Improving the Cognitive Agent Intelligence by Deep Knowledge Classification

Amine Chemchem, Francois Alin, Michaël Krajecki

International Journal of Computational Intelligence and Applications

Improving The Cognitive Agent Intelligence by Deep Knowledge Classification

Amine Chemchem*, François Alin, Michael Krajecki
CReSTIC Center, University of Reims Champagne-Ardenne
Campus Moulin de la Housse BP 1039
51687 Reims cedex 2, France.

Received (July 6, 2018)
Revised (January 21, 2019)

In this paper, a new idea is developed for improving the agent intelligence. In fact with the presented Convolutional Neural Network (CNN) approach for knowledge classification, the agent will be able to manage its knowledge. This new concept allows the agent to select only the actionable rule class, instead of trying to infer its whole rule base exhaustively. In addition, through this research, we developed a comparative study between the proposed CNN approach and the classical classification approaches. As foreseeable the deep learning method outperforms the others in term of classification accuracy.

Keywords: Deep Learning; Data Mining; Cognitive Agent; Knowledge management; Supervised Classification.

1. Introduction

Since about thirty years, several researches from artificial intelligence community provide competitive approaches to accelerate the Knowledge Discovery from Data (KDD) process. Nowadays, the impressive evolution of Machine Learning and Data Mining (MLDM) methods furnishes a tremendous amount of knowledge. Our challenge in this contribution, is to mine these knowledge in order to extract meta-knowledge and/or meta-models. This will allow access to a more abstract level, and to get only the most important and interesting knowledge. The meta-knowledge extraction process is reached by the extension of data mining techniques to deal with knowledge[1]. However, the meta-model extraction is realised by performing machine learning models to classify knowledge. These various process are shown schematically in figure1.

In this study, a new convolutional neural network approach for knowledge classification is designed, developed, and then compared to the classical classification methods such as: Multilayer Perceptron Networks, Naive Bayes, Support Vector Machine, and K-Nearest Neighbors.

*mohamed-lamine.chemchem@univ-reims.fr

Another challenge is managed by this work, which consists of scaling-up knowledge processing. From this paper, a new parallelization of the proposed CNN model is implemented on the DGX1[a] server, which features 8 GPUs based on the Volta cards. This solution allows to deal with more than one million rows of knowledge, and performs learning step thirty times faster than the sequential model on one CPU.

A very interesting architecture of cognitive agent can be deduced from this study. In this context, the new agent is able to manage its rule base using the proposed knowledge classification package. With this new module, the inference engine select only the actionable rule class at the arrival of a new fact, instead of trying to deduce the whole rule base.

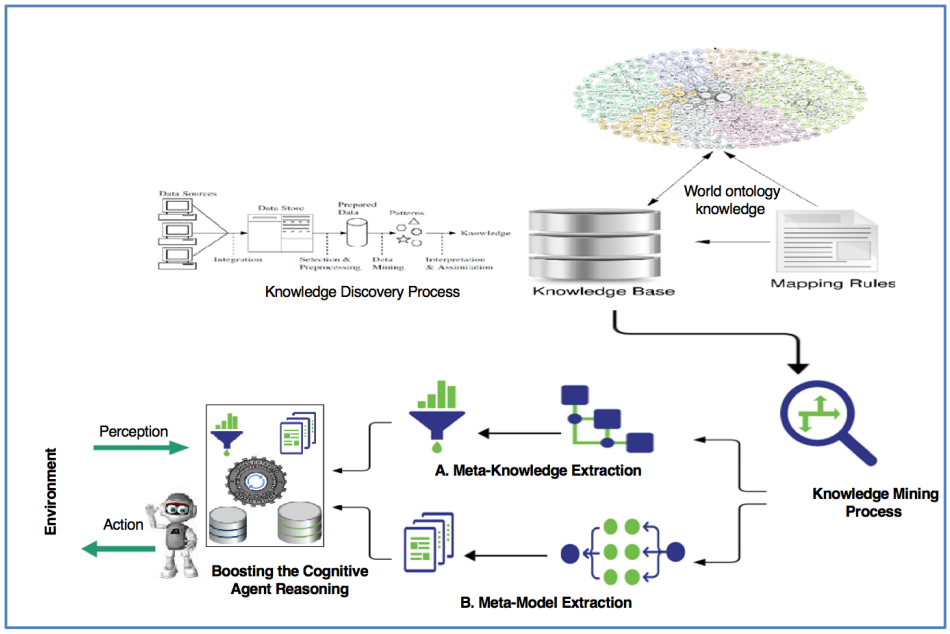

Fig. 1. *The General Workflow*

The rest of this paper is organized as follows: Next section shows the concept of knowledge mining. In section III the suggested algorithms of induction rules mining are described. Then, experimental results are shown in section IV compared to the deep learning algorithms. After that, a new architecture of the intelligent agent is

[a]DGX-1 is a line of Nvidia produced servers and workstations which specialize in using GPGPU to accelerate deep learning applications. `https://www.rave.com/dgx-1/`

introduced. Finally we conclude by making some remarks and talking about future perspectives.

