A Knowledge-Based Platform for the Classification of Accounting Documents*

Alessia Amelio[0000−0002−3568−636X], Alberto Falcone[0000−0002−2660−1432], Angelo Furfaro[0000−0003−2537−8918], Alfredo Garro[0000−0003−0351−0869], Domenico Saccà[0000−0003−3584−5372]

DIMES - University of Calabria
87036 – Rende (CS), Italy
{a.amelio, a.falcone, a.furfaro, a.garro, sacca}@dimes.unical.it

Abstract. Due to the lack of standardization, the task consisting in the classification of invoices' billing entries for accounting purposes has yet not been made fully automatic despite the diffusion of e-invoicing systems. Each accounting firm adopts its own conventions also on the basis of the specific business domain of the invoice owner. Here, we describe a knowledge-based software platform devised to adequately address this issue. The paper focuses on the evaluation of the most appropriate classification algorithms which are suitable to be employed in an adaptive meta-learning approach. The effectiveness of the selected algorithms is experimentally assessed by running them on a real dataset of invoice entries.

Keywords: Automatic Invoice Processing · Meta-Learning · Classification.

1 Introduction

Over the years, business processes and software systems supporting accounting have reached a high level of maturity and, at the same time, have allowed the producers of such systems to achieve an excellent and consolidated level of diffusion in the market. The ever increasing pervasiveness of new ICT technologies, mainly due to the diffusion of mobile devices and "smart" and Internet-enabled objects, has determined, in all application domains, a strong need for innovation for IT products and services, which in turn, has contributed to the increase in competitiveness among companies. Also in the specific domain of systems and

* Supported by project P.O.R. CALABRIA FESR-FSE 2014-2020 - "Smart Electronic Invoices Accounting – SELINA" (CUP J28C1700016006)

services supporting the processing of accounting documents, the demand for innovation has determined the definition of effective strategies to evolve the offer in order to maintain the customer portfolio and potentially increase it.

In this context, one of the aspects most susceptible of innovation is the classification of invoices' billing entries, a process which is not fully automated. One of the main issues is the lack of a standard for the categorization and association of the information related to the invoices entries in the corresponding accounting categories. Often, each accounting firm adopts its own conventions by modifying and adapting them to its client portfolio, to its internal management processes and to other contingent needs.

Most of the scientific works related to the processing of accounting documents has been devoted to the extraction of information from scanned documents, e.g. in [6] a flexible form-reader system has been devised for such purpose. The work described in [4] focuses on the feature extraction process for the supervised classification of scanned invoices obtained from printed documents. An approach based on incremental learning techniques, where some experiments were carried out by applying it to the classification of a small data set of invoices (a learning set of 324 documents, a testing set of 169 documents and 8 classes), has been proposed in [10].

This paper proposes a knowledge-based software platform devised to address the issue of automatic classification of invoices entries which is able to tackle the lack of standardization by allowing the building of specific classification models tailored on the peculiarities/preferences of a given accounting firm. The most appropriate classification algorithms suitable to be employed in such a domain are evaluated and their effectiveness is experimentally assessed by running them on a dataset of 910874 invoice entries. The architecture has been devised to support a meta-learning approach consisting in the integration of more classification algorithms in order to achieve better prediction performances.

The rest of the paper is organized as follows. Section 2 presents the architecture of the proposed classification platform. Section 3 describes the adopted methodology and the dataset used for its validation. Section 4 illustrates the experimental setting whose results are shown in Section 5. Finally, Section 6 draws the conclusions.

2 Platform Architecture

Figure 1 shows the architecture of the proposed platform. It is characterized by a set of modules, each of which is in charge of specific functionalities supporting the classification process of invoices entries.

The *Data Pre-Processing* module aims at performing from the raw input data: (i) cleaning, (ii) numerical feature extraction, and (iii) normalisation. The clean data are used to feed both the *Classification* and *Knowledge update* modules. The first module is composed of a set of classification models, each specifically defined to take into account the details of a given user profile along with related product information (see Section 3) during the classification process (the

