Comparative Performance of Neural Networks and Ensemble Methods for Command Classification in ALEXA Virtual Assistant

Li lI
Artificial Intelligence and Big Data College, Chongqing Polytechnic University of Electronic Technology
Chongqing 401331, China
E-mail: 200708036@cqcet.edu.cn

Keywords: ALEXA virtual assistant, deep learning models, neural networks, random forest, command classification

Received: December 2, 2024

Our study investigates the classification of commands for the ALEXA virtual assistant using various machine-learning models. The dataset includes 16,521 samples, and data preprocessing steps, such as vectorization and remove all stop words and punctuation, were applied before training. Decision Trees, Random Forest, Hist Gradient Boosting, AdaBoost, and Neural Networks are employed to classify textual commands into respective classes. The dataset consists of commands and their classes, transformed into feature vectors using the TF-IDF method. Our neural network architecture comprises three dense layers and two dropout layers, totaling 272,850 trainable parameters, and uses RMSprop for optimization and categorical cross-entropy as the loss function. Performance is evaluated utilizing metrics like accuracy, precision, recall, and F1 score. Results have shown that neural networks perform better in comparison to classical algorithms and outperform AdaBoost explicitly in all metrics. The comparative results between neural networks and AdaBoost in evaluation metrics are, respectively, as follows:(0.851695 / 0.620157), (0.857729 / 0.771549),(0.851695 / 0.62057) and (0.85236 / 0.639389). Therefore, deep learning will indeed provide many promises toward solving challenging NLP tasks in a virtual assistant system like Alexa. The findings provide enormous insight into effective methodologies regarding the classification of commands and further establish the relevance of neural networks within extending virtual assistant technology. Further research may consider discussing more recent neural network structures and exploring their scalability and generalizability across several domains and languages.

Povzetek: Raziskava primerja učinek nevronskih mrež in ansambelskih metod pri razvrščanju ukazov v Alexa asistentu, kjer nevronske mreže dosegajo najboljše razultate.

1 Introduction

Whereas virtual assistants gave a whole new dimension to human-computer interaction, in facilitating technology towards and for people in every respect, the pioneering leader is ALEXA by Amazon. It grants users the ability to perform voice-controlled functions that range from managing smart home appliances to even playing music. The central role of ALEXA involves understanding and classifying commands.

Intent classification for virtual assistants is especially challenging because of the variability of natural language and the extensive range of tasks ALEXA can do. For a virtual assistant to function reliably and provide a seamless user experience, it is very important to accurately classify the command.

These models represent a wide variety of machine learning techniques used in this investigation, from the more traditional decision tree-based approaches to advanced ensembles and neural networks. Decision trees provide interpretability and simplicity, while ensemble methods, such as Random Forest, Histogram Gradient Boosting, and AdaBoost, use the power of bundling several weak learners together to improve classification performance. Furthermore, the neural network-based model gives a versatile structure that can fit complex patterns in data.

The aim is to find out the best model that classifies voice commands in virtual assistant ALEXA through hard experimentation and analysis. Some of the key parameters used in the evaluation include classification accuracy, computational efficiency, and scalability to handle high volumes. Other than this, we discuss the strengths and weaknesses of each model, which also constitutes insights into their performance and suitability against real-world deployment.

These findings contribute to the extension of machine learning knowledge in virtual assistant systems that have natural language processing tasks. The outcome is better represented in reality by designing and implementing a more efficient command classification system that is highly accurate to enhance the user experience with ALEXA and other similar voice assistants.

1.1 Main novelty

The most important novelty of this research includes an in-depth comparative analysis of machine learning models to

classify commands within ALEXA. The work will detail the performance of different models of the command classification in ALEXA, whereas other previous works may have focused on individual models or just general natural language processing tasks. We present the comparison of various models, starting from classic decision tree-based approaches and ending with ensembles such as Random Forest, Histogram Gradient Boosting, and AdaBoost, and one neural network-based model. The selection thus enables diverse exploration of methodologies and their suitability for the task at hand. We restrict our focus to the virtual assistant ALEXA; hence, these research findings should apply directly to this widely used and practical context and thus hopefully also contribute to tangible improvements in user experience and system reliability.

Furthermore, our study lets the models be scrutinized along several dimensions: not only for the accuracy of the classification but also for computational efficiency and scalability. This provides a more complete picture of the performance and practical viability of each model in real-world applications. By distilling the strengths and weaknesses of each model, we give great insight into the performance characteristics of the models, shedding light on why some of these models may be more suitable for command classification in ALEXA compared to others. Our study's results can further help in designing and implementing more efficient and effective command classification systems for ALEXA and similar virtual assistants. In turn, this may improve user experience and interaction with these platforms.

1.2 Related studies

With the improvement in techniques of machine learning and deep learning and also with the development of strong tools such as AI and virtual assistants, the researchers and the engineers went deep inside the aspects of text recognition, classification, speech recognition, and all those areas. It is a vast field covering a whole range of techniques that inspire new approaches toward solving these problems comprehensively.

Zhou et al. [1] present a new model, C-LSTM, that combines CNN and RNN architectures for sentence modeling in text classification. In C-LSTM, CNN is used to extract the representation of phrases, which is fed into an LSTM to produce the sentence representation. This model captures both local phrase features and global sentence semantics. These results indeed show that, for both sentiment and question classification tasks, C-LSTM outperforms CNN and LSTM. The results obtained were excellent for all tasks using the C-LSTM model.

Lai et al. [2] propose a new kind of recurrent convolutional neural network for text categorization that does not rely on human-designed features. The model learns the representation of words by composing the meaning of a sentence with a recurrent structure and a max-pooling layer to automatically identify keywords during classification. Experimental results on four datasets show that this approach outperforms the existing techniques, especially on document-level datasets.

A simple approach for detecting forge news is proposed by Granik & Mesyura [3] based on a naive Bayes classifier. They implemented such a system and performed its testing on the Facebook news posts dataset. The classifier achieved an accuracy of roughly 74% on the test set, impressive given the simplicity of the model. The paper reviewed several lines of research improving these results and pointed out a direction in which artificial intelligence approaches may help in dealing with the problem of fake news detection.

Tavčar et al. (2016) [4] introduce a web-based system designed to enhance virtual museum tours through the use of intelligent virtual assistants. This system analyzes user preferences to generate personalized exhibition recommendations. Furthermore, it incorporates a natural language interface, enabling users to ask questions and receive contextually relevant responses**.**

Kim et al. [5] investigate capsule networks for text classification, which is not well researched, although capsule networks have been very successful in image classification. This paper showed very promising results using capsule networks for text classification and their advantages when compared to convolutional neural networks. They further propose a simple routing method to reduce the computational complexity. Experiments on seven benchmark datasets show that capsule neural networks, using the routing technique presented in this paper, match the performance of traditional methods.

