Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Guadalajara

School of Engineering and Sciences

The more the better? A comparative review of how hyperparameter values impact the performance of Rule-based, Neural Networks and Deep Learning models

Presented by

Karen P. Navarro Arroyo

Submitted to the
School of Engineering and Sciences

Guadalajara, Jalisco, Dec, 2024

The more the better? A comparative review of how hyperparameter values impact the performance of Rule-based, Neural Networks and Deep Learning models

by

Karen P. Navarro Arroyo

Abstract

In Natural Language Processing (NLP) and Machine Learning, the accuracy of a model depends heavily on the value assigned to its hyperparameters. This paper aims to demonstrate whether assigning high values to these hyperparameters (e.g. more epochs, which would be translated as the number of times the model iterates over the entire training set) improves its accuracy and how different combinations of values in the hyperparameters impact performance.

These tests were performed using three different approaches: rule-based, neural networks (NN) and deep learning (DL), which were trained with the same datasets for sentiment analysis. Each of these models used different sets with different values for their hyperparameters in order to then make a comparative analysis between their accuracy results and training time, what values they used and how this impacted these results and whether using high values really improves accuracy.

The results showed that increasing the value of a hyperparameter generally does not improve accuracy and, in some cases, also negatively impacts training times. For example, epochs (a parameter shared by all three models) result in a model with poor performance if they are extended too long.

The results of these experiments highlight the importance of finding optimal values for hyperparameters taking into account which model is being used, what tasks it must perform, and what computational resources are available.

List of Figures

List of Tables

Contents

Chapter 1

Motivation

In the field of Natural Language Processing (NLP) and Machine Learning, hyperparameter tuning plays a crucial role in the performance of a model. It may at first seem feasible to use high values for certain hyperparameters, such as epochs, batch size, or the number of hidden layers in a neural network, in order for the model to give more accurate results. Not only may this not be universally applicable, but it can also impact training time and waste computational resources. This is why it is important to look for when a hyperparameter value should be *high* and when it should not, also considering that models generally use a number of hyperparameters that can be tuned in an infinite number of combinations.

Strategies to address this problem have already been developed, such as Bayesian optimization [10] [8], hyperband [6], or restricted tree-structured Parzen estimator (c-TPE) [9]. The problem with these is that they aim to find the "best" combination of values for the hyperparameters, which may not always be the best option if few computational resources are available, the task that the model seeks to solve does not demand very high precision but does require speed, among others that will depend on the particular case. Furthermore, while these approaches have advanced the field, the assumption that higher values produce better results has not been rigorously questioned in many cases. Consequently, there is a gap in understanding the nuanced effects of hyperparameter scaling on accuracy, especially across different datasets and model architectures.

This study aims to investigate whether increasing hyperparameter values consistently translates to higher accuracy. This will be demonstrated by testing different combinations of hyperparameter values (sets) for each planned model to obtain contrasting results and analyzing these results for accuracy, training, testing, validation and total time, identifying patterns and relationships between the values and the outputs. Our findings will contribute to a deeper understanding of hyperparameter tuning, ultimately enabling more efficient and effective model optimization strategies.

Chapter 2

Literature Review

Various studies related to parameter optimization have been carried out over the years and from different approaches. These are the main approaches that relate to the objective of this research:

- **Hyperparameter optimization methods**: Some of the most notable methods of this approach are grid search, random search, Bayesian optimization, and gradient-based methods. Bergstra and Bengio highlighted that random search often outperforms grid search due to its ability to explore hyperparameter spaces more efficiently [2]. Similarly, Snoek et al. [7] introduced Bayesian optimization as a way to find optimal hyperparameter configurations with fewer iterations. These studies provide valuable insights but primarily focus on optimization techniques rather than the effects of scaling individual hyperparameters.
- **Impact of hyperparameter scaling on model performance**: Research on hyperparameter scaling often assumes a direct correlation between larger values and better performance. An example is given by Goodfellow et al. [3] who suggested that deeper architectures generally achieve higher accuracy, although this comes with the risk of overfitting and diminishing returns. On the other hand, Keskar et al. [11] showed that excessively large batch sizes can degrade generalization performance, challenging the assumption that "bigger is better."

Research so far in this field generally focuses on finding the most optimal values of hyperparameters to find the best results, however, there is still a gap in understanding the effects of scaling these values.

Chapter 3

Experimentation

3.1 Models and dataset

For this experiment, three different approaches were used to process datasets of text labeled with its respective emotion:

- Rules-Based: This model process the training dataset to build an ”emotion lexicon” using a counter to record the frequency of each word in the texts associated with an emotion. The most frequent words for each emotion are then selected using the hyperparameter counter_most_common. To do the prediction, it reads the test dataset for each sentence and a score is accumulated for each emotion based on the number of words in the text that match words in the lexicon for that emotion. The emotion with the highest score will be the one returned as the prediction result. Accuracy is measured by comparing the correct predictions to the total number of predictions (Figure 3.1).

- Neural Networks: This NN model is based on supervised learning and uses text representations via embeddings. The training dataset is processed to break the text into tokens and convert the words into indices, which are transformed into vectors in the embedding layer. These vectors are averaged (now that the words are represented as numbers) to create another vector representing the entire text. This vector is processed by the linear layer, which generates a score for each emotion class. Finally, the class with the highest score is selected as the model's prediction (Figure 3.2).

- Deep Learning: This model uses LSTM and embeddings. It first processes the text to convert words into indices using a vocabulary dictionary. The indices are transformed into vectors in the embeddings layer. These sequences of vectors are processed in the LSTM layer to capture the context of words within a sequence. The final layer calculates probabilities for emotions using Softmax, and the model predicts the emotion with the highest probability.

The datasets for the experiments with the three models were the same. They have the format ”Sentence; emotion” (in this case, the emotions are the labels). Each of the three datasets has a different function; training (16,000 data), testing (2,000) and validation (2,000). NN and DL approaches use all three, but Rules-based only uses the training and testing data.

Both the model codes and the datasets were provided by the Institution [5].

3.2 Methodology

For each of the models, different tests were performed with different combinations of values for each hyperparameter. The hyperparameters and data outputs of each model are described in Table 3.1.

