Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 6, pp. 5322–5330, DOI 10.12694/scpe.v25i6.3307

THE APPLICATION OF DEEP LEARNING IN SPORTS COMPETITION DATA PREDICTION

JI CHEN[*] AND PENGTAO CUI[†]

Abstract. In order to predict sports competition data, the author needs to implement the structure and related processes of the relevant competition victory and defeat prediction system, and specifically introduce and plan the implementation of each functional module. The data collection and storage module adopts Alibaba Cloud servers and combines Python to remotely and automatically collect data on a scheduled basis, according to the actual situation of game wins and losses, data cleaning and filtering are carried out, and multiple encoding forms are used to vectorize the data in order to find the best model. The data is divided according to the standard training and testing sets, and multiple classifiers are used for model training and saved locally for direct use next time; Test the above model using the training set; Compare the advantages and disadvantages of each vectorized encoding and classifier based on the final performance evaluation module. Based on the relevant experimental results, a detailed analysis was conducted to compare the advantages and disadvantages of each model, proving that introducing word vectors (word embeddings) into the competition data analysis system is worthwhile. We have obtained an excellent performance prediction model with a highest accuracy P of 0.825, a recall R of 0.729, and a corresponding F1 value of 0.774. For a prediction model that only knows the initial lineup allocation as a prerequisite, this already has sufficient practical guidance significance.

Key words: Deep learning, Sports competitions, Data prediction, Application, machine learning

1. Introduction. In recent years, the training data generated in competitive sports training has shown explosive growth, resulting in a massive amount of training data. For the massive amount of training data, athletes or coaches only focus on the valuable part of the training data [1]. Therefore, how to find the desired data in massive amounts of data, conduct timely and effective analysis, and apply it to training and competitions is an urgent problem that needs to be solved. By utilizing the powerful data processing, mining, and analysis capabilities of deep learning, the massive data generated in competitive sports training is trained and applied to competitive sports training, committed to promoting the accuracy and refinement of analysis in competitive sports training, providing technical guidance for athlete training, and promoting the scientific and information-based development of competitive sports training in China, provide some reference methods for the research and application of deep learning in competitive sports training. With the deepening development of educational modernization, the application of big data and artificial intelligence technology has become a hot direction in the field of education. An increasing number of studies are utilizing data analysis and artificial intelligence algorithms for the evaluation and analysis of educational processes and outcomes. However, current research mostly emphasizes the importance and application value of data, and lacks specific evaluation systems and methods to support it [2-3]. In physical education teaching, algorithmic models can be used to evaluate students' athletic performance and teaching effectiveness.

The following are some possible algorithm models and evaluation indicators. Machine learning models can predict student sports performance and performance scores by analyzing student sports performance data, such as athlete posture, movement, speed, strength, etc. These models can be trained based on a large amount of data to achieve higher accuracy; By using data mining models, students' behavior patterns during exercise can be obtained, and teaching effectiveness evaluations can be obtained from them. For example, data such as interaction information between athletes and coaches, athlete performance files, and live streaming athlete videos are collected for analysis and to determine the quality of coach instruction; Artificial intelligence models: In AI based models, deep learning techniques can provide coaches with more intuitive and accurate evaluations, while models that predict teaching effectiveness can predict athletes' learning tendencies and propose effective

[*]Zhejiang Yuexiu University, Shaoxing, Zhejiang, 312000, China
[†]Shaoxing University Yuanpei College, Shaoxing, Zhejiang, 312000, China (Corresponding author, chenji24118@ 126.com)

Fig. 2.1: Schematic diagram of the structure of the competition victory and defeat prediction system

learning strategies based on their personal information and performance data; The video analysis model can discover many typical movements such as single movements, movements, and jumps from video playback, and based on the analysis results, a comprehensive score can be obtained to evaluate student performance and provide targeted guidance to students. These algorithm models can evaluate the teaching effectiveness of physical education based on multiple evaluation indicators. Some common indicators include comparative analysis, which is a technique used to determine whether a student's current performance is better or worse than their past performance. Teachers can compare students' current performance scores with their previous performance scores; Noise to noise ratio (SNR) is the relative noise level used to compare student performance evaluation models. If the SNR of the model is higher, its prediction accuracy and precision will be higher; Relative Gain refers to the relative improvement amplitude for all participants using a specific method; Training convergence speed refers to the current application model and its accuracy, using different parameters and configurations can improve the convergence speed of the model.