Qiao et al. [6] introduce a novel approach in this work to obtain task-specific distributed representations of n-grams for text classification by using "region embeddings." Using two different models to generate the embeddings, each word representation is combined through a weighting matrix to interact with the local context. These serve as parameters for the neural network classifier. It outperforms the existing methods on several benchmark datasets and effectively captures the important phrasal expressions present in texts, as evidenced by the experimental results.

Table 1: Examples of related works along with data

Reference	Task Type	Data	Accuracy
Lai et al. [2]	Convolutional Neural Networks	20Newsgroup/ 7520	CNN/94.79% RCNN/96.49%
		Fudan University document/ 8823	CNN/94.04% RCNN/95.20%
Kim et al. [5]	Classification using Capsules	20Newsgroup/10182	Capsule-A/80.39% Capsule-B/80.03%
		Reuters10/6472	Capsule-A/87.74% Capsule-B/87.96%
Qiao et al. [6]	Region Embedding for Text Classification	Amazon Review Polarity/ 3,000,000	VDCNN/95.7% D-LSTM/ -
		Sogou News/ 450,000	VDCNN/96.8% D-LSTM/94.9%

Islam et al. [7] present the Semantics Aware Random Forest (SARF) classifier, which selects the features relevant to the predicted classes. They assess SARF's performance on 30 real-world text datasets, comparing it with leading ensemble selection methods. The results show that SARF excels in textual information retrieval, indicating a promising new avenue for research on classifier interpretability.

Chen et al. [8] introduce NPMM, a nonparametric model designed for online short-text analysis. It utilizes auxiliary word embeddings to estimate topic numbers and tackle sparsity in topic-word distributions. NPMM automates the process of determining document-topic association by employing squared Mahalanobis distance, thereby eliminating the need for hyperparameter tuning in a new topic generation. Additionally, they suggest a nonparametric sampling strategy for identifying key terms within each topic. Inference is performed using a one-pass Gibbs sampling algorithm, augmented by a Metropolis-Hastings step. Experimental results demonstrate NPMM's superior performance compared to existing methods.

Yao et al. [9] present a text classification model that utilizes fastText. The model employs feature engineering to extract key information from the text and generates high-quality, low-dimensional, continuous representations using the fastText algorithm. The
experiment involves classifying a text dataset from the Baidu Dianshi platform's "user comment data emotional polarity judgment" using Python. Results from the emotional polarity judgment task indicate that the model outperforms traditional machine learning algorithms in precision, recall, and F-values, showcasing its strong classification performance.

Gargiulo et al. [10] explore Deep Learning architectures for text classification, focusing on extreme multi-class and multi-label tasks with hierarchical labels. They introduced the HLSE method for adjusting the data
label and evaluating a variety of WE model. Evaluation of the PubMed collection shows the effectiveness of HLSE and the importance of various WE model for such tasks.

Recently, Wang et al. [11] proposed Hierarchy-guided Contrastive Learning (HGCLR) to incorporate hierarchy into a text encoder. While training, HGCLR prepares the positive samples from the label hierarchy and lets the text encoder learn hierarchy-aware representation on its own. While in testing, the HGCLR-enhanced encoder can discard redundant hierarchy. The effectiveness of HGCLR has been well verified by intensive experiments on three benchmark datasets.

Sun et al. [12] introduced Clue and Reasoning Prompting (CARP), a method that uses a step-by-step reasoning approach tailored for text classification. CARP guides Language Models (LLMs) to detect basic clues such as keywords, tones, semantic connections, and references, which then trigger a diagnostic reasoning process for making final decisions. To tackle the issue of limited tokens, CARP utilizes a fine-tuned model on a supervised dataset for KNN demonstration search in in-context learning. This method combines the generalization ability of LLMs with task-specific evidence from the complete labeled dataset.

2 Utilized models, data and methods

This section presents a brief description of the executed dataset as well as the models. Also, the methodology of this study is explained.

Model Hyperparameters (default values)

We used Decision Tree, Random Forest, AdaBoost, and Histogram-based Gradient Boosting (Hist GB) models with their default parameters, as these values are commonly used in studies and provide reliable results. Additionally, details regarding the experimental setup, computational environment, and software libraries used have been provided for better understanding. The key default parameters are.

Model Hyperparameters (default values)

Decision Tree: max_depth=30, min_samples_split=2, min_samples_leaf=1, criterion="gini"

Random Forest: n_estimators=100, max_depth=30, min_samples_split=2, min_samples_leaf=1, criterion="gini"

AdaBoost: n_estimators=50, learning_rate=1.0, base_estimator=DecisionTreeClassifier(max_depth=1)

Histogram-based Gradient Boosting (Hist GB): learning_rate=0.1, max_iter=100, max_depth=30, l2_regularization=0.1, min_samples_leaf=20

Data Splitting Strategy

- The dataset was split into **75% training and 25% testing**.
- This process was repeated **5 times with random splits**, and the final results were reported as the **mean values** across these runs.

Computing Environment

- **RAM:** 16GB
- **CPU:** Ryzen 7
- **GPU:** NVIDIA RTX 3060
- **Software:** Python 3.10.*, TensorFlow, Scikit-learn

Neural Network Configuration

- **Train-Validation Split:** 10% of the training data was set aside for validation.
- **Batch Size:** 32
- **Epochs:** 30
- **Dropout Rate:** 0.2
- **Dense Layers:** Three layers, each with **128 neurons**.
- **Optimizer:** RMSprop with learning rate 2.5e-4 due to its fast convergence.

2.1 Data description

Aiming to classify the commands of ALEXA, they gathered various information related to the ALEXA virtual assistant. The dataset contains 16,521 records collected from the MASSIVE dataset' containing both the text of the commands and their corresponding class labels. The dataset encompasses approximately 20 distinct classes. Classification techniques were used to sort and categorize the data based on these labels.

Fig. 1 demonstrates the dispersion of the data in each category.

