to outliers in the data. This can lead to poor prediction if there is only one outlier in the column.
We therefore decided to propose a new method of feature extraction more robust to outliers. To do this we removed the features based on the presence or absence of a letter and introduced robust statistics instead of the old statistics in the feature vector.
First the minimum and maximum are replaced by the 15th and 85th percentile. Then the central tendency statistics (mean and median) are replaced by the trimean denoted here TRI [15].

$$TRI = \frac{Q_1 + 2 * Med + Q_3}{4} \tag{7}$$

Finally the variance is replaced by the biweight midvariance [16] denoted BWMV.

$$BWMV = n * \frac{\sum_{|u_i|<1}(x_i - Med)^2(1-u_i^2)^4}{(\sum_{|u_i|<1}(1-u_i^2)(1-5u_i^2))^2} \tag{8}$$

$$u_i = \frac{(x_i - Med)}{c * MAD} \tag{9}$$

with $c = 9$ and MAD is the median absolute deviation.

Another improvement we propose is the addition of bigrams to the list of counted characters (while removing the characters ‘, ˜and \0c). In order to avoid that these bigrams are dependent on a single language. We have used the top 10 most found bigrams within the following 6 languages: Polish, French, German, Spanish, English, Italian. The bigrams thus chosen are: ’an’, ’en’, ’es’, ’er’, ’re’, ’ar’, ’de’, ’el’, ’he’ ’in’.
The new robust vectors lengths are 287 for the normal version and 327 for the extended version. The extended version of the standard feature vector is 582.
In order to illustrate the difference between the feature extraction methods (Sherlock, standard version, robust version), we will use a column of 30000 elements containing French movie titles (FILM-FRENCH). We will then draw randomly 500 subsets of sizes {5, 10, 20, 30, 50, 100, 200, 500, 1000, 2000, 5000}. We will then extract from each of these subsets the features using the 3 methods (depriving Sherlock of the features ”Word embeddings” and ”Paragraph vectors”), then compute the Euclidean norm of the feature vector. The average of the norms for each size for both methods is shown in the figures 4 and 5.
These representations allow us to observe that our methods create feature vectors whose values do not tend to infinity as

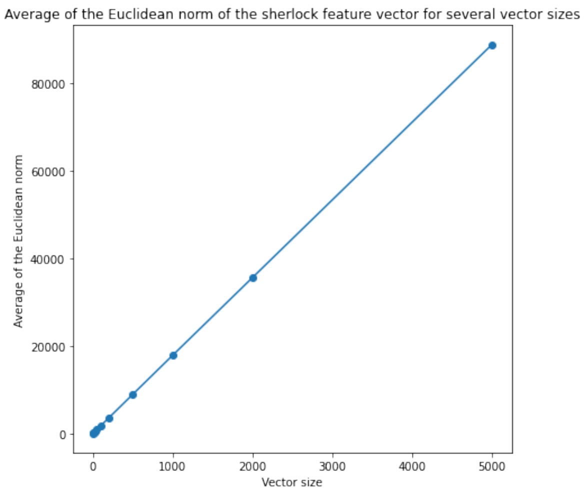

Figure. 4: Average of the Euclidean norm of the Sherlock feature vector for several vector sizes (for a column of semantic type FILM-FRENCH)

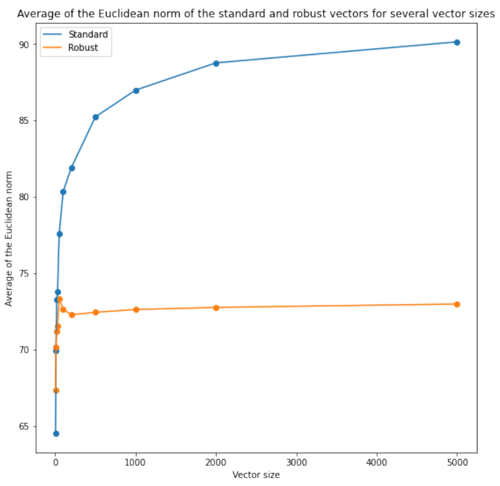

Figure. 5: Average of the Euclidean norm of standard and robust vectors for several vector sizes (for a column of semantic type FILM-FRENCH)

the size of the target column increases, while several features from Sherlock’s method grow indefinitely.

size	std Sherlock	std Standard	std Robust
5	34	35	40
10	27	26	27
20	34	20	20
30	43	17	16
50	55	14	12
100	74	11	9
200	106	8	6
500	159	5	4
1000	231	3	3
2000	374	2	2
5000	559	1	1

Table 1: Standard deviation of the Euclidean norm of the Sherlock feature vector for several vector sizes (for a column of semantic type FILM-FRENCH) for both methods part a.