Due to the youthful nature of esports and its rugged nature in the past, not many scholars have devoted their academic energy to it. Traditional NBA, tour de france, and other technologies such as game replay have not had significant effects in esports. Therefore, it has not been until recent years that relevant scholars have studied this area, and the research methods used are generally still some basic ones. So some of the machine learning methods adopted by the author, whether it is traditional machine learning algorithms such as k-nearest neighbor (KNN), popular neural network algorithms, or even deep neural network algorithms of deep learning, are less applied in relevant research and analysis. So for the entire traditional machine learning algorithm, such as KNN, SVM, logistic regression, and so on, which have been studied since the last century and have been developed and improved so far, we will not elaborate on it here. However, if we consider changing the perspective of the entire game win loss analysis model and transforming it into a special type of natural language processing model (NLP) in the form of small dictionaries and heavy correlations, we will discover the adaptability of some natural language processing methods today. Natural language recognition models have also been proposed and continuously studied since the last century. In the 1950s, expert rules based on syntactic and lexical features were commonly used for modeling and processing. Later, due to the rapid increase in the amount of data to be processed, at the beginning of this century, supervised feature engineering methods based on these features were mostly used for modeling. As time enters the second decade of the 21st century, the scale of data further expands exponentially, and the network data generated in 2020 alone is the sum of all years before 2019. For such a large-scale data, a series of problems such as how to minimize supervised label training without utilizing feature engineering have emerged.

2. Methods.

2.1. Overall System Framework. Generally speaking, such a framework is quite complex. We follow the general method of decomposing and processing related complex transactions, and build the entire system in a modular and procedural manner. The modules of the entire system have been divided into the following 5 modules in the form of Figure 2.1: Data collection and storage, data and processing, model selection, data classification, and performance evaluation [4].

Below is a brief introduction to the functions of the relevant modules: Data collection and storage is a module used to achieve real-time and dynamic collection of data from the network, laying a solid foundation for subsequent data preprocessing modules. Data and processing involves cleaning and preprocessing data into different formats to make appropriate inputs for different classifier models. Model selection refers to selecting different models for classification processing based on different data formats and types. The data classification module is based on the selected model and uses different classifiers for classification. The performance evaluation module is a display module designed to accurately evaluate the performance of classification results.

2.2. System workflow and description. In order to better determine which classifier to use to build the final model, the modules operate independently and use multiple methods to work, resulting in high flexibility of the system. The following section provides detailed modules of the entire system that work together [5].

(1) Data collection and storage module. Thanks to the open attitude of the developer and publisher of DOTA2 game, Valve, it is not difficult to collect game data related to DOTA2. The official API data collection interface is provided, and there are also related DOTA2 APIs that can be called in Python related libraries, that is, through the relevant competition ID, we can obtain specific data for the competition. At the same time, considering that the author's goal is to establish a complete and separable system for predicting competition outcomes, we have established the following standards and methods for data collection:

- Determine the start time of the first game of collection;
- Determine the total amount of collected data;
- Save the above variables as input;
- Start the remote Alibaba Cloud server, execute relevant Python code, read the input from the previous section, complete data collection and storage, and automatically write it into a CSV format file as output.

Additionally, due to the complexity and size of competition data, as well as the communication efficiency of remote servers, our collection and storage speed is approximately 0.05 seconds per piece of data.

(2) Data cleaning module. Due to the complexity of competition data, not every game is a valid match. For example, non pure 5V5 player competitive games (such as players fighting against computers together); Or the competition time may be too short (whether due to network fluctuations or someone giving up the competition early); Or it could be an abnormal competition mode (such as center lane singles, or activity competitions, etc.). So we filtered the relevant data and only retained the game data that met the following conditions: the game had 10 human players [6]; The competition time is greater than 20 minutes; Only consider the game mode of all hero ladder match. The reason for selecting the above three points is based on the data obtained from an effective game; It should be composed of 10 human players; And there should be no early abandonment or network fluctuations (time greater than 20 minutes); And based on the fact that all heroes are in a selectable rank mode, the maximum level of confrontation and equality between players is crucial.