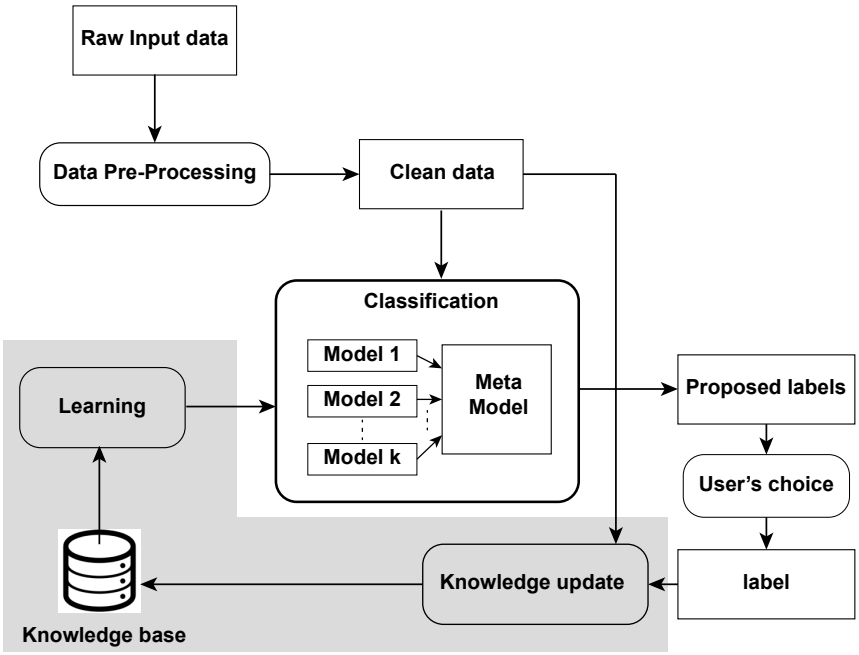

Fig. 1. The architecture of the platform

predicted labels depend on both these information). The second module allows, starting from the *Clean data* and *User's choice*, to update the *Knowledge base* that is exploited by the *Learning* module to re-train the classification models according to the user profiles.

The labels are predicted by the *Classification* module through the *Meta Model* that combines the contributions of each classification model. The *Classification* module produces a ranked list of labels which are fed to the end-user that, in turn, either selects one of them or specifies a new one. In both cases, the resulting labels along with the input data are used by the *Knowledge update* module to update the *Knowledge base*. This feedback allows both to increase the prediction performance of the classification process and to adapt it to *concept drift* issues [15].

3 Data and Methodology

The original dataset consists of a sample of 910874 anonymous invoices' billing entries with 92 numerical and string attributes related to products sold in the period 2016-2018. Specifically, the first 8 are related to the kind of invoice's billing entry along with its occurring date; the following 23 attributes concern the product supplier; the following 2 ones affect the details of the payment;

the next 20 attributes concern the recipient of the goods and related transport details. The latest 39 attributes are related to the product details, i.e. weight, description and the user (invoice owner) information. The group of the invoice entry, which has been defined by the dataset owner, represents the class attribute, for a total of 71 groups.

The dataset keeps information for the following 7 user profiles: (i) *Driver*, (ii) *Change of heading*, (iii) *Direct agent*, (iv) *Fair event*, (v) *Supplier*, (vi) *Internet* and (vii) *Office*.

Starting from the original dataset, the following 2 string attributes have been considered for the analysis: (i) *business activity type* (e.g. "cafe" and "hotel"), and (ii) *description of invoices' billing entry*.

3.1 Preprocessing

The dataset has several missing, invalid and inaccurate values, particularly in correspondence with the product information (e.g. negative weight values, non-homogeneous descriptions and units of measure). To address this issues, it was necessary to perform data cleaning and preparation procedures through the *Data Pre-Processing* module in order to make this data usable for the subsequent classification steps [11].

The data pre-processing task consists of three parts: (i) *data cleaning*, (ii) *feature extraction*, and (iii) *normalization*.

Data Cleaning. The defined *data cleaning* process, which aims to eliminate inaccurate and corrupt data, is composed of three steps which are performed on the 2 selected string attributes:

1. Remove duplicate, redundant and invalid text;
2. Remove useless words and sentences for classification purposes (e.g., "transport cost" or "closed on Tuesday").
3. Identification through regular expressions of invalid text patterns, and further deletion of numbers, symbols, special characters and units of measurement to avoid unexpected behaviours during the classification phases (e.g. overfitting, underfitting).

Feature Extraction. After *data cleaning*, for each entry, a feature vector of 10 numerical values is built from the 2 string attribute values. Specifically, the *business activity type* is characterised by a single word, while the *description of invoices' billing entry* is converted to a sequence of words. All distinct words in the dataset form a vocabulary which associates a unique integer value to each word.

