Contrasting Neural Network Techniques for Enhanced Natural Language Understanding in Conversational Agents – CoN^{2}LUC

Arnulfo Alanis[1(✉)], J. Ascención Guerrero-Viramontes[2], Gulliermo Daniel Prieto[1], and Bogart Yail Marquez[1]

[1] Systems and Computer Department, National Technology of México, Campus Tijuana, Calzada del Tecnológico S/N, Fraccionamiento Tomas Aquino, Baja California, C.P. 22414, Tijuana, B.C, México
{alanis,gulliermo.prieto17,Bogart}@tectijuana.edu.mx

[2] Division of Postgraduate Studies and Research, Tecnológico Nacional de México, Campus Aguascalientes, Aguascalientes, Mexico
jaguerrero@aguascalientes.tecnm.mx

Abstract. Language is one of the processes that human beings use to be able to communicate effectively, this helps to share not only information, but elements of everyday life, to process all kinds of situations, ranging from instructions for a certain event to show and identify and express emotional states. With the passage of time when individuals have an inter-relationship with other people, through various elements, such as work and educational environments, the factors of collaboration and exchange of ideas become more important, since as more communication takes place, emotional processes can be identified, as these end up becoming factors. With the above, it can be assumed and indicated that emotional intelligence takes a greater relevance, since it contributes in a very important way to the way in which individuals process and manage emotions in various social contexts. In this article, a development using machine learning is presented, the main objective is to be able to have a simple and easy understanding, based on natural language. Contemplating a DataSet that contains conversational elements, for this study two strategies will be applied to classify and predict the intentions and emotions of users, the study seeks to determine which of these two approaches, have better performance to support the understanding of natural language.

This type of research work, contemplating neural networks, is intended that the creation of tools such as conversational agents, can have a better response, be more accurate and supported by various techniques to be empathetic to user conversations.

Keywords: Convolutional Neural Networks (CNN) · Recurrent Neural Networks (RNN) · LSTM (Long Short-Term Memory) LSTM

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15891, pp. 370–380, 2026.
https://doi.org/10.1007/978-3-031-97617-9_24

1 Introduction

Currently in many applications that have AI, the PLN has a very important relevance, as this ranges from personal assistants, attention assistants to text analysis [1]. The PLN has the main function of classifying text entered by users, ranging from short instructions to complex commands (i.e., e.g., user intentions and sentiments).

In order to determine and find complex elements and to be able to learn the representation of complex data without the need for very complex requirement analysis processes, neural networks support these processes as well as others[2].

Initially, convolutional neural networks (CNN) were proposed to solve problems and situations of pattern identification in the area of computer vision, because they take advantage of two-dimensional convolution operations in images.

However, for text classification, one-dimensional (1D) convolution can be applied to token embedding sequences. This 1D approach mitigates the noise that could arise from forcing text into a 2D structure, while preserving the advantages of CNN in local feature extraction [3].

There are also the recurrent neural networks (RNN), and in particular those that have an improvement with the LSTM, these are of great relevance in those processes of the capture that contemplates long term dependencies [4]. Something to consider is that when performing sequential processing of tokens, this type of neural networks can maintain those hidden states that have longer text information.

A development in Python is presented, with the objective of making a comparison of the deep learning models: CNN 1D and RNN, which perform text classification, which will be in an application known as "chabot". Each of the models has its strengths, for example: CNN 1D performs the data capture efficiently, and with them identifies the patterns, and on the other hand the RNN contains the information and uses it in the context of the work, to be used in the long term. This article will show what considerations should be taken into account in the design of the models, reviewing the data preprocessing and, as in all model development, reviewing the evaluation metrics.

After that, the development of the data preprocessing, its architectures and training will be presented, once the processes are indicated, the performance comparison and an analysis of the results will be made. As a result, the results will be indicated, highlighting the best numbers as well as the advantages and disadvantages.

2 Network Structure

The analysis for this development is based on having text inputs, contemplating two different neural architectures, which were developed in Python, TensorFlow/Keras. In each of the models' different strategies are applied in order to identify the patterns of the word sequences, and ultimately having a prediction.

2.1 Evolutionary Neural Network (CNN)

The CNN had its highest point of disclosure, in environments of pattern and image recognition [2], these networks, have the peculiarity that they can adapt to natural language,

being able to apply one-dimensional filters, in an environment of temporal dimension for the use of tokens [3].

