Comparison of LSTM, GRU and Hybrid Architectures for usage of Deep Learning on Recommendation Systems

Mario Toledo
Instituto de Pesquisas Tecnológicas do Estado de São Paulo (IPT-SP)

Marcelo Rezende
Instituto de Pesquisas Tecnológicas do Estado de São Paulo (IPT-SP)

ABSTRACT

This article shows the results of a performance analysis from LSTM, GRU and Hybrid Neural Network architectures in Recommendation Systems. To this end, prototypes of the networks were built to be trained using data from the user's browsing history of a streaming website in China. The results were evaluated using the metrics of Accuracy, Precision, Recall and F1-Score, thus identifying the advantages and disadvantages of each architecture in different approaches.

CCS CONCEPTS

• **Computing methodologies**; • **Machine learning**; • **Learning paradigms**; • **Supervised learning**;

KEYWORDS

Neural Networks, RNN, LSTM, GRU, Recommendation Systems

ACM Reference Format:
Mario Toledo and Marcelo Rezende. 2020. Comparison of LSTM, GRU and Hybrid Architectures for usage of Deep Learning on Recommendation Systems. In *2020 The 4th International Conference on Advances in Artificial Intelligence (ICAAI 2020), October 09–11, 2020, London, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3441417.3441422

1 INTRODUCTION

Recommendation Systems (RS) aim to suggest items that may be relevant to the user according to their preferences [1]. However, when trying to recommend items to a user for which there is no prior information, the Recommendation Systems can generate recommendations that do not relate to the user's profile of interest or generate no recommendation at all. This type of problem is identified as Cold-Start Problem, appearing frequently in studies related to Recommendation Systems [1-5].

A possible and alternative way to deal with the Cold-Start Problem is to ask the user to rate items in the start of the system's usage [6], or forcing the user to answer questions or force this user to answer questions that allow them to be associated with stereotypes or other sets of users (such as demographics or age data, for example). However, this might create an insatisfaction with the user, who might have a minimal effort to use the system and may answer the questions incorrectly.

One of the solutions adopted to deal with the Cold-Start Problem is the use of the user's browsing history for recommendation. For example, if a page X is accessed frequently before users access the page Y page, then the page Y is more relevant to be recommended to users who accessed the page X.

Considering the sequential format of the browsing history, and the pattern recognition for recommendation, Recurrent Neural Networks has been suggested as an alternative to this problem in studies like [7-13]. Each of these studies uses a specific architecture or proposes a new kind of RNN, evaluating the results and checking the recommended items generated.

The LSTM architecture appears in most of the studies. Kong, Li and Lv [10] specifies that this model out for the possibility of making predictions with long time intervals, standing out over the other Recurrent Neural Networks in relation to the dependence of long data streams. Fu et. al. [11] proposes a new architecture for LSTM, and shows that the model can be evaluated by the metrics of Accuracy, Recall, Precision and F1-Score. But both [10] and [11] compare their models with standard LSTM architectures.

According to Korataev and Lyadova [8], the GRU architecture are shown to be simpler and less redundant than Long Term Memory Networks, with their training time reduced between 20% to 30%. Although there are demonstrations that there is no difference in quality depending on how the LSTM are initialized, the GRU still guarantee superior performance. However, this study does not make any comparison by the usage of metrics with other models.

Ruocco, Skrede and Langseth [12] proposes a model that uses two GRU with the proposal of processing the current session of a user in a Recurring Neural Network and then processing the session history of the same user in a second Recurring Neural Network. Thus, the second Recurrent Neural Network acts as a hidden layer of the first. The model was evaluated using the metrics of Recall and Reciprocal Classification Average, in addition to comparison with other recommendation models, including traditional Recurrent Neural Networks.

Anil et. al. [13] evaluates the best architecture to transform analyzes into evaluations for a product recommendation system through tests on three models of Recurring Neural Networks: LSTM, GRU and a Hybrid Network that unifies both to identify the benefits of such architecture. Such an experiment showed that the three Recurrent Neural Networks are comparable in terms of Accuracy, but the hybrid solution, which unifies a LSTM and a GRU, is more accurate with less loss, while the time to train it is lesser than the other tested architectures. But the study of Anil et. al. [13] does not specifies that kind of metrics were used to compare all kinds of networks.

