Volume: 9 Issue: XI Month of publication: November 2021
DOI: https://doi.org/10.22214/ijraset.2021.38982

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XI Nov 2021- Available at www.ijraset.com

Automate Labeling Of Bugs and Tickets Using Attention-Based Mechanism in Recurrent Neral Networks

Ravi Kauthale[1], Dr. Prashant Nitnaware[2]
[1, 2]Pillai College of Engineering, Maharashtra

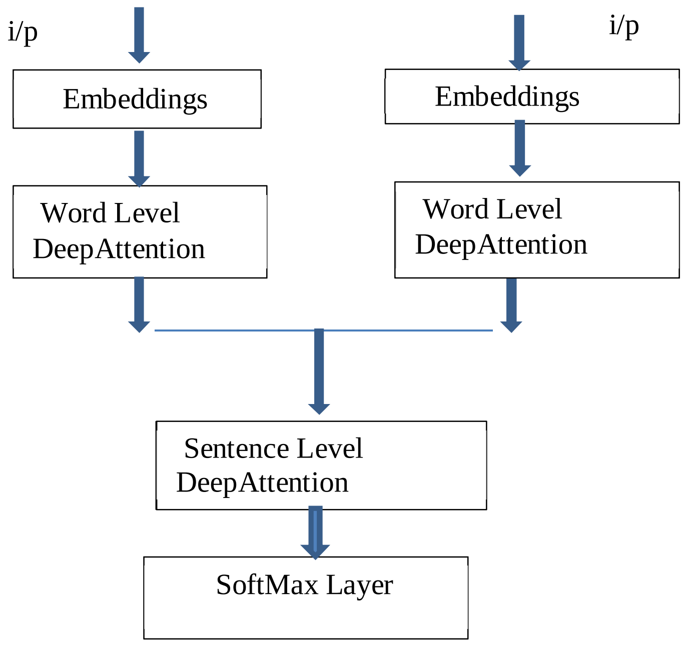
Abstract: The aim here is to explore the methods to automate the labelling of the information that is present in bug trackers and client support systems. This is majorly based on the classification of the content depending on some criteria e.g., priority or product area. Labelling of the tickets is important as it helps in effective and efficient handling of the ticket and help is quicker and comprehensive resolution of the tickets. The main goal of the project is to analyze the existing methodologies used for automated labelling and then use a newer approach and compare the results. The existing methodologies are the ones which are based of the neural networks and without neural networks. In this project, a newer approach based on the recurrent neural networks which are based on the hierarchical attention paradigm will be used.
Keywords: Automate Labeling, Recurrent Neural Networks, Hierarchical Attention, Multi-class Text Classification, GRU

I. INTRODUCTION

While handling any client support ticket, the first thing a representative agent has to do is to mark the ticket with various criterion. There could be a lot of different kinds of criteria e.g., priority, product area, platform, customer etc. Also, the service agent has to choose a specific action for the service ticket to move forward. The action could be from different groups within the company. Some tickets might need assistance from the engineering teams, or some tickets might need an onsite visit.
The main idea behind labelling the tickets is to ensure that the tickets are handled effectively. E.g., if you consider the priority of the ticket, low priority tickets are handled after the high priority ones are handled. The engineering team or onsite technician team will only get involved if the ticket is marked for intervention. Similar situations arise while handling the project management tools such as Jira or Bugzilla or ClearQuest. Team members label the tasks based on the area of work, platform on which the work is carried out, components involved etc. One such example can be the task needing work on Kernel or the task needing work on the front end of the system.
Manually labelling such customer support tickets or tasks might take days or in some cases weeks even, and it will be costly for any organization. If there is a system that labels such tickets or tasks automatically, it would reduce a lot of expediter. It would also help the organization operate speedily by taking quick actions wherever needed. For organizations this would be a strikingly important problem to address. The recent evolution in neural networks and word embedding leverages help address and resolve this problem.
Text classification is the general problem which is handled in this project. Consider a fixed set of known text classes. If we have the body of text, we need to find out the class of the body from the fixed set of classes. On the other hand, if the text classification is to be considered on the customer support tickets or project tasks, there will be some admonition. The reason behind this is that the data will contain a lot of unique fragments which can be complex for the automated system to understand e.g., the data from customer support tickets or project tasks can contain stack traces or some HTML code block or some system images etc. However, such data is usually very well formatted and structured. This kind of structuring or formatting can be helpful in some of the novel methods used to solve this problem including the method this project proposes.
Multi class text classification where the quantity of classes being predicted are more than 2 are of main interest of this method. Binary classification tasks such as sentiment analysis more commonly produce the best results. An example of sentiment analysis is the Twitter data set using which a prediction can be made if the tweet is positive or negative, another example could be using the sentiment analysis to mark an email as spam or clean. Using this method 95% or more times accurate results can be achieved. This can be credited because of the fact that the dual categories usually have a lot of clue words e.g. “good” or “grate” could mean that the tweet is positive, and “sale” would mean the email is spam. This way the classification becomes simpler.