Each of these variables represents the following:

- increment_step: Amount to increase per iteration
- max_word_count (MWC): Maximum number of most common words to be included in the lexicon.
- MAX_SEQ_LENGTH: Maximum number of words to be considered in each text. If a text has more words than this limit, it is truncated, if it has less, it is padded with the special token <PAD>.
- EPOCHS: Number of times the entire training set will be repeated.
- EMBED_DIM, EMBEDDING_DIM: The dimension of the vectors representing each word in the vocabulary.
- BATCH_SIZE: Number of samples processed simultaneously.
- HIDDEN_DIM: The size of the internal representation that the LSTM uses to capture temporal and contextual data in the sequences.

Model	Hyperparameters	Output
Rules-based	increment_step max_word_count	Accuracy (%) Training time (s) Testing time (s)
Neural Networks	MAX_SEQ_LENGTH EPOCHS EMBED_DIM	Accuracy (Testing) (%) Accuracy (Validation) (%) Training time (s) Testing time (s) Validation time (s) Loss (List)
Deep Learning	MAX_SEQ_LENGTH BATCH_SIZE EMBEDDING_DIM HIDDEN_DIM EPOCHS	Accuracy (Testing) (%) Accuracy (Validation) (%) Training time (s) Testing time (s) Validation time (s) Loss (List)

Table 3.1: Hyperparameters and outputs for each model

3.3 Experimental Setup

All the model codes were executed in the same environment, which was Google Colab with default settings. All the models use the same datasets mentioned in the previous section.

3.4 Procedure

Each model was treated differently depending on its hyperparameters:

- Rules-Based: The sets are divided into two parts: Part A, which uses a MWC of 1000, and Part B, which uses 2000. This decision was made because the accuracy results were the same for each iteration regardless of the value of this variable, i.e. it only gives longer results but not different ones. increment_step was kept constant so that a comparison could be made in training and testing times, which were the outputs that could vary in results.

  In total, the model was run six times; 3 in Part A and 3 in Part B.

- Neural Networks: Different value combinations were tested for each set to primarily evaluate accuracy and training time results. The value ranges for each hyperparameter were:

  MAX SEQ LENGTH: 10-100

  EPOCHS: 10-500

  EMBED DIM: 10-300

- Deep Learning: Similar to NN, the ranges of each hyperparameter were:

  MAX SEQ LENGTH: 25-100

  BATCH SIZE: 8-64

  EMBEDDING DIM: 50-300

  HIDDEN DIM: 64-512

  EPOCHS: 5-20

3.5 Data Collection and Analysis

Each model outputs a series of results mentioned in Table 3.1. These were compiled in an Excel sheet (02_DataAnalysis >Data.xlsx) and then processed and graphed using a Python code (available in the repository (02_DataAnalysis >DataAnalysis.ipynb) [1])

