DOI: https://doi.org/10.15276/hait.07.2024.15
UDC 004.91

Optimizing hierarchical classifiers with parameter tuning and confidence scoring

Sergii V. Mashtalir[1)]
ORCID: https://orcid.org/0000-0002-0917-6622; sergii.mashtalir@nure.ua. Scopus Author ID: 36183980100
Oleksandr V. Nikolenko[2)]
ORCID: https://orcid.org/0000-0002-6422-7824; oleksandr.nikolenko@uzhnu.edu.ua
[1)] Kharkiv National University of Radio Electronics, 14, Nauky Ave. Kharkiv, 61166, Ukraine
[2)] Uzhhorod National University, 14, University Str. Uzhhorod, 88000, Ukraine

ABSTRACT

Hierarchical classifiers play a crucial role in addressing complex classification tasks by breaking them down into smaller, more manageable sub-tasks. This paper continues a series of works, focused on the technical Ukrainian texts hierarchical classification, specifically the classification of repair works and spare parts used in automobile maintenance and servicing. We tackle the challenges posed by multilingual data inputs – specifically Ukrainian, Russian, and their hybrid – and the lack of standard data cleaning models for the Ukrainian language. We developed a novel classification algorithm, which employs TF-IDF victimization with unigrams and bigrams, keyword selection, and cosine similarity for classification. This paper describes a method for training and evaluating a hierarchical classification model using parameter tuning for each node in a tree structure. The training process involves initializing weights for tokens in the class tree nodes and input strings, followed by iterative parameter tuning to optimize classification accuracy. Initial weights are assigned based on predefined rules, and the iterative process adjusts these weights to achieve optimal performance. The paper also addresses the challenge of interpreting multiple confidence scores from the classification process, proposing a machine learning approach using Scikit-learn's GradientBoostingClassifier to calculate a unified confidence score. This score helps assess the classification reliability, particularly for unlabeled data, by transforming input values, generating polynomial parameters, and using logarithmic transformations and scaling. The classifier is fine-tuned using hyper parameter optimization techniques, and the final model provides a robust confidence score for classification tasks, enabling the verification and classification results optimization across large datasets. Our experimental results demonstrate significant improvements in classification performance. Overall classification accuracy nearly doubled after training, reaching 92.38 %. This research not only advances the theoretical framework of hierarchical classifiers but also provides practical solutions for processing large-scale, unlabeled datasets in the automotive industry. The developed methodology can enhance various applications, including automated customer support systems, predictive maintenance, and decision-making processes for stakeholders like insurance companies and service centers. Future work will extend this approach to more complex tasks, such as extracting and classifying information from extensive text sources like telephone call transcriptions.

Keywords: Natural language processing; tree-based classification; machine learning; data analysis; applied intelligent systems

For citation: **Mashtalir S. V., Nikolenko O. V. "Optimizing hierarchical classifiers with parameter tuning and confidence scoring".** ***Herald of Advanced Information Technology.*** **2024; Vol. 7 No. 3: 231–242. DOI: https://doi.org/10.15276/hait.07.2024.15**

1. INTRODUCTION AND LITERATURE REVIEW

In the machine learning domain, hierarchical classifiers have become a crucial method for addressing complex classification challenges [1]. These classifiers organize the classification task hierarchically, simplifying a broad, multi-class problem into smaller, more digestible sub-tasks. This method reflects human cognitive decision-making processes [2], which are sequential and structured, making it particularly apt for areas with inherent hierarchical structures, such as taxonomy classification [3], image recognition [4] and [5], medical diagnosis [6], autonomous systems [7], and document categorization [8] and [9].

A perspective area of research in hierarchical classifiers focuses on optimization, specifically on parameter tuning and confidence estimation. Parameter tuning adjusts the model's parameters to improve performance, while confidence scoring quantifies the classifier's predictions certainty, both critical for enhancing the hierarchical classification systems accuracy and dependability [10].

Parameter tuning in hierarchical classifiers is notably more complex than in non-hierarchical (flat) classifiers due to the layered decision-making process [11]. Each layer may require distinct parameters, and the dependencies between these layers must be meticulously managed. On the other hand, confidence scoring addresses the necessity for dependable predictions. In practical applications, it is vital not only to understand the classifier's

predictions but also the confidence level in these predictions. This is especially crucial in hierarchical classifiers, where errors can cascade through the hierarchy.

An intriguing challenge arises when hierarchical classification is combined with natural language processing (NLP) [12].

Over the past few years, NLP has undergone significant advancements, largely fueled by developments in machine learning and deep learning. Nevertheless, NLP systems often struggle with issues related to incomplete or erroneous training data [13], which can lead to skewed model predictions and impact the trustworthiness of the results. Effective data preparation and validation are essential for addressing these concerns and ensuring the robustness of NLP models.

Natural Language Processing encounters significant challenges when applied to technical texts due to the distinctive characteristics inherent to these domains, such as specialized vocabulary, complex syntactic structures, and heightened context dependency [14]. Technical documents often employ domain-specific jargon, abbreviations, and terminology not represented in general-purpose language models, leading to increased risk of semantic misinterpretation. Furthermore, technical writing typically exhibits intricate sentence structures, with multiple clauses and interdependencies, which complicate syntactic parsing and semantic analysis.