ICAAI 2020, October 09–11, 2020, London, United Kingdom

ACM ISBN 978-1-4503-8784-2/20/10…$15.00
https://doi.org/10.1145/3441417.3441422

However, all these studies do not identify which Recurrent Neural Network architecture perform better within in the context of Cold-Start Problem or what are the advantages and disadvantages among the other architectures. Taking this into consideration, this article shows the results of a performance analysis between LSTM, GRU and Hybrid Networks architectures, frequently identified in other researches related to the context of using the user session data to generate recommendations.

In order to identify the architecture and variables that has the best performance for this kind of situation, prototypes were built based on each of the architectures, using the same database as an input, obtained through a public database of a Chinese website of live streaming [11], and evaluating the results from the metrics of Accuracy, Recall, Precision and F1-Score used in [11].

2 METHODOLOGY

The experiment had different stages. Firstly, the data went through an analysis process, identifying the peaks of access and the cases that may interfere with the final results of this work. From the analysis, the data was adapted for the execution of the prototypes, divided into a base for training and another for testing. Then, the construction of the prototypes was carried out based on the data collected through the State of the Art, defining the input parameters for each one and the specific parameters of each architecture. Finally, the three prototypes were executed independently using the adapted data and generating their results in specific directories.

The data used in this experiment represents the browsing history of a Chinese live streaming website, as used in the work of [11] and made publicly available through the Kaggle website. Each line of the data file represents a user's navigation sequence, containing numeric identifiers for each page visited in chronological order. To analyze the data, a script was created, using the Python language in version 2.7.15, in order to quantify the page accesses per user and the frequency of access. This script was executed on an Ubuntu version 18.04.2 LTS system with 16 GB of RAM and an Intel Core i7-7700 processor.

The purpose of this script is to count and group the values of metrics that allow data analysis before adapting them. Iteratively, each line of the file was processed, accounting for the number of sessions per user, total pages visited by all users and the number of pages visited per session. In addition, the frequency in which a page is accessed and the number of times a user accesses the same page was also considered. Then, the average number of hits per user was calculated, alongside the highest frequency of hits on a single page, the lowest frequency of hits on a single page, the largest sequence of pages that a user accessed and the lowest sequence of pages that a user accessed.

In total, the data file accounts for 1,806,204 lines representing unique users, 18,927,700 page accesses and 453,423 distinct pages accessed. The number of accesses varies between 1 and 1,060 pages in the same session, with an average of 10 different accesses for each user, in which the least visited page had at least 1 user access and the most visited had 341,471 users.

To calculate the frequency of accesses for each page, the number of times each page was accessed was counted, and then the number of pages per number of accesses was grouped. In order to calculate the frequency of users by the number of accesses per page, the number of pages accessed by each user was counted, and then, the number of users was grouped by the number of accesses per page.

Considering that sessions with access to a single page have no value for this research, since such sessions have no learning value in relation to the user's trajectory to the destination page, they were discarded from the entry base. Therefore, only sessions that have at least two pages accessed by the same user were considered.

In addition to removing the strings with only a single access from the same user, there was no need for other changes to the input data for use in the prototypes. Initially, the data were divided into two groups containing 80% of the data for the training base and 20% for the test base.

To build the prototypes, the Keras library [12] was used alongside the Python language on version 2.7.15, running on an Ubuntu 18.04.2 LTS with 16 GB of RAM and an Intel Core i7-7700 processor.

Considering the Recurring Neural Network architectures to be used in this experiment, a script was created to train a LSTM Network, a GRU Network and a Hybrid Network that would unify both models.

The data was downloaded into a text file and then was read in the model. Each line of the input file was separated into two datasets, where the set X represented the sequence of pages traversed by each user and the set y represented the last page accessed by each user. Then, the data was transformed using the One-Hot Encoding algorithm, allowing the use of categorical data as numeric data. In this process, the number of strings and the number of pages accessed were stored for identification, such as the size of the number of sentences and the size of the system's vocabulary for comparison in the results.