Figure 3.1: Rule-based model flowchart

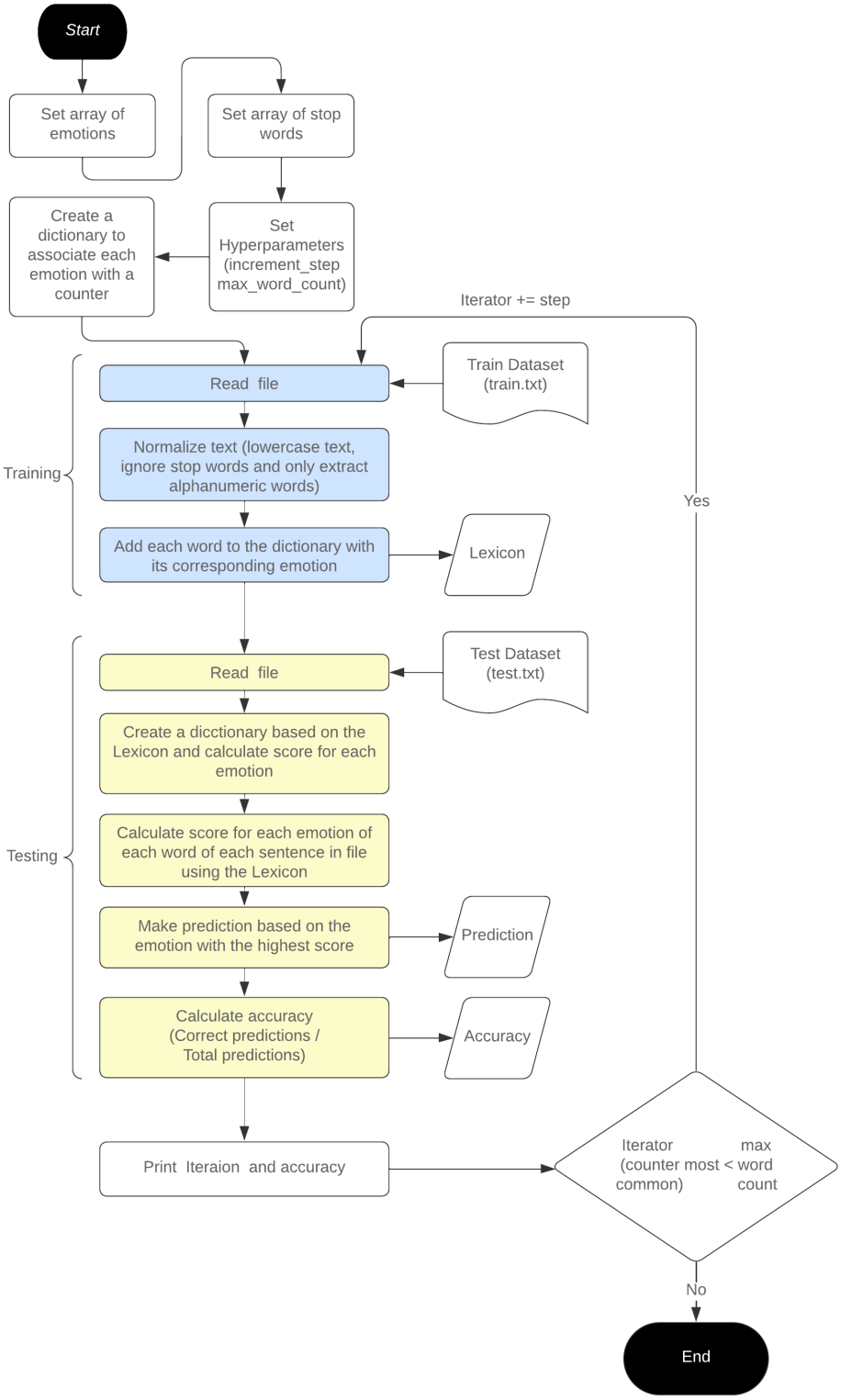
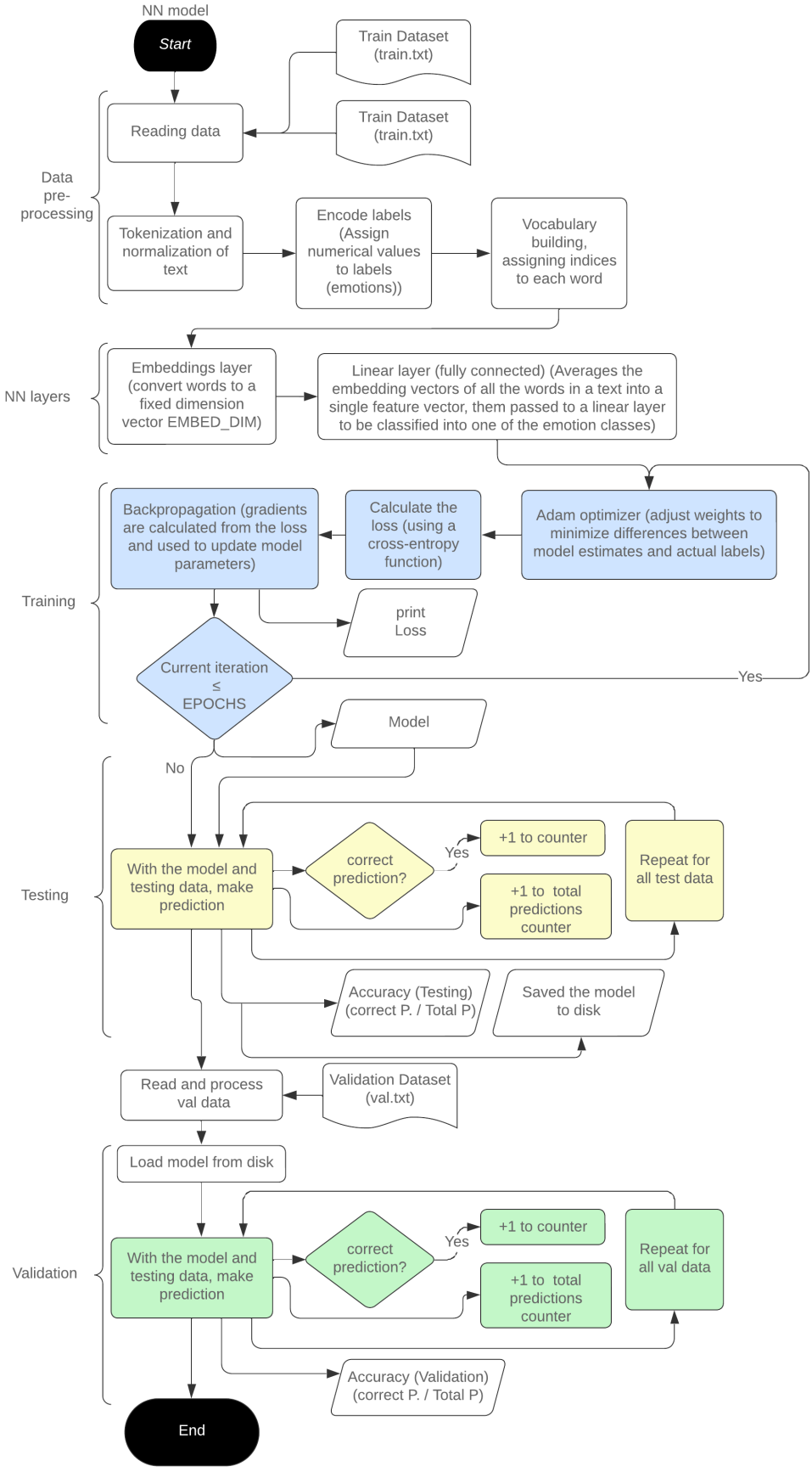


Figure 3.2: Neural Networks model flowchart



Chapter 4

Results

The complete results can be found in the repository of this research (02_DataAnalysis >Data.xlsx) [1]

4.1 Rules-based

In each trial, the combination of hyperparameters had zero impact on the results, each given iteration invariably returned the same accuracy. The largest MWC used was 2000, despite this, the point at which the model reached maximum accuracy was at MWC=300 and 350 with 60.6% (Table 4.7), with an average total time of 0.676 and 0.717 respectively.

4.2 Neural Networks

The results with the highest accuracy were given by set 10, with 87.75% and 87.75% in test and validation accuracy respectively (Figure 4.8). This set also had the lowest training time, 63.04 seconds (Figure 4.9), well below the average which is 340.47 seconds (Table 4.1).

It is worth mentioning that in set 10, values for the hyperparameters Epochs and Embedding Dimensions were considerably lower than the others.

4.3 Deep Learning

The highest accuracy was achieved in set 13, with 91.5% and 91.20% for testing and validation respectively (Figure 4.2), although it also had the longest training time, 4371.71 s (1.2144 h), well above the average, 820.60 s (13.6767 min). Five other sets gave very close results; 4, 5, 8 and 10, and with a shorter training time compared to set 10, as shown in Figure 4.11. A notable behavior given the results of the model is that it is either highly accurate (based on Accuracy (Testing) 88.70 - 91.50%) or it is very bad (34.20 - 34.80%), as can be seen in Figure 4.10.

Hyperparameters	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7
MAX_SEQ_LENGTH	10	10	25	25	50	10	10
EPOCHS	10	20	50	200	100	50	100
EMBED_DIM	50	10	100	100	300	100	300
Results							
Accuracy (Testing)	69.80%	66.70%	81.05%	77.60%	81.55%	66.15%	63.15%
Accuracy (Validation)	68.55%	66.85%	83.80%	79.75%	83.60%	66.05%	63.70%
Training time (s)	7.54	10.83	112.82	538.08	829.48	99.53	416.66
Testing time (s)	0.04	0.04	0.04	0.04	0.09	0.08	0.06
Validation time (s)	0.02	0.03	0.03	0.03	0.06	0.09	0.06
Total time (s)	8.01	11.3	113.34	538.67	830.22	100.01	417.11