The first element of the feature vector is the integer value in the vocabulary connected with the *business activity type*'s word. The other elements of the feature vector are the integer values in the vocabulary associated with the

description of invoices' billing entry's words. Accordingly, the size of the feature vector is equal to 1 plus the size of the longest possible sequence of words composing a *description of invoices' billing entry* in the dataset, which is equal to 9.

Due to the predefined format of the *description of invoices' billing entry* in the dataset, the feature vectors are consistent and comparable. The whole procedure has been performed through the *Tokenizer* utility class provided by the Keras library [12].

Normalisation. In each feature vector, missing numerical values for a given feature are replaced with the mean of the distribution of the remaining ones for that feature. Then, the feature values are bounded to be included in the range $[0, 1]$ using the following *min-max* procedure:

$$\hat{x_i}^j = \frac{{x_i}^j - min_j}{max_j - min_j}, \tag{1}$$

where ${x_i}^j$ is the value of j^{th} feature of feature vector x_i, and min_j and max_j are the minimum and maximum values of j^{th} feature, respectively.

This step is essential for distance-based classifiers to standardize the range of the numerical features so that each feature contributes approximately proportionately to the final distance.

3.2 Classification Models

Although different algorithms have been employed and tested to define the set of classification models composing the *Classification* module, a subset of them were selected which obtained the best performance on the feature vectors related to the different user profiles. Specifically, they are:

- Bayes Net,
- Decision Tree,
- Random Forest,
- K-Nearest Neighbour,
- Deep Neural Network,
- Repeated Incremental Pruning to Produce Error Reduction.

3.3 Bayes Net

A Bayes Net (BN) is a probabilistic graphic model that represents a set of stochastic variables with their conditional dependencies through the use of a Direct Acyclic Graph (DAG). In the DAG, each node represents a feature of the dataset, whereas the conditional dependencies among them are represented as edges [14]. The BN model can be used to classify an entry a of the dataset; specifically, a is predicted to belong to the class c whether c maximizes the posterior probability $p(c|a)$.

3.4 Decision Tree

A Decision Tree (DT) is a classifier based on a tree structure that is learned from a training dataset. In the tree, leaf nodes represent the class labels, whereas the no-leaf nodes correspond to the feature decision points which spit an entry a of the dataset according to its feature values. After evaluating all the decision points, the entry a is predicted to belong to the class c, which is the leaf node along the path [3].

3.5 Random Forest

A Random Forest (RF) generates a set of tree-based classifiers, each of which is built starting from a random subset of the training dataset. A tree-based classifier can be a decision tree as previously described. The classification of an entry a is done by combining the contributions of each tree-based classifier through a majority voting strategy [3].

3.6 K-Nearest Neighbour

A K-Nearest Neighbour (KNN) is a classifier that aims at finding the *K-Nearest* entries of the training dataset from a test entry a based on a given distance function. At the end, a is predicted to belong to the class c, which represents the most frequent class in the selected *K-Nearest* entries [2].

3.7 Deep Neural Network

A Deep Neural Network (DNN) can be defined as a weighted oriented graph $G =< V, E >$ where nodes represent the artificial neurons $V(G) = \{v_1, v_2, ..., v_n\}$, whereas the weighted edges $E(G) = \{e_1, e_2, ..., e_m\} \subseteq V \times V$ represent the relationships between neurons. A DNN is composed of a layer of input neurons, one or more intermediate layers of neurons and a layer of output neurons that provides the classification result. The model learns a non linear-transformation by tuning the layer weights so as to associate the input entry to another space where it becomes linearly separable [2].

3.8 Repeated Incremental Pruning to Produce Error Reduction

Repeated Incremental Pruning to Produce Error Reduction (*RIPPER*) is a rule-based classifier which learns rules by considering all the entries of a given judgment in the training dataset as a class label, and detecting a set of rules that include all the entries belonging to that class. After that, the algorithm continues considering the next class label and does the same steps, iterating until all class labels have been considered [7].

4 Experiments

The classification task consists in predicting the group of invoices' billing entries related to user profiles from the feature vectors in the dataset.

The experiments involving *BN, DT, RF, KNN* and *RIPPER* have been performed in Weka [1], whereas the *DNN* has been implemented in Python through well-known deep learning libraries, i.e. Pandas, Sklearn, Keras with Tensorflows as backend.

For *KNN*, the experimented values for K ranged from 1 to 8. In the end, K was set to 1 since it provided the best results on the given dataset. Concerning the other parameters, they were set to the default Weka values.

