Distributed Optimization of Classifier Committee Hyperparameters

Sanzhar Aubakirov[1], Paulo Trigo[2] and Darhan Ahmed-Zaki[1]
[1]*Department of Computer Science, Al-Farabi Kazakh National University, Almaty, Kazakhstan*
[2]*Instituto Superior de Engenharia de Lisboa, Biosystems and Integrative Sciences Institute Agent and Systems Modeling, Lisbon, Portugal*

Keywords: Distributed Computing, Text Processing, Machine Learning, Hyperparameters Optimization.

Abstract: In this paper, we propose an optimization workflow to predict classifiers accuracy based on the exploration of the space composed of different data features and the configurations of the classification algorithms. The overall process is described considering the text classification problem. We take three main features that affect text classification and therefore the accuracy of classifiers. The first feature considers the words that comprise the inputtext; here we use the N-gram concept with different N values. The second feature considers the adoption of textual pre-processing steps such as the stop-word filtering and stemming techniques. The third feature considers the classification algorithms hyperparameters. In this paper, we take the well-known classifiers K-Nearest Neighbors (KNN) and Naive Bayes (NB) where K (from KNN) and a-priori probabilities (from NB) are hyperparameters that influence accuracy. As a result, we explore the feature space (correlation among textual and classifier aspects) and we present an approximation model that is able to predict classifiers accuracy.

1 INTRODUCTION

The rapid progress on computer–based communications and information dissemination generates large amounts of data that are daily available in many domains. The field of machine learning (ML) has seen unprecedented growth due to a new wealth of data, to the increases in computational power, to new algorithms, and a plethora of exciting new applications. In this dissertation we are focusing on text classification algorithms. Text classification is the process of organizing data into categories for its most effective and efficient usage.

Nowadays there is a great choice of text classification algorithms. There are 179 methods of classification by machine learning algorithms (Fernández-Delgado et al., 2014). As researchers tackle more ambitious problems, the models they use are also becoming more sophisticated. However, according to the researches (Sebastiani, 2002) classifier ensembles outperforms the best individual classifier. Classifier ensembles (committees) are based on the idea that, given a task that requires expert knowledge to perform, k experts may be better than one if their individual judgments are appropriately combined.

The growing complexity of ML models inevitably comes with the introduction of additional parameters, which is often expressed via a vector of model parameters. Additional parameters express higher-level properties of the learning model and therefore they cannot be learned directly from the regular training process; such parameters are designated as hyperparameters. Hyperparameters change the way the learning algorithm itself works (e.g. hyperparameters are used to describe a thresholds, the number of neurons in a hidden layer, the number of data points that a leaf in a decision tree must contain to be eligible for splitting). Each classification algorithm is tuned via hyperparameters that affect the learning process and the final accuracy of the prediction model. In the context of text classification, the input (text) is pre–processed by a set of operators and therefore each operator also influences the prediction accuracy. These hyperparameters are typically optimized in an outer loop that evaluates the effectiveness of each hyperparameter configuration using cross-validation. Taking into account each algorithms hyperparameters, there is a staggeringly large number of possible alternatives overall.

The design decisions range from the classification algorithm, optimization parameters such as learning rates and text pre–processing parameters such as stemming. Proper setting of these hyperparameters is critical for performance on difficult problems. There

Aubakirov, S., Trigo, P. and Ahmed-Zaki, D.
Distributed Optimization of Classifier Committee Hyperparameters.
DOI: 10.5220/0006884101710179
In *Proceedings of the 7th International Conference on Data Science, Technology and Applications (DATA 2018)*, pages 171-179
ISBN: 978-989-758-318-6

are many methods for optimizing over hyperparameter settings, ranging from simplistic procedures like grid or random search (Bergstra et al., 2011; Bergstra and Bengio, 2012), to more sophisticated model–based approaches using random forests (Hutter et al., 2011) or Gaussian processes (Snoek et al., 2012).

Hence, in the context of the text classification problem, we can formulate a feature space composed of: a) data set, b) hyperparameters of the classification algorithms, and c) textual pre–processing operators. According to (Garey and Johnson, 1990) the task of selecting the right features is nontrivial and checking all possible combinations is an NP–complete task. The task of selecting the classifier algorithm becomes resource intensive for the ensemble of classifiers and it requires expert knowledge from various areas of Computer Science (Machine Learning, Natural Language Processing, High Performance Computing).