II. METHODOLOGY

This project proposes the usage of hierarchical attention archetype with assorted GRU (Gated Recurrent Units) cell sizes, and a shallow network that would be used apace with. This approach allows networks to outperform regular hierarchical attention on datasets where straightforward term-based approach works well.
In this approach, there are multiple steps involved. Sentence hierarchy and attention vectors are used. The networks architecture is changed to use hierarchical attention blocks and shallow network is introduced which takes a word embedding as input and generated a vector. In the end in depth trainings are performed on the data sets to automate the labelling.



A. Pre-processing data

A set of preprocessing is done on the data that is used to analyse the results. This preprocessing is done across all the approaches mentioned in the paper. One of the most important activity which needs to be carried out which applying any machine learning is data cleaning. This activity should similarly be done on the data received in customer support tickets and project tasks. The data set used in this paper has broadly two categories of data i.e stack trace or error messages and HTML snippets. This information at times only adds noise to the date and should be cleaned. Below mentioned steps are used to clean the data:

1) Casting everything to lower case
2) Removing stop words
3) Filtering data set specific garbage using custom regular expressions

B. Model Building

This approach comprises of two ideas:

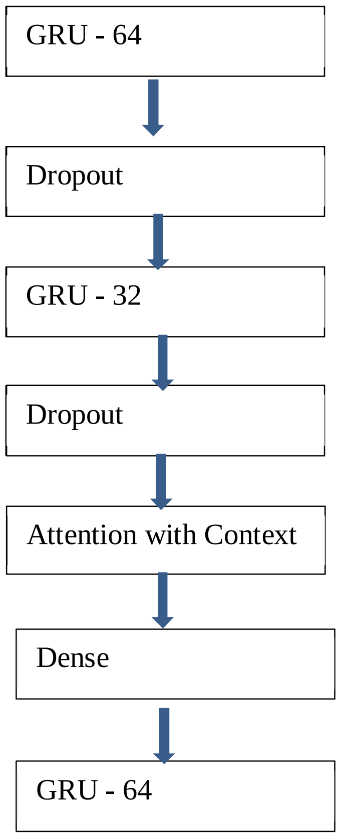
1) Utilizing sentence hierarchy
2) Utilizing attention vectors

By sentence hierarchy it is meant that one RNN will accept word embeddings from a particular sentence as input and enumerate additional vector which will act as the characterization of that sentence. And then the seconds RNN will use the sentence vectors and quantify a concluding vector for the document. The final vector will be fed to SoftMax layer to obtain final probabilities. This paradigm will work nicely as the language is structured in sentences.
Documents in a dataset can frequently follow structured pattern e.g., the most predominant information is usually found at the end of the document. This makes a good case to have pertinent coefficients for outputs of both word and sentence encoders. This task is achieved by instigating attention vectors which are marked as us and uw in the figure. These attention vectors are divided across all outputs at their level, and they are trained alongside other parts of the model. When it comes to combining sentence or word vectors into one, the coefficients which are going to be used will be a dot product of a suitable attention vector with sentence or word.