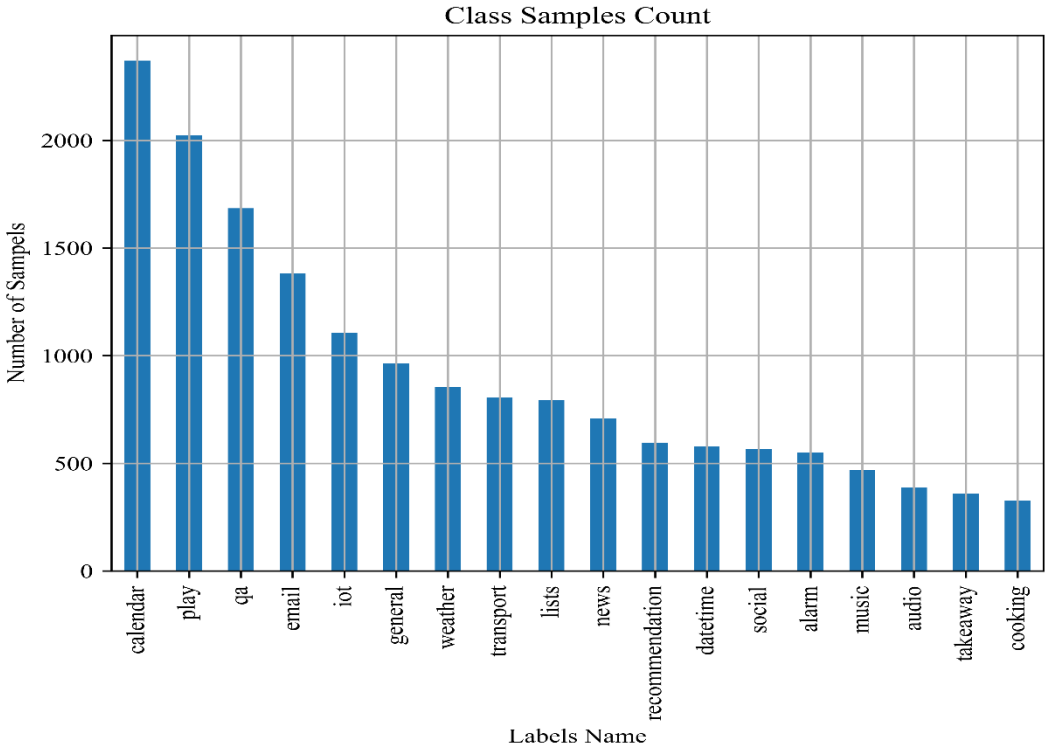

Figure 1: Number of samples in each label

According to the graphs in Fig. 1, the data with labels of calendar and play are in the majority, and labels such as cooking and takeaway contain the least amount of data. Each label contains various words with different frequencies. The following figures demonstrate word frequency in some of the labels presented in the word cloud as an example.

Figure 2: Word frequency in the label of the calendar

Fig. 2 displays that the label named calendar contains several words with close relation to the calendar. It displays those words such as calendar, meeting, event, remind, and today have been repeated more.

Figure 3: Word frequency in the label of cooking

The related words to cooking are demonstrated in Fig. 3. Accordingly, words such as recipe, cook, make, chicken, and find are more frequent in the commands related to cooking. This indicates that ALEXA users mostly used this VA to learn food recipes.

Figure 4: Word frequency in the label of transport

According to Fig. 4, in the commands related to transport, words like ticket, train, traffic, and book are more frequently, denoting the fact that people mostly used ALEXA to buy tickets and check the traffic.

Figure 5. Word frequency in the label of play

According to Fig. 5, the play label involves the words related to the commands for playing music and other media. Hence, words such as play, song, music, podcast, radio, and start are in the most repeated commands.

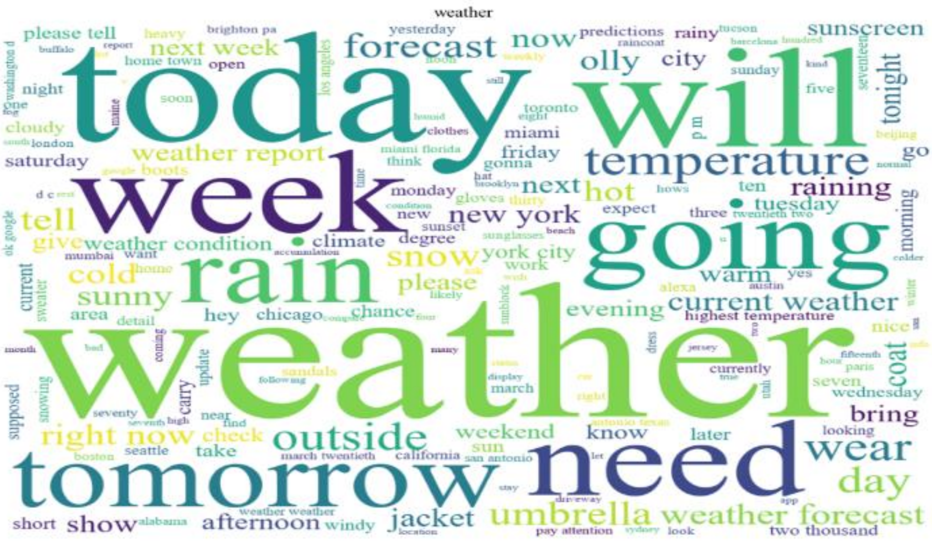

Figure 6. Word frequency in the label of weather

The label of weather contains all the command texts related to weather. According to the word cloud in Fig. 6, weather, today, will, week, need, rain, tomorrow, and will are the most frequent words in this class.

The following sections describe the compared models in this study.

2.2 Decision tree model

A decision tree classifier is a popular algorithm for classification, using a tree-like model where each internal node tests a specific feature, branches represent the decision outcome of those tests, and leaf nodes represent the class labels. This structure makes it intuitive to understand how decisions are made based on input features [13].

The algorithm begins by choosing the optimal feature from the dataset using specific criteria like information gain or Gini impurity. It then splits the dataset into subsets, starting with the selected feature. Each subset represents a different value of the feature. Each subset follows this iterative feature selection and partitioning process, recursively creating a tree structure until a stopping condition has been reached, such as when a maximum tree depth is reached or a minimum number of samples in a node.

Once the tree has been constructed, any instance can be classified by starting at the root and working its way down to a leaf. At each node, it considers the test on the feature and heads down the appropriate branch until it reaches a leaf. This gives the instance the class label of the

leaf that it ends up at. The ease with which this process can be followed makes decision trees easily understandable and interpretable and forms one of the major advantages of the technique.

With their great power, however, comes great susceptibility to overfitting in the case of large decision trees. To overcome this, several methods can be performed: pruning and tuning of parameters such as maximum depth or minimum samples per leaf. Then there are the ensemble methods, which include Random Forests and GBMs, where many trees are combined with the intent of reducing overfitting and further improving performance. Generally, decision tree classifiers offer a versatile and interpretable approach to classification tasks, with several strategies that may be used to improve performance and generalization capabilities [14], [15].

2.3 Random forest model

The Random Forest ensemble is resistant and involves learning for both classification and regression problems. It constructs a large number of decision trees during training and combines their outputs to yield the outcome. Each tree in the Random Forest is trained on a different subset of the data through a process called bootstrap sampling, meaning that random samples are drawn with replacement from the training data.

In the decision trees formed in a Random Forest, each node will not consider all features on which to split; instead, a random subset of the features alone would be considered for a split. That injects randomness into the model and helps with decorating the trees, yielding an ensemble that's more robust and less prone to overfitting. The best split at each node is determined based on a chosen criterion such as Gini impurity or information gain. This is an iterative process that repeats either until the trees are full or the stopping criterion is reached. For classification tasks, each tree in the Random Forest "votes" for a class, and the class receiving the most votes is chosen as the final prediction. For regression tasks, the final prediction is gained by averaging the predictions from all the trees.

Random Forests provide several benefits. They are resilient to noise and outliers due to the averaging effect of multiple trees. They also offer insights into feature importance, helping identify which features significantly impact predictions. Further, the training of individual trees can be parallelized, thus making them efficient for large datasets.