2. Related Works

Our interest in this study revolves around two main subjects which are; scalable cognitive agent, and knowledge classification in general which involves induction rules mining.

For the first subject, we found very few papers with ideas about the notion of scalable cognitive agent. As in [2], the authors develop a scalable multi-agent learning algorithms for solving the winner determination problem in combinatorial double auctions. Or in [3], where the authors present a new simulation model using agent based model, under cognitive agent-based computing approach for self-organized shape formation in swarm robots. And, nothing about the paradigm that we would like to cover in this article. However, we notice that biologists and psychologists are showing interest in the study of scalable brain [4].

What can be said about the second topic, is that the literature offers a large spectrum of detailed research on pattern recognition. Knowledge classification including simple data and other patterns have been examined intensively over the last decade. In the following, we will review some pattern classification studies.

The authors in [5] apply text categorization for political blog posts classification. Their idea of identifying left versus right political alignment is quite different from our classification process, since we are interested by multi-label classification. Using a Naive Bayes classifier coupled with forward feature selection, they were able to outperform SVMs. Given their success, we also implement a Naive Bayes approach for the comparative study. In the same topic, in paper[6], an efficient convolutional neural network is applied to extract lexical and sentence level features for relation classification.

In Biomedicine domain, the authors of [7] proposed an efficient convolutional neural networks approach, for biomedical text classification. In [8] a new hybridization of SVM classification algorithm based on the information entropy with particle swarm optimization is proposed in order to improve the classification accuracy of biomedical datasets. Their results demonstrate that the proposed algorithm achieved good accuracy in biomedicine prediction.

More recently, several research studies are exploring the outcomes of the application of machine learning approaches on social networks. Such as [9][10][11] in which several new algorithms are developed for twitter sentiments classification.

The idea in [12] is very close to the concept that we would like to introduce. The authors develop a new evolutionary approach through ontological model for data and knowledge classification. The proposed genetic algorithm allows to obtain an effective solution of the classification problem in multidimensional space, and different variants of classes sorting in the ontology. In our study, we compare the performances of classification methods on big knowledge base. The result of this

process can be used as an ontology, since the aim of our study is to discover existing relations between knowledge classes.

3. The Knowledge Based Agent Concept

Nowadays, in computer science, the word "agent" is used in various domains. For this reason, many researchers defined the intelligent agent in different ways. We quote some of the following definitions:

"An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." [13]

"Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program, with some degree of independence or autonomy, and, in doing so, employ some knowledge or representation of the user's goals or desires." [14]

"Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions"[15]

According to these definitions, we can say that the agent in artificial intelligence community, is an autonomous entity with some degree of intelligence. Its main goal is to develop its knowledge by interacting with the environment for solving problems.

3.1. *The Cognitive Agent Architecture*

The knowledge based agent, known also as "cognitive agent", is an agent which has a reasoning ability on its knowledge base, with a capacity to manage interactions with other agents and/or its environment.

It receives knowledge from its environment, in order to develop them using a knowledge based system like -usually- an expert system. Figure2 illustrates the architecture of the cognitive agent, which is composed of three principals components:

- The inference engine: applies logical rules to the knowledge base and deduces new knowledge. This process would iterate as long as it can infer rules.
- The facts base: is a set of knowledge called "facts" and considered as true. From these facts, the inference engine will apply the rules from its rule base to deduce other facts and solve a problem of logic.
- The rules base: consists of some domain encoding of expertise for the system. This can be in the form of semantic nets [16], procedural representations [17], or production rules [18]. In general case, it is represented by production rules, called also "induction rules". These rules occur in sequences and are expressions of the form:

$$if < condition > then < action >$$

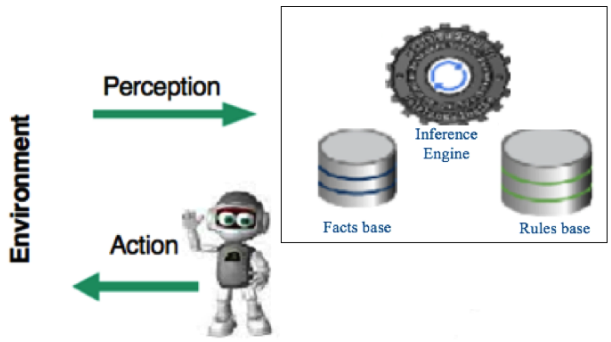

Fig. 2. *The Cognitive Agent Architecture*

where if the conditions are true then the actions are executed.

3.2. *Induction Rules Representation*

Induction rules are in the core of the cognitive agent architecture. 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 3. For these reasons, this work is designed for induction rules processing.

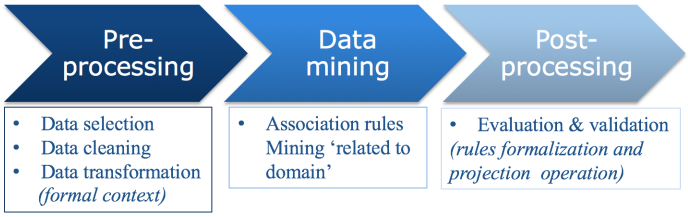

Fig. 3. *Knowledge Discovery Process [19]*

An induction rule is a boolean formula of the form: $R : X \rightarrow Y$, where X and Y are sets of clauses. X is called the premise part of the rule and Y its consequence [20].
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, and V is a set of values.