Some characteristics, per layer:

- Convolutional: In these layers are applied different filters, also called "Kernels", which are used to detect features.
- Activation: Their function is to add nonlinearity.
- Pooling: These layers are designed to reduce the dimensionality and to have a low computational cost.
- Fully connected: These layers are designed for the process of performing the qualification, which is based on the extracted features [6].

This model has an uncomplicated but not simple implementation for those who are starting, the model is composed: First, by an embedding layer, whose task is to assign each token index to a vector. Second, by a 1D convolutional layer (Conv1D) which has the task of scanning the embedding vectors, and captures the features as well as n-gram patterns. Third, it performs an operation called "GlobalMaxPooling1D", which is responsible for selecting the activations, thereby merging the maps into a single vector. Fourth, in the dense layers, it performs the transformation of the vector into an output distribution over the class labels:

$$Y(i,j) = \sum_{m} . \sum_{n} .K(m,n) \cdot X(i+m, j+n) + b \tag{1}$$

Its efficiency and very low computational cost performance position it as one of the most useful models in tasks such as voice recognition, sentiment identification, biometric data, and power electronics [7].

2.2 Recurrent Neural Network (RNN) with LSTM

RNNs are a type of network that are applicable to sequential data, this because they can have a hidden state that is maintained through temporal processes [4]. The layer that is in charge of embedding is to convert the indexes of each token into embedding vectors, that is why an LSTM layer can process the vectors in order, considering the relationships across multiple.

LSTM, being a variant of RNNs, use gate structures that help them to control and regulate the information and, moreover, a very important feature is the ability to retain the information "dependencies" in the long term.

This type of network, with the passage of time has helped the problem of the weakness of the gradient, which allows the RNN to have long-term dependencies in those data sequences that can be considered complicated [8].

2.3 Model Compilation and Trainingdebilidad

Both the CNN and RNN, the Adam optimizer is used, which is a good tool as it supports adaptivity and training efficiency. A cross-entropy is used to analyze the loss, which is considered to be highly efficient when the labels are encoded with integers [2]. A

callback called “EarlyStopping” is implemented which function is to monitor the loss that occurs in the training when a high point is reached, this will help to mitigate the overfitting.

3 Comparison of Architectures

3.1 CNN

It focuses on the extraction of features in the text, for this, convolutional filters are used. By applying the 1D filter, the training minimizes the noise of the additional dimensions, this is very convenient as it makes it an accurate method, and strong enough to identify textual patterns [3].

3.2 RNN (LSTM)

Its main feature is the sequentiality of the language, turning token contexts into easily identifiable values, and thus with them, being able to better interpret subsequent words [4]. Another feature is to be able to identify long term dependencies, this gives developers a point to consider, as support in queries or statements, regardless of whether they depend on contextual clues.

Since the 2 models will be trained and the dataset will be the same, it will be possible to better visualize the performance of both, and thus evaluate which method has a more robust prediction for a given application. From the literature it could be indicated that the CNN can be efficient, the RNN having the characteristic of visualizing the context in a wider way can also be a good prediction. In the end, to determine which one would be the correct one, several factors should be considered such as: the size of the data set, the length of the words and phrases, the analysis of speed and accuracy.

3.3 Summary

A summary of both models is made, as show in Table 1.

Table 1. Comparation RNN with LSTM and CNN [4, 6, 9, 10]

Characteristics	RNN with LSTM (Recurrent Neural Network with Long-Term Memory)	CNN (Convolutional Neural Network)
Architecture	It has cells that regulate the information in its sequence	It has convolutional and pooling layers that are used to extract characteristics
Data	Are sequential or temporal, for example: audio, text, etc	Structured, e.g. images
Previous data memory	Can store relevant information	It does not store information, so it has no memory of previous data
Training	Its processing is sequential	It is parallelizable
Long-term dependency	Can model long-term dependencies	Not suggested for long term dependencies
Use	PLN, for time series	Clasificacion de imágenes, detencccion de objetos, etc
Advantages	Can capture context and temporal dependency	Learns spatial patterns

4 Algorithms

This paper presents two deep learning approaches, (CNN) and (LSTM) to identify and classify user data in a chatbot environment [2–4]. Each of the models are prohramed and trained using tools such as: TensorFlow/Keras, this is to take advantage of working with embedded word vectors and specialized layers.

Onsidering NumPy and Pandas, supports the work of numerical operations and therefore the management of data, while NumPy allows to have efficient calculations, pandas help to have the data and help in the preprocessing of the data, such as CSV files.