However, with these strengths, Random Forests are quite computationally expensive. In cases of high linearity among the features-target variables relationships or high-dimensional data, its performance may not be at par with other techniques. Nevertheless, because it has robustness and effectiveness in solving a wide variety of problem types, Random Forests find wide use in practice in a wide array of domains [16], [17].

Random Forests can be represented mathematically, although it's a bit more complex than a single decision tree due to the ensemble nature of the model.

Let $X = (X_1, X_2, \dots, X_n)$ be the input features and Y be the target variable.

A Random Forest consists of B decision trees $T_1, T_2, \dots, T_B$ each trained on a bootstrap sample of the data. The prediction for a new instance x in a classification problem is obtained by taking a majority vote among the predictions of all trees:

$$\hat{y}_{RF}(x) = \boldsymbol{mode}\left(\hat{y}_1(x), \hat{y}_2(x), \dots, \hat{y}_B(x)\right) \quad \textbf{(1)}$$

Where $\hat{y}_i(x)$ is the predicted value of the i-th tree.

2.4 Histogram gradient boosting

The Histogram-based Gradient Boosting Classifier is a robust machine learning algorithm for classification tasks, especially on big data. It falls into the category of a gradient-boosting family that combines a sequence of weak learners, usually decision trees, into a robust classifier.

Perhaps the most salient feature of the Hist Gradient Boosting Classifier is the unique approach to feature binning via histogramming. This avoids the use of exact algorithms that usually find split points in decision trees and instead discretizes features into intervals, hence making it way faster during training, particularly for datasets with a large number of samples.

Similar to other boosting gradient-type methods, the Hist Gradient Boosting Classifier builds trees consecutively in a greedy manner such that the newest tree in each step minimizes the previously incurred loss. Adding more trees in a sequence reduces some loss of objective function—e.g., deviance or exponential loss for classification—leading to an increase in the predictive power of the model. The Histogram Gradient Boosting Classifier uses regularisation to avoid overfitting by limiting the tree depth and the minimum number of samples to create a leaf and to split a node for good generalization to unseen data.

Another advantage of the Hist Gradient Boosting Classifier involves the handling of missing values in the dataset, both during training and prediction, without needing to take prior care of imputation. This facility makes it highly convenient when dealing with real datasets that are inherently bound to show some data loss.

Due to its histogram-based approach, the Hist Gradient Boosting Classifier is highly scalable and efficient and is hence indicated in problems with a high number of samples and features. Like other gradient-boosting methods, it is also robust to outliers in data and can handle mixed types of features, including categorical and numerical variables [18], [19].

2.5 AdaBoost model

AdaBoost-Adaptive Boosting is a classification ensemble learning technique that combines several weak learners into one robust learner. The basic principle behind AdaBoost lies in training multiple weak learners—usually simple decision trees—on a training set sequentially while focusing on instances difficult to classify.

All instances in the dataset have an equal weight to start training. A weak learner is then fitted on this data and tested for performance. Weights are increased for those instances that have been classified incorrectly, while the weights are decreased for the instances correctly classified in successive rounds. The adaptive fashion of this method will make subsequent weak learners focus more on instances that were challenging to classify in earlier rounds. It involves training each of these weak learners sequentially in a manner that each of them tries to correct mistakes made previously by the other weak learners. Several of these weak learners are trained one after another, and all their predictions are combined to have the final prediction. They all contribute to the final prediction. Their contribution is weighted by their accuracy.

AdaBoost works effectively for both binary and multiclass classification problems. It is very effective to deal with complicated data since it can capture complex boundaries of the decisions. AdaBoost may be sensitive to outliers and noisy data, which could degrade the performance. Though limited, AdaBoost is still popular owing to its simplicity and efficiency. Besides, it often acts as a base algorithm in advanced ensemble methods like GBM and XGBoost, further enhancing its performance and robustness [20], [21].

2.6 Neural Network model

A neural network model is a computational framework of interconnected nodes, or neurons, in layers inspired by the structure and function of biological neural networks, such as the human brain. Each neuron receives an input, undergoes processing of that input, and sends it to the next layer; this is accomplished with the interlinking of neurons through weights according to their strength.

It is trained in a procedure called supervised learning, where the model learns to relate input data to output labels by adjusting the weights of the connections among neurons. By repeatedly exposing the model to training data, it learns to recognize patterns and, eventually, to make predictions or classifications. They find applications in many areas, such as image and audio recognition and natural language processing, because they can find complex patterns from enormous data [22].

The neural network model in this study comprises three dense layers and two layers of dropout. In total, the neural network model contains 272,850 trainable parameters.

Dense layer: It forms one of the major components in neural network architectures. Every neuron in this layer is normally connected to every neuron of the previous layer, thus creating a dense connectivity structure. Hence, each neuron's output is dependent on all neurons in the preceding layer. Each such connection is also associated with a learned weight in training. The most common places for which dense layers occupy are in the middle of the neural network, allowing it to capture intricate patterns within the data. Normally, after each of them comes an activation function that may introduce non-linearity into the network, hence enabling it to learn and represent complex relationships among features of data in a more effective way.

Dropout in neural networks is a regularization approach that works by randomly deactivating a portion of the neurons during training to prevent overfitting. During each training iteration, dropout arbitrarily turns a portion of the outputs from the neurons to zero. This forces the network to learn redundant representations of its data, reducing the risk of relying too heavily on any particular feature or combination of features. Dropout effectively simulates training multiple networks with different architectures simultaneously, resulting in a more robust model that generalizes better to unseen data. Typically, dropout is applied to hidden layers of the network, with a dropout rate parameter controlling the fraction of neurons to deactivate.

We also utilized the categorical cross entropy as the loss function in the neural network model. Categorical cross-entropy stands as a frequently employed loss function in machine learning, particularly for tasks involving multi-class classification. Its role is to evaluate the discrepancy between the actual distribution and the predicted distribution of categorical variables. In classification problems, the output variable is categorical, meaning it falls into one of several classes. The true distribution represents the actual classes, typically encoded as a one-hot vector, where only one element (the true class) is 1, and the rest are 0s. The predicted distribution, on the other hand, represents the model's probabilities for each class. Mathematically, categorical cross-entropy is calculated using the following formula for a single example:

$$\boldsymbol{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_i \boldsymbol{y}_i \log(\hat{\boldsymbol{y}}_i) \quad \textbf{(2)}$$

Where y is the true distribution (the one-hot encoded vector). $\hat{y}$ is the predicted distribution (the vector of probabilities). Class i's true probability is denoted by y_i. The anticipated probability of class i is shown by $\hat{y}_i$.

The sum is taken over all classes. This formula is applied for every example within the dataset and then averaged across all examples to make a final calculation of loss.

That's because it is trained to penalize the model when it has a lower probability of the true class by quantifying the difference between predicted and true distributions. This is appropriate for problems with mutually exclusive classes, in which each instance is assigned only to one class. Minimizing this categorical cross-entropy loss, therefore, allows the model to generate probabilities that align well with the actual distribution, hence increasing its accuracy of classification [23].