These difficulties are compounded by the scarcity of annotated domain-specific corpora and the limited ability of NLP models to generalize across different technical fields. Each domain possesses unique terminologies and structural patterns, necessitating domain-specific adaptation and expert-annotated datasets, which are both time- and resource-intensive to develop. These issues are particularly pronounced for less-represented languages [15], where the availability of linguistic resources is limited, restricting the development of multilingual models.

This work is the third in a series [16] and [17] focused on the technical Ukrainian texts hierarchical classification. The primary issue addressed in these studies is the classification of repair works and spare parts used in the automobiles' maintenance and servicing.

2. THE GOAL OF PAPER

The goal of this study is to develop a model for the hierarchical classification of technical texts in Ukrainian and Russian and to identify effective optimization approaches through parameter tuning. Additionally, the research seeks to evaluate the classification quality by calculating key confidence metrics, which will enhance the model's efficiency and reliability.

The paper explores a novel method for calculating confidence scores, utilizing probability estimation and ensemble techniques, enabling the classifier to deliver more detailed outputs crucial for decision-making processes in applications such as automotive quality assurance.

The interplay between parameter tuning and confidence scoring forms this paper's central theme. Effective parameter tuning improves the classifier fundamental performance, while precise confidence scoring ensures the predictions reliability. Integrating these two elements fosters a more robust and dependable hierarchical classification system.

Through extensive experimentation and case studies, this paper demonstrates the proposed methods practical advantages. It highlights improvements in classification accuracy and confidence calibration across various datasets in the automotive repairs domain. The results emphasize the comprehensive model training significance that adequately incorporates both parameter tuning and confidence scoring.

3. PROBLEM DEFINITION

3.1. Automotive sector issues

The significance of repair works and spare parts classification cannot be overstated. Out of approximately 1.5 billion vehicles globally, only 20% have detailed, centrally collected, stored, and analyzed repair information [18] and [19]. This data pertains primarily to new vehicles, up to four years old, which are serviced at authorized service centers. Automotive manufacturers utilize specialized software in the original equipment service centers (OES), where all repairs and spare parts are accurately classified. This allows for the collection of precise statistical data on the individual components and assemblies reliability, their operational characteristics, warranty cases, and more [20].

Unfortunately, once a vehicle leaves the official service network, its subsequent repair and maintenance history becomes fragmented and often lost. Non-authorized service centers lack a unified classifier for repairs and spare parts, let alone a single information system. Furthermore, there are thousands of such systems worldwide, each with different languages and data formats. In Ukraine alone, dozens of similar programs are used.

Fig. 1 illustrates the automotive fleet structure and highlights the data problem concerning repairs.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

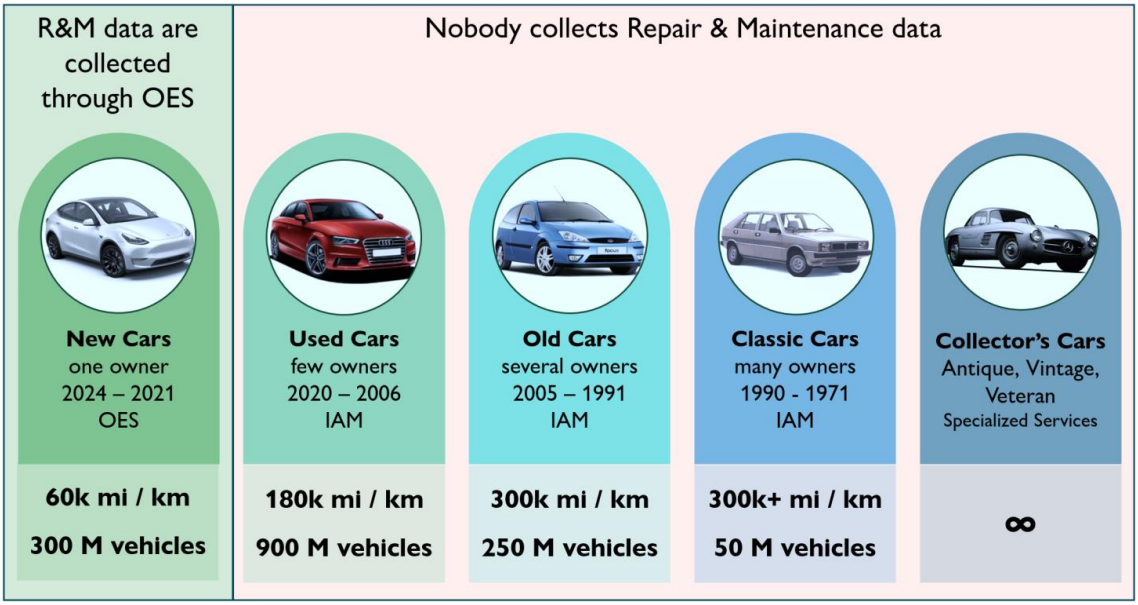

Fig. 1. **Global car park structure**
Source: **compiled by authors**

Repair data is valuable not only to automotive manufacturers, but also to various other sectors. For example, insurance companies can benefit from this data to determine repair costs and residual vehicle values. Similarly, service centers and even car owners would find it beneficial to have information not only about the current repair costs but also about future expenses related to repair and maintenance.

3.2. Data collection and data structure

The input data used for this study comprised information on repair works and spare parts from a Garage Management System (GMS).

This data was divided into four sets:

1) Classes: hierarchical structures for repair works (3 levels) and car parts (4 levels).