The *DNN* has a 1D tensor 10-dimensional input vector (corresponding to the size of the feature vector), an *Embedding* layer that allows to represent words with their meanings, a *Flatten* layer that transforms the output coming from the *Embedding* so as to be exploited by the following *Dense* hidden layer. Also, a *BatchNormalization* layer has been introduced to normalize the activation function of neurons, followed by a *Dropout* layer, which randomly deactivates neurons to prevent overfitting issues [13]. Finally, a *Dense* output layer composed of 71 neurons (corresponding to the number of groups of invoices' billing entries) provides the predicted class. Figure 2 shows the *DNN* architecture.

In order to evaluate the performance of the classifiers, *accuracy*, *error* and *kappa statistics* have been computed from the multiclass confusion matrix. The *accuracy* represents the percentage of correctly classified entries; the *error* quantifies the percentage of incorrectly classified entries; finally, the *kappa statistics* is based on a comparison between the observed and expected accuracy [11].

To make the evaluation independent from the choice of training and test datasets, a k-fold cross validation ($k = 10$) has been used, which is a valuable literature technique to achieve statistically significant results [11].

5 Results

Table 1 reports the classification results obtained by *BN, DT, RF, KNN, RIPPLE* and *DNN* in terms of *accuracy*, *error* and *kappa statistics* for one randomly selected test dataset related to the *Supplier* user profile.

It is worth noting that all the presented classification models obtained an *accuracy* in predicting the group of invoices' entries exceeding 95%. Specifically, *DNN* obtains the best result of 99.20% for *accuracy*, 0.80% for *error* and 0.99 for *kappa statistics*. Hence, it can be definitively adopted as classification model for the *Supplier* user profile. This is followed by *RF*, *KNN*, *DT*, *RIPPLE* and *BN*.

Concerning the best classification method, *DNN*, Figure 3 shows the trend of *accuracy* and *error* over the epochs on training and test datasets.

As depicted in Figure 3, the *error* decreases at each epoch, whereas the *accuracy* increases at each epoch. This is what it is expected using an optimization function based on the descending gradient, which tries to minimize the *error*

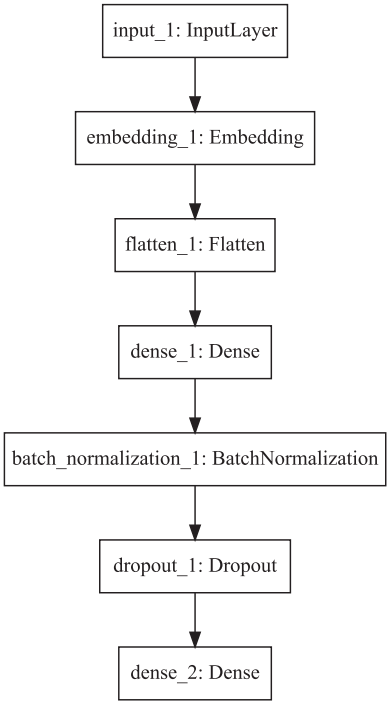

Fig. 2. The architecture of the adopted *DNN*.

Table 1. Classification results for the user profile *Supplier*.

Classification model	**Accuracy (%)**	**Error (%)**	**Kappa statistics**
BN	95.64	4.35	0.95
DT	96.82	3.17	0.96
RF	98.02	1.98	0.98
KNN	97.60	2.40	0.97
RIPPLE	96.58	3.42	0.96
DNN	99.20	0.80	0.99

value at each epoch [13]. An opposite trend can be observed for *accuracy* where its value increases over the epochs. It worth noting that the defined *DNN* is able to learn the key patterns characterizing the dataset already from the first epochs; it is visible from the trend of the *accuracy* and *error* obtained on the dataset which rapidly increases and decreases, respectively, and then does not change significantly over the epochs.