Hyperparameters	Set 8	Set 9	Set 10	Set 11	Set 12	Set 13
MAX_SEQ_LENGTH	25	25	50	50	100	10
EPOCHS	20	100	20	50	50	500
EMBED_DIM	50	200	100	200	300	100
Results						
Accuracy (Testing)	84.45%	78.45%	87.75%	85.50%	86.05%	61.85%
Accuracy (Validation)	86.40%	80.85%	87.75%	87.15%	87.85%	62.90%
Training time (s)	32.9	506.93	63.04	419.79	495.23	893.3
Testing time (s)	0.06	0.08	0.08	0.08	0.12	0.06
Validation time (s)	0.06	0.09	0.08	0.09	0.13	0.06
Total time (s)	33.52	507.74	63.73	420.75	496.35	898.34

Table 4.1: Table of results for the NN model

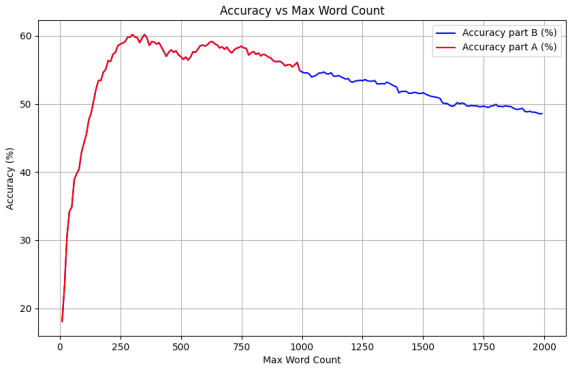


Figure 4.1: Rules-based model Accuracy vs MWC results

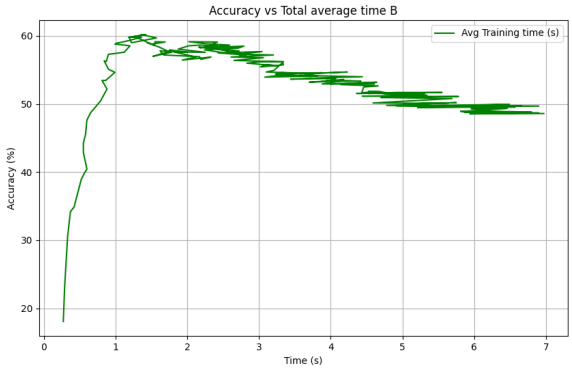


Figure 4.2: Rules-based model Accuracy vs Total time

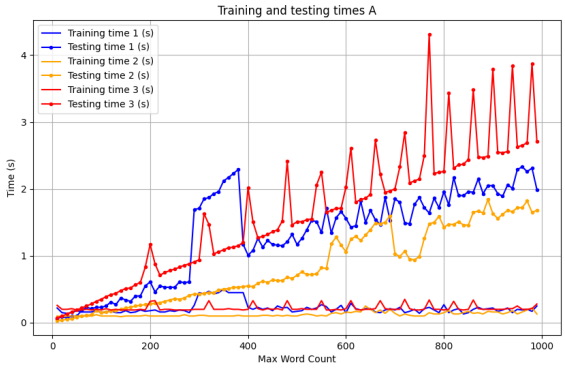


Figure 4.3: Rules-based model Total time vs MWC (Part A)

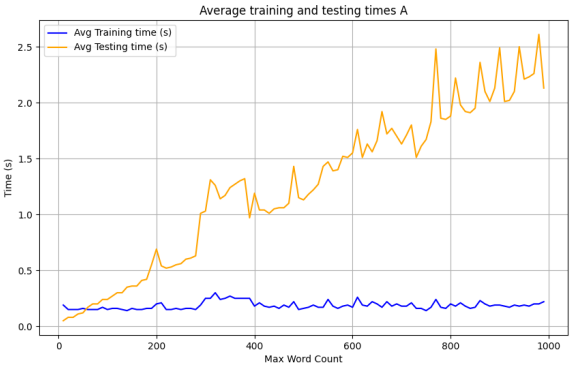


Figure 4.4: Rules-based model Average times vs MWC (Part A)

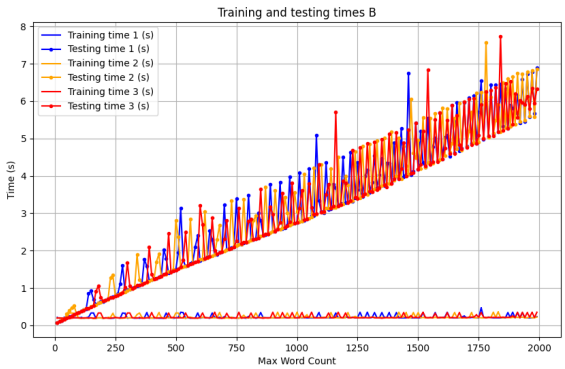


Figure 4.5: Rules-based model (Part B)

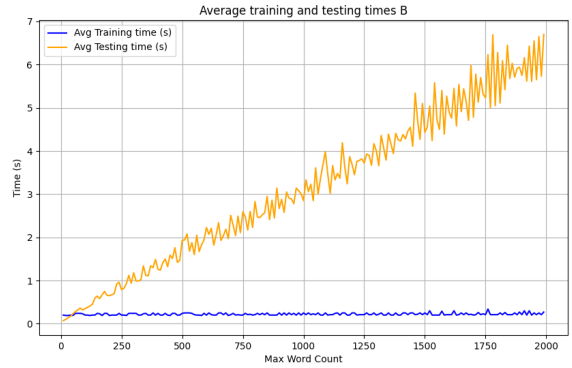


Figure 4.6: Rules-based model Average times vs MWC (Part B)