2) Training Data: manually labeled data from the GMS, consisting of 6,000 entries.

3) Test Data: also manually labeled data from the GMS, consisting of 11,000 entries.

4) Operational or Input Data: comprising tens of millions of entries from the GMS.

Let us examine these types in greater detail.

Car repair works are organized in a three-level tree structure as shown in the data extract in Table 1.

Car components are classified within a hierarchical four-level tree structure. The classification begins with the most general component types, such as mechanical and body parts, oils and fluids, wheels and tires. These broad categories are then divided into their corresponding systems, including filters, power transmission, braking, suspension, steering, engine, cooling, electric and electronic systems.

Within each system, the classification is further refined into specific components. For example, the suspension is divided into subcategories such as damping, arms, wheel hubs, bearings, etc. Finally, the lowest classification tier consists of specific spare parts detailed lists, such as shock absorbers, struts, coil and leaf springs. The car parts data tree extract is presented in Table 2.

Table 1. **Repair works classes' data tree extract**

ID	Parent ID	UA	EN
1000	0	Діагностичні роботи	Diagnostic work
1100	1000	Діагностика	Diagnostics
1101	1100	Ручна діагностика	Manual diagnostics
1102	1100	Комп'ютерна діагностика	Computer diagnostics
1200	1000	Тестування	Testing
2000	0	Загальні роботи	General works
2100	2000	Заміна	Replacement

Source: **compiled by authors**

Table 2. **Car parts classes' data tree extract**

ID	Parent ID	UA	EN
1000000	0	Механічні деталі	Mechanical parts
1070000	1000000	Амортизація	Suspension damping
1070100	1070000	Амортизатори і стійки	Shock absorbers & struts
1070101	1070100	Амортизаторі підвіски	Shock absorbers
1070102	1070100	Стійки підвіски	Struts
1070105	1070100	Пневмо-амортизатори	Pneumatic shocks
1070300	1070000	Опори амортизаторів	Strut mountings
1070400	1070000	Пружини підвіски	Coil & leaf springs

Source: **compiled by authors**

The training and test datasets comprise combined repair works and parts lists. For example, an entry might be "Pneumatic damping diagnostics on shock-tester". These lists have been manually labeled specifically for this study.

Operational or input data consists of arbitrary text, which may include information from garage management systems (GMS), phone calls transcriptions, messages from messengers, emails, etc. The objective is to determine whether the input text contains information related to car repairs and to correctly assign it to one of the predefined classes for works and parts.

A classification is considered successful if:

1) At least 90 % of car repair works are identified and extracted from the incoming unlabeled texts.

2) Of these, no less than 90% of works and parts are correctly assigned to their appropriate classes.

For example, the text "Pneumatic damping diagnostics on shock-tester" should be accurately classified into class 1102 for works and 1070105 for parts

For this study purposes, we simplify the task by assuming that the input text contains information about works and components. Therefore, only the second criterion of successful classification is considered. The task of extracting relevant information about repair works and automotive parts from arbitrary text will be addressed in future studies.

In summary, this series of works aims to address the critical issue of technical texts hierarchical classification, focusing on the automotive industry. By improving the classification and analysis of repair and maintenance data, we can enhance this information's reliability and accessibility for multiple stakeholders, ultimately contributing to better decision-making and resource management in the automotive sector.

4. CLASSIFICATION ALGORITHM

4.1. Data preprocessing

Our previous work [16] extensively examined the challenges of processing technical texts based on Ukrainian and Russian languages, including their hybrid form known as "surzhyk". Here, we briefly summarize the key points.

In datasets composed of manually input texts, we frequently encounter numerous errors and technical terms, often presented in a mixture of two languages. Moreover, standard data cleaning models are not always available for the Ukrainian language.

Therefore, we adapted the classical NLP approach as follows:

1) Language identification based on language-specific letters and terms.

2) Data normalization by removing unnecessary characters and excluding stopwords. The stopwords list was meticulously revised to avoid omitting important repair-related abbreviations, e.g., ТО – технічне обслуговування (technical maintenance).

3) Translation of all texts into Ukrainian, the primary language for our research. Simultaneously, a Russian-Ukrainian dictionary of terms was automatically compiled.

4) Tokenization and subsequent splitting of merged tokens into their constituents.

5) Automatic correction of grammatical errors based on the generated dictionary and existing tokens, utilizing the Jaro-Winkler metric for word matching.

6) Lemmatization using comprehensive online dictionaries of the Ukrainian language.

7) Separation of specific prefixes such as auto-, electro-, pneumatic-, etc.

8) Deciphering abbreviations and replacing synonyms.

As a result of these operations, the original parts classes' tree consisting of 10.288 sentences

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

was transformed into 6.062 tokens, which were then compressed into 2.484 tokens after applying the described sequence of steps. The proposed preprocessing methodology resulted in a 59 % reduction in dictionary size, thereby significantly accelerating data processing in all subsequent stages.

4.2. Classification

As described in our previous work [17] initial attempts to classify the data using standard algorithms like Naive Bayes, k-Nearest Neighbors (kNN), and logistic regression yielded unsatisfactory results, with the best performance being slightly above 80%.

To achieve the required classification accuracy, we developed our own classification algorithm.

The process involves:

1) Initialization and Full Name Construction: For each node, a full name is created by combining the node's name with its children's full names.