In the prototype using the LSTM architecture, a sequential model with three layers was created: a Long Term Memory layer with 128 units, a Dense layer with activation from the linear function a(x) = x and an activation layer varying the activation function used, as described in Table 1

In the prototype using the GRU architecture, a sequential model with three layers was also created: a layer of Recurring Blocking Units with 128 units, a Dense layer with activation from the linear function a(x) = x and an activation layer varying the activation function used, as described in Table 2

In the prototype using the hybrid architecture, a sequential model with four layers was created: a layer of LSTM with 128 units, a layer of GRU with 128 units, a Dense layer with activation from the linear function a(x) = x and an activation layer varying the activation function used, as described in Table 3

Table 4 shows the variables and the set of values used during the experiment of this work. These values were taken into consideration from the values used by studies in [7-13], also considered base values for each of the used variables.

In total, 108 experiments were generated from the combination of variables. The prototypes were executed simultaneously, training the 3 network architectures at the same time. Performance during training of the networks varied according to the architecture used and the input parameters. The LSTM and GRU networks had equivalent training times, while the hybrid network took longer to finish its execution. Likewise, an increase in the execution time of each period of the networks was noticed with the change in the size of

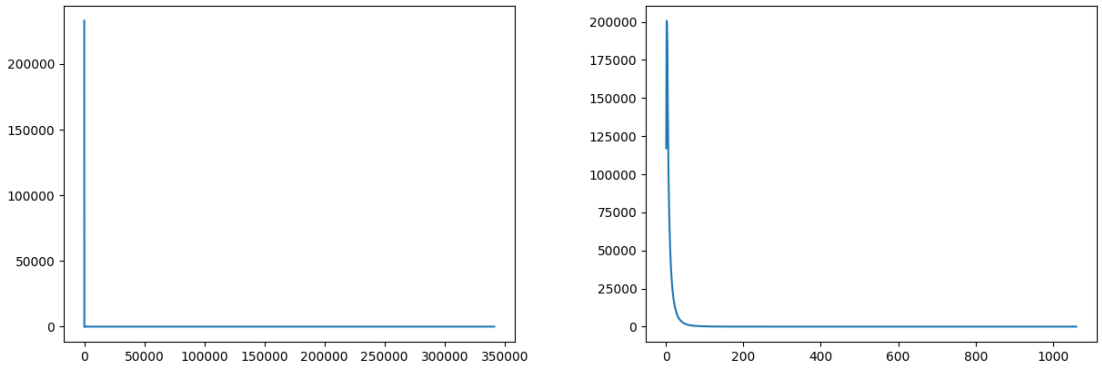

Figure 1: Left image: the frequency of visits to each page by the number of times the page was accessed. Right image: frequency of visits to each page by the number of users.

Table 1: LSTM prototype architecture

Layer	Input	Output
Input Layer	(None, 40, 4862)	(None, 40, 4862)
LSTM	(None, 40, 128)	(None, 128)
Dense	(None, 128)	(None, 4862)
Activation	(None, 4862)	(None, 4862)

Table 2: GRU prototype architecture

Layer	Input	Output
Input Layer	(None, 40, 4862)	(None, 40, 4862)
GRU	(None, 40, 128)	(None, 128)
Dense	(None, 128)	(None, 4862)
Activation	(None, 4862)	(None, 4862)

Table 3: Hybrid prototype architecture

Layer	Input	Output
Input Layer	(None, 40, 4862)	(None, 40, 4862)
GRU	(None, 40, 4862)	(None, 128)
LSTM	(None, 40, 128)	(None, 128)
Dense	(None, 128)	(None, 4862)
Activation	(None, 4862)	(None, 4862)

Table 4: List of variables and values used in this experiement

Variable	Values
Activation Function	softmax, tanh and sigmoid
Optimizer	SGD, Adam, RMSProp and Adagrad
Batch Size	64, 128, 256
Epochs	20, 50, 100