The table 1,2 represents the standard deviation and coeffi-

size	c_v Standard	c_v Robust
5	0.54	0.61
10	0.37	0.39
20	0.27	0.28
30	0.23	0.22
50	0.19	0.17
100	0.14	0.12
200	0.1	0.08
500	0.06	0.05
1000	0.04	0.04
2000	0.02	0.02
5000	0.01	0.01

Table 2: Standard deviation of the Euclidean norm of the Sherlock feature vector for several vector sizes (for a column of semantic type FILM-FRENCH) for both methods part b.

cient of variation of the results from the previous experiment. We can see that our feature vectors have a high standard deviation when the number of rows in the column is low. This observation will lead us to use two different machine learning models.

C. *Method for learning*

The proposed method uses two classifiers, one for columns containing many rows named "high", the other for columns containing few rows named "low". For each case, we chose to try two types of classifiers, a random forest (RF) [17] as it is shown in Sherlock that this algorithm performs well in this task and Catboost [18] whose results exceed those of RF for other tasks using similar features [19, 20, 21, 22, 23].
Each model is calibrated using the isotonic method with 5 cross-validations (the final probability is the average of the predictions of the 5 classifiers) [24]. The two RF use two different datasets as training data. One is generated with small column sizes ranging from 5 to 30 rows and the other with a number of rows ranging from 31 to 1000. Each of these datasets is composed of 57 classes (one part of the classes is in French and the other in English, all data and their descriptions are available on github [5]), each class has 21000 examples. Among these 21000 examples generated with the algorithm 1, 7000 are generated with an α equal to 0.00001 , 7000 with an α equal to 0.001 and finally 7000 with an α equal to 1. We have thus at the end two data sets of 1197000 examples each.

III. Experimental results and discussions

This section will present all the results and experiments that have been performed. The experiments are performed on a Colab instance with a Xeon 2.30GHz 4-core, 25GB of Ram and a Tesla P100 (16GB). The parameters used for the RF are: a maximum depth of 18 and 200 estimators. The parameters used for Catboost are: 1500 iterations, a depth of 9 and a learning rate of 0.04.

[5]https://github.com/Marc-Chevallier/SOCPAR22

A. *Experiments and results with standard features*

Our first experiment aims at evaluating the accuracy and the F1-score of our two classifiers on data that we will generate again with the Algorithm 1. Two test data sets are then created for each of the two models. One will be generated with columns whose size varies from 5 to 30 rows, the other with a number of rows between 31 and 9000. The α values vary from 10^{-7} to 100 and are modified by power of 10. For each α value, 50 examples are generated per class, so we have a total of 28500 examples for each test dataset. The results of these first experiments are presented in table 3.
In almost all configurations, Catboost outperforms RF with an overall accuracy rate of 0.9413 vs. 0.9310 (for the "high" version, the "low" version being less interesting due to the instability of the characteristics). This is not surprising since Catboost is considered a more powerful classifier than RF in this kind of problems. Both models show excellent results for α values above 10^{-3}. Below this value, the results decrease sharply, despite training examples generated with α equal to 10^{-5}. This phenomenon can be explained by the data generation process. Indeed, the lower α is, the more we obtain columns that will be unbalanced (having an extremely over-represented value compared to the others). This imbalance does not uniformly reduce the accuracy rate of all classes. Semantic types containing a large number of different values as well as a large number of possible characters are the most affected (e.g. first names). On the contrary, those with few different values and few different characters are not affected (e.g. genders).
The tables 4, 5, 6 and 7 show the most important features for both models and both classifiers, they are sorted in descending order. Although they represent only a small portion of the total features, the "global statistics" are highly represented (in particular, features related to string length). Using the mean, maximum and minimum string size, it is easy to separate many classes. For example, gender or blood type columns will have shorter strings on average, as well as smaller maximum sizes than phone number or URL column strings. Another important type of feature is the number of characters (or cells) containing numeric or alphabetic values. Indeed, with this information, it is possible to separate classes containing only alphabetical characters, such as names and surnames, from classes containing only numbers, such as postal codes or the insee code. Finally, at the level of the characteristics concerning the characters, we find separation characters such as ' ' ',' or '-'. It is likely that the ' ' character is used to separate classes containing several words like university names from classes containing only one word like continent names. The '-' character is probably used to distinguish several types of dates from each other.

B. *Experiments and results with robust and extended features*

In this section, the tests focus on the robust and extended features using the same metodology as before (the results presented are only for the high version). However, during the generation of the synthetic test data, we generated an alternative test dataset. In this alternative dataset, 10% of the columns are polluted. The pollution rate of the polluted columns is randomly chosen between 1 and 3% (uniformly).