2) Vectorization using TF-IDF: Nodes are vectorized using the TF-IDF method to represent term importance, including both unigrams and bigrams.

The TF-IDF metric is calculated using the formula [21]:

$$TF - IDF = TF \times IDF$$
$$TF\ (t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

where $f_{t,d}$ is raw count of a term t in document d; $\sum_{t' \in d} f_{t',d}$ is number of words in document d.

$$IDF(t,D) = log\frac{N}{|\{d\ \in D : t\ \in d\}|},$$

where *N* is total number of documents in the corpus, $N = |D||\{d\ \in D : t\ \in d\}|$ is number of documents, where term t appears, i.e. $TF\ (t, d) \neq 0$.

3) Node Matrices: Constructing for each node unique matrices based on the TF-IDF child nodes vectors.

4) Keyword Selection: Keywords and super keywords are selected based on document frequency values. Keywords are unique to a class, while super keywords are names consisting of two words, weighted more heavily to aid classification accuracy.

5) Training Data Vectorization: Each training string is vectorized, creating matrices with TF-IDF values. Bigrams and their permutations enrich the feature set, accommodating different word orders.

This step is detailed in the chapter "5. Model Training and Evaluation" of this work.

5) Classification: Using cosine similarity to measure the distance between input strings and matrix rows and iteratively refining classification probabilities through training.

The cosine similarity metric between vectors *A* and *B* is calculated by the formula [22]:

$$S_C(A,B) = \cos(\theta) = \frac{AB}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}}.$$

Cosine similarity was chosen because it is a widely used similarity measure for real-valued vectors, which is especially important for parts classification. Additionally, cosine has the nice property that it is 1.0 for identical and 0.0 for orthogonal vectors [23].

7) Confidence scores calculation: The confidence score is the ratio of the probabilities of the most and second most likely classes. If no matches are found in any child classes, an arbitrarily high confidence value is assigned.

This step is detailed in Chapter "6. Unified Confidence Score for Labeled and Unlabeled Data" of this work.

5. MODEL TRAINING AND EVALUATION

Our prior research [17] delineated the comprehensive process of hierarchical classification for automotive works and parts. However, constraints on space precluded detailed discussions of the training and evaluation processes, despite their critical importance in the classification algorithm, which significantly enhanced the classification accuracy.

Additionally, the confidence scores computation and optimization were not thoroughly examined. Given that 99 % of our data was unlabeled, the confidence issue was especially pivotal in our study.

This paper addresses these omissions by providing a detailed training and scoring processes exposition in the subsequent two chapters.

Following the tree construction and the initialization of the vectorizer and classifier for each node, a parameter tuning method for the tree is initiated, which in turn launches, in several threads, the parameter tuning method within each node, acting as potential sub-classifiers of our tree.

Prior to the parameter tuning iterations commencement, several preparatory steps are undertaken:

1) A training dataset is initialized, upon which each parameter values set will be evaluated at each iteration. This set includes input data – vectorized names from the sets comprising the full_name of child nodes, as well as additional manually

annotated names (which are given greater weight), and the output data – the corresponding child nodes and annotations classes.

2) An initial parameter values set at the node is evaluated. The evaluation function launches a one-step training data classification on the node classifier and checks the accuracy percentage of the resulting classes against the true class values.

The training occurs through the weights (parameters) optimization for the node's matrix tokens in the class tree, as well as weights (parameters) for the input (training) vector.

Initially, all class node matrix elements are assigned preliminary weights according to the following rules:

- super keyword – 5.0, keyword – 2.5;
- direct child tokens – 2.0 (direct descendants tokens are given more attention than those of further descendants);
- bigrams – 1.5;
- the first token in the name – 1.5;
- adjectives – 0.5;
- others – 1.0.

The exact values for parameters during initialization are not critically important. What matters is that they are greater than 1 or less than 1, and subsequently, the iterative training algorithm will determine the optimal weights.

As with the matrices for class tree nodes, initial weights are determined for the input strings matrices. However, different rules apply here:

- bigrams – 1.5;
- the first token in the name – 1.5;
- tokens created from words in parentheses – 0.5;
- bigrams created from words on the edge of parentheses from both sides – 0.0.

Parameter tuning iterative cycle then commences. If it does not conclude within 20 iterations, it halts at the last result.

At each iteration, a parameter tuning step is performed:

1) for each parameter, its values are iterated from a possible values predefined set (for example, for most weights >1, parameters from 1 to 10 are iterated in steps of +0.5, and for weights < 1, parameters from 1 to 0 in steps of -0.05);

2) as we are changing parameter values, for each of the training rows, a re-initialization of the weighting parameters is pre-launched, as well as a re-vectorization of the input names (if the value of a parameter related to input vectorization was changed) or a re-initialization of the classifier (if parameters based on which the classifier vectors are built were changed);

3) these values are then evaluated on the training data – through classification and calculating Accuracy – the matches percentage between found and real classes;

4) the parameter and its value that achieve the maximum classification accuracy rating are selected;

5) a check for value update is performed:

- if a change in parameter value led to an increase in accuracy compared to the previous iteration, or accuracy remained the same but the parameter value became closer to 1 – update the node parameter values and proceed to the next iteration;
- if the parameter values iteration did not find a better value for any of the parameters – stop the cycle.

After completing the parameter tuning method on all nodes, the tree can be considered “trained” and used for further classification.

