Deep Learning And Data Mining Classification Through the Intelligent Agent Reasoning

Amine Chemchem, Francois Alin, Michaël Krajecki

Deep Learning And Data Mining Classification Through the Intelligent Agent Reasoning

1st Amine Chemchem
CReSTIC Center, University of Reims Champagne-Ardenne
Campus Moulin de la Housse BP 1039, 51687 Reims cedex 2, France
mohamed-lamine.chemchem@univ-reims.fr

2nd François Alin
CReSTIC Center, University of Reims Champagne-Ardenne
Campus Moulin de la Housse BP 1039, 51687 Reims cedex 2, France
francois.alin@univ-reims.fr

3rd Michael Krajecki
CReSTIC Center, University of Reims Champagne-Ardenne
Campus Moulin de la Housse BP 1039, 51687 Reims cedex 2, France
michael.krajecki@univ-reims.fr

***Abstract*—Over the last few years, machine learning and data mining methods (MLDM) are constantly evolving, in order to accelerate the process of knowledge discovery from data (KDD). Today's challenge is to select only the most relevant knowledge from those extracted. The present paper is directed to these purposes, by developing a new concept of knowledge mining for meta-knowledge extraction, and extending the most popular machine learning methods to extract meta-models. This new concept of knowledge classification is integrated on the cognitive agent architecture, so as to speed-up its inference process. With this new architecture, the agent will be able to select only the actionable rule class, instead of trying to infer its whole rule base exhaustively.**

***Index Terms*—data mining, machine learning, intelligent agent, knowledge mining.**

I. Introduction

Currently, the induction rules have become indivisible pattern of artificial intelligence thanks to their existence as the basis for many disciplines, such as the agent technology, data mining and knowledge discovery. This paper describes how to extend MLDM methods to induction rules.

In this work, we are interested in supervised learning, which is used on many applications, such as patterns recognition and intrusion detection.

The classification of a collection consists of dividing the items that comprise the collection into categories or classes [1]. In the context of data mining, classification is done using a model that is built on historical data. The goal of predictive classification is to accurately predict the target class for each record in new data, i.e. data that are not in the historical set.

A classification will start with the definition of training set, for which target values (or class assignments) are known. Classification algorithms use various techniques to find relations between input data and target values in the training data.

This paper is organized as follows: The next section introduces the concept of knowledge mining. Section III describes the algorithms of induction rules mining. Then, experimental results are shown in section IV, which resumes a comparative study carried out on the MLDM performance approaches for rules classification. After that, the new architecture of the intelligent agent is presented. Finally, we conclude by making some remarks and talking about future perspectives.

II. Knowledge Mining

The web features huge amounts of data from different fields. By applying data mining, and KDD process to these data, many large knowledge bases are extracted. Our study explores the issue to analyze these knowledge bases in order to discover a new hidden pattern.

According to the authors in [2], the different representations of knowledge can be classified into two approaches.

1) **The procedural approach** invokes simplicity and ease of understanding, represented by algorithms simulating real behaviors. In addition, the procedural representation allows to handle problems with algorithmic style, which is fully analyzable and understandable.

2) **The declarative approach** is more flexible because it provides heuristic expressions using statements. The declarative representation allows specify constraints and learns independently from methods of use. The control structure is separated from the knowledge entered as rules on bulk data.

A. Induction Rules Representation

The procedural form is imposed by a limited grammar; therefore, it is more interesting to deal with the most declarative representation. Induction rules are the closest representation form to natural language phrases. In addition, the results of KDD process with some data mining methods, such as decision trees or association rules mining, are declarative knowledge as induction rules, shown on figure 1.

Fig. 1. *Knowledge Discovery Process [3]*

For these reasons, this work is designed for induction rules processing.

An induction rule is a boolean formula of the form:

978-1-5386-7810-7/18/$31.00 ©2018 IEEE
DOI 10.1109/W-FiCloud.2018.00009

$R : X \rightarrow Y$, where X and Y are sets of clauses. X is called the premise part of the rule and Y its consequence [4].
The clause is a comparison between two elements as the form: $a\ operator\ b$; where $(a, b) \subset (A, V)$ i.e A is a set of variables, V is a set of values.

B. Induction Rules Preliminaries

Defining the mathematical preliminaries of induction rules is a necessary step before the mining process. In the previous works [5] [6], we proposed a new similarity measure and new gravity center computation formulas. In this study, the induction rules are considered as text. Therefore, this work summarizes solving the well-known problem of text classification.

Like the document classification problem [7] [8], we define intuitively the induction rules classification (IRC) problem as the task of classifying rules under a predefined category. More formally, if r_i is a rule from the rule base RB and $\{c_1, c_2, ..., c_n\}$ is the set of all the categories, then IRC assigns one category c_j to an induction rule r_i .

As in each supervised machine learning task, a rule may be assigned to more than one category (Ranking Classification), but in this paper only researches on Hard Categorization (assigning a single category to each rule) are taken into consideration.