Figure 4.7: Rule-based model results table

		Part A (MWC=1000)						Part B (MWC=2000)					
		Set A1		Set A2		Set A3		Set B1		Set B2		Set B3	
MWC	Accuracy (%)	training time (s)	testing time (s)	training time (s)	testing time (s)	training time (s)	testing time (s)	training time (s)	testing time (s)	training time (s)	testing time (s)	training time (s)	testing time (s)
10	18.05	0.22	0.06	0.1	0.03	0.26	0.07	0.21	0.07	0.2	0.07	0.19	0.07
20	23.3	0.15	0.09	0.09	0.04	0.2	0.1	0.19	0.1	0.19	0.1	0.2	0.11
30	30.5	0.14	0.07	0.1	0.05	0.2	0.13	0.19	0.13	0.19	0.15	0.19	0.13
40	34.2	0.15	0.1	0.1	0.06	0.21	0.16	0.2	0.19	0.2	0.16	0.19	0.16
50	34.85	0.18	0.1	0.1	0.08	0.19	0.19	0.19	0.2	0.2	0.3	0.19	0.2
60	38.9	0.16	0.18	0.1	0.1	0.19	0.22	0.19	0.23	0.33	0.4	0.19	0.22
70	39.75	0.16	0.23	0.1	0.11	0.2	0.25	0.21	0.25	0.33	0.47	0.19	0.25
80	40.45	0.16	0.21	0.1	0.12	0.19	0.28	0.2	0.29	0.32	0.52	0.2	0.28
90	42.9	0.18	0.23	0.12	0.18	0.2	0.32	0.2	0.33	0.28	0.31	0.2	0.32
100	44.25	0.15	0.23	0.1	0.15	0.19	0.35	0.2	0.35	0.19	0.35	0.2	0.34
110	45.5	0.16	0.25	0.1	0.16	0.21	0.39	0.21	0.38	0.19	0.38	0.19	0.38
120	47.65	0.18	0.31	0.1	0.17	0.19	0.42	0.19	0.42	0.19	0.41	0.2	0.4
130	48.7	0.15	0.27	0.1	0.18	0.2	0.44	0.2	0.44	0.19	0.45	0.2	0.45
140	50.45	0.15	0.37	0.09	0.2	0.19	0.48	0.22	0.85	0.2	0.46	0.19	0.46
150	52.2	0.18	0.34	0.1	0.22	0.19	0.51	0.33	0.92	0.2	0.5	0.2	0.49
160	53.45	0.15	0.32	0.1	0.23	0.19	0.52	0.33	0.69	0.18	0.53	0.19	0.53
170	53.45	0.16	0.4	0.1	0.25	0.19	0.57	0.19	0.57	0.19	0.55	0.19	0.9
180	54.65	0.19	0.4	0.1	0.26	0.2	0.6	0.2	0.59	0.2	0.59	0.33	1.06
190	55.1	0.17	0.55	0.11	0.27	0.19	0.83	0.2	0.63	0.2	0.61	0.33	0.75
200	56.35	0.18	0.61	0.1	0.28	0.32	1.17	0.19	0.64	0.19	0.66	0.19	0.65
210	56.25	0.19	0.46	0.1	0.29	0.33	0.88	0.2	0.67	0.19	0.67	0.2	0.68
220	57.3	0.16	0.55	0.1	0.3	0.19	0.71	0.2	0.71	0.21	0.69	0.19	0.7
230	57.6	0.16	0.53	0.1	0.32	0.2	0.75	0.19	0.75	0.21	1.27	0.19	0.74
240	58.55	0.18	0.53	0.1	0.34	0.2	0.78	0.19	0.77	0.32	1.35	0.2	0.75
250	58.8	0.17	0.53	0.1	0.36	0.19	0.8	0.2	0.8	0.2	0.8	0.21	0.78
260	58.95	0.19	0.61	0.1	0.35	0.19	0.83	0.2	0.83	0.2	0.82	0.2	0.8
270	59.15	0.18	0.6	0.12	0.37	0.19	0.86	0.19	1.11	0.19	0.85	0.2	0.84
280	59.8	0.15	0.61	0.1	0.41	0.19	0.88	0.33	1.6	0.19	0.88	0.2	0.87
290	59.8	0.26	1.69	0.1	0.43	0.2	0.91	0.32	0.91	0.2	0.89	0.2	1.02
300	60.2	0.45	1.71	0.11	0.43	0.2	0.94	0.19	0.95	0.2	0.93	0.33	1.67
310	59.85	0.43	1.85	0.11	0.44	0.2	1.63	0.2	0.97	0.2	0.96	0.33	1.05
320	59.75	0.47	1.87	0.1	0.45	0.33	1.47	0.2	1	0.2	0.99	0.2	0.99
330	59	0.43	1.93	0.1	0.45	0.2	1.03	0.19	1.02	0.2	1.02	0.2	1.01
340	59.7	0.45	1.96	0.1	0.49	0.2	1.06	0.2	1.06	0.29	1.89	0.19	1.08
350	60.2	0.5	2.12	0.1	0.5	0.2	1.09	0.19	1.08	0.33	1.21	0.21	1.08
360	59.7	0.45	2.17	0.1	0.51	0.21	1.12	0.2	1.11	0.21	1.11	0.19	1.1
370	58.6	0.45	2.23	0.1	0.53	0.2	1.13	0.2	1.77	0.2	1.12	0.19	1.12
380	59.15	0.45	2.29	0.11	0.53	0.2	1.15	0.34	1.58	0.2	1.15	0.2	1.16
390	59.1	0.45	1.17	0.1	0.54	0.19	1.2	0.19	1.2	0.21	1.18	0.2	2.1
400	58.8	0.23	1.01	0.1	0.55	0.2	2.02	0.2	1.21	0.19	1.22	0.34	1.36
410	59	0.2	1.08	0.1	0.54	0.33	1.51	0.19	1.26	0.2	1.23	0.21	1.23
420	58.35	0.23	1.26	0.1	0.59	0.21	1.27	0.2	1.26	0.21	1.7	0.19	1.26
430	57.7	0.2	1.13	0.11	0.62	0.19	1.29	0.19	1.29	0.34	1.91	0.2	1.3
440	57	0.22	1.23	0.11	0.6	0.21	1.32	0.19	1.33	0.21	1.31	0.2	1.31
450	57.6	0.18	1.17	0.1	0.64	0.2	1.36	0.19	2.03	0.2	1.37	0.19	1.36
460	57.95	0.25	1.16	0.11	0.63	0.21	1.39	0.33	1.79	0.21	1.37	0.19	1.38
470	57.6	0.22	1.15	0.1	0.63	0.19	1.52	0.2	1.4	0.2	1.43	0.33	2.46
480	57.8	0.23	1.21	0.11	0.68	0.33	2.41	0.21	1.43	0.2	1.42	0.2	1.42
490	57.2	0.16	1.32	0.1	0.66	0.2	1.46	0.19	1.46	0.2	1.5	0.21	1.45
500	56.95	0.17	1.17	0.1	0.71	0.2	1.51	0.2	1.5	0.33	2.81	0.2	1.47
510	56.55	0.18	1.26	0.12	0.76	0.21	1.51	0.21	1.94	0.33	2.36	0.2	1.51
520	56.9	0.23	1.39	0.13	0.72	0.2	1.54	0.35	3.14	0.2	1.54	0.2	1.57
530	56.45	0.2	1.53	0.12	0.72	0.2	1.55	0.34	1.75	0.2	1.56	0.2	1.74
540	56.9	0.22	1.51	0.1	0.73	0.2	2.06	0.19	1.58	0.19	1.58	0.33	2.49
550	57.65	0.27	1.35	0.11	0.82	0.33	2.25	0.21	1.6	0.21	1.61	0.21	1.6
560	57.6	0.24	1.71	0.1	0.81	0.19	1.65	0.19	1.65	0.22	2.85	0.2	1.64
570	58.05	0.16	1.34	0.14	1.18	0.19	1.68	0.2	1.69	0.2	1.66	0.19	1.66
580	58.55	0.2	1.57	0.15	1.28	0.2	1.71	0.19	2.1	0.19	1.69	0.2	1.69
590	58.65	0.26	1.66	0.13	1.16	0.19	1.71	0.33	2.4	0.19	1.72	0.19	1.71
600	58.5	0.15	1.56	0.16	1.06	0.21	2.03	0.19	1.73	0.2	1.76	0.21	3.2