In this research, we propose a methodology to deal with this complex task using an optimization model that explores the feature space with the goal of maximizing the ensemble of classifiers accuracy. We outline the optimization model and describe each step and show that we can approximate the maximization goal using a regression model.

2 RELATED WORKS

The task of optimizing classification algorithms hyperparameter is addressed by many authors (Bergstra et al., 2011; Forman, 2003; Lim et al., 2000; Dasgupta et al., 2007; Thornton et al., 2012). A large study of classification algorithms shows that not only accuracy of the algorithm depends on selected features and input data, but training time, scalability and interpretability of algorithm (Lim et al., 2000). Another research (Dasgupta et al., 2007) points out the challenges associated with automated text, such as a) choosing an appropriate data structure to represent the documents, b) choosing an appropriate objective function to optimize in order to avoid overfitting, c) obtain good generalization, and d) dealing with algorithmic issues arising because of the high formal dimensionality of the data. This last challenge can be addressed via a prior selection of a subset of the features available for describing the data (Dasgupta et al., 2007). Such selection occurs before applying a learning algorithm and setting its operational parameters. A large number of studies on feature selection have focused on text domains both for binary and multi-class problems. This fails to investigate the best possible accuracy obtainable for any single class (Forman, 2003).

Those studies deal with feature selection and provide an in–depth analysis of the problem of simultaneously selecting a learning algorithm and setting its hyperparameters. In the work (Thornton et al., 2012) researchers provide a tool that effectively identifies machine learning algorithms and hyperparameter settings. Proposed approaches still require high computational resources to evaluate each model. Most feature selection studies are conducted in a non-automatic way or in semi-automatic way. This fails to explore all possible features, attributes and algorithms.

In the next chapter we will present a methodology to build effective automatic feature and algorithms selection model. These include testing platform performance against several types of classification algorithms, training datasets and document representation method.

Thus, problem solution has to be designed considering all defined features. We propose optimization model that depends on (i) the quality of the sample sets, (ii) on classification algorithm hyperparameter and (iii) on the document representation (text preprocessing). In this context, the complexity of the task comes from the size of feature space and computing resources needed to explore all domain. We do note the past work (Bergstra et al., 2011; Luo, 2016; Friedrichs and Igel, 2005; Snoek et al., 2012) that discuss more the theoretical aspects of optimization, presenting algorithms, but not concrete implementations on a distributed computing architecture.

We also note the recent work that gives an extensive analysis of the domain (Schaer et al., 2016). This work shows that HPC tools and frameworks available nowadays does not fit following requirments: (i) provide a full simulation of optimization process, (ii) address hyperparameter optimization directly and (iii) provide implementations for classification algorithms. As a result of the work (Schaer et al., 2016), distributed implementation of hyperparameter optimization for medical image classification was developed. Nevertheless, the same problem in the text classification domain remains open. Thus, in the chapter 4 we introduce MapReduce and MPJ hybrid architecture of the fully automatic optmimization algorithm. In the chapter 3 we provide implementation details as news classification case study.

Presented arguments lets one conclude that the effectiveness of classification algorithm depends on the following parameters: (i) ML algorithm itself, (ii) training data set and (iii) document representation (text pre-processing). Most ML algorithms further expose hyperparameters, which change the way the learning algorithm itself works. Hyperparameters express higher-level properties of the model such as its convergence, tresholds or how fast it should learn. Hyperparameters are usually fixed before the actual training process begins. The complexity of the task comes from the size of: (i) algorithm hyperparameter space, (ii) size of data set, (iii) high dimensionality of document feature space and (iv) computing resources needed to explore this. The relevance of studying such phenomena is confirmed by time costs of setting up and evaluating each model, as well as by high ownership costs of conventional high-performance infrastructures. This suggests a challenge for ML and HPC domains: given a dataset, to automatically and simultaneously choose a document representation feature space, learning algorithm and set its hyperparameters to optimize effectiveness.

In the next sections we will justify research motivation and give basic research background to serve the bridge for the reader of the paper.

3 METHODS

Text classification (TC) is the task of assigning a Boolean value to each pair $\langle d_j, c_i \rangle \in D \times C$, where D is a domain of documents, $C = \{c_1, \ldots, c_{|C|}\}$ is a set of predefined categories and $|C|$ is the cardinality of the set C. We will assume that the categories are just symbolic labels, and no additional knowledge (of a procedural or declarative nature) of their meaning is available. Classifier $\varphi_\lambda : D \times C \to \{T, F\}$ approximates unknown target function $\hat{\varphi} : D \times C \to \{T, F\}$, where λ is algorithm hyperparameter and $\lambda \in \Lambda$. Suppose classifier has n hyperparameters $\{\lambda_1, \ldots, \lambda_n\}$ in the domain $\Lambda_1, \ldots, \Lambda_n$. The hyperparamerer set Λ is subset of the vector product: $\Lambda \subset \Lambda_1 \times \cdots \times \Lambda_n$.