III. Induction Rules Classification Algorithms

The main contribution of this study is to extend popular MLDM approaches for induction rules classification, in order to extract meta-models.

Afterwards, this set of algorithms is incorporated as a new package through the intelligent agent knowledge base. Finally, the inference engine reasoning of the cognitive agent is improved. It is enough to classify the new knowledge in order to infer only the corresponding rules class, instead of inferring the entire rule based at each arrival of a new acquaintance.

In this part, we show the extension of the most popular text classification approaches for dealing with induction rules.

A. Naive Bayes Classification Approaches

The Naive Bayes classifier is a simple probabilistic approach, which is based on Bayes theorem with strong and naive independence assumptions [9]. It is one of the most basic text classification techniques with various applications, such as email spam detection, personal email sorting, document categorization and sentiments detection. Despite the naïve design and oversimplified assumptions that this technique uses, Naive Bayes performs well in many complex real-world problems.

Even though it is often outperformed by other techniques, such as boosted trees, random forests and support vector machines. Naive Bayes classifier is very efficient since it is less computationally intensive (in both CPU and memory) and it requires a small amount of training data. Moreover, the training time with Naive Bayes is significantly smaller as opposed to alternative methods [10].

There are several Naive Bayes variations. Here we will discuss about two of them: (1) the multinomial naive Bayes and (2) the Bernoulli naive Bayes. Note that each can deliver completely different results since they use different models.

In a text classification problem, we use the words of the document (in our case, the items of clauses attributes/values of the rule) in order to classify it on the appropriate class. By using the maximum a posteriori (MAP) decision rule, we come up with the following classifier:

$$C_{map} = \arg\max_{c \in C}(P(c|d)) = arg\max_{c \in C}[P(c) \prod_{1 \leqslant k \leqslant nd} P(t_k|c)]$$

Where t_k are the tokens (attribute or value) of the rule. C is the set of classes that are used in the classification. $P(c|d)$ is the conditional probability of class c given rule d. $P(c)$ is the prior probability of class c, and $P(t_k|c)$ is the conditional probability of token t_k given class c.

Due to the fact that computers can handle numbers with specific decimal point accuracy, calculating the product of the above probabilities will lead to float point underflow. This means that we will end up with a number so small, that it will not be able to fit in memory and thus it will be rounded to zero, rendering our analysis useless. To avoid this problem, we will maximize the sum of their logarithms instead of maximizing the product of the probabilities.

$$C_{map} = arg\max_{c \in C}[\log P(c) + \sum_{1 \leqslant k \leqslant nd} \log P(t_k|c)]$$

Thus, instead of choosing the class with the highest probability, we choose the one with the highest log score. Given that the logarithm function is monotonic, the decision of MAP remains the same.

The last problem that we address is if a particular attribute/value does not appear in a particular class, then its conditional probability is equal to 0. If we use the first decision method (product of probabilities) the product becomes 0; however, if we use the second (sum of their logarithms) the log(0) is undefined. To avoid this, we will use add-one or Laplace smoothing by adding 1 to each count:

$$P(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'}) + B'}$$

Where B' is equal to the number of terms contained in the vocabulary V.

Now we examine two common Naive Bayes variations which differ on the way that they calculate the conditional probabilities of each feature and on the scoring of each category.