3.3. *Induction Rules Preliminaries*

Defining the mathematical preliminaries of induction rules is a necessary step before the mining process. In [21] similarity measure formula between two induction rules r_1, r_2 is presented as follows:

$$Similarity(r_1, r_2) = \frac{|C_1 \cup C_2| - |C_1 \cap C_2| + |V_1 \cup V_2| - |V_1 \cap V_2|}{|r_1 + r_2|}$$

Where: C_i is the set of clauses of R_i, and V_i is the set of variables of R_i.

Afterwards, this formula has been improved in a previous work [22], in order to speed-up the mining process.

In this study, the induction rules are considered as text. Therefore, this work summarizes solving the well-known problem of text classification.

Like documents classification problem [23][24], we define intuitively the induction rules classification (IRC) problem as the task of classifying rules under a predefined category. More formally, if r_i represents a rule of 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.

4. Data Mining Methods For Knowledge Classification

Classification of a collection consists of dividing the items that make up the collection into categories or classes [25, 26]. 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. A classification task begins with build training data for which the target values (or class assignments) are known. Many classification algorithms use different techniques for finding relations between the predictor attributes values and the target attributes values in the build data.

According to [27] the algorithms: K-Nearest Neighbours (KNN), Support Vector Machine (SVM) and Naive Bayes belong to the top ten classification algorithms of data mining. In the following subsections, a summarised overview of these algorithms is reported.

4.1. *Naïve Bayes Classification Approaches*

The naïve Bayes classifier is a simple probabilistic approach, which is based on Bayes theorem with strong and naïve independence assumptions [28]. 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 naive 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. Naïve 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[29].

There are several naïve Bayes variations. Here we will discuss about two of them: (1) the multinomial naive Bayes, and (2) the Bernoulli Naïve Bayes. Note that each can deliver completely different results since they use various 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 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, examine two common naïve Bayes variations which differ on the way that they calculate the conditional probabilities of each feature and on the scoring of each category.

4.1.1. *Bernoulli Naïve Bayes*

The Bernoulli variation, as described by [30], 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.

Algorithm 1 Bernoulli Naïve 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 $argmax_{c \in C} score[c]$

4.1.2. *Multinomial Naïve Bayes Algorithm*

This variation, as explained by [30], 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 2 Multinomial Naïve 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 $argmax_{c \in C} score[c]$

4.2. *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)[31, 32].

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 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" [33]. 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.

4.3. *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[34]. A more sophisticated approach, k-nearest neighbor (kNN) classification [27], 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 neighbours are identified, and the class labels of these nearest neighbours 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.

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".

Algorithm 3 KNN for induction Rules Classification [35]

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').

5. Machine Learning Techniques overview

5.1. *Multilayer Perceptron Neural Networks Classifiers*

The basic unit in a neural network is called "neuron" or "unit". Each neuron receives a set of inputs, which are denoted by the vector $\overline{X}_i$ [36][37], 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 [37]

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.

5.2. *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 [38, 39]. 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 classical Multilayer Perceptron Networks, by minimizing a loss function using gradient descent-based methods and back-propagation of the error.

5.3. *The CNN Model for Knowledge Classification*

Our CNN model is inspired by the contribution of [40] in text classification. Since its model demonstrated well 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 soft-max classification layer. In the following, we give a brief explanation of the main components of our network: rules matrix, activations, convolutional, pooling and soft-max layers.

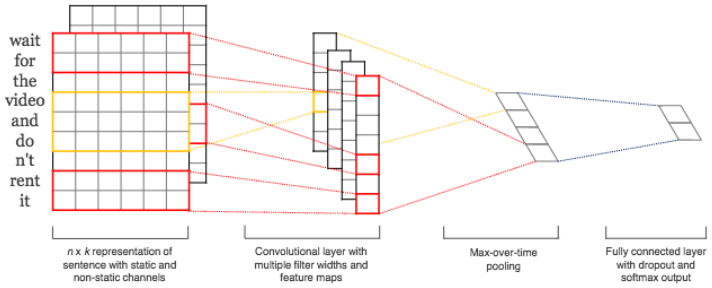

Fig. 4. *Model architecture with two channels for an example sentence [41]*

5.3.1. *The input matrix*

First, the input to the model are induction rules represented as a matrix. Each row of this matrix corresponds to one token,(typically a variable or a value of

the rule clauses). Typically, these vectors are word embeddings (low-dimensional representations) like word2vec or GloVe, but they could also be one-hot vectors that index the word into a vocabulary like in our case. For a 10 word rule using a 100-dimensional embedding we would have a 10*100 matrix as our input.

5.3.2. *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. This operation is explained schematically in figure5.

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[42].

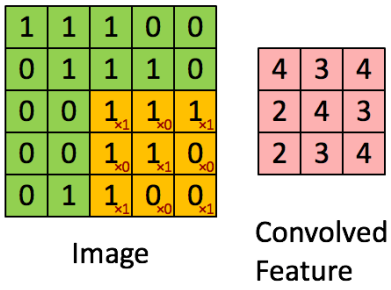

Fig. 5. *Example of a Convolution*

5.3.3. *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. For example, figure6 shows max pooling for a 2*2 window. In our case, pooling is applied over the complete output, yielding just a single number for each filter.