α	RF H	Catb H	Rf L	Catb L
10^{-7}	0.8032	**0.8217**	0.8169	**0.8306**
10^{-6}	0.8379	**0.8689**	0.8521	**0.8306**
10^{-5}	0.8515	**0.8978**	0.8620	**0.8696**
10^{-4}	0.8800	**0.9211**	**0.9024**	0.8996
10^{-3}	0.9585	**0.9762**	0.9448	**0.9593**
10^{-2}	0.9397	**0.9684**	0.9466	**0.9607**
10^{-1}	0.9969	**0.9989**	0.9683	**0.9759**
1	0.9996	**1**	0.9716	**0.9806**
10	**1**	**1**	0.9755	**0.9810**
100	**0.9996**	**0.9996**	0.9775	**0.9830**

Table 3: F1-score 'macro' of the models (H stands for high and L stands for low).

Catboost "high"
Mean value length
Mean number of '-'
Mean number of 'c'
Max value length
Mean number of ','
Mean number of 'f'
Mean number of 'm'
Min value length
Fraction of cells with num chars
Mean number of '0'

Table 4: Top 10 features for catboost models, ranked in descending order by their gini impurity score part a.

Catboost "low"
Mean value length
Mean number of '-'
Min value length
Mean of alphabetic chars in cells
Mean number of ','
Mean number of 'c'
Max value length
Mean number of 'm'
Mean number of 't'
Fraction of cells with num chars

Table 5: Top 10 features for catboost models, ranked in descending order by their gini impurity score part b.

RF "high"
Mean value length
Max value length
Mean of alphabetic chars in cells
Median value length
Min value length
Mean of numeric chars in cells
Mean of words in cells
Mean number of ' '
Mean number of '-'
Mean number of 'e'

Table 6: Top 10 features for RF models, ranked in descending order by their gini impurity score part a.

RF "low"
Mean value length
Mean of alphabetic chars in cells
Max value length
Median value length
Min value length
Mean of numeric chars in cells
Mean number of 'm'
Mean number of ' '
Mean number of 'e'
Mean number of '-'

Table 7: Top 10 features for RF models, ranked in descending order by their gini impurity score part b.

The pollution is of two types, either the creation of a new element in the column containing a random character repeated p times (p chosen randomly between 0 and 100 according to a uniform distribution), or the creation of a new element containing p randomly picked characters (p chosen randomly between 0 and 100 following a uniform law).

Tables 8, 9, 10, 11, 14, 15, 12, 13 as well as figures 6 and 7 summarize all the results of experiments using robust and or extended features. The F1-score is used to evaluate the performance of the models.

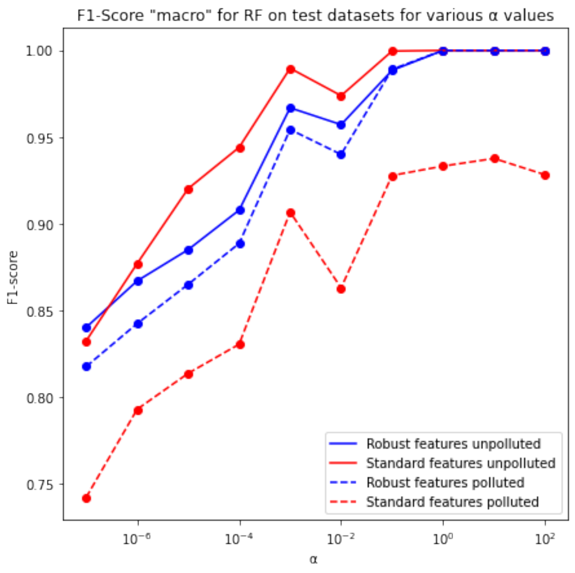

Figure. 6: F1-Score "macro" for RF on test datasets for various α values

The results vary depending on the model used. First with the RF on unpolluted data : in this configuration the only notable difference is that the robust features over-perform when the columns are very unbalanced and under-perform when the columns are balanced compared to the standard features. This result is to be expected, as robust features have the disadvantage of being less efficient than non-robust statistics on noiseless data.