Only when the above three points are met simultaneously can it be a game match with significant data analysis value. So next, we will perform data cleaning operations to match the model. Read the CSV file obtained from the previous step of data collection and storage, and follow the previous three filtering rules to remove the observation data corresponding to the non matching match ID. Keep the hero selection for the R and D sides, with 5 IDs for each. Keep the relevant victory and defeat data, R's victory is recorded as 1, and D's victory is recorded as 0. Remove other irrelevant data, such as match time, match ID, etc. Save the cleaned data as a CSV file for output.

(3) Data segmentation. The specific function of this module is to divide it into training set and test set for subsequent training and testing, and select the optimal model for use. The specific approach is to read the relevant vectorized data as input to the model, and randomly select 70% of it as the training set and 30% as the test set. The data is divided and saved as two separate outputs, among them, the training set is prepared to be placed into the classifier model for training, and then a training model library is generated; The corresponding test set is used to test the model library.

(4) Classifier training. We will not provide a detailed introduction to the training process of each classifier here. Below, we will train them one by one according to the input. The goal and steps here are to obtain the training set data as input, train the model, and output and save the relevant models to the classifier model library.

(5) Classifier Model Library. The general steps are: In order to continue the classifier training from the previous step, save each trained model, and then convert the data from each test set into a format and send

Table 3.1: Distribution of Various Types of Samples in the Sample Set

Category	1 (Radiant side wins)	0 (die wins)
Number of categories	470473	413876

it to the relevant classification model to obtain the final output, which is the prediction results for the test set. The output is saved as model related files for direct use in the next round of victory or defeat prediction classification.

(6) Performance display and evaluation. For each classifier itself, we need a specific method or even module to evaluate the performance of each classifier. We generally select three indicators: accuracy P, recall R, and F1 value. These three indicators are quantitatively calculated together and used as specific evaluations and displays for algorithms of different classifiers.

The calculation formula for the accuracy P, recall R, and F1 values is as follows:

$$P = \frac{1}{2}\left(\frac{TP}{TP+FP} + \frac{TN}{TN+FN}\right) \tag{2.1}$$

$$P = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+F}\right) \tag{2.2}$$

$$F_1 = \frac{2 * P * R}{P + R} \tag{2.3}$$

The general steps for its specific calculation are to input the test set as a test sample into the classifier model library, calculate the relevant accuracy P, recall R, and F1 value quantification indicators, write the file and output it. It should be clarified that the performance evaluation module is an important test of classifier performance, and it is definitely not an optional part. Without this module, we will not be able to determine the performance of the algorithm. Only by designing this module can we continuously improve the model, select more appropriate classifiers as our final classifier, and only in this way can we have the possibility of moving towards higher performance.

3. Results and Analysis.

3.1. Source of data used in the experiment. The dataset used by the author mainly comes from the official API interface data provided by Valve (Dota2), the developer and publisher of this game [7]. The author uses the - dota2 API, a Python library that integrates relevant API interface code, as the main means of obtaining data, and places the code on a remote Alibaba Cloud server. They regularly upload their relevant data files (in CSV format) to the specified email. The main collection of game data is between October 1st, 2021 and October 31st, 2021, with a total of 884347 matches. This is mainly due to the fact that players are familiar with the characteristics of the relevant versions during this period, and the quality of the matches is relatively high, which allows them to obtain the most useful win or loss information. From the actual collected relevant competition data in Table 3.1, we can see that during this period, the distribution of the dataset in the competitions that meet the data cleaning requirements was relatively uniform. The probability of the samples belonging to Class 1 (i.e. the radial side winning) was approximately 53.2%, and the probability of the corresponding samples belonging to Class 0 (i.e. the direct side winning) was approximately 46.8%. Basically, it is a evenly distributed sample set, so each classifier should have good performance.