More formally, the task is to approximate the unknown target function $\hat{\varphi} : D \times C \to \{T, F\}$ by means of a function $\varphi : D \times C \to \{T, F\}$ called the classifier. Then the goal is to minimize the value of $|\varphi - \hat{\varphi}|$, how to precisely define and measure this value (called effectiveness) will be discussed in section 3.1.

There is no clear guideline to choose a set of learning methods and it is rare when one has a complete knowledge about data distribution and also the about the details of how the classification algorithm behaves. Therefore, in practical pattern classification tasks it is difficult to find a good single classifier. Classifier ensembles are based on the idea that k classifiers may be better than one if their individual judgments are appropriately combined. In TC, the idea is to apply k different classifiers $\varphi_1, \ldots, \varphi_k$ to the same task of deciding whether $d_j \in c_i$, and then combine their decision. A classifier committee is then characterized as simplest majority voting (MV), whereby the binary outputs of the k classifiers are pooled together, and the classification decision that reaches the majority of $\frac{k+1}{2}$ votes is taken, where k needs to be an odd number. In the rest of the thesis we will deal with the idea of majority voting classifier ensembles.

The combination of expert opinions is a topic studied since the second half of the twentieth century. In the beginning the studies were devoted to applications such as democracy, economic and military decisions. Multiple classifier combination methods can be considered some of the most robust and accurate learning approaches (Jr., 2011). The fields of multiple classifier systems and ensemble learning developed various procedures to train a set of learning machines and combine their outputs. Such methods have been successfully applied to a wide range of real problems, and are often, but not exclusively, used to improve the performance of unstable or weak classifiers. It is known from the ML literature that, in order to guarantee good effectiveness, the classifiers forming the committee should be as independent as possible (Kagan and Joydeep, 1996).

Classifier ensemble is a set of learning algorithms which decisions are combined to improve effectiveness of the pattern recognition system. Much of the efforts in classifier combination research focus on improving the accuracy of difficult problems, managing weaknesses and strenghts of each model in order to give the best possible decision taking into account all the ensemble. The use of combination of multiple classifiers was demonstrated to be effective, under some conditions, for several pattern recognition applications and scenarios were also experimentally and theoretically studied in the past years (Brown and Kuncheva, 2010).

Many studies showed that classification problems are often more accurate when using combination of classifiers rather than an individual base learner (Sebastiani, 2002). For instance "weak" classifiers are capable of outperform a highly specific classifier (Kuncheva et al., 2001). These methods were widely explored, for example, to stabilize results of random classifiers and to improve the performance of weak ones. Neural-network based classifiers that are unstable can be stabilized using MCS techniques (Breve et al., 2007). Also, noisy data can be better handled since the diversity of classifiers included in the ensemble increases the robustness of the decisions. Besides, there are many classifiers with potential to improve both accuracy and speed when used in ensembles. All these advantages can be explored by researches on the field of pattern recognition and machine learning. The formal definition of the ensemble of classifiers is given below.

The **classifier ensemble** is defined as $\bar{\varphi} = \langle\varphi_\lambda^{(1)},\dots\varphi_\lambda^{(k)}\rangle$, where k is the number of classification algorithms in the ensemble. The decision to assign a Boolean value is taken on the basis of a simple majority $\frac{k+1}{2}$, where k is always odd. The set of all ensembles is defined as $\bar{\Phi}$.

Given a set of classification algorithms $\Phi = \{\varphi_\lambda^{(1)},\dots,\varphi_\lambda^{(|\Phi|)}\}$, appropriate hyperparameter space $\Lambda^{(1)}\dots\Lambda^{(|\Phi|)}$ and the set of ensembles of classifiers $\bar{\Phi} = \{\bar{\varphi}^{(1)}\dots\bar{\varphi}^{(|\bar{\Phi}|)}\}$. Then the optimization of the classification algorithm is determined by the formula 1.1, and the optimization of the hyperparameters of the ensemble of classification algorithms according to the formula 1.2:

$$\varphi_{\lambda^*}^* = \operatorname*{argmax}_{\bar{\varphi}^{(j)}\in\Phi,\lambda\in\Lambda^{(j)}} E(\varphi_\lambda^j) \tag{1}$$

$$\bar{\varphi}^* = \operatorname*{argmax}_{\bar{\varphi}^{(j)}\in\bar{\Phi}} E(\bar{\varphi}^j) \tag{2}$$

, where $E(\ )$ the function of evaluating the accuracy of the classifier.