To enhance the performance of the neural network, this model is optimized using the RMSprop algorithm. ***RMSprop***, or Root Mean Square Propagation, is a popular choice for training neural networks since it avoids the pitfalls of normal stochastic gradient descent by adapting the learning rate of each parameter based on the magnitudes of the gradients. Such adaptiveness provides

faster convergence and better handling of non-stationary and sparse gradients in tasks involving deep learning.

The RMSProp works by giving each parameter a different learning rate, meanwhile computing the exponential moving average of the squared gradients. This is achieved through an exponentially decaying average calculation:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2 \quad (3)$$

Here, g_t represents the gradient of the parameter at time step t, and β controls the decay rate of the moving average. Typically, β is set to a value like 0.9. The moving average $E[g^2]_t$ accumulates the squared gradients over time, providing an estimate of the variance of the gradients.

The update rule in RMSprop is then adjusted using the square root of the accumulated squared gradients, effectively scaling the learning rates for each parameter:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \quad (4)$$

When the parameter at time step t is denoted by θ_t, the learning rate is indicated by η, and ϵ is a tiny constant introduced for numerical stability to avoid division by zero. The division by the square root of the accumulated squared gradients normalizes the learning rate such that updates corresponding to large gradients are reduced and vice versa.

RMSProp is particularly suited for deep learning tasks, wherein the data may have different gradients and scales, with its adaptive learning rate. Nevertheless, similar to many of the optimization algorithms, RMSProp has several hyperparameters, such as the learning rate η and the decay rate β, that need further fine-tuning to attain optimal performance on different applications [24], [25].

2.7 Data preprocessing

First, the text is vectorized using one of the most salient techniques for text analysis and document classification: the TF-IDF method. TF-IDF (Term Frequency-Inverse Document Frequency) is a crucial text preprocessing technique used to convert textual data into numerical representations for machine learning models. TF-IDF involves processes such as tokenization (where text is divided into individual words or tokens), stemming/lemmatization (where words are reduced to their root form to standardize variations), TF calculation (where the frequency of each word in a document is computed), IDF calculation (where the inverse document frequency is computed to down weight common words across multiple documents), and TF-IDF score computation (where each word's TF is multiplied by its IDF score to determine its importance within the document). TF-IDF helps highlight important words in a document while reducing the impact of common but less meaningful terms.

The unigrams, bigrams, and trigrams included in the vectorization process incorporated more intricate patterns and connections from within the textual data.

After vectorization, we had 2057 unique words and important phrases for our dataset. Further, in improving the quality of representation, all stop words and punctuation were removed from the text. This removes the noise in the texts and puts the attention of the model on meaningful content.

Then, the data were divided into training and testing sets. There were 13,217 records designated for training the models to learn trends and relationships from them. 3,304 records were set aside to test the performance of the models. This separation of data into training and testing sets helps to assess how well the trained models generalize to unseen data and provides a measure of their predictive accuracy.

3 Analysis results

3.1 Neural network validation

In the first step, we analyzed the loss values and accuracy of the neural network model during training and testing. The Fig. 7 in the following demonstrates the results.

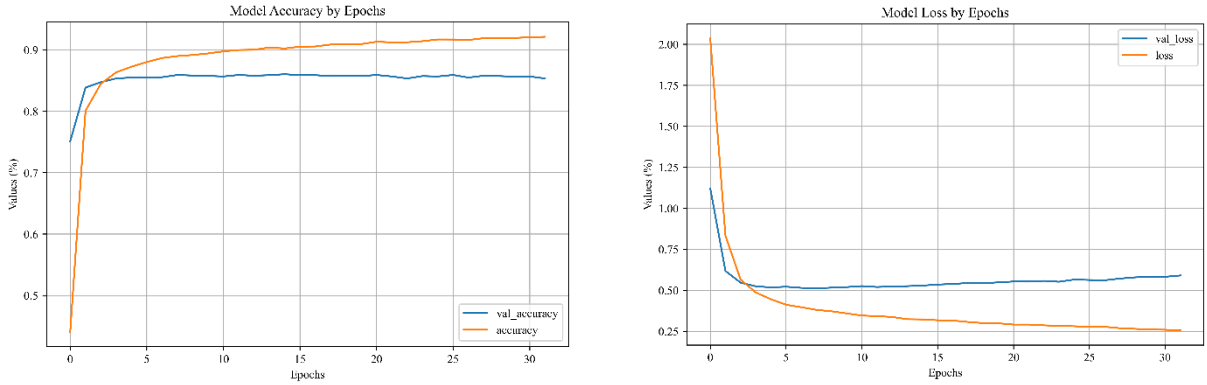

Figure 7: The validation results of the neural network model

Fig. 7a displays the accuracy of the neural network model after 30 epochs. It indicates that the accuracy of the model in total increases in the earlier epochs and stays in the high values as the epochs advance. The training accuracy rapidly rises within the first few epochs and then stabilizes

around 92%, while the validation accuracy follows a similar pattern but remains slightly lower, around 86%. This gap between training and validation accuracy suggests that the model learns well on training data but has some overfitting tendencies.

Furthermore, the loss value of the model during training and testing in Fig. 7b denotes that the loss value of the neural network model in the training process decreases gradually as the epochs go forward. The loss initially drops sharply within the first few epochs, showing rapid learning progress, then continues to decrease more gradually. The training loss consistently reduces, reaching approximately 0.2, while the validation loss exhibits a different trend.

By scrutinizing the loss value of the test (val_loss), it is derived that the loss values at the earlier epochs drop. However, as the epochs advance, the loss value of the model starts increasing to a small number due to overfitting; hence, the validation process after 30 epochs is stopped. This increase in validation loss while training loss continues to decrease suggests that the model is memorizing patterns from the training data rather than generalizing well to unseen data. Techniques such as early stopping, dropout regularization, or data augmentation could help mitigate overfitting and improve the model's ability to generalize.

3.2 Evaluation metrics

The studied models used in this analysis, which are introduced in sections 2.2 to 2.6, are evaluated through well-known metrics such as accuracy, precision, recall, and f1 score. These metrics are used to evaluate the performance of classification models, especially in the context of machine learning and statistics.

Accuracy is the most straightforward metric and measures the ratio of correctly predicted instances to the total instances. It is computed by dividing the total number of forecasts by the number of correct guesses.

$$\boldsymbol{Accuracy} = \frac{(\boldsymbol{TP+TN})}{\boldsymbol{total\ population}} \quad \textbf{(5)}$$

Precision assesses the proportion of accurately predicted positive observations out of all predicted positive observations. It reflects the model's effectiveness in minimizing false positives.

$$\boldsymbol{Precision} = \frac{(\boldsymbol{TP})}{(\boldsymbol{TP+FP})} \quad \textbf{(6)}$$

High precision means that an algorithm returned substantially more relevant results than irrelevant ones, though it might miss some relevant results (low recall).