6. UNIFIED CONFIDENCE SCORE FOR LABELED AND UNLABELED DATA

Following classification, a pertinent question remains: how confident are we in its correctness? This is particularly relevant for unlabeled data, as well as for automated decision-making systems.

As noted, the classification result provides us with classes set along with their probabilities, and confidence scores for each node of the tree. The challenge arises in how to accurately interpret multiple confidence scores. Simple dimensionality reduction methods, such as arithmetic mean or root mean square, which might intuitively be considered, lose crucial information from the tree structure.

In other words, is it better to have confidence closer to the tree's roots or its leaves? Which set provides greater overall confidence, (1000, 0.1, 0.1) or (0.1, 0.1, 1000)? If we were classifying city names, moving through the tree from country to state/region to city, then the set (1000, 0.1, 0.1) would imply high confidence in the country but not in the specific city, whereas (0.1, 0.1, 1000) indicates that we correctly identified Odesa, but it's unclear where exactly – in Ukraine or Texas.

To address this issue, labeled data from the training set are used, on which traditional machine learning is conducted based on three confidence parameters using standard algorithms from the powerful Python library Scikit [24].

In our case, the machine learning process consisted of the following stages.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

6.1. Parameters engineering

In many machine learning algorithms, transforming input values is a necessary condition without which the algorithm won't converge to an optimal result due to excessively extreme input values or too significant difference between feature magnitudes. Moreover, most machine learning models train better and faster on as standardized data as possible:

- clipping is performed (values less than a set minimum become the minimum, and those greater than a set maximum become the maximum) within a range from min=0.000001 to max=1000, to eliminate zero values and the most significant outliers over 1000;
- to capture not only linear dependencies between input parameters and the predicted value but also potential nonlinear input data behaviors, as well as to account for interactions between different input parameters, polynomial parameters up to degree 3 are generated. For example, from input parameters x_1, x_2, x_3, polynomial parameters x_1, x_1^2, x_1^3, x_1x_2, x_1x_3, $x_1^2x_2$, $x_1^2x_3$, $x_1x_2x_3$, x_2, x_2^2, $x_2^2x_1$, etc., are formed;
- logarithmic transformation of polynomial parameters is conducted to reduce the distribution positive skewness, where most values are relatively close to 0, but some highest values reach up to 109, thereby having a long "tail" to the right, and to bring them to values closer to each other and closer to 0;
- values are standardized using a scaler. Typically, values are scaled relative to the mean and variance. In our case, RobustScaler from Scikit was chosen as the scaler, which is more resistant to outliers and uses the median and interquartile range instead of the usual mean and variance.

6.2. Training

GradientBoostingClassifier [25] was chosen as the classifier, which conducts classification based on boosted trees [26].

In practical applications, effectively deploying the GradientBoostingClassifier necessitates the careful adjustment of its hyperparameters, which play a critical role in shaping the model's accuracy and efficiency. This adjustment process typically involves empirical optimization, where methods such as grid search or random search are frequently employed to identify the most suitable hyperparameter settings.

The hyperparameters selected were:

- n_estimators – the simple models number (decision trees) that make up the ensemble
- learning_rate – the value that indicates how significant the contribution of each model in the ensemble is to the overall result
- max_depth – the maximum depth of the decision trees in the model
- max_features – the maximum number of features considered during the tree nodes splits
- min_samples_split – the minimum number of data points in a node (node samples) required to split a node
- subsample – the fraction of data (among all training data) taken for training each of the simple trees

Hyperparameter tuning was performed using GridSearch, i.e., trying all possible parameters combinations among given values sets with cross-validation.

The hyperparameters quality was assessed using BrierScoreLoss, which shows the average squared difference between the predicted class probability (value pred_proba of the model GradientBoostingClassifier, from 0 to 1, corresponding to how confident the model is that the outcome to which the obtained uncertainty scores correspond is correct) and the true accuracy (0 or 1, depending on the correctness of the classification on training data).

6.3. Classification

The GradientBoostingClassifier from the Scikit library is a robust classification algorithm for machine learning tasks, based on the boosting technique. Boosting is an ensemble method that constructs a series of models sequentially, with each subsequent model aiming to correct its predecessors' errors.

Initially, a decision tree model is created, typically a simple one. This model is imperfect, with accuracy slightly better than a random choice. The first step is not crucial; the iterative process is expected to significantly enhance it.

Next, a loss function is determined to evaluate the model's effectiveness. In this case, the function measures the discrepancy between predicted probabilities and actual class labels, specifically the deviation loss between them.

Gradient boosting methodically enhances the model. At each new step, new models are created to rectify the existing ensemble deficiencies:

- the loss function gradient based on the current model predictions is calculated. This gradient indicates the direction in which predictions should be altered to reduce loss;

- a new decision tree is trained to forecast these gradients for each item in the training set. This tree aims to predict the previous model errors;
- this new decision tree is added to the ensemble with a coefficient known as the learning rate. This coefficient controls the speed at which the model learns. The learning rate is a critically important hyperparameter in gradient boosting. It assesses and scales each tree contribution. If it is too high, the model may overfit; if too low, the model may require too many trees to converge to a satisfactory solution;
- the algorithm continues to add trees until the specified number of trees (n_estimators) is reached or until no further improvement can be made on the training set.