3.2. Experimental testing environment. The author's experimental testing is based on: operating system: Windows 1064 bit system; Processor: Intel (R) Core (TM) i7-5500UCPU240GHz (4-core); Memory: 8.00GB; The software platform for the experiment is designed based on Python 3.2.5 and meets the author's requirements for relevant programs [8].

Table 3.2: K-Nearest Neighbor (KNN) Model Results

Group number	Parameter K	Accuracy P	Recall rate R	F1 value
1	K=3	0.624	0.453	0.525
2	K=4	0.636	0.451	0.528
3	K=5	0.631	0.452	0.527

Table 3.3: Results of logistic regression model

Group number	Regularization parameter	Optimization method parameters	Accuracy P	Recall rate R	F1 value
1	L2	1iblinear	0.679	0.489	0.566
2	L2	sag	0.691	0.523	0.595
3	L1	1iblinear	0.664	0.486	0.561

3.3. Experiment.

(1) Construction of word vectors and classifier models. The model that the author will verify is divided into two modules, the word vector module and the classifier module. The author of the classifier module mainly divides the relevant classifiers described in Chapter 2 into five common models: K-nearest neighbor (KNN) model, logistic regression model, support vector machine (SVM) model, xgboost model, and neural network (ANN) model. The reason for using decision trees alone is their poor stability, and the author should use the xgboost model, an integrated decision tree model, as a suitable alternative. The specific tuning of hyperparameters for the above related models, due to their complexity and complexity, will be briefly presented in the next section.

We noticed that the traditional method of using word vectors is to input the sum of related word vectors as the meaning of the sentence into the next layer of classifier. However, in this article, we will propose some new ways of combining word vectors, such as CBOW-3. Considering that the specific usage scenario in this article is competition win/loss prediction, rather than traditional semantic analysis, not only does the sum of word vectors contained in the sentence serve as input, but colleagues also take the difference of word vectors contained in the sentence and the word vectors formed by the overall lineup of the two sides as input, which contains more information than traditional models, let's leave the specific classification effect to the next section for further investigation.

(2) Performance display of each model. For considerations of time and other factors, for the first three simple models, we only use one hot encoding for input processing, while the word vector format is mainly processed by xgboost.

For the K-nearest neighbor (KNN) model, we directly use one hot encoded vectorized data for processing as the most basic comparison. As shown in Table 3.2.

So we observed that when K=4, the balance of various indicators is good, but overall, KNN is a very poor classifier at this time, and its prediction performance is not much different from directly predicting the victory of R (radial), and its time spent is more than 8 hours.

The logistic regression model is shown in Table 3.3.

The regularization parameter (penalty) refers to whether the subsequent penalty term is chosen as L1 regularization or L2 regularization. Optimization method parameter (solver): refers to the optimization method of logistic regression loss function, among them, liblinear is implemented using open-source libraries and internally uses coordinate descent method to iteratively optimize the loss function; Sag (Random Average Gradient Descent), which is a fast improvement variant of SDG, usually has significant advantages for large-scale data. Overall, we have observed significant improvements in its performance compared to the KNN model, especially in terms of accuracy. A relatively good model achieved a P of 0.690, but overall, the distance from us to truly achieve a good prediction of the outcome of the competition is still quite far; At the same time, we noticed that even with the best and shortest parameters of sag and L2, it still took nearly 4 hours to construct the model

Table 3.4: Support Vector Machine (SVM) Model Results

Group number	Kernel function	Kernel function parameters γ	regulari--zation parameter	Penalty coefficient C	Accu--racy P	Recall rate R	F1 value
1	linear	not have	L1	1	0.606	0.443	0.512
2	linear	not have	L1	0.1	0.639	0.457	0.533
3	linear	not have	L2	0.1	0.611	0.441	0.511
4	rbf	0.05	not have	0.1	0.586	0.431	0.497
5	rbf	0.01	not have	0.1	0.596	0.433	0.502
6	sigmoid	0.05	not have	0.1	0.634	0.456	0.531
7	sigmoid	0.01	not have	0.1	0.634	0.456	0.531