In the training and evaluation process, the use of TensorFlow/Keras provides a high level of functionality to support the conversion of token indices into vectors, "EarlyStopping" in order to have the ability to automatically stop training when the loss is too high or no longer improves. On the other hand, the Conv1D and GlobalMaxPooling1D, analyze and take the patterns for the CNN model, the LSTM layer is in charge of processing the sequential data for the RNN. It is contemplated to have tools such as "scikit-learn" to help split the data into training and test sets.

In the FrondEnd environment, "matplotlib" is worked with to plot the training metrics and confusion matrices, this supports the developed to have a clearer view of the performance of each model. For the graphical user interface (GUI) we have "tkinter", which is integrated with the "speech_recognition" library, in order to have voice input and produce real-time transcriptions.

A helpful point is that by separating tasks such as data processing, model building, training, evaluation and user interaction, the system is intended to remain flexible and adaptable to other situations and scenarios.

5 Definition of Model Architectures

This paper generates and analyzes two different deep learning designs for text-based classification: (CNN) and (RNN), both using LSTM. Both CNN and RNN are TensorFlow/Keras implementations, for each an embedding layer is contemplated to transform integer encoded inputs [2–4].

5.1 CNN Model

CNN has as first an embedding layer, second a Conv1D layer that slides a kernel over the sequence of embedding vectors, it is worth mentioning that it focuses on text patterns, third, it has the operation "GlobalMaxPooling1D", which has the function of selecting the maximum value of each feature map, to condense the extracted patterns in a single vector.

Algorithm 1

```
def create_cnn_model(vocab_size, embedding_dim, input_length,
    num_classes):
      model = Sequential([
       Embedding(input_dim=vocab_size, output_dim=embedding_dim, in
       put_length=input_length),
       Conv1D(filters=128, kernel_size=5, activation='relu'),
       GlobalMaxPooling1D(),
       Dense(64, activation='relu'),
       Dense(num_classes, activation='softmax')
  ])
   model.compile(optimizer='adam', loss='sparse_categorical_crossentro
   py', metrics=['accuracy'])
return model
```

One thing to note is that compared to typical 2D convolutions applicable to image processing, the 1D architecture used here is adapted to sequential data. It is good to point out that by restricting the convolutions to a single dimension, the evolution of the model thus avoids noise, which can be generated and focuses on the detection of critical n-gram type features [3].

5.2 RNN Model (LSTM)

On the other hand, the LSTM based model which is responsible for processing the input tokens. And after the embedding layer, the data is fed into a 64-unit LSTM layer, this helps the network to propagate hidden states over time and capture long-range dependencies. A final dense layer generates class probabilities:

LSTMs have a feature that gives them their value, which is being able to retain the relevant context in long sequences [4]. This positions them as a good choice and more so when the order and relationship of tokens are important.

Algorithm 2

```
def create_rnn_model(vocab_size, embedding_dim, input_length,
    num_classes):
      model = Sequential([
      Embedding(input_dim=vocab_size, output_dim=embedding_dim, in
      put_length=input_length),
      LSTM(64),
      Dense(64, activation='relu'),
      Dense(num_classes, activation='softmax')
  ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossen
     tropy', metrics=['accuracy'])
return model
```

5.3 Comparison of Models

While, one may think that both models fulfill the same goal i.e. classification, it would not have much relevance to perform this strategy time, but their differences may generate different performance results. That is, while CNNs quickly extract salient and localized features [3]. For LSTM models, additional computations are required per token [4]. Trying to do this exercise time and stealing these two strategies, tending the same dataset, gives a guideline to be able to identify which method provides higher accuracy or generalization for a particular application, be it speed-sensitive chatbot interactions or more context-aware tasks that require deeper sequential understanding.

6 DataSet

The DataSet that was used, at the beginning was very small, and contained some predefined intents coming from Kaggle [5]. It contemplated some intents such as: greeting, farewell, creator and hours. Upon analysis and observing that it could not satisfy the work, as it did not generate the interlocution intents, nor did it generate enough variability in user expressions, the new approach focused on orienting it towards a wider and more adaptable JSON based scenario, which allowed the incorporation of additional tags (to mention a few: weather, time, joke, news) along with a significantly larger list of possible user inputs.

The new DataSet is composed of several patterns for each intent, with various phrases that users might employ in real conversations. Each of the elements will be processed with the features of the text preprocessing routines, i.e. lower case, punctuation removal, which will be enhanced prior to model training. By doing this, consistency can be guaranteed and noise can be reduced, plus one can integrate user-generated text, which can come from a variety of sources. Also, the new DataSet has a series of response lists for each intent, allowing for more natural and varied responses.