The addition of features from the bigram counting has

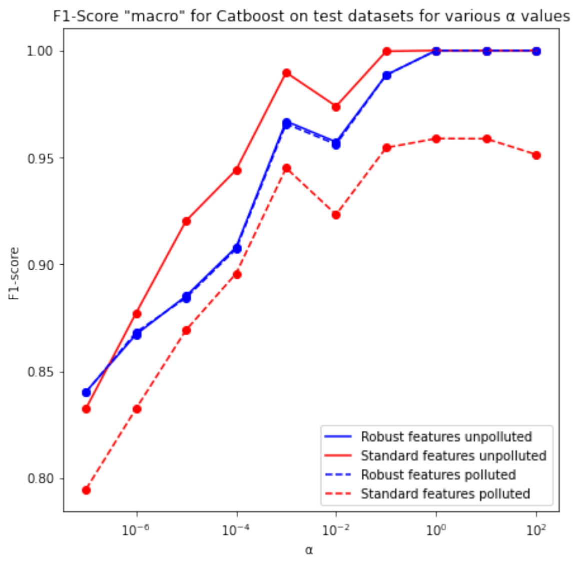

Figure. 7: F1-Score ”macro” for Catboost on test datasets for various α values

α	RF robust	RF robust extend
10^{-7}	0.8176	**0.8344**
10^{-6}	0.8414	0.8463
10^{-5}	0.8651	0.8659
10^{-4}	0.8899	0.8885
10^{-3}	0.9557	0.9598
10^{-2}	0.9399	**0.9426**
10^{-1}	0.9891	0.9895
1	1	1
10	1	1
100	1	1
mean	0.8998	0.9038

Table 8: F1-score ’macro’ of the RF part a.

α	RF	RF extend
10^{-7}	0.7873	0.8227
10^{-6}	0.8489	**0.8706**
10^{-5}	0.8727	**0.8965**
10^{-4}	0.8916	**0.9038**
10^{-3}	0.9666	**0.9673**
10^{-2}	0.9299	0.9234
10^{-1}	**0.994**	0.9933
1	1	1
10	1	1
100	1	1
mean	0.8987	**0.9377**

Table 9: F1-score ’macro’ of the RF part b.

little effect on the robust features, but improves the results obtained with the standard features, the results are particularly improved for low alpha values. However, the features depending on the bigram counts have no real importance in the decision making of the model. It seems therefore that the performance improvement is due to the increase of the number of features that can be used at each split [25].
When the data are polluted, we observe a drastic decrease of the F1 score of the models using the standard features.

α	RF robust	Rf robust extend
10^{-7}	0.8177	**0.8346**
10^{-6}	0.8425	**0.8463**
10^{-5}	0.8651	**0.8669**
10^{-4}	**0.8887**	0.8885
10^{-3}	0.9544	**0.9581**
10^{-2}	0.94	**0.9427**
10^{-1}	0.9891	**0.9895**
1	1	1
10	1	1
100	1	1
mean	0.8996	**0.9038**

Table 10: F1-score ’macro’ of RF classifier on polluted data part a.

α	RF	RF extend
10^{-7}	0.7423	0.7722
10^{-6}	0.7931	0.8154
10^{-5}	0.8138	0.8341
10^{-4}	0.8306	0.8405
10^{-3}	0.9068	0.9074
10^{-2}	0.8631	0.8566
10^{-1}	0.9279	0.9266
1	0.9333	0.9325
10	0.9377	0.9362
100	0.9283	0.9284
mean	0.8396	0.8504

Table 11: F1-score ’macro’ of RF classifier on polluted data part b.

α	Catb robust	Catb robust extend
10^{-7}	0.8403	**0.8434**
10^{-6}	0.8671	0.8678
10^{-5}	0.8851	0.8868
10^{-4}	0.9079	0.9092
10^{-3}	0.9669	0.96773
10^{-2}	0.9572	0.96
10^{-1}	0.9885	0.9896
1	1	1
10	1	1
100	1	1
mean	0.9413	0.94245

Table 12: F1-score ’macro’ of Catboost classifier part a.

Indeed, to perform the classification, the RF relies a lot on statistics based on the length of the strings. But the addition of a single outlier in the column leads to a large modification of the values of some of these statistics. Models using simple and extended robust features are not affected by data pollution.

When the classifier used is Catboost, if the data are not polluted, we see a slight advantage for standard features compared to robust features. We can thus confirm that there is a real cost (loss of f1-score on clean data) associated to the use of these features which is not dependent on the classifier. We can also note that for this classifier the use of features from bigrams has no impact and should be avoided.

α	Catb	Catb extend
10^{-7}	0.8325	0.8347
10^{-6}	0.8771	**0.8788**
10^{-5}	**0.9203**	0.9178
10^{-4}	**0.9441**	0.9424
10^{-3}	**0.9897**	0.9873
10^{-2}	0.9739	0.9703
10^{-1}	0.9996	0.9982
1	1	1
10	1	1
100	1	1
mean	0.95372	0.95295

Table 13: F1-score 'macro' of Catboost classifier part b.