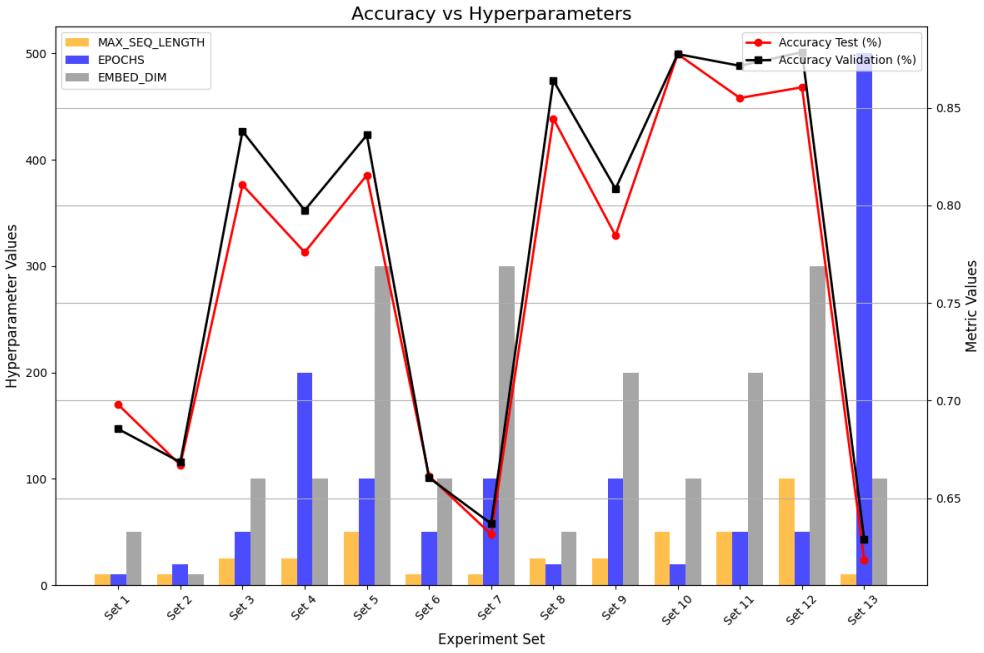


Figure 4.8: NN model Accuracy vs Hyperparameter

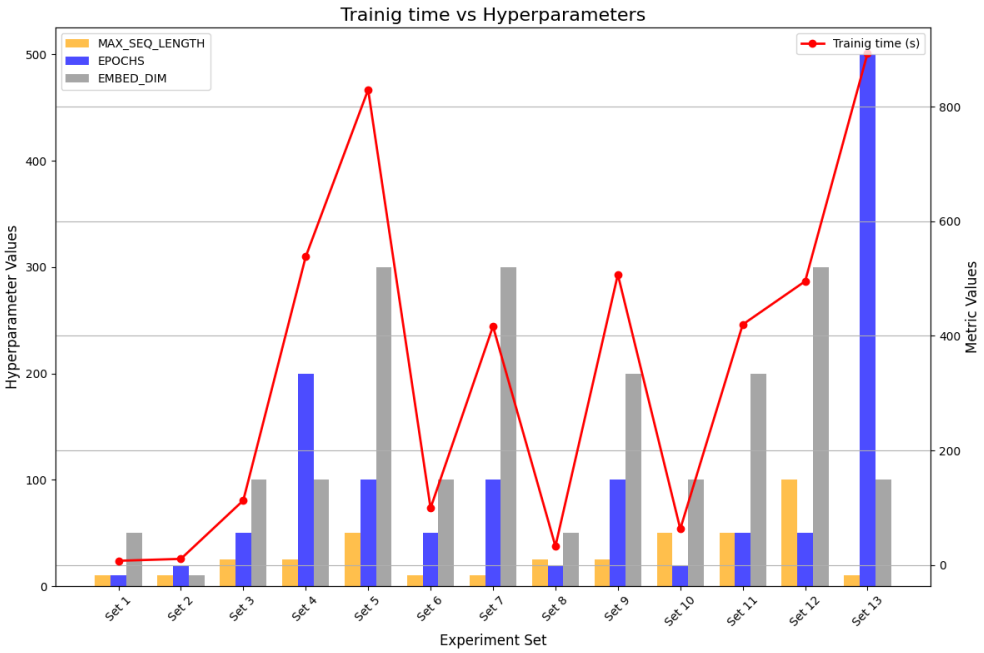


Figure 4.9: NN model Training Time vs Hyperparameter

Hyperparameters	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7
MAX_SEQ_LENGTH	50	50	50	50	50	50	100
BATCH_SIZE	32	32	32	16	32	64	32
EMBEDDING_DIM	100	300	100	200	200	50	100
HIDDEN_DIM	128	128	128	256	256	64	128
EPOCHS	5	5	15	10	10	5	5
Results							
Accuracy (Testing)	34.80%	34.20%	34.80%	88.70%	89.20%	34.50%	34.70%
Accuracy (Validation)	34.80%	31.50%	34.80%	89.85%	88.80%	34.75%	34.80%
Training time (s)	99.61	165.35	99.61	618.93	579.8	36.39	96.26
Testing time (s)	0.75	1.11	0.75	2	2.05	0.33	0.73
Validation time (s)	0.77	1.11	0.77	1.97	2.09	0.33	0.77
Total time (s)	101.6	168.32	101.6	623.41	584.65	37.66	98.25