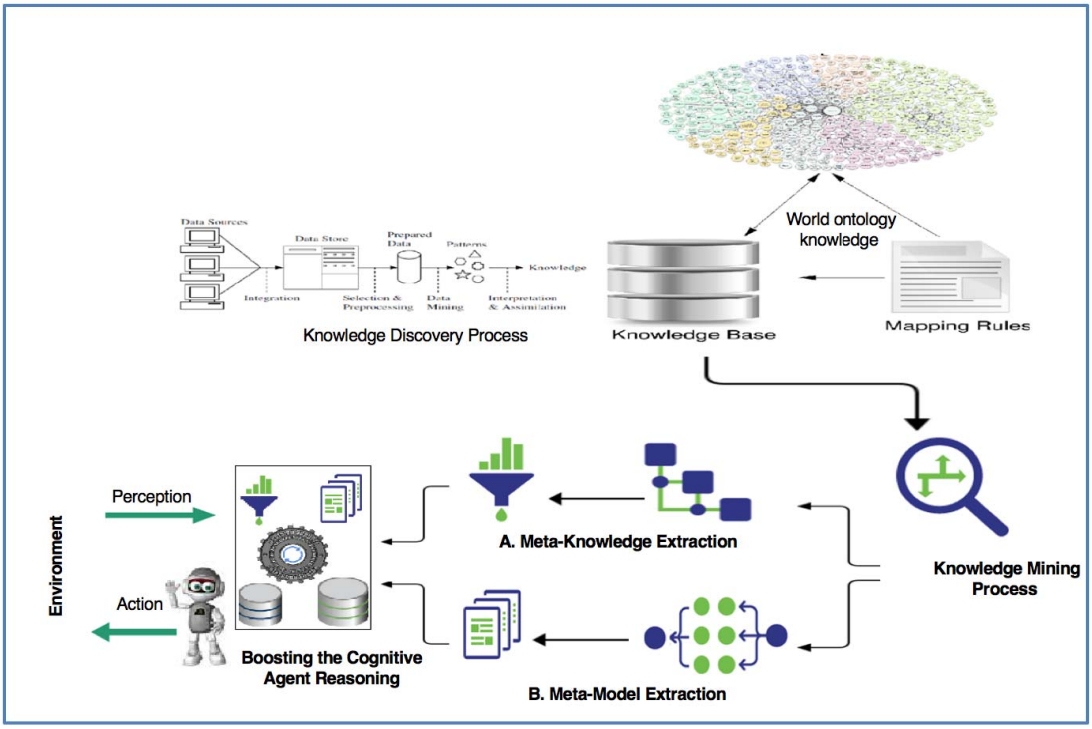

Fig. 2. *The General Framework*

1) Bernoulli Naive Bayes: The Bernoulli variation, as described by [11], generates a boolean indicator about each token of the vocabulary equal to 1 if the token belongs to the examining rule and 0 if it does not. This variation is different from the multinomial model because it takes into account the non-occurring clauses within a rule. While in multinomial variation, only the number of token occurrences is enumerated. Both the training and the testing algorithms are presented in algorithm1.

2) Multinomial Naive Bayes Algorithm : This variation, as explained by [11], estimates the conditional probability of a particular clause (attribute / value) given a class as the relative frequency of term t in rules belonging to class c.

$$p(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

Thus, this variation takes into account the number of occurrences of a term t in training documents from class c, including multiple occurrences. Both the training and the testing algorithms are presented in algorithm2.

Algorithm 1 Bernoulli Naive Bayes Algorithm

Train BernoulliNB(C,D)
1- $V \leftarrow Extract\_Vocabulary(D)$.
2- $N \leftarrow Count\_Docs(D)$.
3- **For each** $(c \in C)$ **do**
4- $N_c \leftarrow Count\_Docs\_In\_Class(D, c)$
5- $prior[c] \leftarrow N_c/N$
6- **For each** $(t \in V)$ **do**
7- $N_{ct} \leftarrow CountDocsInClassContainingTerm(D, c, t)$
8- $condprob[t][c] \leftarrow (N_{ct} + 1)/(N_c + 2)$
9- **return** $V, prior, condprob$
ApplyBernoulliNB$(C, V, prior, condprob, d)$
1- $V_d \leftarrow ExtractTermsFromDoc(V, d)$
2- **For each** $(c \in C)$ **do**
3- $score[c] \leftarrow logprior[c]$
4- **For each** $(t \in V)$ **do**
5- **IF** $(t \in V_d)$ **Then**
6- $score[c] += logcondprob[t][c]$
7- **else**
8- $score[c] += log(1 - condprob[t][c])$
return $argmax_{c \in C} score[c]$

B. Support Vector Machine Approach

Support vector machines (SVM) have exhibited superb performance in binary classification tasks. Intuitively, SVM aims at searching for a hyperplane that separates the two classes of data with the largest margin (the margin is the distance between the hyperplane and the point closest to it) [12] [13].

For example, suppose we are given a vector space representation of n documents (in our case of rules). In the bag-of-words model, each vector d_i has a component for each term feature, which is proportional to its importance (term

Algorithm 2 Multinomial Naive Bayes Algorithm

Train MultinomialNB(C,D)
1- $V \leftarrow Extract\_Vocabulary(D)$.
2- $N \leftarrow Count\_Docs(D)$.
3- **For each** $(c \in C)$ **do**
4- $N_c \leftarrow Count\_Docs\_In\_Class(D, c)$
5- $prior[c] \leftarrow N_c/N$
6- $text_c \leftarrow ConcatenatTextOfAllDocsInClass(D, c)$
7- **For each** $(t \in V)$ **do**
8- $T_{ct} \leftarrow CountTokensOfTerm(text_c, t)$
9- **For each** $(t \in V)$ **do**
10- $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
11- **return** $V, prior, condprob$

ApplyMultinomialNB$(C, V, prior, condprob, d)$
1- $W \leftarrow ExtractTokensFromDoc(V, d)$
2- **For each** $(c \in C)$ **do**
3- $score[c] \leftarrow logprior[c]$
4- **For each** $(t \in W)$ **do**
5- $score[c] += logcondprob[t][c]$
return $argmax_{c \in C} score[c]$

frequency or TFIDF are commonly used). Each rule vector is normalized and associated with one of the two labels, $+1$ or -1. The training data is thus $\{(d_j, c_i), j = 1, ..., n\}, c_i \in \{-1, +1\}$.