Table 3.5: xgboost Model Results 1

Group number	Data format	Learn--ing Rate	Loss para--meters	Maximum tree depth parameter Max-depth	Iterations num_ boost _round	Accu--racy P	Recall rate R	F1 value
1	0-H	0.2	10	6	800	0.696	0.604	0.647
2	0-H	0.2	10	10	800	0.711	0.612	0.658
3	0-H	0.2	5	15	800	0.721	0.616	0.665
4	0-H	0.05	5	15	1600	0.723	0.619	0.667
5	0-H	0.05	5	15	1600	0.739	0.626	0.678
6	0-H	0.01	1	20	3200	0.781	0.659	0.715
7	0-H	0.01	1	20	3200	0.788	0.661	0.719
8	0-H	0.005	1	20	6400	0.789	0.663	0.721

and process game data exceeding 80W.

The Support Vector Machine (SVM) model is shown in Table 3.4.

Kernel function parameters γ only makes sense in the case of non-linear kernel functions; When the regularization parameter specifically corresponds to a linear kernel, is L1 or L2 regularization chosen; The penalty coefficient C refers to the penalty coefficient C in the prototype and dual forms of the SVM classification model, and all of our experiments above were conducted with 10 cross validation (l0 fold). The final result we obtained shows that when the kernel function takes a linear kernel and L1 is regularized, the effect is best, and the effect is better when C is small. However, regardless of the type of model, not only does it have a long computation time (even for linear kernels, model training takes more than 10 hours, while for non-linear kernel functions, the training time is longer), but its performance cannot be separated from logistic regression, the reason for this should be that many matches have identical lineups but different winning or losing labels, which constitute some noise points. SVM systems are well-known for their poor handling of noise points, resulting in poor performance.

Xgboost model: Due to the complexity of this model, we search for its optimal parameters in two steps.

Step 1. First, only use one hot encoded data input for parameter tuning, and strive to find a suitable set of parameters (as shown in Table 3.5).

In the above model, we can see that the maximum subtree depth remains around 20 and the learning rate is maintained η when the number of iterations is small and large, the final accuracy P, recall R, and F1 values are all more appropriate, while for the loss parameter γ is not sensitive.

Step 2. Use the hyperparameters with good performance mentioned above to input data in the form of related word vector encoding, and observe the classification effect. Among them, learning rate η= 0.01, loss parameter y=1, maximum tree depth parameter max_depth=20, maximum number of iterations num_boostround= 3200, this training takes approximately 6000 seconds, which is within an acceptable range (as shown in Ta-

Table 3.6: XGBoost Model Results 2

Group number	Data entry format	Accuracy P	Recall rate R	F1 value
1	GloVe-1	0.793	0.666	0.724
2	CBOW-1	0.794	0.673	0.729
3	skip- gram-l	0.793	0.674	0.729
4	GloVe-2	0.799	0.677	0.733
5	CBOW-2	0.806	0.686	0.741
6	skip-gram 2	0.816	0.691	0.748
7	GloVe-3	0.802	0.683	0.738
8	CBOW-3	0.812	0.688	0.745
9	skip-gram-3	0.819	0.696	0.753

Table 3.7: Results of Neural Network (ANN) Model

Group number	Data entry format	Accuracy P	Recall rate R	F1 value
1	One-hot	0.759	0.638	0.693
2	GloVe-1	0.786	0.678	0.728
3	CBOW-1	0.799	0.68	0.735
4	skip-gram 1	0.806	0.684	0.74
5	GloVe-2	0.798	0.703	0.747
6	CBOW-2	0.821	0.712	0.763
7	skip-gram-2	0.822	0.706	0.76
8	GloVe-3	0.811	0.716	0.761
9	CBOW-3	0.822	0.719	0.767
10	skip-gram-3	0.825	0.729	0.774

ble 3.6).

We can see from the above results that when the input metrics of the model are in the form of skip gram 3 word vectors, all performance metrics perform the best, and in most word vector concatenation forms, the data input generated by the skip gram method corresponds to better performance metrics.