Recall or sensitivity measures the proportion of accurately predicted positive observations compared to all actual positive instances in the dataset. It evaluates the model's ability to detect all positive cases.

$$\boldsymbol{Recall} = \frac{(\boldsymbol{TP})}{(\boldsymbol{TP+FN})} \quad \textbf{(7)}$$

High recall means that an algorithm returned most of the relevant results (low false negatives), though it might also bring back many irrelevant results (low precision).

The ***F1 score*** represents the harmonic mean of precision and recall, offering a balanced single score that considers both precision and recall.

$$\boldsymbol{F1\ score} = \mathbf{2} \times \frac{(\boldsymbol{Precision} \times \boldsymbol{Recall})}{(\boldsymbol{Precision} + \boldsymbol{Recall})} \quad \textbf{(8)}$$

The F1 score is a good measure of a model's accuracy, especially when there is an uneven class distribution (imbalanced data).

The obtained results for the aforesaid metrics are presented in the following table:

Table 2: The metrics results for the models

	Neural Network	Decision Tree	Random Forest	AdaBoost	Hist GB
Accuracy	0.851695	0.81023	0.831114	0.620157	0.804782
Precision	0.857729	0.815996	0.837435	0.771549	0.822176
Recall	0.851695	0.81023	0.831114	0.62057	0.804782
F1 score	0.85236	0.810181	0.830936	0.639389	0.80615

The results show the neural networks' superior performance. Neural Network achieves the highest accuracy (0.8517) and strong performance across all metrics, making it the most effective model overall. Decision Tree and Random Forest also perform well, with Random Forest slightly outperforming Decision Tree in all metrics.

AdaBoost shows significantly lower performance, particularly in accuracy (0.6202) and F1 score (0.6394), indicating it may not be well-suited for this dataset.

Hist Gradient Boosting (Hist GB) performs competitively, with results close to those of the Decision Tree and Random Forest models. To get a better understanding of the obtained results, Fig. 8 presents the graphic comparison.

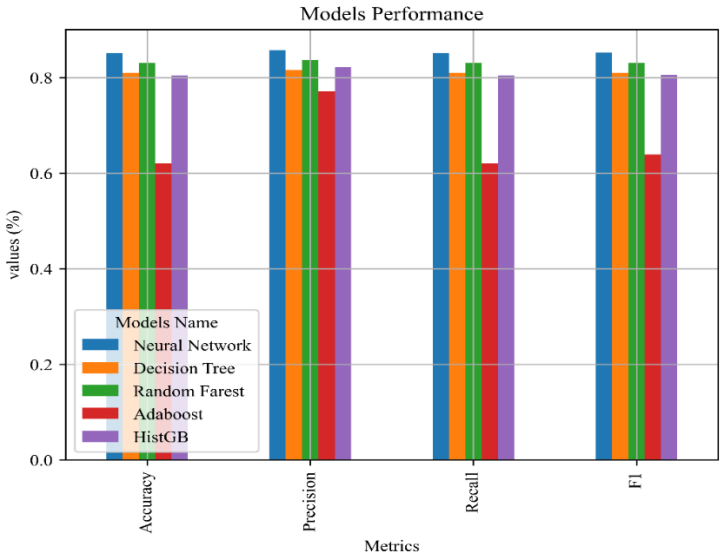

Figure 8: Comparison of the obtained results for the models

By comparing the accuracy of the models, it is deduced that the neural network model holds the highest accuracy value, and the random forest and the decision tree models are the second and third-best models. The precision of the models demonstrates the neural network model shows promising value with the best-obtained value among the studied models. The results of the recall denote the outperformance of the neural network model compared to the other exerted models. The results indicated the disappointing results of the AdaBoost. According to the f1 score of the models, the neural network presents the highest obtained value, and the random forest and decision tree are the next beneficial models.

In total, the neural network consistently outperforms all other models across all metrics utilized in this study. Conversely, AdaBoost exhibits the lowest performance across the board. This highlights the superior effectiveness and robustness of the neural network over its counterparts, making it the clear choice for the task at hand.

3.3 Class correlation in the neural networks

Since the neural network model was the best model for accomplishing the given tasks, the class characteristics of the neural networks are delved into, rigorously. The following figure demonstrates the confusion matrix among the classes.

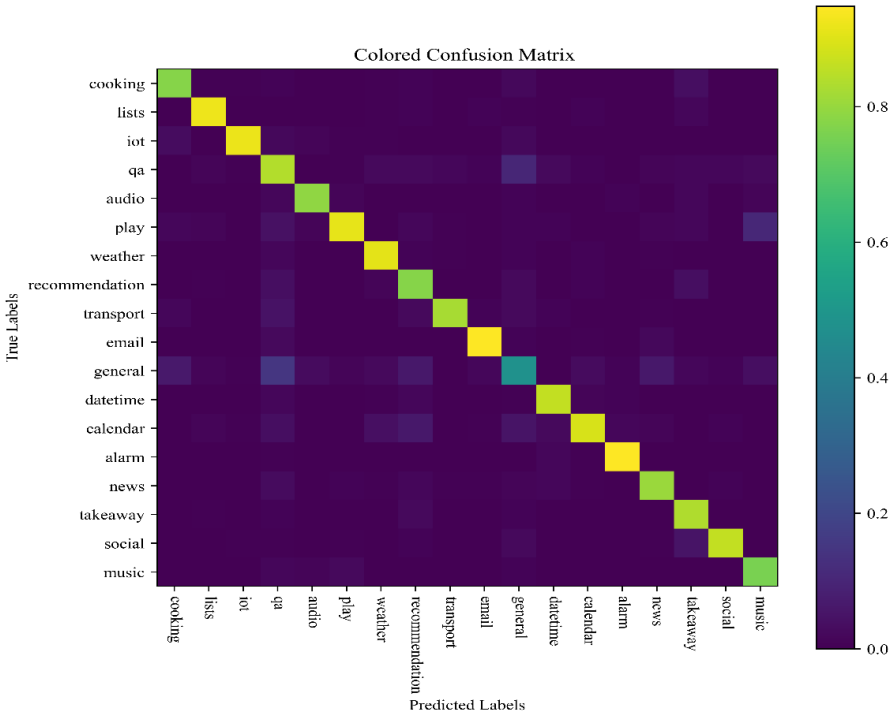

Figure 9: Confusion matrix of neural network classes

Fig. 9 displays the confusion matrix, which denotes which classes are mistaken with each other. This happens due to the similarity of data between the two classes. In Fig. 9, the lighter colors represent more similarity between classes. As an example, the general and qa contain resembling textual data and are sometimes mistaken with each other by the models. In other words, "light colors" in the confusion matrix indicate higher values of classification accuracy, representing a stronger correlation between true and predicted labels.