A linear SVM finds a vector w and a scalar constant b, such as: $\forall i, c_i(w_{ci}.dj + b) \geq 1$, and $||w||$ is minimized.

Most discriminative classifiers, including SVMs, are essentially two-class classifiers. A standard method of dealing with multi-class problems is to create an ensemble of yes/no binary classifiers, one for each label. This method is called "one-vs-others" [14]. For each label l_i, the positive class includes all documents which have l_i as one of their labels and the negative side includes all other documents. During application, the set of labels associated with a document d_j is $\{k\}$, such as: $w_k.d_j + b_k > 0$. This is the basic SVM method that serves as a baseline against which we compare other methods.

C. K-Neighbors Approach

One of the simplest, and rather trivial classifiers is the rote classifier, which memorizes all training data and performs classification only if the attributes of the test object match one of the training examples exactly [15]. A more sophisticated approach, k-nearest neighbor (KNN) classification [16], finds a group of k objects in the training set that are closest to the test object and bases the assignment of a label on the predominance of a particular class in this neighborhood. There are three key elements of this approach: a set of labeled objects, a distance or similarity metric to compute distance between objects and the value of parameter k, which represents the number of nearest neighbors.

To classify an unlabeled object, the distance of this object to the labeled objects is computed, its k-nearest neighbors are identified, and the class labels of these nearest neighbors are then used to determine the class label of the object.

Algorithm3 provides a high-level summary of the nearest neighbor classification method. Given a training set DR and a test object $z = (x', y')$ the algorithm computes the distance (or similarity) between z and all the training objects $(x, y) \in DR$ to determine its nearest-neighbor list: D_z. (x_i is the training data of $object_i$, while y_i is its class. Likewise, x' the data of the test object and y' is its class.) Once the nearest-neighbors list is obtained, the test object is classified based on the majority class of its nearest neighbors:

$$Majority\_Voting\_y' = argmax_v \sum_{x_i, y_i \in D_z} I(v = y_i).$$

where v is a class label, y_i is the class label for the i^{th} nearest neighbors and $I()$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

Algorithm 3 KNN for induction Rules Classification [5]

input: - a training rule set DR
- a parameter k : an integer between 1 and the number of rules already classified DR.
- z : the test rule to classify.
For (each classified rule in DR_i) **do**
Calculate the distance Dist(z, DR_i).
end for
- $D_z \leftarrow$ Select the k nearest neighbors (z);
For (each class) **do**

$$Majority\_Voting : y' = argmax_v \sum_{x_i, y_i \in D_z} I(v = y_i).$$

end for
- Attribute to z the class (y').

The basic nearest neighbors classification uses uniform weights. Namely, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors, such as nearer neighbors contribute more to the fit. This is why we implemented these two variants, and we name them respectively "Uniform_knn" and "Distant_knn".

D. Neural Networks Classifiers

The basic unit in a neural network is called "neuron" or "unit". Each neuron receives a set of inputs, denoted by the vector $\overline{X}_i$ [17] [18], which in this case, corresponds to the token frequencies in the i^{th} rule. Each neuron is also associated with a set of weights A, which are used for computing a function $f()$ of its inputs. A typical function which is often used in the neural network is the linear function as follows: $p_i = A.\overline{X}_i$. We assume that the class label is denoted by y_i. The goal of this approach is to learn the set of weights A with the use of the training set. The idea is to start off with random weights, and gradually update them when a mistake is done by applying the current function on the training example. The magnitude of the update is regulated by

a learning rate μ. This forms the core idea of the perceptron algorithm.

Algorithm 4 Perceptron Algorithm [18]

inputs: Learning Rate: μ
Training rules $(\overline{X}_i, y_i) \forall i \in \{1...n\}$.
Initialize weight vectors in A to 0 or small random numbers.
Repeat
– Apply each training rule to the neural network
– **if** $((A.\overline{X}_i)$ does not matches $y_i)$ **then**
update weigts A based on learning rate μ.
until weights in A converge.

E. Convolutional Neural network Classifier

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers, and then followed by one or more fully connected layers as in a standard multilayer neural network. The neurons of a convolutional layer are grouped in feature maps sharing the same weights, so the entire procedure becomes equivalent to convolution [19] [20]. Convolutional layers are usually followed by a nonlinear activation-layer, in order to capture more complex properties of the input data. Pooling layers are used to subsample the previous layer, by aggregating small rectangular subsets of values. Maximum or average pooling is often applied by replacing the input values with the maximum or the average value, respectively. Finally, one or more dense layers are put in place, each followed by an activation-layer, which produce the classification result.

The training of CNNs is performed similarly to that of other ANNs, by minimizing a loss function using gradient descent-based methods and back-propagation of the error.

Our CNN model is inspired by the contribution of [21] in text classification. Since this model demonstrated performant results, we adapted it in our study for induction rules classification. It is mainly composed of three convolutional layers followed by a non-linearity, max pooling and a softmax classification layer.