α	Catb robust	Catb robust extend
10^{-7}	0.8401	**0.8431**
10^{-6}	**0.8681**	0.8675
10^{-5}	0.8843	**0.8854**
10^{-4}	0.9071	**0.9084**
10^{-3}	0.9657	**0.9663**
10^{-2}	0.9561	**0.958**
10^{-1}	0.9885	**0.9896**
1	1	1
10	1	1
100	1	1
mean	0.9409	**0.9418**

Table 14: F1-score 'macro' of Catboost classifier on polluted data part a.

α	Catb	Catb extend
10^{-7}	0.7947	0.7999
10^{-6}	0.8324	0.8363
10^{-5}	0.8692	0.8681
10^{-4}	0.8956	0.8962
10^{-3}	0.945	0.9448
10^{-2}	0.9232	0.9219
10^{-1}	0.9545	0.9553
1	0.9588	0.9608
10	0.9587	0.9625
100	0.9513	0.9557
mean	0.9083	0.9101

Table 15: F1-score 'macro' of Catboost classifier on polluted data part b.

When the data are polluted, we observe, as before, a decrease in results for the models using the standard features. In the same way, when the features used are robust, no decrease in performance is observed. We can thus conclude that the robust features allow to make the models resistant to some types of outliers.

C. Experiments and results in the open set condition

In the following experiment, we tried to address an aspect often neglected in the state of the art. The fact that the semantic type detection problem is in most cases an open set problem [26, 27]. Sherlock does not directly address this problem but proposes to reject the 10% of examples with the lowest exit probabilities in order to improve the performance of the model [7].

We will reuse the characteristics (standard) and the method described in the first experiment, adding to our datasets new unknown examples to reject. These new examples to reject come from two semantic types "Cheese name" and "Association name". For each of these two types, 1000 examples are generated following the same process as for the test data. The class predicted by our models is the one with the highest output probability. Since our models are calibrated, we will consider these exit probabilities as a measure of the model's confidence in its prediction. By using a minimum confidence threshold to make a prediction, we can reject examples.

In this experiment, we will consider in each case that we have two data sets. One containing the data to be rejected (containing the unknown classes) and the other the test data (containing the known classes) to be classified. We will measure the accuracy rate of the classifiers on each of the two datasets by varying the rejection threshold. Thus, we will consider that a member of a known class that is rejected is an error, and that an unknown element predicted as belonging to a known class is an error. The results of this experiment are presented in Tables 16, 17, 18 and Figures 8 and 9.

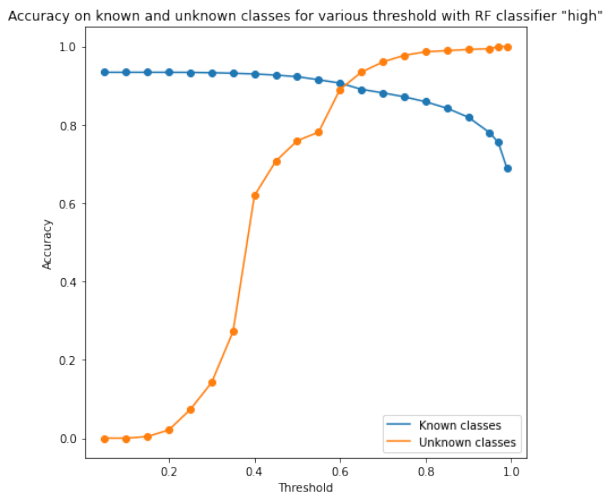

Figure. 8: Accuracy rate of known and unknown classes for different thresholds with the 'high' RF classifier.

threshold	0.35	0.4	0.45	0.5	0.55
Catb know	0.95	0.95	0.95	0.94	0.94
Catb unknow	0.14	0.18	0.21	0.28	0.34
RF know	0.93	0.93	0.93	0.92	0.91
RF unknow	0.27	0.62	0.70	0.76	0.78

Table 16: Accuracy rate of known and unknown classes for different thresholds with RF and Catboost classifiers classifier part 1

We have presented the results only for the "high" version of the algorithm for both classifiers, the results being similar for the "low" versions.