3.1 The Effectiveness of Classifier

The evaluation of document classifiers is typically conducted experimentally, rather than analytically (Manning et al., 2008). The experimental evaluation of a classifier usually measures its effectiveness, that is, its ability to take the right classification decisions. TC effectiveness is usually measured in terms of the classic IR notions of precision (P) and recall (R) (Manning et al., 2008). These are first defined for the simple case where an TC algorithm returns a set of classified documents.

These probabilities may be estimated in terms of the contingency table for c_i on a given test set. Here, FP (false positives, a.k.a. errors of commission) is the number of test documents incorrectly classified under c_i; TN (true negatives), TP (true positives), and FN (false negatives, a.k.a. errors of omission) are defined accordingly.

Precision P for a class c_i is the number of TP divided by the total number of documents labeled as belonging to the c_i.

$$Precision = \frac{TP}{TP+FP} = P \tag{3}$$

Recall (R) for a class c_i in this context is defined as the number of TP divided by the total number of documents that **actually** belong to the class c_i

$$Recall = \frac{TP}{TP+FN} = R \tag{4}$$

Our final goal is to evaluate the classifiers relative to each other, rather than simply assess the effectiveness of a single classifier. An extensive analysis of various evalution metrics specifically for TC domain based on contingency table, P and R may be found in (Carletta, 1996; Fawcett, 2006; Manning et al., 2008; Powers, 2008; Powers, 2012). Research (Carletta, 1996) states that the kappa statistic measure are easily interpretable and allows different classifiers results to be compared. The work (Manning et al., 2008) states that different systems or variants of a single system which are being compared for effectiveness have little impact on the relative effectiveness ranking using kappa statistics. The kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves.

In the (Powers, 2008; Powers, 2012) authors states that Kappa does attempt to renormalize a debiased estimate of Accuracy, and is thus much more meaningful than Recall, Precision, Accuracy, and their biased derivatives. But they also assume that it is intrinsically non-linear, doesn't account for error well, and retains an influence of bias. Furthermore, in the (Powers, 2012) authors suggests Matthews correlation coefficient (MCC) instead of Kappa saying that for comparison of behaviour, Matthews Correlation is recommended. Matthews Correlation coefficient is appropriate for comparing equally valid classifications or ratings into an agreed number of classes.

Thus, in our work we use Kappa Cohen's Coefficient (*Kappa* 11) and Matthews correlation coefficient (*MCC* 10) as a classifier evaluation metrics. We will also show standard F1 score (F_1 8), Accuracy (*ACC* 6), Precision (*P* 3), Recall (*R* 4) and Error Rate (*ER* 5) because it is often used as an evalutaion metrics in most text classification researches and it could be used by other researchers as a benchmark. The following shows the formulas for calculation the coefficients.

$$\mathrm{ER} = \frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}} \tag{5}$$

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{6}$$

$$\mathrm{EACC} = \frac{\frac{(\mathrm{TP+TN})*(\mathrm{TP+FP})}{\mathrm{TP+TN+FP+FN}} + \frac{(\mathrm{FP+TN})*(\mathrm{FN+TN})}{\mathrm{TP+TN+FP+FN}}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \tag{7}$$

$$F_1 = 2\cdot\frac{\mathrm{PPV}\cdot\mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \tag{8}$$

$$\mathrm{Kappa} = \frac{ACC - EACC}{1 - EACC} \tag{9}$$

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \tag{10}$$

$$\text{Kappa} = \frac{ACC - EACC}{1 - EACC} \quad (11)$$

where

- *IDF* is inverse document frequency
- $f(S_i, R)$ is occurrence frequency of a word i from sentence S in sentence R
- $|R|$ is a length of sentence R
- *avgDL* is an average length of sentences in the document
- $k1$ and b are parameters, $k1 = 1.2$, $b = 0.75$