Boosted trees are prone to overfitting. Therefore, several regularization techniques are integrated into the GradientBoostingClassifier:

- limiting the depth of trees with max_depth;
- a fraction of the training data (subsample) is randomly selected to train each tree. This randomness enhances the model robustness;
- learning rate reduction – the learning_rate parameter scales the contribution of each tree, lowering the overfitting risk by diminishing the updates.

The data obtained after training allow for calculating a single confidence score for unlabeled data. The obtained confidence scores can be sorted from top to bottom. In doing so, homogenous names will have the same score and be located nearby, which is convenient for verifying the classification correctness. If the result is correct/incorrect for one name, it will be the same for all similar names. This allows for creating new classes or optimizing the algorithm immediately for a large number of input data.

Table 3 shows the top and bottom five results of the parts classification, confidence scores by tree levels, and the final unified confidence.

7. RESULTS ACHIEVED AND CONCLUSIONS

Based on the proposed approach, a function library in Python was developed. The brief classification times – up to 125 ms for a single row and up to 56 seconds for eleven thousand rows – permit the use of the algorithm in an online mode for wide variety of problems.

Accelerations by more than an order of magnitude are achieved for data comprising thousands of rows, thanks to powerful Python algorithms optimized for working with large matrices. The library we developed is also optimized for rapid computation of large data arrays and utilizes all built-in Python optimization techniques.

The classification accuracy varied across different datasets from 85% to 98% for works and from 87% to 96 % for parts names.

As shown in Table 4, the overall classification accuracy of the proposed algorithm nearly doubled after training, reaching 92.38%.

Table 3. **Top and bottom five results of the parts classification**

Testing data sample	Labeled	Predict.	Result	Confidence by levels			
				L-1	L-2	L-3	Unified
Заміна зовн. ручки і приводу замка чи двері	2011300	2011300	True	36	119	97	99.9 %
Зняття і установка консолі склоочисника	1200500	1200500	True	66	146	73	99.8 %
Замена сцепного шкворня	1050900	1050900	True	31	670	100	99.8 %
Установка обігрівального елементу сидіння	2030200	2030200	True	21	1.16 9	100	99.8 %
Зняття та встановлення маховика інерційн.	1080300	1080300	True	100	198	100	99.8 %
…	…	…	…	…	…	…	…
Заміна газонаповнених амортизат. капота	2010300	2011100	False	2	18	3	7.5 %
Ремонт клапана привода передней двери	2011300	1140400	False	6	5	15	5.5 %
Ремонт ПЖД	1160900	1120000	False	156	1	1	5.3 %
Замена клапана моторного тормоза	1031600	1140400	False	376	4	17	4.4 %
Проверка клапана моторного тормоза	1031600	1140400	False	376	4	17	4.4 %

Source: **compiled by authors**

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

Table 4. **Parts names classification results**

Model type	Vectorization	Accuracy, training data	Accuracy, test data
Custom model without weighting and training	count vectors	—	0.5174
	tf-idf vectors	—	0.6684
Main model with weighted parameters after training	count vectors	0.9365	0.9184
	tf-idf vectors	0.9552	0.9238

Source: **compiled by authors**

The classification of works related to mechanical parts was most effective, while the classification of specialized tasks, such as transmission repair or truck repair works, was less accurate. The partial attribution of this variability to the incomplete directories for certain tasks points towards an potential enhancement area through the expansion and refinement of class directories.

One of the significant ancillary benefits observed from our algorithm implementation is the missing terms identification that necessitate inclusion in the directories, thereby improving the comprehensiveness and the classification system accuracy. This outcome also contributes valuable insights for domain-specific knowledge bases.

The research presented in this paper has successfully demonstrated the application of tree-based classification methodologies to the domain of Ukrainian technical text analysis, specifically focusing on the automotive industry. Through the development of a Python function library; we have showcased our proposed approach capability to efficiently classify technical texts related to automotive repairs and parts, achieving classification times that support real-time application scenarios. This efficiency opens the algorithm up for a wide array of practical uses, from enhancing the call centers operational quality to the creation of automated chatbots and digital assistants for service advisors in automotive service stations.

In conclusion, the research underscores the profound potential of tree-based classification in navigating the complexities of technical text analysis within the automotive sector. By bridging the gap between structured data classification and the nuanced realm of natural language processing, we pave the way for advanced applications that could significantly impact various stakeholders, including insurance companies, automobile manufacturers, and vehicle owners as shown on Fig. 2.

Fig. 2. **Practical implementation of automotive works and parts accurate classification**

Source: **compiled by authors**

The ability to accurately predict maintenance costs and reliability of vehicle components from aggregated, labeled big data represents a substantial stride towards demystifying the vehicle ownership total cost, thereby empowering consumers and industry players alike with valuable, actionable insights.

From the perspective of automotive manufacturers, this approach could substantially impact vehicle design, component reliability and safety, production processes, and warranty policies. Insurance companies may benefit from precise repair cost calculations and accurate assessments of residual vehicle value, leading to reduced expenses. Automotive repair shops can enhance their services by implementing automated chatbots and digital assistants for service managers. Additionally, car owners will be able to determine not only the purchase price of a vehicle but also the total cost of ownership for specific models.

Looking forward, we aim to extend our research to encompass more complex tasks, such as the extraction, identification, and classification of automotive-related works from extensive text bodies, including transcriptions of telephone calls.