Neural Network (ANN) Model: Due to limitations in space and model training time, some good numerical values for each parameter are directly provided here. We ultimately adopted a neural network model with two hidden layers (fully connected form), with the first layer consisting of 20 nodes and the second layer consisting of 5 nodes, all using sigmoid response functions. The training results are roughly shown in Table 3.7.

It can be seen that the performance of these models is relatively excellent, and we will discuss their comparison with other models in detail in the next section.

(3) Comparison of performance and summary of advantages and disadvantages of various models. Next, we will draw some graphs using data from various tables to facilitate our comparison of various models. For convenience and intuitiveness, we will only compare the optimal performance indicators of each model as shown in Figure 3.1.

Next, we will continue to plot the training and prediction time of each model as a graph for comparison, as shown in Figure 3.2.

We can draw the following conclusion from Figure 3.2. Using traditional one hot encoding input data, combined with traditional machine learning algorithms (excluding neural network models), the ensemble tree regression algorithm represented by xgboost performs the most brilliantly, with an accuracy level of 0.788 and the fastest computational speed, taking only 1.67 hours [9,10]. On the other hand, compared to other algorithms with strong explanatory power, the logistic regression algorithm has the highest accuracy of only 0.691 and takes 4.17 hours. The KNN algorithm, due to its simplicity, results in unsatisfactory performance and time

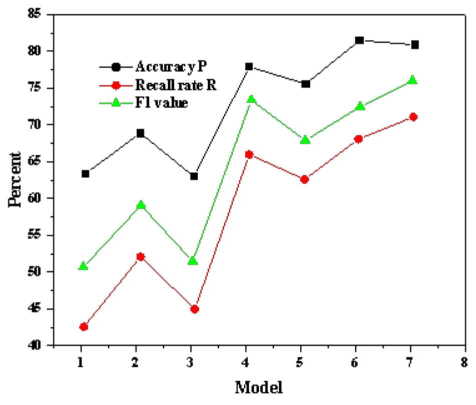

Fig. 3.1: Comparison of performance indicators of various models

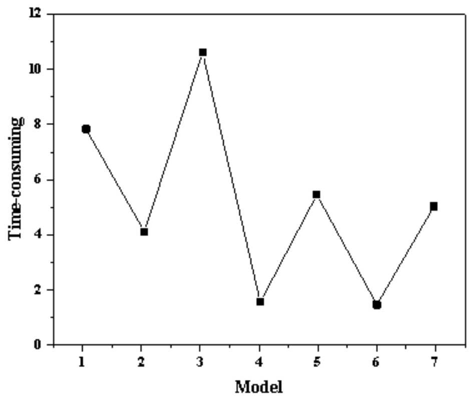

Fig. 3.2: Comparison of time consumption among different models

consumption. In general, SVM support vector machine algorithms with excellent performance not only have poor performance (accuracy of only 0.639), but also take the longest time (10.56 hours) among all algorithms. A reasonable explanation for this phenomenon should be that there are many duplicate points in the data, but the labels are different, these confusion points seriously affect the search for hyperplanes and support vectors, resulting in low model performance. The new data format is generated by transforming the data into a word vector pattern. For neural network models, the best performing group of neural network models achieved an astonishing accuracy P of 0.825, with an error rate (1-P) of only 0.177. Compared to the accuracy P of the original neural network combined with one hot encoding (P=0.693), the corresponding error rate (1-P) was 0.309, which is equivalent to an improvement of 42.87%. Even for the already impressive xgboost algorithm, using only word vectors as an improvement method has increased the accuracy P from the original 0.789 to 0.819, and the error rate (1-P) has decreased by 14.16%. From this point, it can be seen that introducing word vectors is indeed appropriate. From the above models, we can see that in terms of related time consumption, xgboost is the fastest among all models due to its algorithm's parallelism and multithreading, processing around 80W of data in about 1.5 hours. However, the processing time of the neural network (AN) model is time-consuming.