Hyperparameters	Set 8	Set 9	Set 10	Set 11	Set 12	Set 13
MAX_SEQ_LENGTH	100	25	50	50	75	100
BATCH_SIZE	16	64	8	32	32	8
EMBEDDING_DIM	300	50	200	100	150	300
HIDDEN_DIM	512	64	256	512	256	512
EPOCHS	10	5	10	10	10	20
Results						
Accuracy (Testing)	90.30%	34.60%	91.35%	34.95%	87.55%	91.50%
Accuracy (Validation)	91.15%	35.20%	91.05%	35.40%	88.15%	91.20%
Training time (s)	1745.76	37.85	850.4	1429.44	536.71	4371.71
Testing time (s)	6.5	0.33	2.2	5.3	1.94	6.76
Validation time (s)	6.74	0.33	2.16	5.95	1.95	7.6
Total time (s)	1759.9	38.98	855.62	1441.27	541.09	4386.61

Table 4.2: Table of results for the DL model

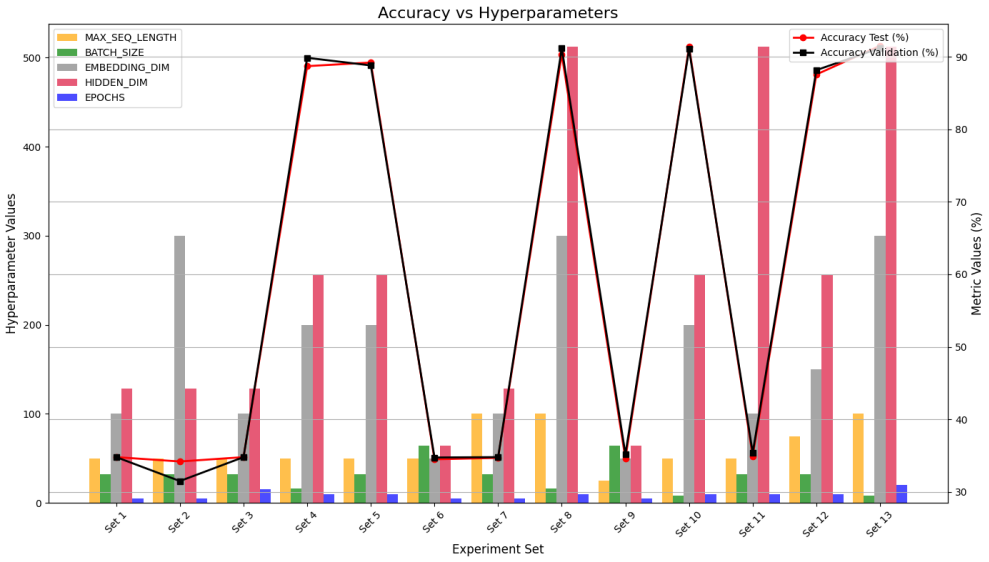


Figure 4.10: DL model Accuracy vs Hyperparameter

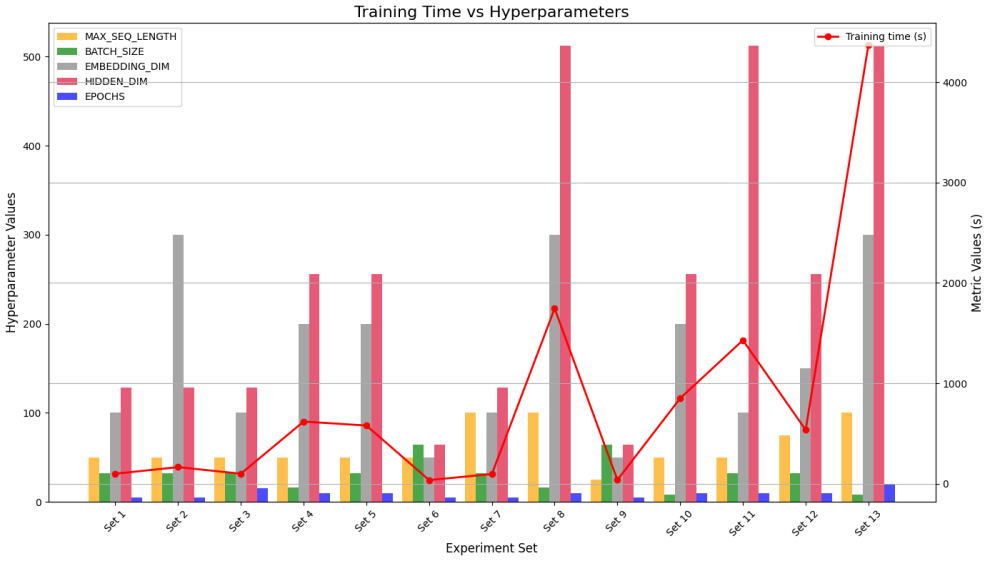


Figure 4.11: DL model Training Time vs Hyperparameter

Chapter 5

Discussion

5.1 Interpretation

What stands out first in the results of the three models is that, in general, giving a high value to the epoch hyperparameter has negative repercussions on the accuracy and training time, giving a very poor performance.

For the **rule-based model**, Part A was run with a MWC of 1000, which, as can be seen in Figure 4.1, the accuracy of each iteration increases rapidly but, when reaching its maximum point at MCW=300 and 350, the accuracy gradually decreases, which is verified in the following sets of Part B (MWC=2000), where this decrease continues.

Very similar to Figure 4.1, Figure 4.2 now compares training time against accuracy. As can be interpreted from this graph, accuracy is also decreasing even though training time is increasing. As can be seen in the four remaining graphs of this model (Figures 4.3, 4.4, 4.5 and 4.1), training time increases linearly with its respective iteration in MWC.

This can also be seen in the experiments performed with the **NN model**. The set with the most inaccurate predictions and the longest training time was set 13, whose hyperparameters were MAX SEQ LENGTH=10, EPOCHS=500 (the largest value of this variable of all the sets) and EMBED DIM=100. On the other hand, the sets (10, 11 and 12) in which minimum values were used for the epochs and the sets with medium-high values for the other parameters performed with good validation accuracies, 87.15 - 87.85% with average training times 419.79 - 495.23s (see Figures 4.8 and 4.9)

For the NN model, we can say that the best performing was set 10, with hyperparameters MAX SEQ LENGTH=50 (medium-high), EPOCHS=20 (midium-low), EMBED DIM=100 (medium) (Table 3.1), which we can consider as ”moderate values”, since none are at either end of the range of values used for each hyperparameter. This set had the best validation and training accuracy and the lowest training time (see Table 4.1).