We can see that the method works and that the threshold needed to reject the majority of the unknown elements is lower for the random tree forest than with Catboost despite

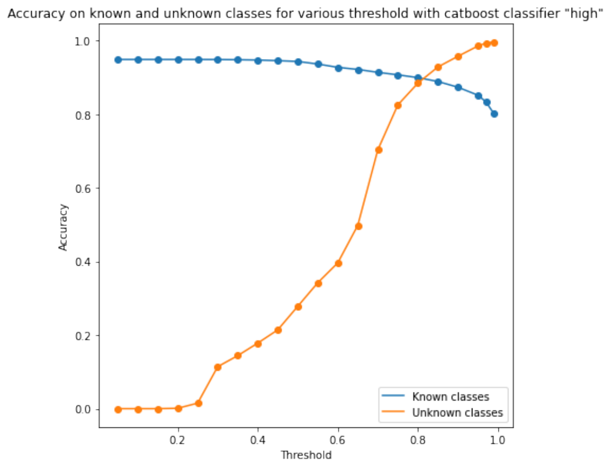

Figure. 9: Accuracy of recognition of known and unknown classes for different thresholds with the Catboost 'high' classifier

threshold	0.6	0.65	0.7	0.75	0.8
Catb know	0.93	0.92	0.91	0.91	0.90
Catb unknow	0.40	0.5	0.70	0.83	0.88
RF know	0.91	0.89	0.88	**0.87**	0.86
RF unknow	0.89	0.93	0.96	**0.98**	0.98

Table 17: Accuracy rate of known and unknown classes for different thresholds with RF and Catboost classifiers part 2

threshold	0.85	0.9	0.95	0.97
Catb know	0.89	**0.87**	0.85	0.83
Catb unknow	0.93	**0.96**	0.98	1
RF know	0.84	0.82	0.78	0.75
RF unknow	0.99	0.99	0.99	1

Table 18: Accuracy rate of known and unknown classes for different thresholds with RF and Catboost classifiers part 3

the calibration. It is interesting to note that depending on the objective we set, the classifier to use is not the same. Indeed, if we want to find the best compromise between the accuracy on the unknown classes and the known classes, we have to choose the RF with a threshold of 0.75, which allows to reject 98% of the unknown individuals while keeping 87% of accuracy. On the other hand, if we want to reject the maximum number of unknown individuals, we have to choose Catboost with a threshold of 0.97. Finally, it should be noted that the rejection performance depends on the data to be rejected; the more similar they are to the learned classes the more difficult it is to reject them efficiently. A possible improvement is to define a threshold for each class and not a global threshold for all the classes [28].

D. Experiments and results on real data

Our next experiment aims at testing our model with new data and evaluating its performance against another algorithm. The new data consist of 32 semantic types belonging to the 57 types we used in training. These data come from other real data sources than those used for training and were not generated with our method.

For each semantic type we have 20 individuals for a total of 640 columns each containing several hundred rows. The algorithm we use for the comparison is an algorithm using dictionaries and regular expressions.
Thus, for the semantic types where it is possible, a human expert has defined the regular expressions. For semantic types that cannot be defined using a regular expression (e.g., first and last names), dictionaries were defined using the same data that was used to generate the training data for our machine learning models. The search using the dictionaries is first performed in an exact manner and in case of failure in an approximate manner using the Jaro-Winkler [29, 30] distance. To make a prediction, the algorithm associates a class to each element of the column. This class is determined by searching first among the regular expressions, then in case of failure in the dictionaries. The final predicted class for the column is the one that is in the majority among the elements of the column. On this new dataset, the accuracy rate for RF is 0.97 and 0.98 for Catboost versus 0.91 for the model using regular expressions and dictionaries. The method using machine learning shows a perfect score for semantic types only composed of alphabetic characters but is in trouble with numeric types, in particular with phone numbers which are confused with currencies. Indeed, the features used are not perfectly adapted to numeric type [11]. Moreover, we were able to confirm the efficiency of the method on an internal database of the company Synaltic containing noisy data. We tested the model using a RF on two tables that contained 41 and 52 columns with each more than one million rows. The accuracy rate was respectively 0.93 and 0.88.

IV. Conclusion

In this study, we introduced the topic of automatic annotation of semantic types. Semantic types detection is a critical task because they are used by many other algorithms. One obstacle to the industrial use of machine learning algorithms to perform this task is the need for too much data to train the algorithms. In order to facilitate the access to this type of algorithms as well as the use of custom semantic types (not being in the most common datasets), we have proposed a method to generate training data using only a small number of real examples. In addition, we proposed a simplification of the extracted features to make them more versatile without compromising the efficiency of the method. As real data often contain outliers we have introduced an alternative feature set that is robust to certain types of outliers. Since the problem of automatic semantic type detection can be an "open set" problem, we have introduced a method to handle this scenario.
More research needs to be conducted to improve the accuracy of the method. First of all, the impact of new features specifically defined for the detection of numerical semantic types or using multilingual transformers [31, 32] should be studied. Furthermore, to improve the results in the "open set" context, further studies should be conducted on the use of classifiers specialized for this task [33].