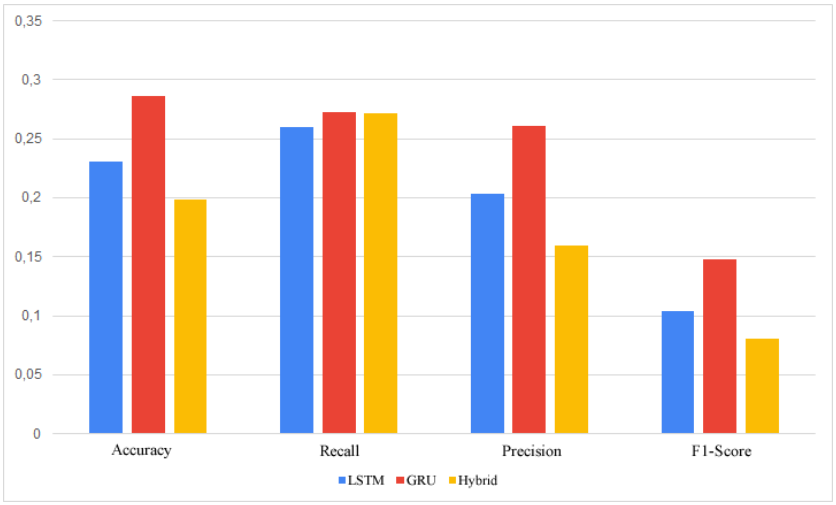

Figure 2: Graphic representation of the general average of the metrics

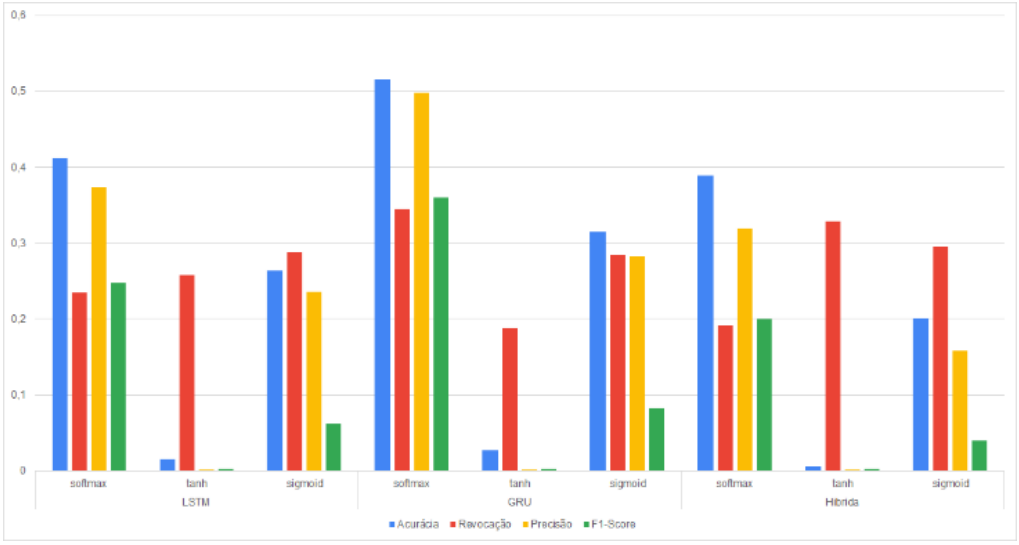

Figure 3: Graphic representation of the metrics by activation function and architecture.

the batch, while the parameters such as activation function and optimizer had low variation in terms of performance during the training.

3 RESULTS

From the results generated, the general average of each of the evaluation metrics was calculated, as shown in Figure 1. The data demonstrated a greater generalization of the GRU architecture as for the other variables in the model, having a good performance in a greater number of scenarios than the other architectures, while the Hybrid Network architecture had a good performance in a smaller number of scenarios.

Grouping the results by the activation function, it is possible to analyze the performance of the architectures from the functions used in training the model of this experiment. Figure 2 shows that the softmax function presents better performance compared to the other functions used, while the hyperbolic tangent had the lowest performance among the other functions used.

When analyzing the results from the number of epochs used, it is clear that the greater the number of epochs, the greater the quality of the results. This happens regardless of the other parameters used, but the value has little influence on increasing the values of the metrics, slightly improving the results, as described in Figure 4

The optimizers were the values that most influenced the evaluated metrics. The results with higher values are related to the use of the RMSProp optimizer, especially when used in conjunction with