3.2 The Feature Space

The first feature is classifier Λ_1. Each classifier has hyperparameters that affect the classification effectiveness. In this work, we consider the well-known classifiers K-Nearest Neighbors (KNN) and Naive Bayes (NB) where K (from KNN) and -priori probabilities (from NB) are hyperparameters that influence accuracy. Our experiments have shown that KNN classifiers accuracy dramatically low for $K > 200$, that is why we decide to bound K from 1 to 200. We have a set of hyperparameters features C = (0-200), where values $(1-200)$ mapped to K in KNN classifier and value 0 mapped to NB classifier. We have a set Λ_1 of 201 different features:

$$\Lambda_1 = [0, 1, 2...199, 200] \quad (12)$$

The second feature is pp_3 - an input data text feature selection by extracting n-grams. For our research, we choose 1-gram, 2-gram, 3-gram, 4-gram, 5-gram and altogether from 1 to 5 gram extraction. As a result, we have a set Λ_2 of 6 different features:

$$\Lambda_2 = \{1Gram, 2Gram, 3Gram, 4Gram, 5Gram, 1to5Gram\} \quad (13)$$

The third features that affects classifier effectiveness is an input data pre-processing. The goal of data pre-processing (pp) is to clean and prepare the text for classification and the whole process is as pipeline with several stages, namely: (pp_1) stemming, (pp_2) short words filtering and (pp_3) n-gram extraction. We consider that all input text is already pipelined through text cleaning and stop words stages.

Thereby, we have a set Λ_3 consisting of pp_1 and pp_2 operations. On pp_1 stage we decide whether input text will be processed with stemmer (S) or not (NS). On stage pp_2 we decide whether input text will be processed with word length filter (LF) or not (NLF). Length filter is dropping out the word that is shorter than four symbols. There are combinations of features pp_1 and pp_2, for example, if we want to apply both stemming and short word filtering. As a result, we have a set Λ_3 of 4 different features:

$$\Lambda_3 = \{S + LF, S + NLF, SN + LF, SN + NLF\} \quad (14)$$

The next is data set feature, for this research we use different document text fields Λ_4. As there are two fields available, title and body, we have a set, Λ_4, of three different features:

$$\Lambda_4 = \{body, title, body + title\} \quad (15)$$

Totally, we have a combination of 24 features text pre-processing features, 201 classifiers hyperparemeter features and 3 data set features. More formally we have space φ_λ, which is space of 14472 different combinations of all features. By combining all features, we assemble classifier. Assembled classifier is a combination φ_λ that can be expressed using formula as follows:

$$\varphi_\lambda = \{\lambda_i, \lambda_j, \lambda_k, \lambda_l\} \quad (16)$$

where,

$$i \in [0-200], \lambda_i \in \Lambda_1, \quad (17)$$

$$j \in [0-6], \lambda_j \in \Lambda_2, \quad (18)$$

$$k \in [0-3], \lambda_k \in \Lambda_3, \quad (19)$$

$$l \in [0-3], \lambda_l \in \Lambda_4 \quad (20)$$

We propose tuple notation to describe each assembled classifier, it is shown in figure 1.

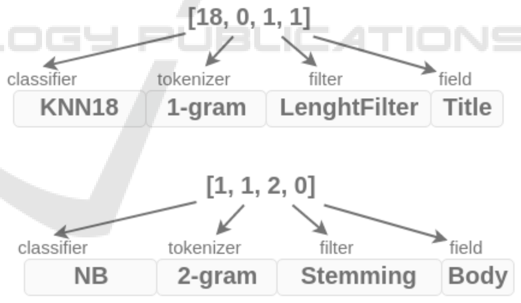

Figure 1: Tuple notation describes each classifier with all features.

4 ENSEMBLE OF CLASSIFIERS EFFECTIVENESS OPTIMIZATION ARCHITECTURE

We built a classifiers ensemble based on work (Bauer and Kohavi, 1999) to combine classifiers. To make a decision we need a simple majority of voters. Suppose we have five voters, then combination of all possible classifiers would be the combination of given

14472 set of classifiers of 5 elements. Which is $5 * 10^{18}$ possible combinations. We use a classical genetic algorithm to deal with this complexity. The chromosome will consist of 5 genes, where the gene is one of the combinations of the classifier, analyzer and field. By assigning each characteristic to a number, you can define a set of genes. Thus, the gene is an array of 4 digits, where each of the digits is responsible for the characteristics of the classifier. An example of chromosome shown in Figure 2.