Acknowledgements

I gratefully acknowledge Astrid Balick for her generous support.
Supported by organization Synaltic : https://www.synaltic.fr/

References

[1] Rahm, E. & Bernstein, P. A survey of approaches to automatic schema matching. *The VLDB Journal*. **10**, 334-350 (2001,12)

[2] Kandel, S., Paepcke, A., Hellerstein, J. & Heer, J. Wrangler. *Proceedings Of The SIGCHI Conference On Human Factors In Computing Systems*. (2011,5)

[3] Ilyas, I. & Chu, X. Data Cleaning. (2019,7)

[4] Trabelsi, M., Cao, J. & Heflin, J. SeLaB: Semantic Labeling with BERT. *2021 International Joint Conference On Neural Networks (IJCNN)*. pp. 1-8 (2021)

[5] Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., Chen, C. & Tan, W. Annotating Columns with Pre-Trained Language Models. *Proceedings Of The 2022 International Conference On Management Of Data*. pp. 1493-1503 (2022)

[6] Arik, S. & Pfister, T. TabNet: Attentive Interpretable Tabular Learning. *Proceedings Of The AAAI Conference On Artificial Intelligence*. **35**, 6679-6687 (2021,5)

[7] Hulsebos, M., Hu, K., Bakker, M., Zgraggen, E., Satyanarayan, A., Kraska, T., Demiralp & Hidalgo, C. Sherlock. *Proceedings Of The 25th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining*. (2019,7)

[8] Zhang, D., Hulsebos, M., Suhara, Y., Demiralp, Li, J. & Tan, W. Sato. *Proceedings Of The VLDB Endowment*. **13**, 1835-1848 (2020,8)

[9] Hulsebos, M., Gathani, S., Gale, J., Dillig, I., Groth, P. & Demiralp Making Table Understanding Work in Practice. *ArXiv*. **abs/2109.05173** (2022)

[10] Hu, K., Gaikwad, S., Hulsebos, M., Bakker, M., Zgraggen, E., Hidalgo, C., Kraska, T., Li, G., Satyanarayan, A. & Demiralp VizNet. *Proceedings Of The 2019 CHI Conference On Human Factors In Computing Systems*. (2019,5)

[11] Khurana, U. and Galhotra, S. Semantic Concept Annotation for Tabular Data. *Proceedings Of The 30th ACM International Conference On Information and Knowledge Management*. (2021,10)

[12] Ng, K., Tian, G. & Tang, M. Dirichlet and Related Distributions. (Wiley,2011,4)

[13] Wang, X., Wang, S., Liang, Y. & Lei, Z. Decisive Vector Guided Column Annotation. *SSRN Electronic Journal*. (2023)

[14] Huber, P. & Ronchetti, E. Robust Statistics. (Wiley-Blackwell,2009,1)

[15] Doyle, J. & Catherine Huirong Chen On the efficiency of the Trimean and Q123. *Journal Of Statistics And Management Systems*. **12**, 319-323 (2009)

[16] Wilcox, R. Comparing the Biweight Midvariances of Two Independent Groups. *The Statistician*. **42**, 29 (1993)

[17] Breiman, L. Random Forests. *Machine Learning*. **45**, 5-32 (2001,10)

[18] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. & Gulin, A. CatBoost: Unbiased Boosting with Categorical Features. *Proceedings Of The 32nd International Conference On Neural Information Processing Systems*. pp. 6639-6649 (2018)

[19] Chevallier, M., Boufares, F., Grozavu, N., Rogovschi, N. & Clairmont, C. Near duplicate column identification: a machine learning approach. *2021 IEEE Symposium Series On Computational Intelligence (SSCI)*. (2021,12)

[20] Chevallier, M., Rogovschi, N., Boufarès, F., Grozavu, N. & Clairmont, C. Detecting Near Duplicate Dataset. *Proceedings Of The 13th International Conference On Soft Computing And Pattern Recognition (SoCPaR 2021)*. pp. 394-403 (2022)

[21] Chevallier, M., Rogovschi, N., Boufarès, F., Grozavu, N. & Clairmont, C. Detecting Near Duplicate Dataset with Machine Learning. *International Journal Of Computer Information Systems And Industrial Management Applications*. **14**, 374-385 (2022,7)

[22] Chevallier, M., Rogovschi, N., Boufarès, F. & Grozavu, N. Semantic Type Detection in Tabular Data via Machine Learning using semi-synthetic data. *Proceedings Of The 14th International Conference On Soft Computing And Pattern Recognition (SoCPaR 2022)*. (2023)