In this way the attention vector acts as "the ideal vector" which if present would achieve perfect score. Using attention vectors in this network makes a lot of sense as data is often well formatted with a noteworthy portion of items in the Linux bugs dataset filling a pre-defined template for their bug report. These kinds of scenarios are best to be used with attention-based mechanisms.
The concepts of hierarchy for detection and classification are previously used as well e.g., in visual recognition.

C. Network Architecture

Depending on the results as mentioned in following sections, the hierarchical attention works best on the datasets which follow some structure. It is also evident that hierarchical attention does not produce good results for dataset which are not well structured. To solve this problem two changed are proposed.
The first change involves using several attention blocks similar to the ones in Fig II with each of them having a different GRU cell size. Hierarchical attention uses GRU cells instead of the more common Long short-term memory (LSTM) cells, which is turn produces higher performances although with a small margin. The architecture of one such block is portrayed in below



III. TRAINING DETAILS

For data training below mentioned cross-entropy loss function is used.

$$L_{y'}(y) := -\sum y'_i \log(y_i)$$

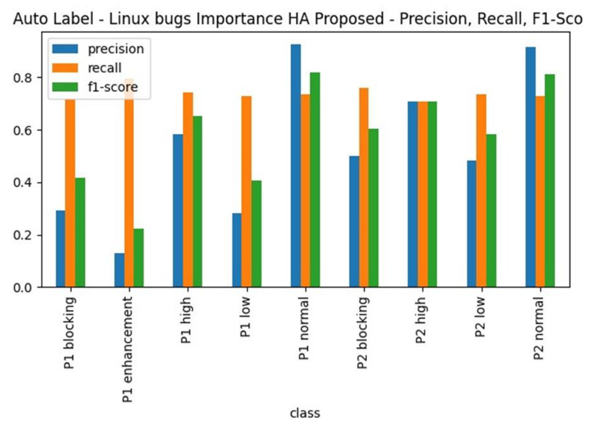
where yi is the predicted probability value for class i and y′i is the accurate probability for that class.
Training deep neural networks can be a meticulous task. In this section details about the training used on the datasets are mentioned. First of all, dropout is widely used to avoid overfitting. Overfitting can happen very easily depending on the size of the datasets. Even more captivating fact is that dropout probability to work best when set at around ½ was found, which is
higher than typical values. In this solution, dropout layers are between any two RNNs of affine layer in this solution as depicted in Fig III. Secondly, RMSprop is also used for optimization. Lastly Word2vec library is used to compute the word embeddings.

IV. RESULTS AND DISCUSSION

Two live data sets are used to evaluate the performance of the proposed model. One if from the Arch Linux bug tracking tool and the other one is from the Chromium bug tracker tool.