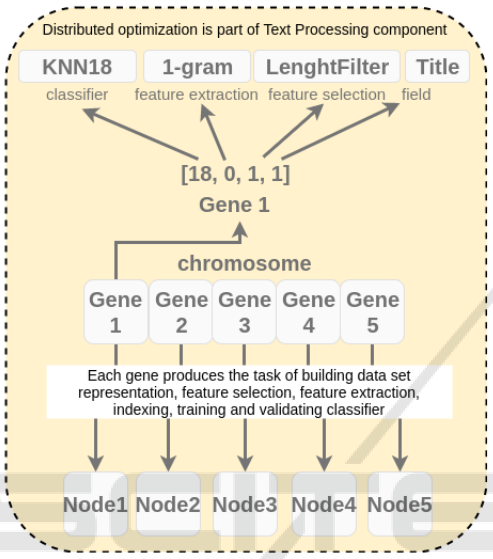
Distributed optimization is part of Text Processing component
KNN18
classifier
1-gram
feature extraction
LenghtFilter
feature selection
Title
field
[18, 0, 1, 1]
Gene 1
chromosome
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Each gene produces the task of building data set representation, feature selection, feature extraction, indexing, training and validating classifier
Node1
Node2
Node3
Node4
Node5

Figure 2: An example of MapReduce processing of one chromosome for voting classifier.

Each generation chromosome distributed using MPJ to the slave nodes. Each slave nodes distributes their task to the CPU cores using MapReduce Java implementation. Slave nodes responsible for document representation, feature selection, feature extraction, indexing, training and validating classifier. In the end all slaves sends back K-fold cross validation results to master Node, which is producing next generation of chromosomes. Generation algorithm schema is shown on figures 3.

We will use the value of the Kappa coefficient as a fitness function. The stopping criteria for the genetic algorithm is set to $Kappa = 0.95$. Selection will occur using the roulette selection method. This method increases the probability of selecting an individual with a high fitness function. The higher the Kappa coefficient, the more likely it is to get into the next generation. Optimization schema is shown in figure 4.

We use simple One-Point Crossover as a crossover function. The point at which the crossing will occur selected randomly. The mutation of each chromosome triggered with a 35% chance. This means

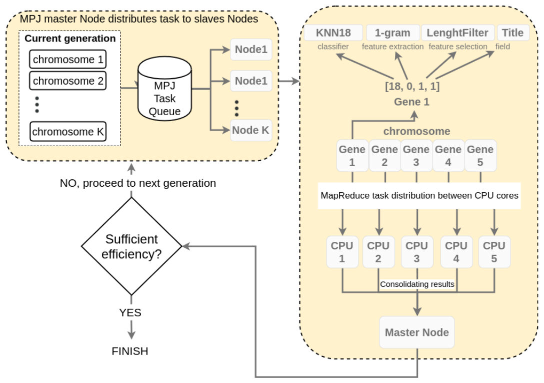

Figure 3: MPJ distributes chromosomes to slave nodes. Java based MapReduce application computes classifier effectiveness.

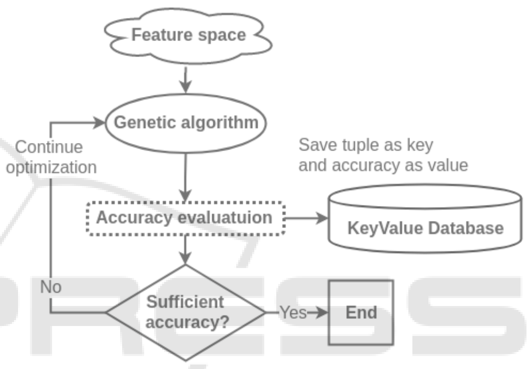

Figure 4: Distributed optimization schema workflow.

that after each cycle of the algorithm, only 35% of the population can mutate, and only 2 of the 5 genes will mutate.

This simple genetic algorithm produces well-stratified data to train prediction model. Figure 4 shows that we save each tuple as a key and accuracy as the value. We execute algorithm until it generates enough results to approximate accuracy function.

Proposed voting classifier description is a vector in an N-dimensional space, where N is a number of features of the classifier. Nearly any regression model can be used to approximate relationship between features and accuracy. As proof of concept, we built multilayer perceptron (MLP) with one hidden layer in order to predict voting classifier accuracy without computation, training and testing classifier. The prediction-learning schema is in figure 5.

Multilayer perceptron was trained using data that genetic algorithm produced. In order to archive maximum performance, we implement a genetic algorithm in a distributed computing manner using hybrid of Java Message Passing Interface (MPJ-Express) and MapReduce paradigm.