6. 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 asso-

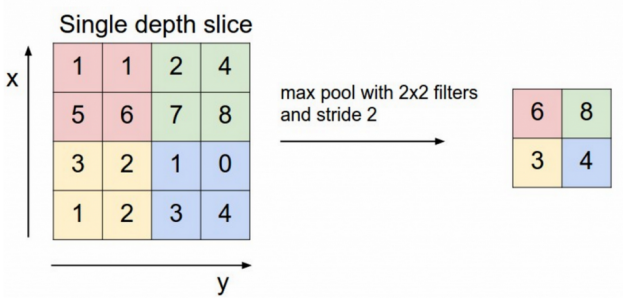

Fig. 6. *Example of max pooling in CNN [43]*

ciation rules mining) such as the input of our process. But given the slowness of this procedure, we directly adapt our rule bases from public datasets using the following transformation formula:

IF $(attribute_1 = value_1)$ and $(attribute_2 = value_2)$ and . . . Then $(attribute_n = value_n)$

Where n is the columns number of the dataset. For example, if we have a data row as in table 1. After transformation step, the obtained rule is shown on table 2.

Product Id	Profile Name	Helpfulness Denominator	Summary	Text	Classes (score 0..5)
B00CH1	Carol A.	1	Healthy Dog Food	A very healthy dog food. Good for their digestion. Also for...	4

Rule	Classes
IF (product Id = B00CH1) and (Profile Name = Carol A.) and (Helpfulness Den =1) and text =(A very healthy dog food. Good for their digestion. Also for...) then (summary = Healthy Dog Food)	4

The authors would like to notice that rules representation as explained in table 2 is just taken as text for classification aims, and not for the inference process. However, if the goal was a knowledge inference system, we have to select standardized rules, classify them manually, in order to be able to perform the experimental study.

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	06	568,454	06

6.1. *Public Datasets Collection*

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**[b]. It 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**[c]. 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).
- Address : the approximate street address of the crime incident.
- X : Longitude.
- Y : Latitude.

(3) The third dataset which is called **News Aggregator Dataset**[d] is provided by Artificial Intelligence Lab at the Faculty of Engineering, Roma Tre University

[b]https://www.kaggle.com/moose9200/data-sms/data
[c]https://www.kaggle.com/c/sf-crime/data
[d]https://archive.ics.uci.edu/ml/datasets/News+Aggregator

- Italy, and it is published on UCI Website [44].

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**[e] 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 [45]. 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.

6.2. *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.

$$Acc_i = \frac{t}{n} * 100 \tag{1}$$

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

6.3. *Experimental Results*

In this study we use Python language, Tensorflow tool, Keras library with Jupyter Notebook programming model. All the algorithms are executed and compared using

[e]https://www.kaggle.com/snap/amazon-fine-food-reviews

various number of GPUs devices of The **ROMEO NVIDIA DGX-1**[f] **AI supercomputer**. The programs execution process is shown in figure7. The results of all the developed MLDM approaches for induction rules classification are summarized on table4.

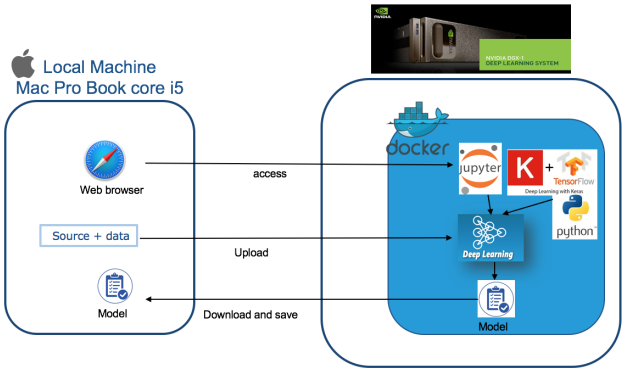

Fig. 7. *Experimental Environment Devices*

Classification Approach	Balanced SMS	Unbalanced SMS	San Francisco Crimes	News Aggregator	Amazon Reviews
Bernoulli Naive Bayes	Time(sec)=0.028 Accuracy= 95.65%	0.114 59.85%	0.096 54.84%	0.471 92.40%	0.629 70.98%
Multinomial Naive Bayes	Time(sec)=0.022 Accuracy= 98.98%	0.058 67.70%	0.086 46.03%	0.243 92.22%	0.525 64.17%
Linear SVM	Time(sec)=0.190 Accuracy= 100%	1.043 64.56%	1.271 57.27%	5.525 95.11%	27.209 77.92%
KNN Uniform	Time(sec)=1.057 Accuracy= 95.21%	13.354 63.91%	15.083 42.02%	– –	– –
KNN Distant	Time(sec)=1.055 Accuracy= 96.30%	12.173 61.38%	14.538 42.72%	– –	– –
MPL NN	Time=2sec/epoch Accuracy= 100%	36s/ep 67.94%	67s/ep 53.13%	446s/ep 95.34%	1430s/ep 79.49%
CNN	Time=43sec/epoch Accuracy= 98.73%	1711s/ep 95.19%	544s/ep 92.18%	22461s/ep 76.19%	21899s/ep 90.03%