REFERENCES

1. Silla, C. & Freitas, A. “A survey of hierarchical classification across different application domains”. *Data Mining and Knowledge Discovery*. 2011; 22 (1): 31–72. DOI: https://doi.org/10.1007/s10618-010-0175-9.

2. Homenda, W., Jastrzebska, A. & Pedrycz, W. “Multicriteria decision making inspired by human cognitive processes”. *Applied Mathematics and Computation*. 2016; 290 (1): 392–411. DOI: https://doi.org/10.1016/j.amc.2016.05.041.

3. Shen, J. & Han, J. “Taxonomy-guided classification, automated taxonomy discovery and exploration”. *Synthesis Lectures on Data Mining and Knowledge Discovery*. Springer, Cham. 2022. p. 83–100. DOI: https://doi.org/10.1007/978-3-031-11405-2_5.

4. Sun, Y., Wang, X., Peng, D., Zhenwen, R. & Shen, X. “Hierarchical hashing learning for image set classification”. *IEEE Transactions on Image Processing*, 2023; 32: 1732–1744. DOI: https://doi.org/10.1109/TIP.2023.3251025.

5. Chen, J., Stouffs, R. & Biljecki, F. “Hierarchical (multi-label) architectural image recognition and classification”. *Proceedings of the 26th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA).* 2021; 1: 161–170. DOI: https://doi.org/10.52842/conf.caadria.2021.1.161.

6. Yang, C., Harjoseputro, Y. & Chen, Y. “A hybrid approach of simultaneous segmentation and classification for medical image analysis”. *Multimed Tools Appl*. 2024. DOI: https://doi.org/10.1007/s11042-024-19310-9.

7. Yang, F, Li, X., Liu, Q., Li, X. & Li, Z. “Learning-based hierarchical decision-making framework for automatic driving in incompletely connected traffic scenarios”. *Sensors*. 2024; 24 (8): 2592. DOI: https://doi.org/10.3390/s24082592.

8. Schopf, T., Braun, D. & Matthes, F. “Semantic label representations with Lbl2Vec: A similarity-based approach for unsupervised text classification”. *Web Information Systems and Technologies.* 2023; 469: p. 59–73. DOI: https://doi.org/10.1007/978-3-031-24197-0_4.

9. Zhang, L., Ding, J., Xu, Y., Liu, Y. & Zhou, S. “Weakly-supervised text classification based on keyword graph”. *The 2021 Conference on Empirical Methods in Natural Language Processing*. 2021. p. 2803–2813. DOI: https://doi.org/10.18653/v1/2021.emnlp-main.222.

10. Esmaeili, A., Ghorrati, Z. & Matson, E. T. “Hierarchical Collaborative Hyper-Parameter Tuning”. *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection. PAAMS 2022. Lecture Notes in Computer Science.* 2022; 13616; p. 127–139. DOI: https://doi.org/10.1007/978-3-031-18192-4_11.

11. Williams, L., Anthi, E. & Burnap, P. “Comparing hierarchical approaches to enhance supervised emotive text classification”. *Big Data and Cognitive Computing*. 2024; 8 (4): 38. DOI: https://doi.org/10.3390/bdcc8040038.

12. Zangari, A., Marcuzzo, M., Rizzo, M., Giudice, L., Albarelli, A. & Gasparetto, A. “Hierarchical text classification and its foundations: A review of current research”. *Electronics*. 2024; 13 (7): 1199. DOI: https://doi.org/10.3390/electronics13071199.

13. Arenas, M., Botoeva, E., Kostylev, E. & Ryzhikov, V. “A note on computing certain answers to queries over incomplete databases”. *In: CEUR Workshop Proceedings. Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web*. Montevideo: Uruguay. 2017.

14. Li, Y., Currim, F. & Ram, S. “Data completeness and complex semantics in conceptual modeling: The Need for a Disaggregation Construct”. *Journal of Data and Information Quality*. 2022; 14 (4): 1–21, https://www.scopus.com/authid/detail.uri?authorId=15122156300. DOI: https://doi.org/ https://doi.org/10.1145/3532784.

15. Blasi, D., Anastasopoulos, A. & Neubig, G. “Systematic inequalities in language technology performance across the world’s languages”. *In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 2022; 1: 5486–5505. DOI: https://doi.org/10.18653/v1/2022.acl-long.376.

16. Mashtalir, S. & Nikolenko, O. “Data preprocessing and tokenization techniques for technical Ukrainian texts”. *Applied Aspects of Information Technology*. 2023; 6 (3): 318–326. DOI: https://doi.org/10.15276/aait.06.2023.22.

17. Mashtalir, S. & Nikolenko, O. “Advancing automotive technical text analysis: A Tree-based classification approach for Ukrainian texts”. *The 7th International Conference on Computer Science, Engineering and Education Applications.* 2024.

18. Mohammad, A. “Using blockchain for data collection in the automotive industry sector: A literature review”. *Journal of Cybersecurity and Privacy*. 2022; 2 (2). DOI: https://doi.org/10.3390/jcp2020014.

19. Danielkiewicz, R. & Dzieńkowski, M. “Analysis of user experience during interaction with automotive repair workshop websites”. *Journal of Computer Sciences Institute*. 2024; 30: 39-46. DOI: https://doi.org/10.35784/jcsi.5416.

20. Hemphill, T., Longstreet, P. & Banerjee, S. “Automotive repairs, data accessibility, and privacy and security challenges: A stakeholder analysis and proposed policy solutions”. *Technology in Society*. 2022; 71(3): 102090. DOI: https://doi.org/10.1016/j.techsoc.2022.102090.