As I am using an AMD graphics card, I cannot use the GPU provided by the relevant NVIDIA graphics card for acceleration, and can only use the CPU platform for processing. Therefore, the training time is about 4 hours. According to existing online data, using GPU acceleration can increase the speed by approximately 20 times, and it only takes about 15 minutes to complete the training.

4. Conclusion. Due to the urgency of introducing modern statistical analysis methods such as machine learning into the esports industry, the author's main goal and content around Valve's DOTA2 game is to build a relatively complete DOTA2 game victory prediction system based on deep learning. We use unsupervised learning methods such as word vectors (word embeddings), combined with various supervised machine learning classifiers, in order to ultimately achieve victory or defeat prediction of competition data, and propose their own new model in it, and ultimately compare the performance of different word vectors (word embeddings), semantic combinations, and various supervised learning classifiers. The structure of the entire competition victory and defeat prediction system has been constructed to support the verification of the aforementioned theories, and each module has been functionalized and independent. Finally, a series of related experiments were conducted using the system, and the advantages and disadvantages of different models were compared, and the structure of the relevant system was ultimately determined. Although we have completed the construction of the relevant victory or defeat prediction system, in the field of machine learning, we usually have a relatively recognized view that the relevant features used by the algorithm generally give a ceiling that a machine learning model can reach; However, the selection of specific classifier models can generally affect the difficulty of approaching this ceiling. Therefore, although we have used features from primary representations such as single hot encoding and advanced representations such as word embeddings, these advanced features are all constructed based on the distribution hypothesis, i.e. words with similar contextual context, its linguistic meaning should also be similar to this hypothesis theory. The meaning that these models can express ultimately comes from the information of some words around them (in this case, hero IDs).

REFERENCES

[1] Zjavka, L. (2022). Power quality statistical predictions based on differential, deep and probabilistic learning using off-grid and meteo data in 24-hour horizon. International journal of energy research1863(8), 46.

[2] Jun, K., & Yeon, L. J. (2022). Development of a cost analysis-based defect-prediction system with a type error-weighted deep neural network algorithm. Journal of Computational Design and 899(2), 2.

[3] Khairuddin, J., Malik, A. M. A., Hiekata, K., Siow, C., & Ali, A. (2022). Web application with data centric approach to ship powering prediction using deep learning. Softw. Impacts, 1(1453), 100226.

[4] Liou, H. I., & Huang, K. C. (2023). A deep learning model for stock price prediction in swing trading. 2023 9th International Conference on Applied System Innovation 222(ICASI), 154-156.

[5] Chen, M., Kang, X., & Ma, X. (2023). Deep learning–based enhancement of small sample liquefaction data. International journal of geomechanics879(9), 23.

[6] Sun, W., Li, J., Yuan, Q., & Jiang, M. (2023). Supervised and self-supervised learning-based cascade spatiotemporal fusion framework and its application. ISPRS journal of photogrammetry and remote sensing364(Sep.), 203.

[7] Yeow, L. Y., Teh, Y. X., Lu, X., Srinivasa, A. C., Tan, E., & Tan, T. S. E., et al. (2023). Prediction of mycn gene amplification in pediatric neuroblastomas: development of a deep learning-based tool for automatic tumor segmentation and comparative analysis of computed tomography-based radiomics features harmonization. Journal of computer assisted tomography76(5), 47.

[8] Reddy, S., Reddy, K. V. N., Rao, S. N. T., & Kumar, K. V. N. (2023). Diabetes prediction using extreme learning machine: application of health systems. 2023 5th International Conference on Smart Systems and Inventive Technology 7856(ICSSIT), 993-998.

[9] Gao, N., Wang, M., & Cheng, B. (2022). Deep auto-encoder network in predictive design of helmholtz resonator: on-demand prediction of sound absorption peak. Applied acoustics86(Mar.), 191.

[10] Zheng, G., Chai, W. K., & Katos, V. (2022). A dynamic spatial-temporal deep learning framework for traffic speed prediction on large-scale road networks. Expert Systems with Application742(Jun.), 195.

Edited by: Hailong Li
Special issue on: Deep Learning in Healthcare
Received: Feb 2, 2024
Accepted: Mar 18, 2024