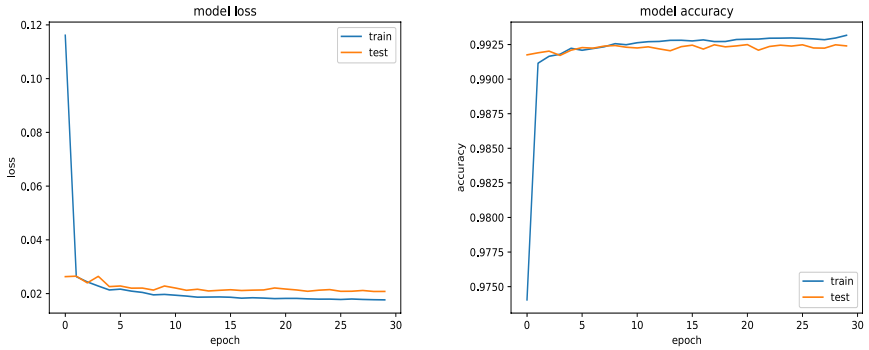

Fig. 3. Accuracy and error obtained by *DNN* on invoices' entries related to the *Supplier* user profile.

6 Conclusions and Future Works

This paper presented the architecture of a platform for the automatic classification of invoice entries for accounting purposes and the evaluation of the performance of the classification algorithms that can be suitably employed in such a domain. The architecture has been designed to enable a meta-learning approach where issues such as adaptation to invoice owner profiles and concept drift can be effectively handled. Toward this goal, the future work will involve the analysis of ensemble learning techniques [15, 16] based on new boosting and bagging strategies to combine the classification results in the *Meta Model*, different feature representations for modelling the invoice entries, and the active learning approach proposed. Another potential future research direction concerns the extension of the architecture in order for it to work in distributed environments e.g. through the use of Model-Driven approaches [5] and its evaluation by means of virtual environments [8, 9].

References

1. Weka 3: Data Mining Software in Java, `https://www.cs.waikato.ac.nz/ml/weka/index.html`
2. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician **46**(3), 175–185 (1992)
3. Amelio, L., Amelio, A.: Classification methods in image analysis with a special focus on medical analytics. In: Machine Learning Paradigms, pp. 31–69. Springer (2019)
4. Bartoli, A., Davanzo, G., Medvet, E., Sorio, E.: Improving features extraction for supervised invoice classification. In: Artificial Intelligence and Applications. ACTAPRESS (2010). https://doi.org/10.2316/P.2010.674-040
5. Bocciarelli, P., D'Ambrogio, A., Falcone, A., Garro, A., Giglio, A.: A model-driven approach to enable the distributed simulation of complex systems. In: Complex

Systems Design & Management, Proceedings of the Sixth International Conference on Complex Systems Design & Management, CSD&M 2015, Paris, France, November 23-25, 2015, pp. 171–183. Springer International Publishing (oct 2015). https://doi.org/10.1007/978-3-319-26109-6_13
6. Cesarini, F., Gori, M., Marinai, S., Soda, G.: INFORMys: a flexible invoice-like form-reader system. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(7), 730–745 (jul 1998). https://doi.org/10.1109/34.689303
7. Cohen, W.W.: Fast effective rule induction. In: Machine Learning Proceedings 1995, pp. 115–123. Elsevier (1995)
8. Furfaro, A., Argento, L., Parise, A., Piccolo, A.: Using virtual environments for the assessment of cybersecurity issues in IoT scenarios. Simulation Modelling Practice and Theory **73**, 43–54 (apr 2017). https://doi.org/10.1016/j.simpat.2016.09.007
9. Furfaro, A., Piccolo, A., Parise, A., Argento, L., Saccà, D.: A cloud-based platform for the emulation of complex cybersecurity scenarios. Future Generation Computer Systems **89**, 791–803 (dec 2018). https://doi.org/10.1016/j.future.2018.07.025
10. Hamza, H., Belaid, Y., Belaid, A., Chaudhuri, B.B.: Incremental classification of invoice documents. In: Proc. of 19th International Conference on Pattern Recognition. IEEE (dec 2008). https://doi.org/10.1109/icpr.2008.4761832
11. Han, J., Pei, J., Kamber, M.: Data mining: concepts and techniques. Elsevier (2011)
12. Ketkar, N., et al.: Deep Learning with Python. Springer (2017)
13. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436 (2015)
14. Solares, C., Sanz, A.M.: Different bayesian network models in the classification of remote sensing images. In: Intelligent Data Engineering and Automated Learning - IDEAL 2007, 8th International Conference, Birmingham, UK, December 16-19, 2007, Proceedings, pp. 10–16. Springer Berlin Heidelberg (2007). https://doi.org/10.1007/978-3-540-77226-2_2
15. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD'03. ACM Press (2003). https://doi.org/10.1145/956750.956778
16. Zhang, C., Ma, Y.: Ensemble machine learning: methods and applications. Springer (2012)