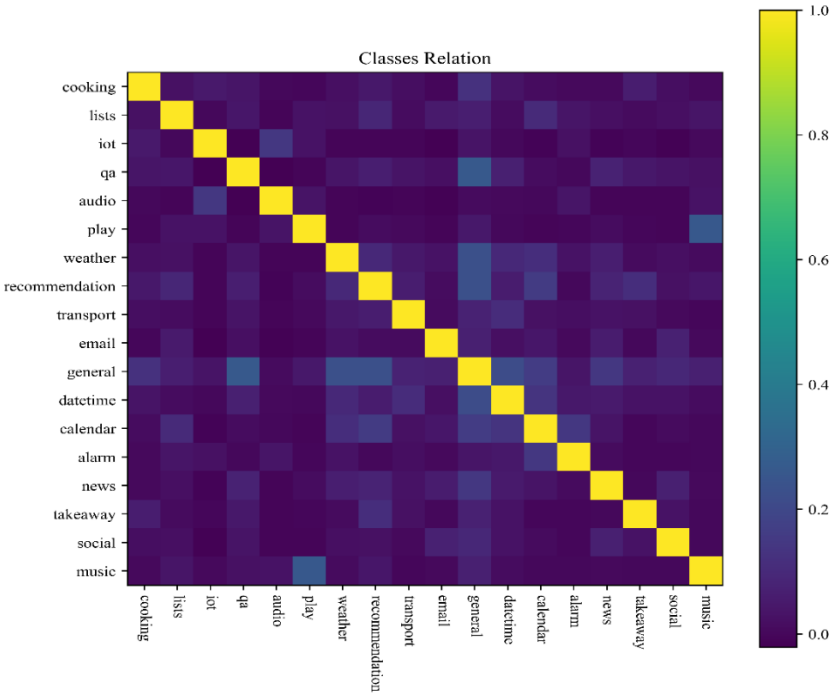

Figure 10: Class relation in the neural network model

Fig. 10 demonstrates the relationship between classes in the neural network model. The lighter colors indicate the higher relation between the two classes. Two classes with high relations denote that the classes share the same commands and contain resembling textual data. According to the matrix, the general has a high relationship with classes such as weather, recommendation, and qa. Also, play and music share similar commands.

Analyzing the confusion matrix, we can observe areas where the model misclassified instances. The off-diagonal elements in the matrix highlight these misclassifications.

For example, there is a noticeable overlap between the "lists" and "recommendation" classes. This could stem from shared contextual words in their respective command texts, such as "add," "suggest," or "list." Enhancing the feature extraction process or incorporating contextual embeddings may help differentiate these classes better.

The confusion between "news" and "general" is evident, likely because commands labeled as "general" might occasionally reference current events or updates, leading the model to associate them with "news." Using additional semantic analysis might mitigate this issue.

4 Discussion

Highest Values Across All Metrics: It can be observed that the Neural Network outperforms other models across all evaluation metrics, achieving the highest scores. This indicates that the neural network has the lowest error in its predictions and is the most optimized model for this task.

Traditional models like Decision Trees and Random Forests have structural limitations that prevent them from effectively capturing complex relationships in the data. In contrast, a neural network, with multiple hidden layers, can better learn nonlinear and intricate patterns within the dataset. In other words, the flexibility in learning complex patterns is one of the advantages of this model.

Deep learning also offers advantages in natural language processing (NLP). Neural networks are particularly well-suited for Natural Language Processing (NLP) tasks. Unlike classical models based on decision trees, neural networks can better understand semantic relationships between words and phrases. Given that the goal of this research is to classify voice commands in ALEXA, this capability makes the neural network a superior choice.

The proposed model utilizes improved optimization and overfitting prevention techniques. The neural network architecture benefits from techniques such as Dropout and RMSprop optimization, which improve the model's generalization ability. This ensures that the model maintains high performance on new, unseen data.

Neural networks often excel in text classification because they can automatically learn complex and hierarchical representations from raw text data. In contrast with traditional machine learning methods, which usually rely on manually engineered features or simpler representations, neural networks (especially architectures

like CNNs, RNNs, or Transformers) can capture intricate semantic relationships and contextual nuances, leading to improved performance.

One possible reason AdaBoost underperforms in this context is its reliance on weak learners (often decision trees) that might not capture the complexity of text data as effectively as neural networks. Text data is typically high-dimensional and sparse, making it challenging for boosting methods that build models sequentially. If the individual weak learners cannot handle the intricacies of word order, context, and nuanced semantics, the ensemble may struggle even after several iterations. Additionally, AdaBoost is sensitive to noisy data and outliers, which are common in natural language tasks, further impeding its performance.

Additionally, the performance and scalability of the models used in the article were compared with each other. Decision Trees Fast in training and prediction but prone to overfitting. Good scalability for small datasets Random Forest Higher accuracy than decision trees with reduced overfitting but slower training. Moderate scalability. Hist Gradient Boosting More efficient than classical boosting methods, suitable for large datasets, but requires careful hyperparameter tuning. AdaBoost Performs well on small datasets but is sensitive to noise and less scalable than modern boosting methods. Neural Networks Capable of learning complex patterns but computationally expensive and requires large datasets for optimal performance. High scalability with GPU/TPU acceleration.

Statistical

The following is the t-test for neural network and random forest for accuracy precison and recall

T-Test Results:
Accuracy: t = 4.1309, p = 0.0006
Precision: t = 3.8258, p = 0.0012
Recall: t = 6.3499, p = 0.0000
F1-Score: t = 2.5812, p = 0.0188

5 Conclusion

In conclusion, our research offers an in-depth analysis of command classification for the ALEXA virtual assistant, utilizing various machine learning models and methods. By utilizing Decision Trees, Random Forest, Hist Gradient Boosting, AdaBoost, and Neural Networks, we aimed to discern the most effective approach for accurately categorizing user commands.

Our study capitalized on a dataset containing text-based commands and their corresponding classes, which were transformed into feature vectors using the TF-IDF method. Notably, our neural network architecture was composed of three dense layers and two dropout layers, comprising a total of 272,850 trainable parameters. The loss function used was the categorical cross-entropy, while RMSProp was used to ensure sound optimization for the training of the neural network model.

In this respect, our results unambiguously prove the superiority of neural network models compared to other conventional machine learning algorithms through extensive evaluation by using precision, accuracy, F1 score, and recall metrics. More precisely, the neural networks were always ranked first compared to AdaBoost in terms of classification performance for all measured parameters.

This work provides insight into the performance of various machine learning models for command classification but points out the very important potential use of neural networks in dealing with complex natural language processing tasks. The top performance given by the relative performance of neural networks in this work points to their suitability for real-world applications within virtual assistant frameworks such as ALEXA.

Furthermore, our findings point out how state-of-the-art techniques, such as deep learning, will have to be explored for high accuracy and robust performance in command classification tasks. The success of neural networks here indicates how these approaches are relevant within virtual assistant technology, opening new paths toward the development of more fluid user experiences and natural ways of interaction.