[f]https://www.nvidia.fr/data-center/dgx-1/

From Table4 we can report many remarks:

- The KNN algorithm cannot handle large rule bases (crashes when dealing with over then 300,000 rows: out of memory error).
- There is a big difference between algorithms performances when processing balanced datasets and the others that are not.
- In terms of execution time, the Multinomial Nave Bayes approach gives the best results, followed by the Bernoulli variant.
- Classification accuracy varies radically depending on the type of processed rule base.
- In general, the best accuracy scores are obtained by the CNN approach, since the latter does not yield less than 76%, whatever the type of processed dataset.

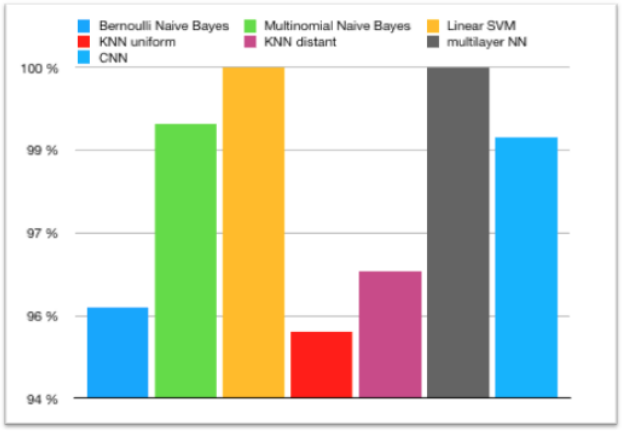

Fig. 8. *Classification Accuracy on Balanced SMS Rules*

Figure 8 shows the classification accuracy comparison of the MLDM approaches applied on the balanced SMS dataset. According to this figure we remark that in general, all the compared approaches give very good results, since they get all a score better than 95,2% on their test set.

In figure 9 the classification accuracy comparison of the MLDM approaches applied on news aggregator dataset is shown. From this figure we remark that also here, all the compared approaches get a good classification accuracy. Nevertheless, if we want details, Linear SVM and MPL Neural Networks are more efficient than the other algorithms. So, when SVM and MPL NN reach 100% of accuracy, the CNN approach do not exceed 76.16%.

Figure 10 resumes the classification accuracy scores of the compared MLDM approaches, applied on three multi-class unbalanced rule bases. Which are from the left to right; unbalanced sms, San-Francisco Crimes and Amazon Fine Food Reviews

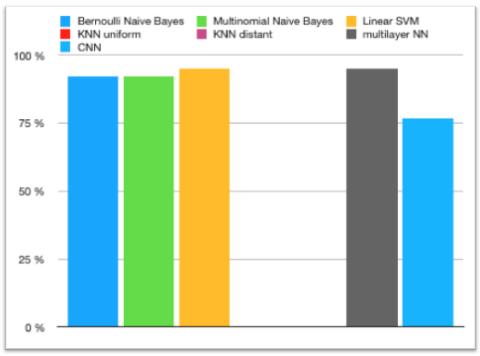

Fig. 9. *Classification Accuracy on Aggregator News Rules*

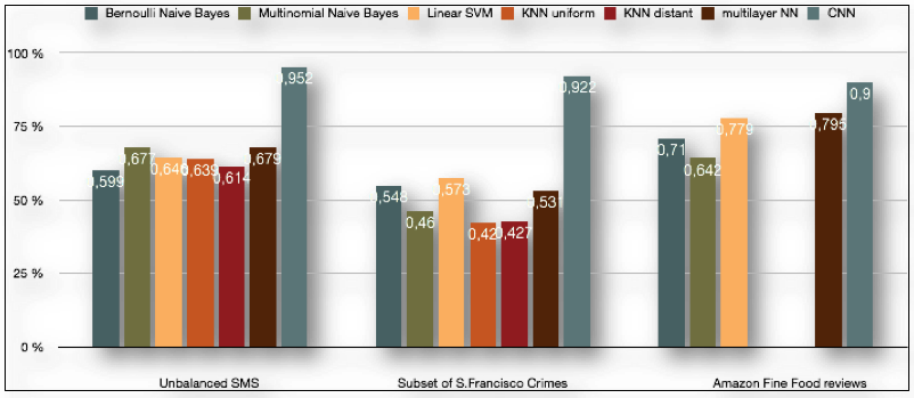

Fig. 10. *Classification Accuracy on Multi-class Unbalanced Rule Bases*

datasets. We notice on all of them, that the CNN algorithm outperforms clearly the other approaches. In fact, when CNN algorithm does not fall below 90%. The best of the other algorithms does not exceed 79.57%.

6.4. *Discussion*

According to the obtained results, we notice many remarks. The first one is that 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 duo to the fact that KNN algorithm computes all the distances between its test set and the whole instances of training set, contrary to the other algorithms.