1) The Convolution: CNNs are responsible for major breakthroughs in image classification and are the core of most computer vision systems today. We can define convolution operation as a sliding window function applied to a matrix.

In vision, the filters slide over local patches of an image, but in Natural Language Processing, filters are typically used to slide over full rows of the token matrix. Thus, the "width" of filters is usually the same as the width of the input matrix. The height, or region size, may vary, but sliding windows in general terms over two to five words at a time.

2) Pooling Layers: Pooling layers are typically applied after the convolutional layers. Pooling layers subsample their input. The most common way to do pooling it to apply a max operation to the result of each filter. It is not necessarily to pool over the complete matrix. In our case, pooling is applied over the complete output, yielding just a single number for each filter.

IV. Evaluation and Experimentation

The collection of our rule bases is done from public benchmarks. We should have taken the results of data mining approaches (like: decision tree algorithms, or association rules mining) such as the input of our process. But given the slowness of the procedure, we directly adapt our rule bases from public datasets as follows:

IF $(attribute_1 = value_1)$ and $(attribute_2 = value_2)$ and . . . Then $(attribute_{n-1} = value_{n-1})$
where n is the columns number of the dataset.
The last attribute value represents the class column.

A. Public Datasets Collection

TABLE I
Benchmark Construction

Benchmark	Attributes	Rows	Classes
SMS Balanced DataSet	03	$12,000$	12
SMS Unblanaced DataSet	03	$42,000$	15
San Francisco Crime Classification Dataset (a subset)	09	$33,028$	12
News Aggregator Data Set	09	$422,937$	04
Amazon Fine Food Reviews	10	$568,454$	06

We have implemented our approaches on four public datasets, where the first dataset is divided into two variants rule bases: Balanced & Unbalanced SMS Datasets.

1) The first dataset known as **data_sms**[1], contains a total of 42,000 instances, distributed unequally over 15 categories: info, spam, ham, pickup, payment, bus, reservation, delivery, train, Cab, Hotel, Expiry, Appointment, Movie, Flight.
We have created a variant of this dataset, by removing the first three categories and taking exactly $1,000$ instances from each of the remaining 12 categories. We call this variant the SMS Balanced Dataset.
2) The second one is a subset of **San Francisco Crime Classification**[2]. This dataset is brought by SF Open Data, the central clearing-house for data published by the City and County of San Francisco. The benchmark contains incidents derived from San Francisco Police Department Crime Incident Reporting system. The data ranges from 1/1/2003 to 5/13/2015. The chosen subset includes 33,028 rows with 9 attributes which are:
   - Dates: timestamp of the crime incident.
   - Category: category of the crime incident (only in training file), it includes 39 different categories. This is the target variable we are going to predict.
   - Descript: detailed description of the crime incident (only in training file).
   - DayOfWeek: the day of the week.
   - PdDistrict: name of the Police Department District.
   - Resolution: how the crime incident was resolved (only in training file).

[1]https://www.kaggle.com/moose9200/data-sms/data
[2]https://www.kaggle.com/c/sf-crime/data

- Address: the approximate street address of the crime incident.
- X: Longitude.
- Y: Latitude.

3) The third dataset which is called **News Aggregator Dataset**[3] is provided by Artificial Intelligence Lab at the Faculty of Engineering, Roma Tre University - Italy, and it is published on UCI Website [23].
In this dataset, the news are grouped into clusters that represent pages discussing the same news story. The dataset includes also references to web pages.
   - 422,937 news pages and divided up into four classes.
   - 152,746 news of entertainment category.
   - 108,465 news of science and technology category.
   - 115,920 news of business category.
   - 45,615 news of health category.
   - Each class is represented by the news category (b = business, t = science and technology, e = entertainment, m = health)
4) The fourth benchmark is known as: **Amazon Fine Food Reviews**[4] and it consists of reviews of fine foods from Amazon. The data span a period of more than 10 years, including all 568,454 reviews up to October 2012. Reviews include product and user information, ratings and a plain text review. It also includes reviews from all other Amazon categories [22]. Data includes:
   - Reviews from Oct 1999 - Oct 2012.
   - 568,454 reviews.
   - 256,059 users.
   - 74,258 products.
   - 260 users with $>$ 50 reviews.
   - 6 classes: from 0 to 5 stars. which represents the satisfaction degree of the reviewer.

B. Evaluation Pattern

All the implemented algorithms take 80% of dataset for training and the remaining 20% for test, in which they are evaluated on the two criteria: execution time and classification accuracy.

The classification accuracy Acc_i of an individual algorithm i depends on the number of samples correctly classified (true positives plus true negatives) and is evaluated by the formula1.

$$Acc_i = \frac{t}{n} * 100 \quad (1)$$

where t is the number of sample cases correctly classified and n is the total number of sample cases.