These findings have significant practical implications. By leveraging neural networks, ALEXA's command classification can be directly improved in real-world scenarios. For instance, better classification accuracy can lead to enhanced user satisfaction by reducing misinterpretation of commands. The findings also suggest that integrating more complex neural architectures could allow ALEXA to handle a broader variety of commands, including those involving nuanced or multi-intent queries. Additionally, the scalability demonstrated in this research implies that the models can adapt to diverse languages and accents, improving accessibility for users worldwide.

In further research, more neural network architectures can be compared and more feature extraction methods could be explored, or even ensemble techniques could be considered to improve the performance on command classification tasks. Additionally, the scalability and generalizability of the proposed models could be evaluated across different domains and languages to assess their broader applicability. Overall, our study contributes valuable insights into the field of natural language processing and lays the foundation for continued advancements in virtual assistant technologies.

Nomenclature

Abbreviation	Description	Abbreviation	Description
VA	Virtual Assistant	X	Input feature
AdaBoost	Adaptive Boosting	Y	Targe value
AI	Artificial Intelligence	$\hat{y}_i(x)$	Predicted value of i-th tree in random forest

CNN	Convolutional Neural Network	y_i	True probability of class i
LSTM	Long Short-Term Memory	$\hat{y}_i$	Predicted probability of class i
SARF	Semantics Aware Random Forest	ϵ	Small constant number
NPMM	nonparametric model	g_t	The gradient of the parameter at iteration t
HLSE	Hierarchical Label Set Expansion	β	Parameter to control the decay rate
HGCLR	Hierarchy-guided Contrastive Learning	θ_t	Parameter at time step t
WE	Word Embedding	η	Learning rate
CARP	Clue and Reasoning Prompting	TP	True positive
GBM	Gradient Boosting Machines	TN	True negative
RMSprop	Root Mean Square Propagation	FP	False positive
SGD	stochastic gradient descent	FN	False negative

Acknowledgements

I would like to take this opportunity to acknowledge that there are no individuals or organizations that require acknowledgment for their contributions to this work.

Competing of interests

The authors declare no competing of interests.

Authorship contribution statement

The author contributed to the study's conception and design. Data collection, simulation, and analysis were performed by "Li LI ". Also, the first draft of the manuscript was written by Li LI commented on previous versions of the manuscript.

Data availability

Data can be shared upon request.

Declarations

Not applicable

Conflicts of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Author statement

The manuscript has been read and approved by all the authors, the requirements for authorship, as stated earlier in this document, have been met, and each author believes that the manuscript represents honest work.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Ethical approval

The research paper has received ethical approval from the institutional review board, ensuring the protection of participants' rights and compliance with the relevant ethical guidelines.

References

[1] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, "A C-LSTM Neural Network for Text Classification," Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.08630

[2] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification," 2015. [Online]. Available: www.aaai.org

[3] M. Granik and V. Mesyura, "Fake News Detection Using Naive Bayes Classifier," in IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), 2017.

[4] A. Tavčar, C. Antonya, and E. V. Butila, "Recommender System for Virtual Assistant Supported Museum Tours," 2016. [Online]. Available: http://www.projekt-

[5] J. Kim, S. Jang, S. Choi, and E. Park, "Text Classification using Capsules," Aug. 2018, [Online]. Available: http://arxiv.org/abs/1808.03976

[6] C. Qiao et al., "A NEW METHOD OF REGION EMBEDDING FOR TEXT CLASSIFICATION," 2018.

[7] M. Z. Islam, J. Liu, J. Li, L. Liu, and W. Kang, "A semantics aware random forest for text classification," in International Conference on Information and Knowledge Management, Proceedings, Association for Computing Machinery, Nov. 2019, pp. 1061–1070. doi: 10.1145/3357384.3357891.

[8] J. Chen, Z. Gong, and W. Liu, "A nonparametric model for online topic discovery with word embeddings," Inf Sci (N Y), vol. 504, pp. 32–47, Dec. 2019, doi: 10.1016/j.ins.2019.07.048.

[9] T. Yao, Z. Zhai, and B. Gao, "Text Classification Model Based on fastText," in IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), 2020.

[10] F. Gargiulo, S. Silvestri, M. Ciampi, and G. De Pietro, "Deep neural network for hierarchical extreme multi-label text classification," Applied Soft Computing Journal, vol. 79, pp. 125–138, Jun. 2019, doi: 10.1016/j.asoc.2019.03.041.

[11] Z. Wang, P. Wang, L. Huang, X. Sun, and H. Wang, "Incorporating Hierarchy into Text Encoder: a

Contrastive Learning Approach for Hierarchical Text Classification," Mar. 2022, [Online]. Available: http://arxiv.org/abs/2203.03825

[12] X. Sun et al., "Text Classification via Large Language Models," 2023.

[13] I. Gounari and M. Kanzilieris, "Wireless Sensor Networks Focusing on Predicting Average Localization Error through Machine Learning Applications," Journal of Artificial Intelligence and System Modelling, vol. 01, no. 04, pp. 104–118, 2024, doi: 10.22034/jaism.2024.474005.1055.

[14] A. Navada, A. Nizam Ansari, S. Patil, and B. A. Sonkamble, "Overview of Use of Decision Tree algorithms in Machine Learning," IEEE Control and System Graduate Research Colloquium, 2011.

[15] Y. Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," Shanghai Arch Psychiatry, vol. 27, no. 2, pp. 130–135, Apr. 2015, doi: 10.11919/j.issn.1002-0829.215044.

[16] G. Biau and E. Scornet, "A Random Forest Guided Tour," Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.05741

[17] G. Louppe, "Understanding Random Forests: From Theory to Practice," Jul. 2014, [Online]. Available: http://arxiv.org/abs/1407.7502

[18] Y. J. Ong, Y. Zhou, N. Baracaldo, and H. Ludwig, "Adaptive Histogram-Based Gradient Boosted Trees for Federated Learning," Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.06670

[19] A. Guryanov, "Histogram-Based Algorithm for Building Gradient Boosting Ensembles of Piecewise Linear Decision Trees," in 8th International Conference, AIST 2019 Kazan, Russia, July 17–19, Goos Gerhard and Hartmanis Juris, Eds., Kazan: Springer, Jul. 2019, pp. 39–50.

[20] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class AdaBoost *," 2009.

[21] R. E. Schapire, "Explaining AdaBoost," 2013.

[22] Y. Goldberg, "Neural Network Methods for Natural Language Processing," Synthesis Lectures on Human Language Technologies, vol. 10, no. 1, pp. 1–311, 2017, doi: 10.2200/S00762ED1V01Y201703HLT037.

[23] Z. Zhang and M. R. Sabuncu, "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels," 2018.

[24] T. Kurbiel and S. Khaleghian, "Training of Deep Neural Networks based on Distance Measures using RMSProp," Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.01911

[25] R. Elshamy, O. Abu-Elnasr, M. Elhoseny, and S. Elmougy, "Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning," Sci Rep, vol. 13, no. 1, Dec. 2023, doi: 10.1038/s41598-023-35663-x.