The second remark is about execution time; Nave Bayes approach gives the best results on this criterion. Effectively, since the parameters of Naive Bayes model (i.e., a-priori and conditional probabilities) are learnt or rather 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. There is no iterations, no epochs, no error back-propagation. All These reasons make the Naive Bayes classifier very fast.

Moreover, we can report that the classification accuracy depends strongly on the type of 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.

6.5. *The New Intelligent Agent Architecture*

As a result of this study, we introduce the new architecture of knowledge classification based agent, by adding to the classical architecture the two new packages, knowledge mining and meta-models, as shown in Figure11.

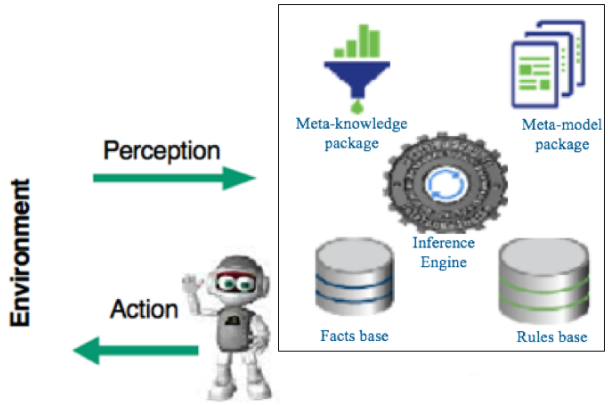

Fig. 11. *The Architecture of Knowledge Classification Based Agent*

- 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 its rules sequentially and exhaustively, the knowledge classification based agent is able to deduce only the (action-

able) class of rules of the new knowledge, using its new packages. The module-selection process from knowledge mining package depends on the type of rule base and is guided by Algorithm5.

Algorithm 5 Knowledge Mining Module-Selection

inputs: Agent Rule Base *Rb*
If (Balanced (*Rb*) = *true*) **then**
  meta_model ← Apply_Linear_SVM_Classifier(*Rb*);
else
  if (*nbr_classes* < 5) **then**
    meta_model ← Apply_Linear_SVM_Classifier(*Rb*);
  else
    meta_model ← Apply_CNN_Classifier(*Rb*);.

6.6. *Parallel Model Implementation on multi-GPU*

From the obtained results, convolutional neural networks have demonstrated their ability to achieve the best accuracy rates on almost all of rule bases. However, their major disadvantage is that they take a lot of time during the learning step, especially when processing large rule sets. To accelerate the learning stage of the CNN model, we have developed a parallel models, using multi GPUs devices of DGX1. The obtained results are summarised in table5.

Dataset	**1 CPU**	**2 GPUs**	**4 GPUs**	**8 GPUs**
Unbalanced SMS	Time/epoch = 1711 sec Time/step = 51 ms	81sec 2 ms	64 sec 2 ms	55 sec 2 ms
San Francisco Crimes (subset of 12 categories)	Time/epoch = 544 sec Time/step= 21 ms	27sec 1 ms	21 sec 0.795 ms	20 sec 0.737 ms
News Aggregator	Time/epoch = 22511 sec Time/step= 69ms	1741sec 5 ms	1447 sec 4 ms	1340sec 4 ms
Amazon Fine Food Reviews	Time(sec) = 21899 sec Time/step= 50ms	2105sec 5ms	1707 sec 4ms	1391 sec 3ms
The large Rule Base (1,065,686 rules)	Time/epoch = 37531sec Time/step= 46ms	2085sec 3 ms	1493 sec 2ms	1230sec 1ms

According to table5 the best results are obtained when using all the available GPUs (the 8) of the DGX1. For instance figure12 shows how execution time of

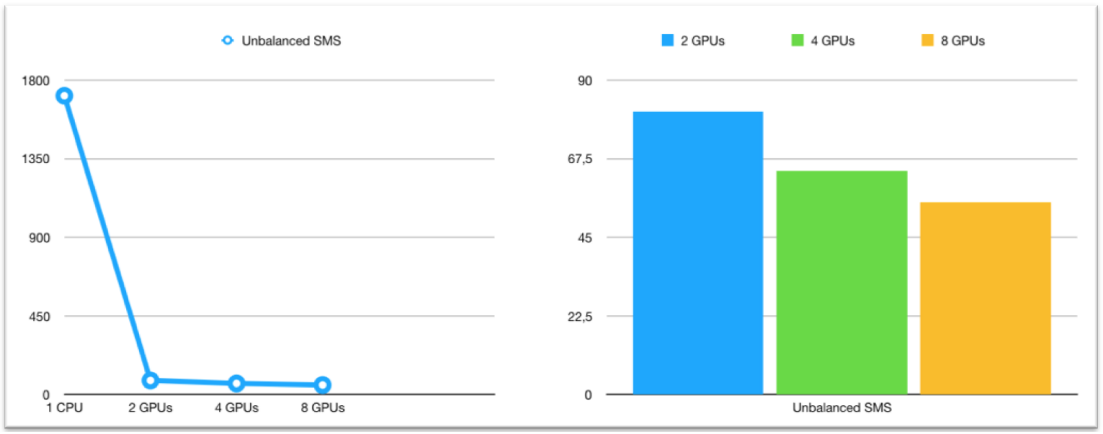

Fig. 12. *Training Time Comparison Using Multi-GPU Model on SMS Dataset*

the training step change, when dealing with 1CPU, 2GPUSs, 4GPUs, and 8GPUs. We can clearly remark that when using 8 GPUs, the training step becomes very faster compared to its execution on the other devices. With numbers, training our model on unbalance sms dataset takes only 55 seconds for one epoch, instead of 1711 seconds when using one CPU.