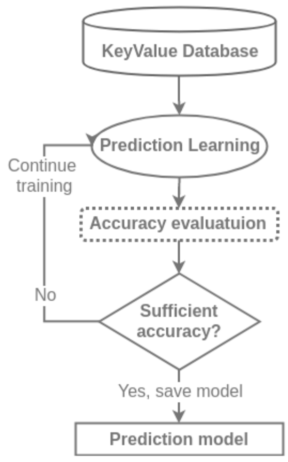

Figure 5: Process of training regression model.

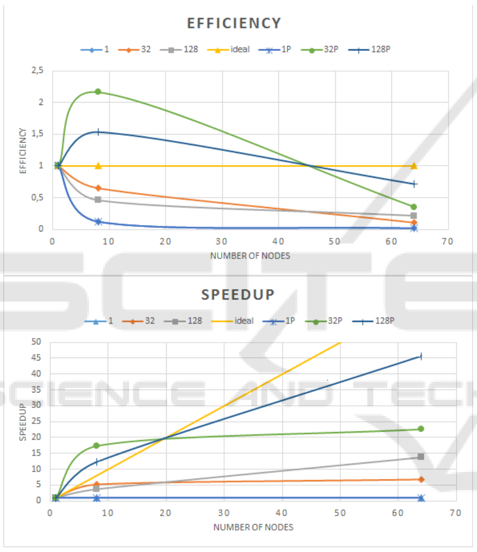

Figure 6: Speedup ratio and parallel efficiency. Plots with prefix “P” shows computation using prediction model.

5 RESULTS

Technical characteristics of the cluster is shown in tables 1 and 2. There are 16 nodes, each node has the same characteristics. Figure 6 shows overall picture for results of the experiments, parallelization gives good efficiency and speedup on all platforms.

5.1 Speedup and Efficiency

We experimented on the task of distributed classifier ensemble effectiveness optimization. The code of classifier committee optimization using genetic algorithm described in previous section. Experimental result shows that the feature selection, feature extraction, data set and classifier hyperparameter has deep influence to the effectiveness of classifier.

The speedup ratio and parallel efficiency plots are shown in figure 6. Speedup and efficiency provide an estimate for how well a code sped up if it was parallelized. The yellow plot shows ideal speedup and efficiency, the ratio of one hundred percent parallelized code.

The green (32P) and blue (128P) plot stand for 32 and 128 classifiers evaluation respectively. The prefix “P” means that optimization process uses prediction model in order to drop out classifiers with insufficient accuracy. This helps to identify weak classifier before implicit training, testing and evaluating. That is why plots sometimes higher than an ideal line.

The proposed workflow finds the voting classifier with the maximum Kappa coefficient of 0.85. The resulted program output is shown in figure 7, chromosome vector representation is shown in listing 1. The effectivness is not perfect, but according to (Landis and Koch, 1977) almost perfect agreement. In addition, the goal of MLP was to predict accuracy and avoid calculation of weak voting classifier. We interpret these results as encouraging evidence for the usefulness of MLP for deciding usefulness of voting classifier.

Table 1: Cluster specification.

CPU	RAM	HDD	Net
Intel Core i5-2500 3.30GHz	16Gb	500GB 7200RPM 6Gb/s	1Gbit/s

Table 2: Software specification.

Name	Version
MPJ Express	0.44
Apache Hadoop	2.6.0
Java	1.8.0_131
Ubuntu OS	14.04

```
  1. 1-GRAM STEMMING_AND_REMOVE_SHORT          NB    ONLY_BODY
  2. 1-GRAM STEMMING_AND_NOT_REMOVE_SHORT      KNN19 ONLY_BODY
  3. 1-GRAM NO_STEMMING_AND_REMOVE_SHORT       KNN29 ONLY_BODY
  4. 2-GRAM NO_STEMMING_AND_REMOVE_SHORT       KNN34 ONLY_BODY
  5. 3-GRAM NO_STEMMING_AND_NOT_REMOVE_SHORT   NB    ONLY_BODY

        Confusion matrix:
                C0      C1
        C0      786     154
        C1      61      4499

        Observed Accuracy:  0.96090
        Expected Accuracy:  0.72745
        Error rate:         0.03909
        Kappa statistics:   0.85657
        Precision:          0.83617
        Recall:             0.92798
        F measure:          0.87968
        MCC:                0.85808
        Total Docs #        5500
==      ==
```

Figure 7: Confusion matrix of the ensemble of classifiers with the Kappa and MCC of 0.85.