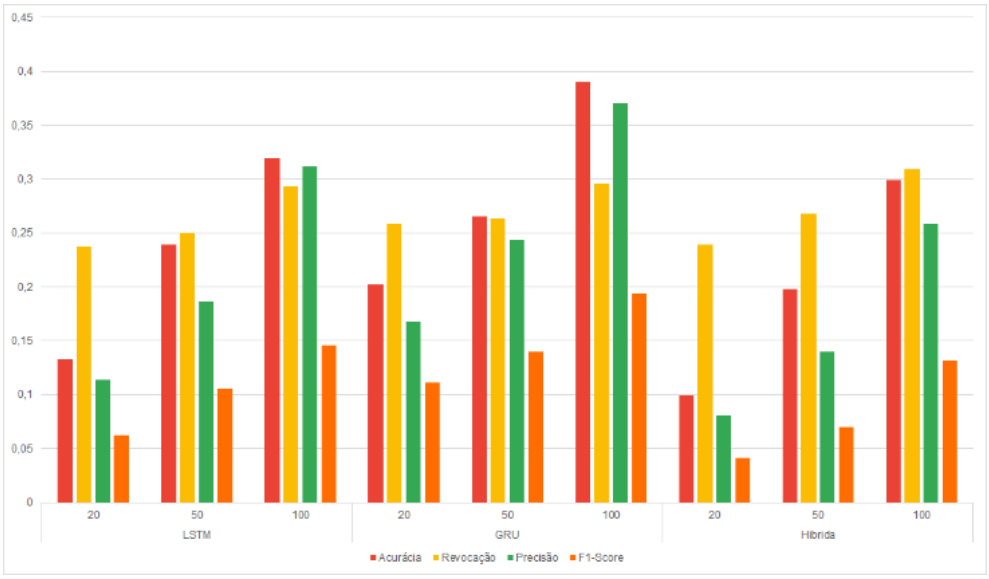

Figure 4: Graphics representation of the metrics by number of epochs and achitecture

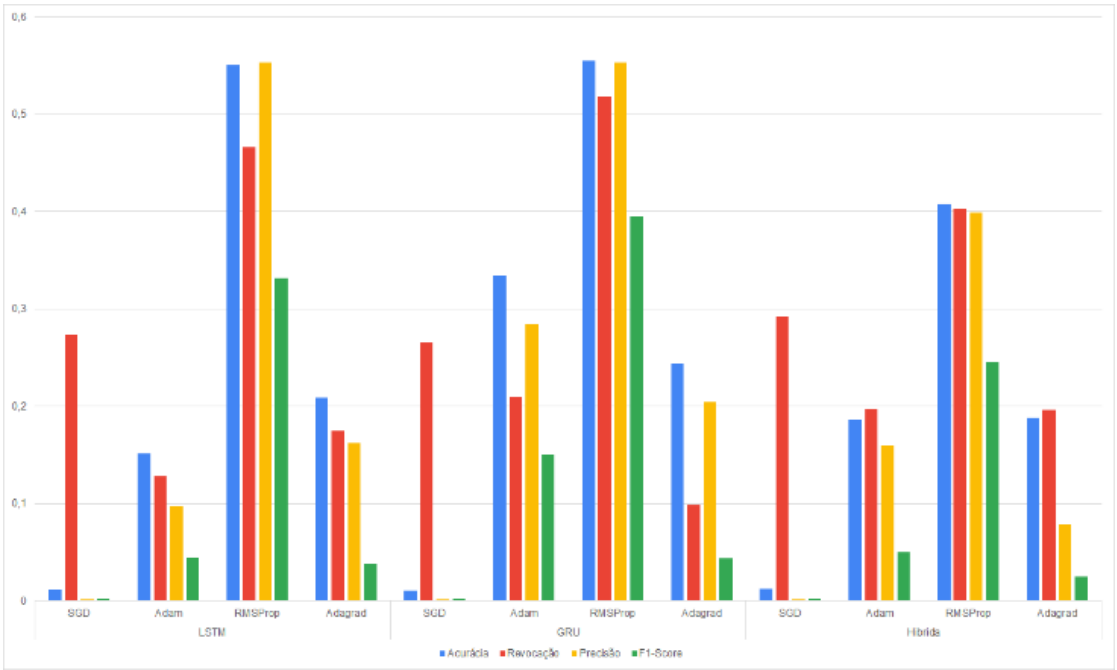

Figure 5: Graphic representation of the metrics by optimizer and architecture

the GRU architecture. However, the results related to the LSTM architecture combined with the use of the RMSProp optimizer showed higher results when a smaller batch size was used.