The DataSet, by having a wider range of intentions, and examples, will be able to improve the chatbot's ability to have the dialog, by being more robust, will be able to support evaluating and contrasting CNN and LSTM architectures.

7 Model Comparison and Reporting

The proposal shown in this paper demonstrates a simple methodology for comparing CNN and RNN (LSTM) architectures, with a wide variety of training conditions. This active function prompts the user to train each model, considering the different epochs: 20, 30, 40 and 50, and the data are divided into separate training and test sets. For each configuration, the following steps are performed, accuracy values: Each training a graph is generated showing the accuracy, accuracy values of the tests: Once the training was performed, confusion matrices: Each of the best performing CNN and RNN models generates its confusion matrix.

The results, are compared based on the above items, and this gives a clearer picture of the performance of the CNN and RNN architectures under different training conditions as shown in Table 2.

8 Result

8.1 CNN Model Results

The ROC curves show that almost all predictions are correct. The CNN model is relevant for its ability to extract correctly the localized features of the text, this supports the conditions could be controlled DataSet, and thus with them generate a classification that could be considered of very high value. However, it should be noted that, during the tests that were performed with real users, it should also be noted that in some occasions it classified erroneously, such as: some ambiguous greetings or some text-dependent queries, as shown in Figs. 1 and 2.

Table 2. Classification Report for Best CNN Model:

Class	Precision	Recall	F1-Score	Support
Goodbye	1	1	1	147
Greeting	1	1	1	163
Help	1	1	1	147
Joke	1	1	1	148
Name	1	1	1	132
News	1	1	1	138
programmer	1	1	1	155
Thanks	1	1	1	150
Time	1	1	1	156
Weather	1	1	1	164
Overall Accuracy	1			1500
Macro Avg	1	1	1	1500
Weighted Avg	1	1	1	1500

8.2 Results of the RNN Model (LSTM)

In this proposal the RNN model, which is based on a LSTM architecture, generates and processes data sequentially, and keeps its states hidden throughout the token work. The design of this architecture is planned to check and have the dependencies and context in the user inputs. At the point when the model was analyzed and trained under the same conditions, i.e., 20 epochs at a learning rate of 0.001, it achieved 100% accuracy on the controlled test set, as shown in Table 3.

When the LSTM processes the complete sequence of words, it allows you to control the sentence structures that are considered more complex. But, however, like CNN, its environment is also controlled and is good, which shows that the DataSet is adequate.

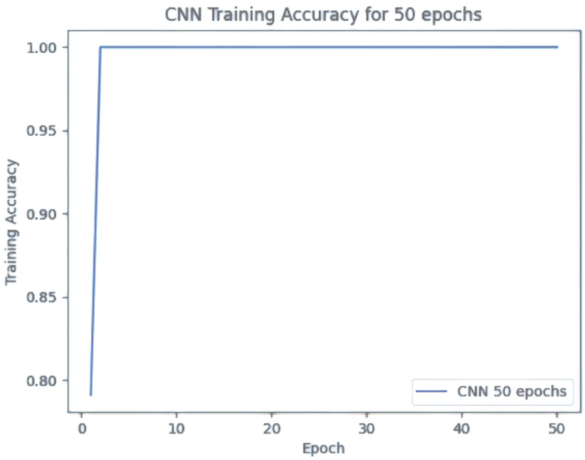

Fig. 1. ROC Curve of CNN accuracy, with 50 epochs

Table 3. Classification Report for Best RNN Model:

Class	Precision	Recall	F1-Score	Support
Goodbye	1	1	1	147
Greeting	1	1	1	163
Help	1	1	1	147
Joke	1	1	1	148
Name	1	1	1	132
News	1	1	1	138
programmer	1	1	1	155
Thanks	1	1	1	150
Time	1	1	1	156
Weather	1	1	1	164
Overall Accuracy	1			1500
Macro Avg	1	1	1	1500
Weighted Avg	1	1	1	1500

When testing with real users, RNN a at times maintains the context better than CNN, but also has challenges with inputs that deviated from the observed values (patterns) that were visualized in it during training. This indicates as a final point, that both models are very accurate in the test DataSet, but when generalized they have some problems, but this can be analyzed.