Listing 1: Best chromosome vector representation in JSON file format.

```
{
  "chromosome": {
    "genes": [
      {
        "classifier": 0,
        "analyzer": 0,
        "filter": 0,
        "field": 0
      },
      {
        "classifier": 19,
        "analyzer": 0,
        "filter": 1,
        "field": 0
      },
      {
        "classifier": 29,
        "analyzer": 0,
        "filter": 2,
        "field": 0
      },
      {
        "classifier": 34,
        "analyzer": 1,
        "filter": 2,
        "field": 0
      },
      {
        "classifier": 0,
        "analyzer": 2,
        "filter": 3,
        "field": 0
      }
    ]
  },
  "fitnessValue": 0.8565710473649101
}
```

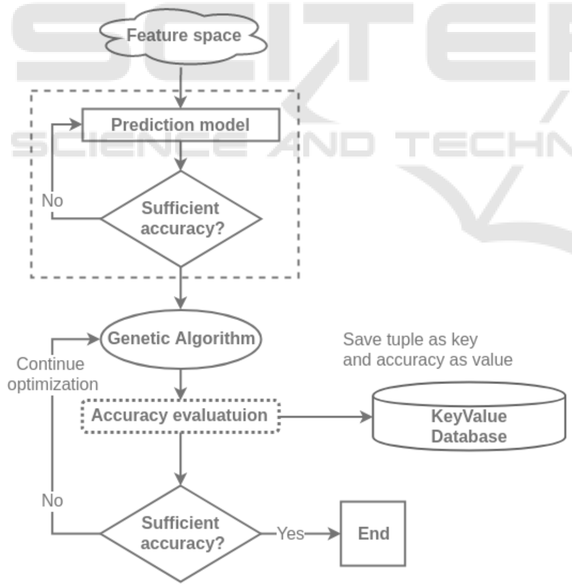

Figure 8: Final optimization workflow. The dotted line highlights the prediction model.

6 CONCLUSION

It is difficult nowadays to decide which classification algorithm to use and how to preprocess text input data. We design a workflow of algorithms that can significantly reduce the amount of time to find out correct attributes of the exact problem. Figure 8 shows proposed workflow of algorithms.

The proposed simulation tool is very effective and has accuracy up to 0.85 value of Kappa and MCC coefficient. Furthermore, the accuracy of the whole workflow can be improved by selecting better approximation model, for example, better MLP architecture.

7 COPYRIGHT FORM

The Author hereby grants to the publisher, i.e. Science and Technology Publications, (SCITEPRESS) Lda Consent to Publish and Transfer this Contribution.

REFERENCES

Bauer, E. and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1):105–139.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc.

Breve, F. A., Ponti-Junior, M. P., and Mascarenhas, N. D. A. (2007). Multilayer perceptron classifier combination for identification of materials on noisy soil science multispectral images. In *XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007)*, pages 239–244.

Brown, G. and Kuncheva, L. I. (2010). *"Good" and "Bad" Diversity in Majority Vote Ensembles*, pages 124–133. Springer Berlin Heidelberg, Berlin, Heidelberg.

Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Comput. Linguist.*, 22(2):249–254.

Dasgupta, A., Drineas, P., Harb, B., Josifovski, V., and Mahoney, M. W. (2007). Feature selection methods for text classification. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 230–239, New York, NY, USA. ACM.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305.

Friedrichs, F. and Igel, C. (2005). Evolutionary tuning of multiple svm parameters. *Neurocomput.*, 64:107–117.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). *Sequential Model-Based Optimization for General Algorithm Configuration*, pages 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg.

Jr., M. P. P. (2011). Combining classifiers: From the creation of ensembles to the decision fusion. In *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*, pages 1–10.

Kagan, T. and Joydeep, G. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–404.

Kuncheva, L. I., Bezdek, J. C., and Duin, R. P. W. (2001). Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34:299–314.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1).

Lim, T.-S., Loh, W.-Y., and Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228.

Luo, G. (2016). Predict-ml: a tool for automating machine learning model building with big clinical data. *Health Information Science and Systems*, 4(1):5.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Polikar, R. (2017). Ensemble learning.

Powers, D. (2008). Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2.

Powers, D. M. W. (2012). The problem with kappa. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 345–355, Stroudsburg, PA, USA. Association for Computational Linguistics.

Schaer, R., Muller, H., and Depeursinge, A. (2016). Optimized distributed hyperparameter search and simulation for lung texture classification in ct using hadoop. *J. Imaging*, 2:19.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2951–2959, USA. Curran Associates Inc.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2012). Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *CoRR*, abs/1208.3719.