A. Arch Linux bugs classification based on Priority/importance:

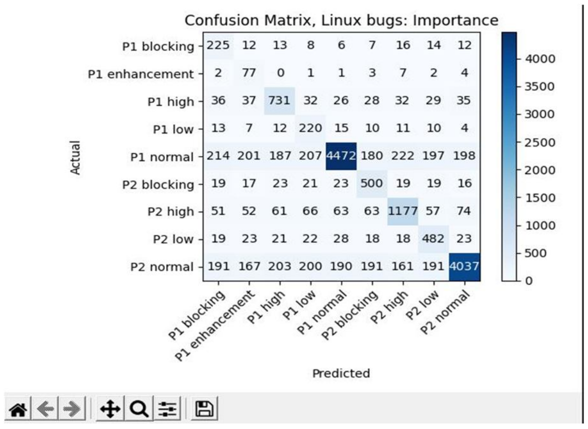
Accuracy: 73.2%



Logistic Regression

	Precision	Recall	F1-score	support
P1 blocking	0.292	0.719	0.416	313
P1 enhancement	0.130	0.794	223	97
P1 high	0.584	0.741	0.654	986
P1 low	0.283	0.728	0.408	302
P1 normal	0.927	0.736	0.820	6078
P2 blocking	0.500	0.761	0.604	657
P2 high	0.708	0.707	0.708	1664
P2 low	0.482	0.737	0.582	654
P2 normal	0.917	0.730	0.813	5531
Accuracy			0.732	16282
Macro avg	0.536	0.739	0.581	16282
Weighted avg	0.816	0.732	0.759	16282

Confusion Matrix(Logistic Regression)

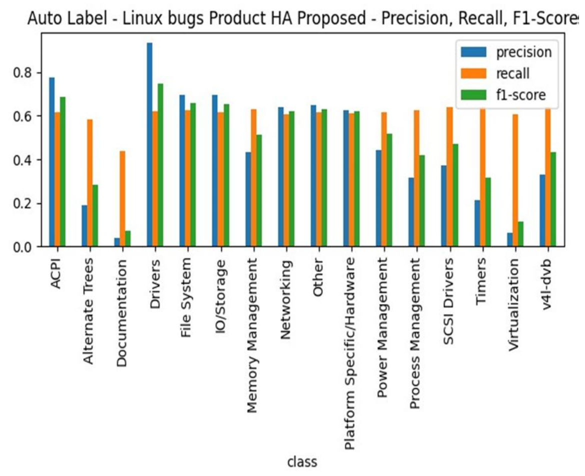


B. ARCH Linux Bugs Classification Based On Product Type

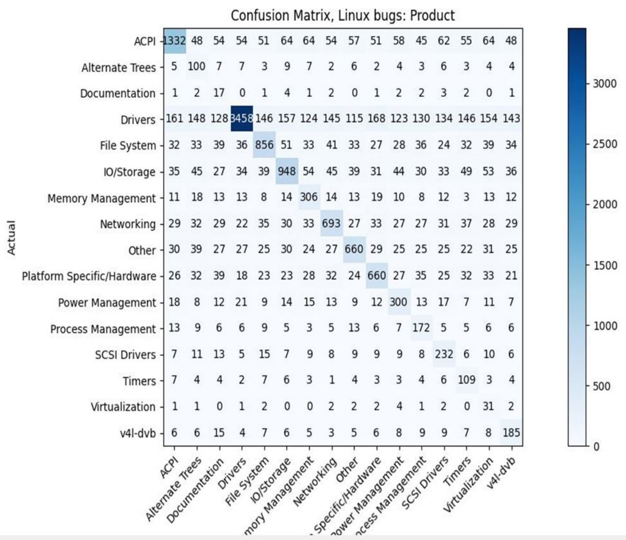
F1-Score Accuracy: 61.8%

Logistic Regression: Classification report

	Precision	Recall	F1-score	support
ACPI	0.777	0.616	0.687	2161
Alternate Trees	0.187	0.581	0.282	172
Documentation	0.040	0.436	0.072	39
Drivers	0.933	0.620	0.745	5580
File System	0.693	0.623	0.656	1374
IO/Storage	0.693	0.615	0.652	1542
Memory Management	0.432	0.628	0.512	487
Networking	0.638	0.607	0.622	1142
Other	0.650	0.616	0.632	1071
Platform Specific/Hardware	0.623	0.612	0.618	1078
Power Management	0.442	0.617	0.515	486
Process Management	0.314	0.623	0.417	276
SCSI Drivers	0.371	0.637	0.469	364
Timers	0.212	0.641	0.318	170
Virtualization	0.064	0.608	0.115	51
V4l-dvb	0.329	0.640	0.434	289
Accuracy			0.618	16282
Macro avg	0.462	0.608	0.484	16282
Weighted avg	0.726	0.618	0.652	16282