The Adam and Adagrad optimizers proved to be strongly linked to the use of the number of epochs used, where the greater the number of epochs, the greater the values of the metrics evaluated. However, when compared to each other, Adam presents higher values of the metrics evaluated in relation to Adagrad. The experiments that used the SGD optimizer had the lowest results regardless of the architecture used, as described in Figure 4

The variation in the batch size showed different behaviors in the evaluated results, according to the values used as input parameters. When using the LSTM architecture with the RMSProp optimizer, the analyzed metrics decreased as the batch size increased, while the use of the GRU architecture had little impact on the results

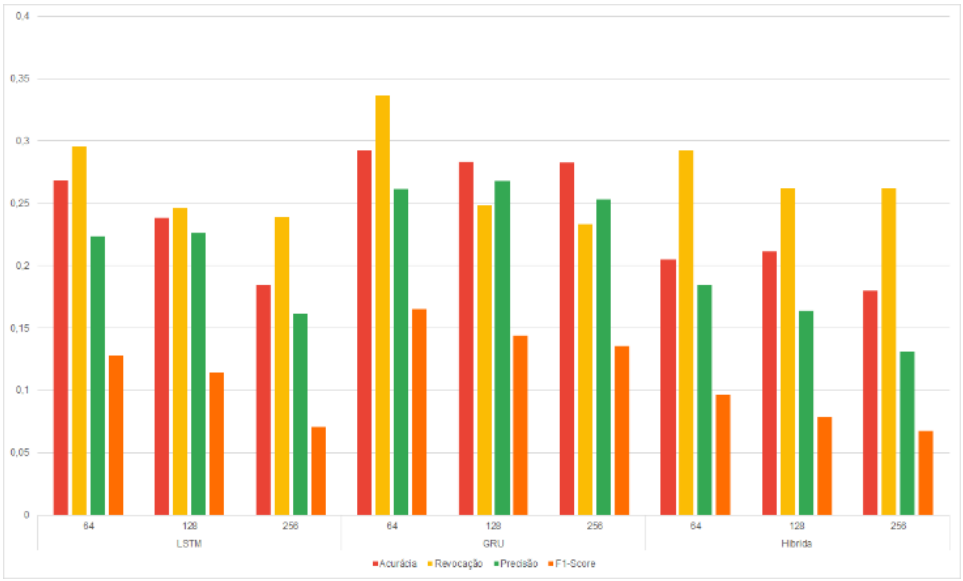

Figure 6: Graphic representation of the metrics by batch size and architecture

using the same parameters. However, the use of other optimizers from the same parameters, varying the batch size, had the same behavior regardless of the architecture used, as described in Figure 5

When analyzing the results from the network architecture used, we can see a superior performance of the GRU architecture, obtaining higher values of Accuracy and F1-Score with greater frequency compared to other architectures. The LSTM architecture is superior in specific cases, while the Hybrid Network does not demonstrate superior performance in any case if compared using the mentioned evaluation metrics.

However, the performance of the architectures varies according to the combination of the parameters used. The GRU architecture has higher values of Accuracy and F1-Score when used in conjunction with the RMSProp optimizer and softmax activation function, also having higher values according to the batch size. In other words: the higher batch size when using GRU Networks together with the RMSProp optimizer, the better its performance will be.

Likewise, the LSTM architecture had a superior performance when compared to the other networks when used in conjunction with the sigmoid activation function and the RMSProp optimizer, regardless of the batch size of the in these cases.

4 CONCLUSION

From the results obtained, in general, the results show a better performance of the architecture of Recurring Blocking Units in most scenarios compared to the other two architectures. While the Recall measurement appears with similar values in most scenarios for the three architectures, the Accuracy and Precision measurements are considerably higher when using GRU, resulting in a higher F1-Score. The Hybrid model, on the other hand, presented lower results than other architectures between all the evaluation metrics.

It was also possible to identify the performance of the architectures from the variables used in the model. The softmax function proved to be more effective than the other activation functions used in the model, obtaining high Accuracy and Precision values when used, while the hyperbolic tangent function had significantly lower results than the other activation functions.

Likewise, the use of the RMSProp optimizer ensured superior results to the other optimizers, while the use of SGD yielded inferior results to the others obtained. The variation in the number of epochs used in the training of the models, on the other hand, improved as the variable grew, while the change in the batch size had little influence on the results.