Fig. 13. *Training Time Comparison Using Multi-GPU Model on the large rule base*

The last line of table5 is schematised on figure13. It represents the comparison of the sequential CNN model to the parallel one on 2,4 and 8GPUs, when processing the big rule base. The latter is obtained by the fusion of all the four previous rule sets. It includes 1,062,000 rules with four classes (each class represents the provided rule set). The classification accuracy here (with CNN model) do not change when

executing on CPU or GPUs and is equal to 98.79%. For the execution time; training one epoch on the 8GPUs takes 1230sec i.e 20 minutes. Instead of 37,531sec i.e more than 10 hours and 25minutes when executed on the CPU1 of DGX.

From these results, we can conclude that, thanks to the parallelization of our CNN model on the 8 GPUs, the training process becomes till 30 times faster than the sequential one on 1 CPU.

when training our model on SMS data set, and using the 8 GPUs, the process becomes 30 times faster than training on one CPU.

7. Conclusion

In this study, a new concept of knowledge rules classification is presented. The contribution is focused on a comparative study of the popular machine learning and data mining algorithms which are developed for induction rules classification. The best results are obtained by the proposed convolutional neural network model, and are very satisfactory in terms of classification accuracy. Nevertheless, this model takes a lot of time in its training step. For this reason, we implemented a parallel version on the GPUs devices of the DGX1, and the results showed that the model training becomes up to 30 times faster than the sequential one.

Furthermore, we integrated the proposal concept on the classical architecture of cognitive agent. By incorporating the new packages of meta-knowledge and meta-models. This new architecture allows to deduce and discover new knowledge very quickly, thanks to its ability for inferring only the actionable class of rules, unlike to the classical agent which tries to deduce its whole rule base.

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 by the deployment on the Hybrid Romeo cluster.

Acknowledgements

The authors would like to thank Arnaud.R & Fabien.B: our colleagues and administrators of the DGX1 & ROMEO supercomputers for their helps and suggestions.

References

1. A. Chemchem, F. Alin, M. Krajecki, Deep learning and data mining classification through the intelligent agent reasoning, in: 2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2018, pp. 13–20. doi:10.1109/W-FiCloud.2018.00009.
2. F.-S. Hsieh, C.-S. Liao, Scalable multi-agent learning algorithms to determine winners in combinatorial double auctions, Applied Intelligence 43 (2) (2015)

308–324. doi:10.1007/s10489-014-0643-9.
URL https://doi.org/10.1007/s10489-014-0643-9

3. Y. R. Darr, M. A. Niazi, Towards self-organized large-scale shape formation: A cognitive agent-based computing approach, CoRR abs/1711.06426. arXiv:1711.06426.
4. C. Eliasmith, O. Trujillo, The use and abuse of large-scale brain models, Current Opinion in Neurobiology 25 (2014) 1 – 6, theoretical and computational neuroscience.
5. K. T. Durant, M. D. Smith, Predicting the political sentiment of web log posts using supervised machine learning techniques coupled with feature selection, in: O. Nasraoui, M. Spiliopoulou, J. Srivastava, B. Mobasher, B. Masand (Eds.), Advances in Web Mining and Web Usage Analysis, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 187–206.
6. D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, Relation classification via convolutional deep neural network, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 2335–2344.
7. A. Rios, R. Kavuluru, Convolutional neural networks for biomedical text classification: Application in indexing biomedical articles, in: Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics, BCB '15, ACM, New York, NY, USA, 2015, pp. 258–267.
8. M. Li, X. Lu, X. Wang, S. Lu, N. Zhong, Biomedical classification application and parameters optimization of mixed kernel svm based on the information entropy particle swarm optimization, Computer Assisted Surgery 21 (sup1) (2016) 132–141.
9. O. Serban, N. Thapen, B. Maginnis, C. Hankin, V. Foot, Real-time processing of social media with sentinel: A syndromic surveillance system incorporating deep learning for health classification, Information Processing & Management.
10. S. Liu, X. Cheng, F. Li, F. Li, Tasc:topic-adaptive sentiment classification on dynamic tweets, IEEE Transactions on Knowledge and Data Engineering 27 (6) (2015) 1696–1709.
11. G. S. Bhathal, G. Gupta, SENTIMENT ANALYSIS OF ENGLISH TWEETS USING DATA MINING: Data Mining, Sentiment Analysis, BookRix, 2018.
12. V. Bova, V. Kureichik, D. Zaruba, Data and knowledge classification in intelligence informational systems by the evolutionary method, in: 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), 2016.
13. M. Wooldridge, An introduction to multiagent systems, John Wiley & Sons, 2009.
14. M. Burgin, G. Dodig-Crnkovic, A systematic approach to artificial agents, CoRR abs/0902.3513. arXiv:0902.3513.
URL http://arxiv.org/abs/0902.3513
15. B. Hayes-Roth, An architecture for adaptive intelligent systems, Artificial In-