C. Experimental Results

In this study, we use NVIDIA devices and Python language with Jupyter Notebook programming model. All the algorithms are executed and compared using one CPU of The **ROMEO NVIDIA DGX-1**[5] **AI supercomputer**. The results of all the adapted MLDM approaches for induction rules classification are summarized on tableII.

From TableII we can report many remarks:

- The KNN algorithm cannot handle large rule bases (crashes when dealing with over then 300,000: out of memory error).
- There is a big difference in algorithms performances when processing balanced datasets and the others that are not.
- In terms of execution time, the Multinomial Naive Bayes approach gives the best results, followed by the Bernoulli variant.
- The classification accuracy varies radically depending on the type of processed rule base.

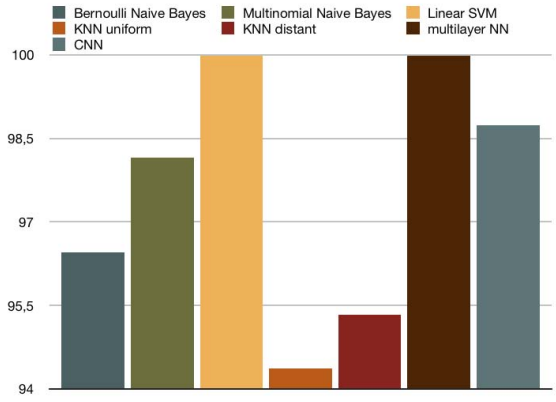

Fig. 3. *Classification Accuracy on Balanced SMS Rules*

Figure 3 shows the classification accuracy comparison of the MLDM classification approaches applied on the balanced SMS dataset. According to this figure, we remark that in general, all approaches give very good results. Nevertheless, if we want details, Linear SVM and Multilayer Neural Networks are more efficient than the other algorithms, so when SVM and Multilayer NN reach 100% of accuracy, the two variants of KNN algorithm do not exceed 95.5% of accuracy.

In figure 4, the classification accuracy comparison of the MLDM classification approaches applied on the aggregator news dataset is shown. From this figure, we remark that Linear SVM is more efficient than the other algorithms, so when SVM reaches 87.26% of accuracy, the CNN gives only 77.19% of accuracy on the test set. Moreover, KNN algorithms get an out of memory error,

Figure 5 resumes the quality of the MLDM classification approaches applied on three large unbalanced rule bases. We remark that the CNN algorithm clearly outperforms the other approaches on all sets of rules. So, when CNN algorithm does not fall below 88.10% accuracy, the best of the other algorithms does not exceed 79.57%.

[3]https://archive.ics.uci.edu/ml/datasets/News+Aggregator
[4]https://www.kaggle.com/snap/amazon-fine-food-reviews
[5]https://www.nvidia.fr/data-center/dgx-1/

TABLE II
DMLM CLASSIFICATION APPROACHES RESULTS

Dataset	Bernoulli Naive Bayes	Multinomial Naive Bayes	Linear SVM	KNN uniform	KNN distant	Multi Layer NN	CNN
Balanced SMS	Time(sec) = 0.028 sec Accuracy (%)= 95.65%	0.022sec 98.98%	0.20 sec 100%	1.59 sec 95.31%	1.46 sec 96.30%	2/epoch 100%	43/epoch 98.73%
Unbalanced SMS	Time(sec) = 0.114 sec Accuracy (%)= 59.85%	0.058sec 67.70%	1.04 sec 64.56%	13.35 sec 63.91%	12.17 sec 61.38%	39/epoch 67.94%	107/epoch 95.23%
San Francisco Crimes (subset of 12 categories)	Time(sec) = 0.096 sec Accuracy (%)= 54.84%	0.086sec 46.03%	1.27 sec 57.27%	14.02 sec 42.02%	14.21 sec 42.72%	67/epoch 53.13%	544/epoch 92.18%
News Aggregator	Time(sec) = 0.47 sec Accuracy (%)= 86.24%	0.24sec 84.67%	5.74 sec 87.26%	–– ––	–– ––	831/epoch 84.41%	7261/epoch 76.19%
Amazon Fine Food Reviews	Time(sec) = 0.62 sec Accuracy (%)= 69.58%	0.52sec 63.39%	27.20 sec 76.24%	–– ––	–– ––	430/epoch 79.57%	4060/epoch 88.10%