Confusion Matrix



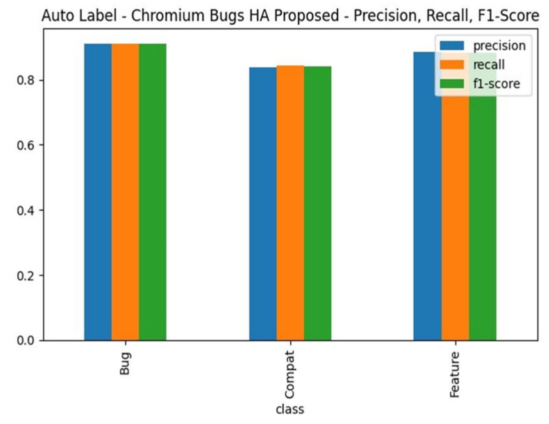
	ACPI	Alternate Trees	Documentation	Drivers	File System	IO/Storage	Memory Management	Networking	Other	Platform Specific/Hardware	Power Management	Process Management	SCSI Drivers	Timers	Virtualization	v4l-dvb
ACPI	1332	48	54	54	51	64	64	54	57	51	58	45	62	55	64	48
Alternate Trees	5	100	7	7	3	9	7	2	6	2	4	3	6	3	4	4
Documentation	1	2	17	0	1	4	1	2	0	1	2	2	3	2	0	1
Drivers	161	148	128	3458	146	157	124	145	115	168	123	130	134	146	154	143
File System	32	33	39	36	856	51	33	41	33	27	28	36	24	32	39	34
IO/Storage	35	45	27	34	39	948	54	45	39	31	44	30	33	49	53	36
Memory Management	11	18	13	13	8	14	306	14	13	19	10	8	12	3	13	12
Networking	29	32	29	22	35	30	33	693	27	33	27	27	31	37	28	29
Other	30	39	27	27	25	30	24	27	660	29	25	25	25	22	31	25
Platform Specific/Hardware	26	32	39	18	23	23	28	32	24	660	27	35	25	32	33	21
Power Management	18	8	12	21	9	14	15	13	9	12	300	13	17	7	11	7
Process Management	13	9	6	6	9	5	3	5	13	6	7	172	5	5	6	6
SCSI Drivers	7	11	13	5	15	7	9	8	9	9	9	8	232	6	10	6
Timers	7	4	4	2	7	6	3	1	4	3	3	4	6	109	3	4
Virtualization	1	1	0	1	2	0	0	2	2	2	4	1	2	0	31	2
v4l-dvb	6	6	15	4	7	6	5	3	5	6	8	9	9	7	8	185

C. Chromium Bugs Classification Based on Class

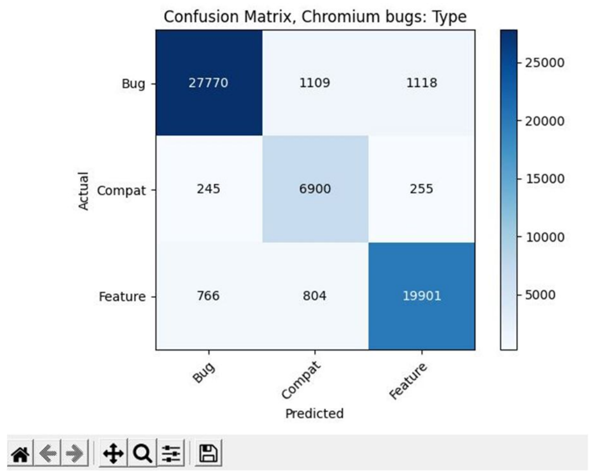
F1-Score Accuracy: 92.7%

Logistic Regression:

	Precision	Recall	F1-score	support
Bug	0.965	0.926	0.945	29997
Compatibility	0.783	0.932	0.851	7400
Feature	0.935	0.927	0.931	21471
Accuracy			0.927	58868
Macro avg	0.894	0.928	0.909	58868
Weighted avg	0.931	0.927	0.928	58868



Confusion Matrix



V. CONCLUSIONS

In this project, a newer approach to solving the complex multi-class classification problem was proposed. The model consisted of various concepts which perform poorly individually but when they are stacked together as proposed, the results outperform the results of the previous methods.