For the **DL model**, the results were very contrasting in terms of training and validation accuracy; they were either very high (87.55 - 91.50 %) or very low (31.5 - 35.40 %), as can be seen in Figure 4.10. The sets that appear in this ”good precision” range were 4, 5, 8, 10, 12 and 13 (Figure 4.10). It has already been pointed out that a large value for the epoch variable negatively

affected the performance of the rule-based and NN models, but what is the common denominator of these DL sets that did have a good performance? The EMBEDDING DIM and HIDDEN DIM hyperparameters are in the medium-high value range (Table 4.2). However, the ”bad” results of the other sets can also demonstrate that using these parameters with high values does not guarantee good model performance either; take as a reference set 11, which uses a high value for HIDDEN DIM, but a very low one in contrast for EMBEDDING DIM. The other hyperparameters of set 11 (resulting in Accuracy (Testing)=34.95%, Training time=1429.44s) are identical to those of set 5 (Accuracy (Testing)=89.20%, Training time=579.8s), which had an excellent performance in comparison. Something similar happened with set 2; a high value for HIDDEN DIM and a low one in contrast for EMBEDDING DIM. It resulted in a ”bad” performance with Accuracy (Testing)=34.20%, Training time=165.35 s. This shows that hyperparameters are critical for the performance of each model and that a balance must be sought between all of them that fits the demands of the tasks and resources available.

While the values assigned to hyperparameters are of utmost importance for the performance of a model, there is no model that achieves 100% accuracy. Hyperparameters are one variable in the success of models, but not the only one. In machine learning and NLP, models also depend on the quality of the data with which the model will be trained and the preprocessing that is given to it, although there is a somewhat limited amount of labeled datasets. In addition, some challenges for sentiment analysis are the understanding of expressions such as irony and sarcasm, the complexity of human emotions and the ambiguity that can appear in written language and other media such as speech or facial expressions [4].

5.2 Limitations of the research

To create the sets, I sought to select the combinations of values that could give the most contrasting results. However, I consider that the sets could have been planned with ”complete combinations” (something similar to the one illustrated in Table 5.1) for all hyperparameters for each model, especially for the more complex ones (NN and DL) to have a broader picture of the performances of each model based on the values of its hyperparameters.

Hyperparameters	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9
HP_A	low	med	high	low	med	high	low	med	high
HP_B	low	low	low	med	med	med	high	high	high

Table 5.1: Example of a table with ”complete combinations”

A possible improvement for rule-based approach sets, although the results are invariably the same in accuracy, could be to perform runs with higher values for MWC with a higher iteration step (to avoid redundancy and achieve shorter training times), although the trend of the results already obtained indicates that the accuracy is decreasing as MWC increases.

Another aspect that can be seen as future work for this research would be to extend the experiments to other models for sentiment analysis such as transformers or even with mixed models.

5.3 Recommendations

As the results of the three models demonstrate that setting a large value to hyperparameters is generally not feasible, especially when it comes to epochs (in fact, larger values for this parameter generally resulted in low accuracy). To find a balance, these values must be moderate, considering which model will be used, what are the specific tasks it must solve and how the adjustment of its hyperparameters impacts its performance, from accuracy to training and testing time.

To find optimal values for hyperparameters (HPO) some strategies have emerged such as Bayesian Optimization [10] [8], Hyperband [6] or constrained Tree-structured Parzen Estimator (c-TPE) [9].

Chapter 6

Conclusion

The test results showed that assigning high values to hyperparameters (especially to the epoch value) can have the opposite effect of improving the performance of a model.

In the first model, rules-based, it can be clearly seen where the model reaches its highest accuracy, at MWC=300 and 350 (Figure 4.1), which means that these two may be the optimal value for this particular parameter and model. In the other two models, NN and DL, being more complex, they depend on more hyperparameters, which will inevitably make it more difficult to find the most optimal combination for all hyperparameters.

It is also worth mentioning that the best accuracy may not always be the best option. For example, in set 12 of NN the best result in validation accuracy was obtained, 87.85%, with a training time of 495.23s. On the other hand, set 10 achieved slightly lower validation accuracy, 87.75%, but its training time was only 63.04s. In other words, the hyperparameters in set 10 made the model 7.85 times faster than set 12 while sacrificing only 0.1% accuracy.

For this reason, it is important to be clear about the priorities that a model must meet in order to find a balance between the accuracy of the results and the training time, which is also important for finding the hyperparameter values that fit these needs.

The main limitation for this study was the number of sets that were tested for each model, however, we sought to use those that could offer the most contrasting results, so that a good comparison could be made between these results and the variables of each set of the models.

The next step for future research will be to test other NLP models (such as transformers). Another step could be to test different datasets for different approaches with the models used in this study (Rules-based, NN and DL) and the following ones.

Bibliography

[1] ARROYO, K. N. W18-final. https://github.com/KarnNA9/RStay/tree/main/W18-Final, 2024.

[2] BERGSTRA, J., AND BENGIO, Y. Random search for hyper-parameter optimization. *Journal of machine learning research 13*, 2 (2012).

[3] GOODFELLOW, I. Deep learning, 2016.

[4] HUSSEIN, D. M. E.-D. M. A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences 30*, 4 (2018), 330–338.

[5] LANGURÉ, A. D. L. Research stay - final project. https://github.com/langure/tec-stay-final, 2023. Accessed: Nov 18, 2024.

[6] LI, L., JAMIESON, K., DESALVO, G., ROSTAMIZADEH, A., AND TALWALKAR, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research 18*, 185 (2018), 1–52.

[7] SNOEK, J., LAROCHELLE, H., AND ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems 25* (2012).

[8] WANG, L., FENG, M., ZHOU, B., XIANG, B., AND MAHADEVAN, S. Efficient hyperparameter optimization for nlp applications. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), pp. 2112–2117.

[9] WATANABE, S., AND HUTTER, F. c-tpe: Tree-structured parzen estimator with inequality constraints for expensive hyperparameter optimization. *arXiv preprint arXiv:2211.14411* (2022).

[10] WU, J., CHEN, X.-Y., ZHANG, H., XIONG, L.-D., LEI, H., AND DENG, S.-H. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology 17*, 1 (2019), 26–40.

[11] YOU, Y., GITMAN, I., AND GINSBURG, B. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888* (2017).