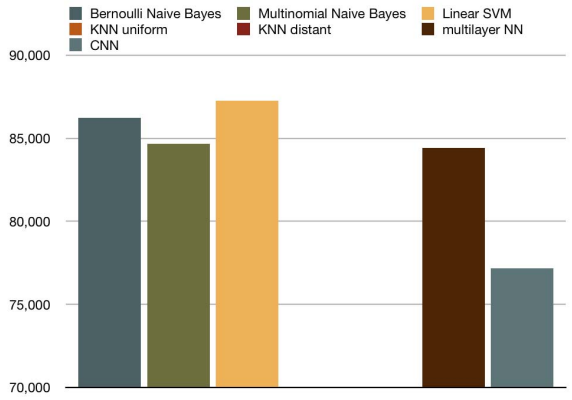

Fig. 4. *Classification Accuracy on Aggregator News Rules*

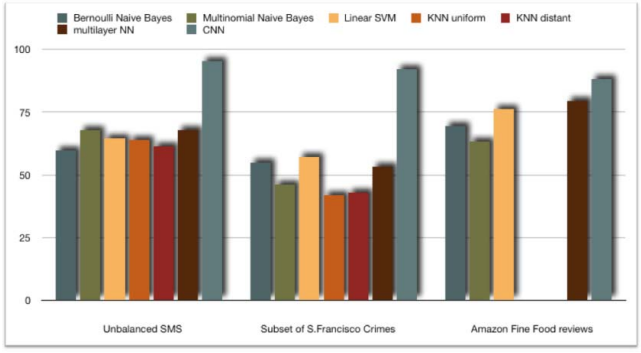

Fig. 5. *Classification Accuracy on Large Unbalanced Rule Bases*

D. Discussion

According to the obtained results, we remark that, the two variants of the KNN algorithm cannot handle large rule sets; in fact, the program crashes with an out of memory error when dealing with a rule base of more than $300,000$. This is due to the fact that KNN algorithm computes all the distances between the test set and the whole instances of training set, contrary to the other algorithms.

In terms of execution time, the Naive Bayes approach gives the best results. We explain that by the fact that the parameters of Naive Bayes classifier model, a-priori and conditional probabilities are determined using a deterministic set of steps, and this involves two very trivial operations that can be blindingly fast on modern day computers: counting and dividing. In addition, there is no iterations, no epochs or no error back-propagation. All these reasons make the Naive Bayes classifier perform very fast.

Moreover, we can report that the classification accuracy depends strongly on the type of the processed rule bases, i.e. there is a big difference between algorithms performances when processing balanced and unbalanced datasets. We observe the same when processing datasets with only four classes and more than six classes.

With numbers for the balanced rule bases that do not exceed 100,000 rules, the SVM and multilayer NN give very good results. In addition, the SVM gives good results for unbalanced rule bases that do not exceed five classes. Otherwise, the best results are obtained using the CNN algorithm for unbalanced rule bases with more than five categories.

Algorithm 5 Knowledge Mining Module-Selection

inputs: Agent Rule Base Rb
If (Balanced $(Rb) = true$) **then**
  $meta\_model \leftarrow$ Apply_Linear_SVM_Classifier(Rb);
else
  if $(nbr\_classes < 5)$ **then**
    $meta\_model \leftarrow$ Apply_Linear_SVM_Classifier(Rb);
  else
    $meta\_model \leftarrow$ Apply_CNN_Classifier(Rb);.

E. The New Intelligent Agent Architecture

As a result of this study, we introduce the new architecture of the intelligent agent, by adding to the classical architecture the two new packages, knowledge mining and meta-models.

- Knowledge Mining Package: composed of all the presented knowledge mining algorithms.
- Meta-models Package: represents the output of the training step of the agent rule base, using a selected classification algorithm from the knowledge mining package.

Contrary to the classical agent that infers all the rules sequentially and exhaustively, the new intelligent agent is able

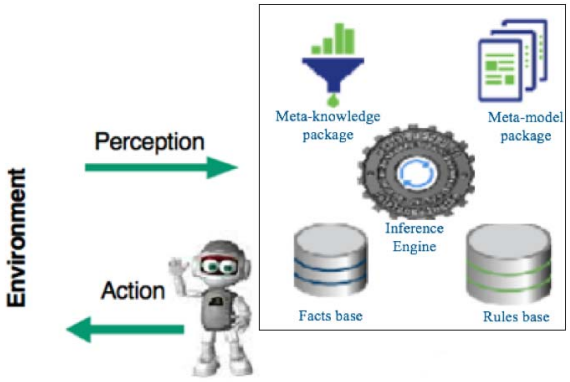

Fig. 6. *The Architecture of Knowledge Classification Based Agent*

to deduct only the class of rules of the new knowledge using its new packages. Module-selection from knowledge mining package is guided by the Algorithm 5.

V. Conclusion & Perspectives

In this paper, we presented a new concept of knowledge mining and meta-models extraction. Our work focused on the study of the supervised learning approaches for classification task. From this research, we have developed an extension of popular MLDM algorithms to deal with induction rules. Afterwards, we have tested and compared these adapted algorithms on large rule bases. The results are very satisfactory and variate depending to the kind of processed rule base.

Furthermore, we integrated the proposal support on the concept of intelligent agent. Hence, the emergence of a new architecture of the intelligent agent which allows to infer and discover new knowledge very quickly.