The proposed model was applied to two real world datasets, one from the Arch Linux bug tracker and other one being the Chromium bug tracker. The existing methodologies and models were also applied on the same dataset and the results were compared. Evidently from the results, the proposed model outperformed the results of most of the existing other models.

REFERENCES

[1] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy, "Hierarchical Attention Networks for Document Classification", in N16-1174 NAACL 2016
[2] Sherstinsky, Alex, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network", 2020, In Physica D: Nonlinear Phenomena, Volume 404, article id. 132306
[3] Haojin Hu, Mengfan Liao, Chao Zhang, Yanmei Jing, "Text classification based recurrent neural network", 2020, IEEE 978-1-7281-4323-1
[4] T. L Burrows, M. Niranjan, "The use of recurrent neural networks for classification", 1994 in IEEE Workshop on Neural Networks for Signal Processing IV
[5] K Gouhara, T. Watanabe, Y. Uchikawa, "Learning process of recurrent neural networks", 1991, 0-7803-0227-3
[6] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, "On the difficulty of training recurrent neural networks", 2013, 30th International Conference on Machine Learning, PMLR 28(3):1310-1318, 2013.
[7] Changshun Du, Lei Huang, "Text Classification Research with Attention-based Recurrent Neural Networks", 2018 IJCCC, 1841-9844
[8] Zhaoyang Niu, Guoqiang Zhong, Hui Yu, "A review on the attention mechanism of deep learning", 2021 NEUCOM 233562906
[9] Adrian Hernandez, Jose M. Amigo "Attention Mechanisms and their applications to complex systems", 2021, ENTROPY 23(3), 283
[10] Andy Brown, Aaron Tuor, Brian Hutchinson, Nicole Nichols, "Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection",2018, MLCS'18 Article
[11] Michal Perelkiewicz and Rafal Poswiata, "Text Language Identification Using Attention-Based Recurrent Neural Networks", 2019, ICAISC, 978-3-030-20912-4_18

