The development of a chatbot using Convolutional Neural Networks

Giorgos Tsakiris[1], *Christos* Papadopoulos[1], *Giannis* Patrikalos[1], *Konstantinos-Filippos* Kollias[1*], *Nikolaos* Asimopoulos[1] *and George F.* Fragulis[1]

[1]Department of Electrical and Computer Engineering, University of Western Macedonia, 501 00 Kozani, Greece

Abstract. Chatbots are artificial intelligence systems that comprehend the intent, context, and sentiment of the user, interact properly with them leading to an increased development of their creation, the past few years. In this study, Convolutional Neural Networks (CNNs) are applied as the classifier and some specific tools for tokenization are used for the creation of a chatbot. Taking into account that it is difficult to apply any algorithm in text, we use a technique called "Word Embedding", which converts a text into numbers in order to run text processing. Specifically, "Word2Vec" Word Embedding technique was applied. AlexNet, LeNet5, ResNet and VGGNet CNN architectures were utilised. These architectures were compared for their accuracy, f1 score, training time and execution time. The results obtained highlighted that there were significant differences in the performance of the architectures applied. The most preferable architecture of our study was LeNet-5 having the best accuracy compared to other architectures, the fastest training time, and the least losses whereas it was not very accurate on smaller datasets such as our Test Set. Limitations and directions for future research are also presented.

1 Introduction

Artificial Intelligence systems called "chatbots", concerning human-machine interaction, have been widely used in the last decades. A chatbot must have the ability to comprehend the intent, context and sentiment of the user and interact properly with them. Machine Learning (ML) has played an important role in the development of chatbots. ML can be described as a subset of artificial intelligence in which a mathematical model based on "training data" is built to make decisions or predictions without being programmed [1]. It has been applied for several scientific purposes, such as education applications [19], sign language learning [2], gaming applications [3] and early and objective autism spectrum disorder (ASD) assessment [4,5]. The structure of a chatbot is based on preprocessed data that can easily be understood by its engine. This process consists of the raw text input, which is separated into single words, called tokens. The tokenized strings are subjected to text processing, creating regular expressions, and removing any punctuation marks and stop words. These entities are classified into predefined classes that, either preexist, or are created by the programmer of the system. Then, there is an intent classification during which the chatbot attempts to understand what the user wants in order to respond with an answer or a solution to the problem. There are two main steps someone should take concerning chatbots, i.e., the way in which the tokens are made and their classification [6]. Chatbots have been developed for several domains such as home automation and Internet of Things [7], customer service [8], banking [9], and elderly care [10]. All these chatbots use Natural Language Processing (NLP) which enables the communication between machine-to-user and user-to-machine utilising human natural language [11]. NLP encodes the information the user inserts into vector. In other words, it aims at text processing with computers for its analysis, information extraction and eventually representation of the same information in a different way [12].

In the current study, the creation of the chatbot is based on Convolutional Neural Networks (CNNs). CNNs are applied as a classifier and some specific tools for tokenization are used. A Convolutional Neural Network is a deep learning algorithm keeping a hierarchical structure in which layers learn from each other. Considering that it is difficult to apply any algorithm in text, a technique called "Word Embedding", which converts text into numbers in order to run text processing, was used. In this paper, "Word2Vec" technique was applied because it offers a great amount of pretrained data and enables users to train their own data set assuming they have enough data for the problem. Additionally, a unique feature of Word2vec is that vectors can be got from other vectors using vector operations [6]. This is the preprocess needed to insert the data into the CNN. AlexNet, LeNet5, ResNet and VGGNet CNN architectures were utilised.

There have been several earlier studies applying CNN for the creation of a chatbot. For example, an emotion recognition algorithm was developed by [13].

* Corresponding author: dece00063@uowm.gr

This algorithm was related to voice-enabled chatbots, and the emotion recognition consisted of two steps. Initially, the user sound received was transformed into a spectrogram by utilising Mel-frequency cepstral coefficient. This spectrogram was analysed by a CNN which classified it into five emotions. In another study by [11], a chatbot which identified users' mental states tried to prevent negative emotions and made users feel positive by urging them to have more constructive thoughts. Three deep learning classifiers were used for emotion detection i.e., Convolutional Neural Network (CNN), Recurrent Neural Network (CNN), and Hierarchical Attention Network (HAN). Therefore, CNN can be applied for chatbot creation of different domains bringing significant results.