As future perspectives, we expect the implementation of a new hyper-heuristic that will automatically update the knowledge mining package and choose the learning method to apply according to the kind of the intelligent agent rule base. In addition, we plan to perform this study on very large scale, using in a first time, a multi GPU architecture, and thereafter, the deployment on the Hybrid Romeo cluster.

Acknowledgment

The authors would like to thank Julien.L, Arnaud.R & Fabien.B: our colleagues and administrators of the supercomputer DGX 1 in **ROMEO HPC Center**[6] for their helps and suggestions.

References

[1] AMATRIAIN, Xavier et PUJOL, Josep M. Data mining methods for recommender systems. In: Recommender systems handbook. Springer, Boston, MA, 2015. p. 227-262.

[2] Ilkka Tuomi, 1999, *Data is More Than Knowledge Implications of the Reversed Knowledge Hierarchy for Knowledge Management and Organizational Memory*, Journal of Management Information Systems Fall 1999, Vol. 16, No. 3., pp 107-121, 1999

[3] RUIZ, Paula Potes, FOGUEM, Bernard Kamsu, et GRABOT, Bernard. Generating knowledge in maintenance from Experience Feedback. Knowledge-Based Systems, 2014, vol. 68, p. 4-20.

[4] GRZYMALA-BUSSE, Jerzy W. A new version of the rule induction system LERS. Fundamenta Informaticae, 1997, vol. 31, no 1, p. 27-39.

[5] CHEMCHEM, Amine et DRIAS, Habiba. From data mining to knowledge mining: Application to intelligent agents. Expert Systems with Applications, 2015, vol. 42, no 3, p. 1436-1445.

[6] CHEMCHEM, Amine, DRIAS, Habiba, et DJENOURI, Youcef. Multi-level Clustering of Induction Rules: Application on Scalable Cognitive Agent. International Journal of Systems and Service-Oriented Engineering (IJSSOE), 2014, vol. 4, no 3, p. 1-25.

[7] KORDE, Vandana et MAHENDER, C. Namrata. Text classification and classifiers: A survey. International Journal of Artificial Intelligence & Applications, 2012, vol. 3, no 2, p. 85.

[8] IKONOMAKIS, M., KOTSIANTIS, Sotiris, et TAMPAKAS, V. Text classification using machine learning techniques. WSEAS transactions on computers, 2005, vol. 4, no 8, p. 966-974.

[9] JAIN, Aaditya et MANDOWARA, Jyoti. Text classification by combining text classifiers to improve the efficiency of classification. International Journal of Computer Application (2250-1797), 2016, vol. 6, no 2.

[10] HUANG, Jin, LU, Jingjing, et LING, Charles X. Comparing naive Bayes, decision trees, and SVM with AUC and accuracy. In: Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. IEEE, 2003. p. 553-556.

[11] MANNING, Christopher D., RAGHAVAN, Prabhakar, SCHTZE, Hinrich, et al. Introduction to information retrieval. Cambridge: Cambridge university press, 2008.

[12] GUYON, Isabelle, WESTON, Jason, BARNHILL, Stephen, et al. Gene selection for cancer classification using support vector machines. Machine learning, 2002, vol. 46, no 1-3, p. 389-422.

[13] VAPNIK, Vladimir. The nature of statistical learning theory. Springer science and business media, 2013.

[14] HSU, Chih-Wei et LIN, Chih-Jen. A comparison of methods for multiclass support vector machines. IEEE transactions on Neural Networks, 2002, vol. 13, no 2, p. 415-425.

[15] ADENIYI, D. A., WEI, Z., et YONGQUAN, Y. Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. Applied Computing and Informatics, 2016, vol. 12, no 1, p. 90-108.

[16] WU, Xindong, KUMAR, Vipin, QUINLAN, J. Ross, et al. Top 10 algorithms in data mining. Knowledge and information systems, 2008, vol. 14, no 1, p. 1-37.

[17] AGGARWAL, Charu C. et ZHAI, ChengXiang (ed.). Mining text data. Springer Science & Business Media, 2012.

[18] AGGARWAL, Charu C. (ed.). Data classification: algorithms and applications. CRC Press, 2014.

[19] ANTHIMOPOULOS, Marios, CHRISTODOULIDIS, Stergios, EBNER, Lukas, et al. Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. IEEE transactions on medical imaging, 2016, vol. 35, no 5, p. 1207-1216.

[20] ABDEL-HAMID, Ossama, MOHAMED, Abdel-rahman, JIANG, Hui, et al. Convolutional neural networks for speech recognition. IEEE/ACM Transactions on audio, speech, and language processing, 2014, vol. 22, no 10, p. 1533-1545.

[21] KIM, Yoon. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[22] MCAULEY, Julian John et LESKOVEC, Jure. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In: Proceedings of the 22nd international conference on World Wide Web. ACM, 2013. p. 897-908.

[23] Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[6] https://romeo.univ-reims.fr/pages/aboutUs