21. Vajjala, S., Majumder, B., Gupta, A. & Surana, H. “Practical natural language processing: A comprehensive guide to building real-world NLP systems”. *Published by O’Reilly Media, Inc*. 2020.

22. Tan, P.-N., Steinbach, M. & Kumar, V. “Introduction to data mining”. 2nd ed. Published by Pearson Education Limited, Harlow. 2019.

23. Singhal, A. “Modern information retrieval: A brief overview”. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering.* 2001; 24 (4): 35–43.

24. Müller, A. & Guido, S. “Introduction to machine learning with python: A guide for data scientists”. *1st ed. Published by O'Reilly Media.* 2016.

25. Géron, A. “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to build intelligent systems”. *3rd ed. Published by O'Reilly Media*. 2022.

26. Bastos, J. “Predicting credit scores with boosted decision trees”. *Forecasting*, 2022; 4 (4): 925–935. DOI: https://doi.org/10.3390/forecast4040050.

Conflicts of Interest: The author declares that there is no conflict of interest

Received: 26.07.2024
Received after revision: 11.09.2024
Accepted: 20.09.2024

DOI: https://doi.org/10.15276/hait.07.2024.15
УДК 004.91

Оптимізація ієрархічних класифікаторів шляхом налаштування параметрів та оцінки впевненості

Машталір Сергій Володимирович[1)]
ORCID: https://orcid.org/0000-0002-0917-6622; sergii.mashtalir@nure.ua. Scopus Author ID: 36183980100

Ніколенко Олександр Володимирович[2)]
ORCID: https://orcid.org/0000-0002-6422-7824; oleksandr.nikolenko@uzhnu.edu.com
[1)] Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна
[2)] Ужгородській національний університет, вул. Університетська, 14. Ужгород, 88000, Україна

АНОТАЦІЯ

Ієрархічні класифікатори відіграють вирішальну роль у вирішенні складних задач класифікації, розбиваючи їх на менші, більш керовані підзадачі. Ця стаття продовжує серію робіт, зосереджених на ієрархічній класифікації технічних українських текстів, зокрема класифікації ремонтних робіт та запасних частин, що використовуються в обслуговуванні та ремонті автомобілів. Ми вирішуємо питання, пов'язані з багатомовними вхідними даними – зокрема українською, російською та їх міксом – і відсутністю стандартних моделей попередньої обробки даних для української мови. У цій статті описується метод навчання та оцінювання моделі ієрархічної класифікації за допомогою налаштування параметрів для кожного вузла в деревоподібній структурі. Процес навчання включає ініціалізацію ваг для токенів у вузлах дерева класів та вхідних рядках, після чого проводиться ітеративне налаштування параметрів для оптимізації точності класифікації. Початкові ваги призначаються на основі наперед визначених правил, а ітеративний процес коригує ці ваги для досягнення оптимальної продуктивності. Стаття також розглядає проблему інтерпретації множинних показників впевненості, отриманих з процесу класифікації, пропонуючи підхід машинного навчання з використанням GradientBoostingClassifier з бібліотеки Scikit-learn для розрахунку уніфікованого показника впевненості. Цей показник допомагає оцінити надійність класифікації, особливо для нерозмічених даних, шляхом трансформації вхідних значень, генерації поліноміальних параметрів та використання логарифмічних перетворень і масштабування. Класифікатор точно налаштовується за допомогою технік оптимізації гіперпараметрів, а фінальна модель забезпечує надійний показник впевненості для задач класифікації, дозволяючи перевіряти та оптимізувати результатів класифікації на великих наборах даних. Загальна точність класифікації майже подвоїлася після навчання, досягнувши 92.38 %. Це дослідження не тільки просуває теоретичну основу ієрархічних класифікаторів, але й надає практичні рішення для обробки великомасштабних, нерозмічених наборів даних в автомобільній індустрії. Майбутні роботи будуть спрямовані на розширення цього підходу на більш складні задачі, такі як знаходження та класифікація інформації з великих текстів, наприклад, транскрипцій телефонних дзвінків.

Ключові слова: обробка природної мови (NLP); деревоподібна класифікація; машинне навчання; аналіз даних; прикладні інтелектуальні системи

ABOUT THE AUTHORS

Sergii V. Mashtalir - Doctor of Engineering Science. Professor, Informatics Department. Kharkiv National University of Radio Electronics, 14, Nauky Ave. Kharkiv, 61166, Ukraine
ORCID: https://orcid.org/0000-0002-0917-6622; sergii.mashtalir@nure.ua. Scopus Author ID: 36183980100
Research field: Image and video processing; data analysis

Машталір Сергій Володимирович - д-р техніч. наук, професор. Професор кафедри Інформатики Харківського національного університету радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

Oleksandr V. Nikolenko - PhD student. Uzhhorod National University, 14, University Str. Uzhhorod, 88000, Ukraine
ORCID: https://orcid.org/0000-0002-6422-7824: oleksandr.nikolenko@uzhnu.edu.com
Research field: Natural language processing; Big Data; machine learning

Ніколенко Олександр Володимирович – здобувач ступеня доктора філософії у Державному вищому навчальному закладі «Ужгородський національний університет», вул. Університетська, 14. Ужгород, 88000, Україна