1.1 Objective

The main objective of this paper is to use Convolutional Neural Networks (CNNs) as a classifier in order to create a chatbot and compare how different architectures can result in different training times and accuracy. Moreover, a specific tool for tokenization is used. Therefore, it can be hypothesised that different CNNs architectures can bring promising results in the creation of a chatbot.

2 Methodology

2.1 Design

There is a certain complexity regarding the choice of the appropriate methodology for making a chatbot understand the user's input, as it needs training based on the context of the conversation. According to the flowchart in Figure 1, the following steps must be taken. Initially, the data need to follow a preprocess procedure, involving the separation of the raw text input from the user into single words, called tokens. Apart from the removal of any punctuation marks and stop words, the tokenized strings go through text normalization, which can be accomplished by two techniques, stemming and lemmatisation. Stemming slices a string into smaller substrings using a set of rules (or a model). The idea is to remove word affixes (especially suffixes) that change the meaning. Lemmatisation, on the other hand, looks up every token in a dictionary and returns the canonical "head" word in the dictionary, which is known as lemma. It can handle unusual cases as well as tokens with diverse parts of speech because it looks up tokens from a ground truth. Both stemming and lemmatisation have benefits and drawbacks. Stemming is faster because it simply asks users to splice word strings. Lemmatisation, on the other hand, dictates a lookup in a dictionary or database and relies on part-of-speech tags to determine a word's root lemma, making it considerably slower but more effective than stemming. The latest is the one that is used in this research [14].

It is difficult to apply any algorithm in text, so a technique called "Word Embedding", which converts text into numbers, is applied. Word Embedding can be done by a variety of tools such as, "Word2Vec", "Count Vector" and "Term Frequency-Inverse Document Frequency (TF-IDF)". In this paper, Word2Vec and more specifically an extension of it, called "Doc2Vec" was utilised [14]. Word2Vec takes single words as input, whereas Doc2Vec takes a whole sentence as input. Word2Vec offers a big amount of pretrained data that can be found online and used on any system. Additionally, it lets users upload their own data to train the network. However, a lot of data is required in order to have a successful training process. Also, a unique feature of Word2vec is that vectors can be got from other vectors using vector operations [6]. These entities are classified into predefined classes that, either preexist or are created by the programmer of the system. Word Embedding is followed by entity classification, a procedure in which there is a recognition of information units derived from unstructured text such as names, including person, organisation names, numeric expressions, and location names. The identification of references to these entities in text was recognised as one of the significant sub-tasks of Information Extraction (IE) [15]. Then, intent classification follows during which the chatbot tries to understand what the user wants before finally responding with an answer or a solution to the problem.

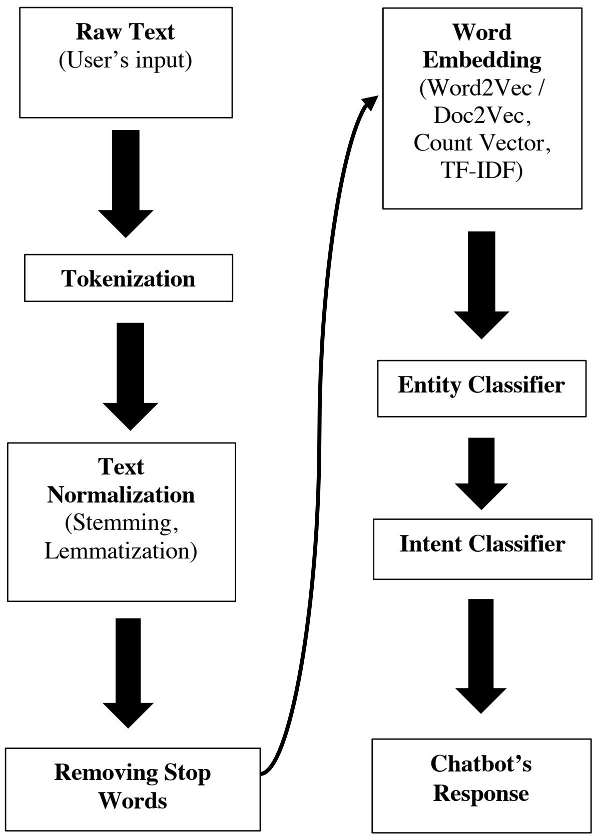
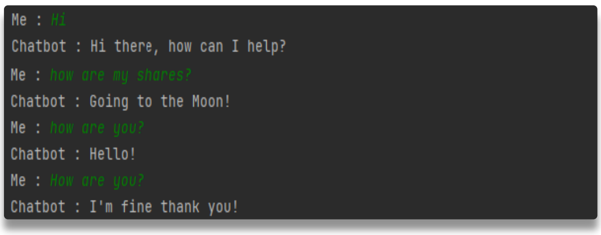