[12] Spyridon Kardakis, Isidoros Perikos, Foteini Grivokostopoulou, Ioannis Hatzilygeroudis, “Examining Attention Mechanisms in Deep Learning Models for Sentiment Analysis” Applied Sciences, app11093883
[13] Sue Han Lee, Hervé Goëau, Pierre Bonnet, Alexis Joly, “Attention-based Recurrent Neural Network for plant disease classification”, 2020, PMC 33381135
[14] Bohang Chen, “Deep Neural Networks for multi-class sentiment classification”, 2018, in the proceedings SPE Annual Technical Conference and Exhibition
[15] Chuanlong Yin “A deep learning approach for intrusion detection Using Recurrent Neural Networks”, IEEE 2017.2762418
[16] Pengfei Liu, Xipeng Qiu, Xuanjing Huang, “Recurrent Neural Network for Text Classification with Multi-task learning.”, 2016 IJCAI 2016
[17] Daler Ali, Malik Muhammad Saad Missen, Mujtaba Husnain, “Multiclass Event Classification from Text”, 2020
[18] Rahul Dey, Fathi Salem “Gate Variants Gated Recurrent Unit Neural Networks”, 2017, IEEE 1558-3899
[19] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, “Empirical evolution of Gated Recurrent Neural Networks on sequence modelling”, 2014, Presented in NIPS 2014 Deep Learning and Representation Learning Workshop
[20] Junaid Hassan, Umar Shoaib, “Multi-class review classification using Deep Recurrent Neural Network”, 2020, in Neural Processing Letters 1031–1048
[21] Indu Chawla, Sandeep K Singh, “Automate labelling of issue reports using semi supervised approach”, 2018 JCM-180779
[22] Ankit Kumar; Reshma Rastogi nee Khemchandani, “Self-Attention Enhanced Recurrent Neural Networks for Sentence Classification”, 2018 978-1-5386-9276-9
[23] Gang Hu; Yi Ye; Yin Zhang; M. Shamim Hossain, “Improved Recurrent Neural Networks (RNN) Based Intelligent Fund Transaction Model”, 2019, 978-1-7281-0960-2
[24] Peikang Lin, Xianjie Mo, Guidong Lin, Liwen Ling, Tingting Wei, Wei Luo, “A News-Driven Recurrent Neural Network for Market Volatility Prediction”, 2017, 978-1-5386-3354-0
[25] Minh-Tri Nguyen, Duong H. Le, Takuma Nakajima, Masato Yoshimi, Nam Thoai, “Attention-Based Neural Network: A Novel Approach for Predicting the Popularity of Online Content”, 2019, 978-1-7281-2058-4
[26] Haizhen H, Jiehan Li, “Attention-Based Deep Neural Network and Its Application to Scene Text Recognition”, 2019, 978-1-7281-2184-0
[27] Li SiChen, “A Neural Network Based Text Classification with Attention Mechanism”, 2019, 978-1-7281-3299-0
[28] Xianlun Tang, Yingjie Chen, Yuyan Dai, Jin Xu, Deguang Peng, “A Multi-scale Convolutional Attention Based GRU Network for Text Classification”, 2019, 978-1-7281-4094-0
[29] Meina Song, Qing Liu; E Haihong, “Deep Hierarchical Attention Networks for Text Matching in Information Retrieval”, 2018, 978-1-5386-5738-6
[30] E A Nismi Mol, M B Santosh Kumar, “Study on Impact of RNN, CNN and HAN in Text Classification”, 2020, 978-1-7281-6453-3
[31] Piyush Singh Parmar, P K Biju, Mani Shankar, Nalinadevi Kadiresan, “Multiclass Text Classification and Analytics for Improving Customer Support Response through different Classifiers”, 2018, 978-1-5386-5314-2
[32] Panuwat Assawinjaipetch, Kiyoaki Shirai, Virach Sornlertlamvanich, Sanparith Marukata , “Recurrent Neural Network with Word Embedding for Complaint Classification”, 2016 in the proceedings of (WLSI/OIAF4HLT2016)
[33] Ivars Namatevsac, Kaspars Sudarsb, Inese Polakaa, “Automatic data labeling by neural networks for the counting of objects in videos”, 2019, in ICTS 2018, Vol 149, 2019
[34] , P., Qiao, M. & Jadav, D., 2018. Large Scale Predictive Analytics for Hard Disk Remaining Useful Life Estimation. IEEE.
[35] B.Y, V. & Borah, A., 2016. Enhanced Rules Framework for Predicting Disk Drives Failures. International Journal of Computer Science and Mobile Computing.
[36] Hamerly, G. & Elkan, C., 2003. Bayesian Approaches to Failure Prediction for Disk Drives. IEEE.
[37] He, X., Wang, Z. & Zhang, J., 2011. Research on security of hard disk firmware. IEEE.
[38] He, Z., Yang, H. & Xie, M., 2012. Statistical modeling and analysis of hard disk drives (HDDs) failure. IEEE.
[39] Hughes, G., 2005. Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application. IEEE.
[40] Jing Li, X., 2014. Hard Drive Failure Prediction Using Classification and Regression Trees. IEEE, p. 12.
[41] Li, J., Li, Z. & Wang, G., 2014. Hard Drive Failure Prediction Using Classification and Regression Trees. 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).
[42] Li, W., 2017. Proactive Prediction of Hard Disk Drive Failure. IEEE.
[43] Mahdisoltani, F., Stefanovici, I. & Schroeder, B., 2017. Improving Storage System Reliability with Proactive Error Prediction. IEEE.