telligence 72 (1) (1995) 329 – 365.
16. P. P. Ruiz, B. K. Foguem, B. Grabot, Improving maintenance strategies from experience feedback, IFAC Proceedings Volumes 46 (9) (2013) 625–630.
17. T. Winograd, Frame representations and the declarative/procedural controversy, in: D. G. BOBROW, A. COLLINS (Eds.), Representation and Understanding, Morgan Kaufmann, San Diego, 1975, pp. 185 – 210.
18. R. Davis, B. Buchanan, E. Shortliffe, Production Rules as a Representation for a Knowledge-Based Consultation Program, Springer New York, New York, NY, 1985, pp. 3–37.
19. P. P. Ruiz, B. K. Foguem, B. Grabot, Generating knowledge in maintenance from experience feedback, Knowledge-Based Systems 68 (2014) 4 – 20, enhancing Experience Reuse and Learning.
20. J. W. Grzymala-Busse, A new version of the rule induction system lers, Fundamenta Informaticae 31 (1) (1997) 27–39.
21. H. Drias, A. Aouichat, A. Boutorh, Towards incremental knowledge warehousing and mining, in: S. Omatu, J. F. De Paz Santana, S. R. González, J. M. Molina, A. M. Bernardos, J. M. C. Rodríguez (Eds.), Distributed Computing and Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 501–510.
22. A. Chemchem, H. Drias, Y. Djenouri, Multilevel clustering of induction rules for web meta-knowledge, in: Á. Rocha, A. M. Correia, T. Wilson, K. A. Stroetmann (Eds.), Advances in Information Systems and Technologies, Springer Berlin Heidelberg, 2013, pp. 43–54.
23. V. Korde, C. N. Mahender, Text classification and classifiers: A survey, International Journal of Artificial Intelligence & Applications 3 (2) (2012) 85.
24. A link-bridged topic model for cross-domain document classification, Information Processing & Management 49 (6) (2013) 1181 – 1193.
25. I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, 2016.
26. S. B. Kotsiantis, I. Zaharakis, P. Pintelas, Supervised machine learning: A review of classification techniques, Emerging artificial intelligence applications in computer engineering 160 (2007) 3–24.
27. X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowledge and Information Systems 14 (1) (2008) 1–37.
28. A. Jain, J. Mandowara, Text classification by combining text classifiers to improve the efficiency of classification, International Journal of Computer Application (2250-1797) 6 (2).
29. J. Huang, J. Lu, C. X. Ling, Comparing naive bayes, decision trees, and svm with auc and accuracy, in: Third IEEE International Conference on Data Mining(ICDM), Vol. 00, 2003, p. 553.

30. C. D. Manning, P. Raghavan, H. Schütze, Introduction to information retrieval: Cambridge: 2008.
31. I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning 46 (1) (2002) 389–422.
32. V. Vapnik, The nature of statistical learning theory, Springer science & business media, 2013.
33. C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, IEEE Transactions on Neural Networks 13 (2) (2002) 415–425.
34. D. Adeniyi, Z. Wei, Y. Yongquan, Automated web usage data mining and recommendation system using k-nearest neighbor (knn) classification method, Applied Computing and Informatics 12 (1) (2016) 90 – 108.
35. A. Chemchem, H. Drias, From data mining to knowledge mining: Application to intelligent agents, Expert Systems with Applications 42 (3) (2015) 1436 – 1445. doi:https://doi.org/10.1016/j.eswa.2014.08.024.
36. B. Liu, L. Zhang, A survey of opinion mining and sentiment analysis, in: Mining text data, Springer, 2012, pp. 415–463.
37. C. C. Aggarwal, Data classification: algorithms and applications, CRC Press, 2014.
38. M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, S. Mougiakakou, Lung pattern classification for interstitial lung diseases using a deep convolutional neural network, IEEE Transactions on Medical Imaging 35 (5) (2016) 1207–1216.
39. O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu, Convolutional neural networks for speech recognition, IEEE/ACM Transactions on Audio, Speech, and Language Processing 22 (10) (2014) 1533–1545.
40. Y. Kim, Convolutional neural networks for sentence classification, CoRR abs/1408.5882. arXiv:1408.5882.
URL http://arxiv.org/abs/1408.5882
41. Y. Kim, Convolutional neural networks for sentence classification, arXiv preprint arXiv:1408.5882.
42. D. Britz, Wildml, artificial intelligence, deep learning, and nlp (2015).
URL http://www.wildml.com/2015/11/
43. A. Karpathy, Convolutional neural networks for visual recognition (2018).
URL http://cs231n.github.io/convolutional-networks/#pool
44. D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017).
URL http://archive.ics.uci.edu/ml
45. J. J. McAuley, J. Leskovec, From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews, in: Proceedings of the 22Nd International Conference on World Wide Web, WWW '13, ACM, New York, NY, USA, 2013, pp. 897–908.