When compared to the results of the works reviewed in the State of the Art of this research, the results obtained through the execution of the built prototypes were superior to the others when specific combinations of parameters were used. It was possible to observe that the works reviewed in the State of the Art did not mention the use of optimizers in their models, while the variation of optimizers during the training of the networks of this research showed a great difference in the increase of the metrics evaluated.

Within the context of this experiment, the use of GRU, combined with the use of RMSProp as the optimizer and the use of softmax as the activation function, is suggested for use of the browsing history as an alternative to Recommendation Systems in a Cold-Start Problem situation, as a result of its performance observed in the results compared to the architectures identified in the works as those of [7-10].

For future works, it is recommended to modify the variables used, being able to add new values or add new variables to the model in order to improve it. In addition to the stipulated variables, other Recurrent Neural Network architectures can be tested under the same conditions as the experiment in this work in order to verify their performance regarding the architectures used. In particular,

the use of other optimizers from the standard values suggested by the authors since they were shown to have a greater influence on the quality of the results.

In addition, other databases containing the user's browsing history can be tested on the model, varying the size, the number of strings or the last state of the user's session. Different databases can be used to identify changing results according to the context of the input database.

REFERENCES

[1] Deng, S., Huang, L., and Guandong, X. 2014. Social network-based service recommendation with trust enhancement. Expert Systems with Applications 41, 8075—8084. DOI= http://dx.doi.org/10.1016/j.eswa.2014.07.012.

[2] Nadimi-Shahraki, M., Bahadorpour, M. 2014. Cold-start Problem in Collaborative Recommender Systems: Efficient Methods Based on Ask-to-rate Technique. Journal of Computing and Information Technology, 22, 105-113. DOI= https://doi.org/10.2498/cit.1002223.

[3] Gomes-Uribe, C., Hunt, N. 2015. The Netflix recommender system: Algorithms, business value. ACM Trans. Manage. Inf. Syst., 6. DOI= http://dx.doi.org/10.1145/2843948.

[4] Shah, K., Sakunke, A., Dongare, S., and Antala, K. 2017. Recommender Systems: An overview of different approaches to recommendations. International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). DOI= https://doi.org/10.1109/ICIIECS.2017.8276172.

[5] Pandey, A. and Rajpoot, D. 2016. Resolving Cold Start problem in recommendation system using demographic approach. 2016 International Conference on Signal Processing and Communication (ICSC). DOI= https://doi.org/10.1109/ICSPCom.2016.7980578.

[6] Masthoff, J. 2011. Group Recommender Systems: Combining Individual Models. Recommender Systems Handbook, 21, 677-702. DOI=https://doi.org/10.1007/978-0-387-85820-3_21.

[7] Tan, Y., Xu, X., and Liu, Y. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. 2016 Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. DOI= https://dl.acm.org/doi/10.1145/2988450.2988452.

[8] Korotaev, A. and Lyadova, L. 2018. Method for the Development of Recommendation Systems, Customizable to Domains, with Deep GRU Network. In Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, 2, 231-236. DOI=10.5220/0006933302310236

[9] Zhou, T., Qian, H., Shen, Z., Zhang, C. Wang, C., Liu, S., and Ou W. 2018. JUMP: a Jointly Predictor for User Click and Dwell Time. Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. DOI= https://doi.org/10.24963/ijcai.2018/515.

[10] Kong, F., Li, J., Lv, Z. 2018. Construction of intelligent traffic information recommendation system based on long short-term memory. Journal of Computational Science, 26, 78-86. DOI= https://doi.org/10.1016/j.jocs.2018.03.010.

[11] Fu H., Li J., Chen J., Tang Y., Zhu J. 2018. Sequence-Based Recommendation with Bidirectional LSTM Network. Advances in Multimedia Information Processing – PCM 2018. 11166. DOI= https://doi.org/10.1007/978-3-030-00764-5_39

[12] Ruocco, M., Skrede, O. and Langseth, H.. 2017. Inter-Session Modeling for Session-Based Recommendation. arXiv e-prints.

[13] Anil, D., Vembar, A., Hiriyannaiah, S., S. G.M. and Srinivasa, K. 2018. Performance Analysis of Deep Learning Architectures for Recommendation Systems. IEEE 25th International Conference on High Performance Computing Workshops (HiPCW), 129-136. DOI=https://doi.org/10.1109/HiPCW.2018.8634192.

[14] Keras. 2015. https://keras.io.