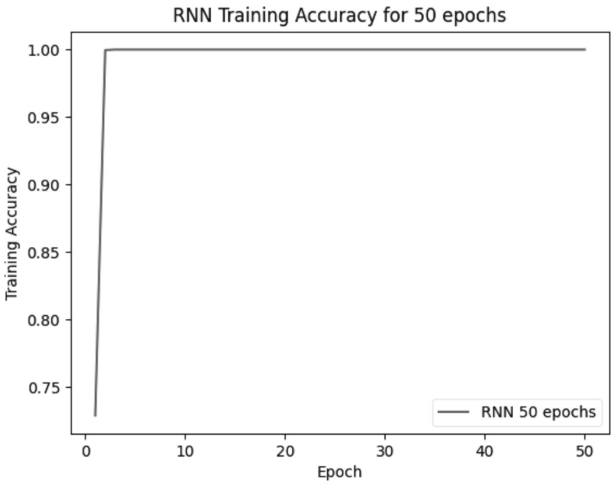

Fig. 2. ROC Curve of RNN accuracy, with 50 epochs

8.3 Model Comparison and Analysis

The following is a list of the results of both models.

The CNN model, by focusing on local patterns, has a high identification of phrases and words, which supports the identification of specific intentions in the context. Being a very efficient architecture, it can respond quickly during the question-answer process. But it can miss some circumstances where some words do not match in order and context, and this is not good because it can change the meaning of the expressions. However, the RNN model (LSTM), whose functionality is designed to maintain the context throughout a sequence, is more appropriate. This makes it more robust in scenarios where language structures are complex or ambiguous.

Both models obtained 100% accuracy in the controlled tests, indicating that the DataSet is consistent with very little inconsistency. But, in real-time user testing, the CNN misclassified some entries. The RNN can be identified as a more accurate model at long dependencies, indicating that the DataSet may not fully reflect the diversity of natural language.

Experiments performed with different numbers of epochs indicated that learning tended to a point where it no longer had growth after a certain point, between 20 and 100 epochs. This may indicate that both models can have convergence in very tight times, this when you have a well selected DataSet. But if you indicate the performance metrics this is high.

Both models show that classification errors are minimal under controlled conditions. When generating the ROC curves, they indicate in each model a good performance, but it was observed that there is a slight better performance in the CNN model versus the RNN in terms of AUC values.

The overall conclusion of the proposal, it can be indicated that for both models, it is identified that they have a very good performance in the control DataSet. If we consider values such as speed and efficiency in feature extraction in CNN model places it as a

good choice for where this is of utmost importance, and in relation to the ability of the RNN to capture elements of the context it may be ineffective in more complex situations.

9 Future Work

The controlled environments, and more when you have a performance considered good, not in all cases, is a point that can be considered equivalent in real environments.

Listed are some solutions that could be part of the proposal.

- To have a wider and more varied Dataset, contemplating real world examples,
- Having a larger DataSet could support having more arguments in the conversation.
- Consider using pre-trained models.
- To have a wider and more varied Dataset, contemplating real world examples,
- To analyze possible improvements in data preprocessing, considering: tokenization, case sensitivity, punctuation marks, to mention a few points.
- Review whether cross validation helps the final results.
- To analyze methods of regularization and argumentation.

For this proposal, the use of controlled tests was proposed to analyze performance, which was positive. In future studies we will perform tests and scenarios applicable to the real world, where the scenarios could be acoustic and analyze how the CNN and RNN models are applicable, but for this as a hypothesis it is proposed that the phases and diversify the training data should be broad and clean. The pipeline should be revised to make the work more transparent and thus achieve a considerable improvement.

References

1. Hirschberg, J., Manning, C.D.: Advances in natural language processing. Science **349**(6245), 261–266 (2015)
2. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
3. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of 2014 Conference on Empirical Methods Natural Language Processing (EMNLP), pp. 1746–1751 (2014)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
5. Vaghani, N.: "Chatbot Dataset," Kaggle (2023). https://www.kaggle.com/datasets/niraliivaghani/chatbot-dataset. Accessed 01 March 2025
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Adv. Neural. Inf. Process. Syst. **25**, 1097–1105 (2012)
7. Li, Z., Yang, W., Peng y F. Liu, S.: A survey of convolutional neural networks: analysis, applications, and prospects. Mech. Syst. Signal Process. **151**, 107398 (2021). https://doi.org/10.1016/j.ymssp.2020.107398
8. Ghojogh y A. Ghodsi, B.: Redes Neuronales Recurrentes y Redes de Memoria a Largo Corto Plazo: Tutorial y Revisión. arXiv preprint arXiv:2304.11461 (2023)
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). https://www.deeplearningbook.org/
10. Ghojogh, B., Ghodsi, A.: Recurrent neural networks and long short-term memory: a tutorial and review. arXiv preprint arXiv:2304.11461 (2023). https://arxiv.org/abs/2304.11461