[23] Chevallier, M. L'Apprentissage artificiel au service du profilage des données. (Université Paris-Nord - Paris XIII,2022,11), Doctoral thesis, https://theses.hal.science/tel-04022895

[24] Niculescu-Mizil, A. & Caruana, R. Predicting good probabilities with supervised learning. *Proceedings Of The 22nd International Conference On Machine Learning - ICML '05*. (2005)

[25] Max Kuhn, K. Feature Engineering and Selection: A Practical Approach for Predictive Models. (Chapman,2019)

[26] Bendale, A. & Boult, T. Towards Open World Recognition. *2015 IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*. (2015,6)

[27] Scheirer, W., Rezende Rocha, A., Sapkota, A. & Boult, T. Toward Open Set Recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **35**, 1757-1772 (2013)

[28] Scherreik, M. & Rigling, B. Automatic threshold selection for multi-class open set recognition. *SPIE Proceedings*. (2017,5)

[29] Jaro, M. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal Of The American Statistical Association*. **84**, 414-420 (1989,6)

[30] Winkler, W. The state of record linkage and current research problems, Statistics of Income Division, Internal Revenue Service Publication R99/04. (1999)

[31] Green, T., Ponzetto, S. & Glavaš, G. BabelBERT: Massively Multilingual Transformers Meet a Massively Multilingual Lexical Resource. (arXiv,2022)

[32] Miao, L., Wu, J., Behre, P., Chang, S. & Parthasarathy, S. Multilingual Transformer Language Model for Speech Recognition in Low-resource Languages. (arXiv,2022)

[33] Dhamija, A., Günther, M. & Boult, T. Reducing Network Agnostophobia. *NeurIPS*. pp. 9175-9186 (2018)

Author Biographies

Marc Chevallier born in 1991 in France has completed a Ph.D. in computer science within the LIPN laboratory at the Sorbonne Paris Nord University, France. His main research topics focus on data profiling using machine learning as well as feature selection using genetic algorithms. He currently holds a position as R&D and Innovation Manager at Synaltic while also being an associate member of the LIPN laboratory.

Nicoleta Rogovschi born in 1983 in Moldova received her Master of Computer Science degree from Paris 13 University in 2006 in Machine Learning. She is currently an Associate Professor in Computer Science at the Paris Descartes University. She completed her Ph.D. in Computer Science (Probabilistic Unsupervised Learning) in 2009 in the Computer Science Laboratory of Paris 13 University and the HdR (Hability to direct researches) in 2021 at Sorbonne Paris Nord University. She's research is with the Data Mining (GFD) Team. Her research interests include Probabilistic Learning, Unsupervised Learning, Clustering and Co-Clustering methods for different types of data in different contextes : anonymization, recommender system, opinion detection,... She is also a member of EGC, AFIA, IEEE, INNS, INNS AML group. Nicoleta Rogovschi codirected 5 PhD students and tens of Master Research students.

Faouzi Boufares is assistant professor in computer science at Paris XIII University since 1990. He is a member of the computer science laboratory of the University of Paris Nord. His research work, since his PhD thesis in 1986, has focused on databases. The main topics recently have been data profiling and semantics to automatically detect and correct anomalies in data when integrating heterogeneous data. Data quality was the main objective to better support decision making. His research interests are in data engineering and data science. He has directed several theses in the field, which have led to several publications, namely on data quality and deduplication. Recently, in his research team, the principles of machine learning are used to serve data quality. His teachings focused on the analysis and design of information systems, databases and data warehouses, and decision support systems.

Nistor Grozavu born in 1984 in Moldova received his HdR degree from Sorbonne Paris Nord University in 2020 and PhD degree in Unsupervised Machine Learning in 2009 from Paris 13 University. He is currently Full Professor in Computer Science at CY Cergy Paris University. His research is with the MIDI team from ETIS Laboratory. His research interests include Unsupervised Learning, Transfer Learning, Dimensionality reduction, Collaborative Learning, Machine Learning by Matrix Factorization and content-based information retrieval, Quantum Machine Learning. These researches are applied in different applications for text mining, visual information retrieval, recommendation systems, fraud detection, etc. via ANR, FUI, PEPS CNRS, AUF projects. He is also a member of IEEE, INNS, and the co-founder of the INNS Autonomous Machine Learning group. Nistor is the co-author of a patent on visual information retrieval, published two book chapters, 12 peer-reviewed journal papers, and more than 40 international conference papers. Nistor Grozavu co-supervised 2 post-doctorants, 8 PhD students and supervise each year 1-2 Master2 internship.