Raw Text (User's input)
Tokenization
Text Normalization (Stemming, Lemmatization)
Removing Stop Words
Word Embedding (Word2Vec / Doc2Vec, Count Vector, TF-IDF)
Entity Classifier
Intent Classifier
Chatbot's Response

Fig. 1. A flowchart showing how a chatbot engine processes an input string and gives a valid reply.

2.2 Proposed Approach

CNNs are a deep learning algorithm keeping a hierarchical structure in which layers learn from each other. They are one of the most popular deep learning classifiers associated with image classification, as well. CNNs have also been used for Relation Classification and Relation Extraction [16,17]. CNNs have multiple layers, such as the convolutional layer and the fully connected layer, which have parameters, as well as the non-linearity layer, and the pooling layer which have no parameters. AlexNet, LeNet5, ResNet and VGGNet CNN architectures were utilised in this study and the differences between them are presented in Table 1. In image classification task, LeNet-5 is the only architecture that gets greyscale images as input.



```
Me : Hi
Chatbot : Hi there, how can I help?
Me : how are my shares?
Chatbot : Going to the Moon!
Me : how are you?
Chatbot : Hello!
Me : How are you?
Chatbot : I'm fine thank you!
```

Fig. 2. An example of our model's performance.

Table 1: The number of the main layers used by the architectures.

	Convolutional Layers	Fully Connected Layers	Pooling Layers	Output Activation Function
AlexNet	5	3	3	Softmax
LeNet-5	2	3	2	Softmax
ResNet-50	49	1	2	Softmax
VGGNet	13	3	5	Softmax

The experimental procedure was utilised on a computer with Intel Core i7-6700HQ (2.60GHz), 16GB of RAM, Windows 10 Home OS (Version 20H2), GeForce GTX 960M Graphics Card. The necessary code was written in Python 3.8 and the Anaconda Virtual Environment was also used to work with the chatbot's environment to avoid files running locally on the computer. NLTK library was added, as well.

CNN models were trained for the implementation of the chatbot [Table 2]. The training was done for 200 epochs on each model, using an existing dataset, which contained 826 questions along with 352 unique answers [18]. As Loss Function we used "sparse_categorical_crossentropy" whereas "Adam" was employed as an optimiser.

A test-dataset was used for model evaluation. The Test Set used 1/3 of the original data of the dataset.

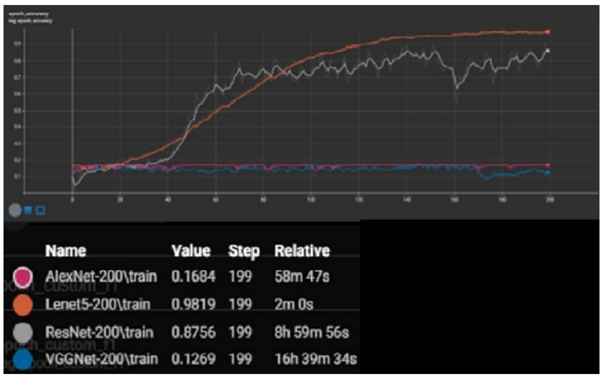
3 Results and Discussion

This study utilised Convolutional Neural Networks (CNNs) as a classifier to create a chatbot and compare how different architectures can result in different training times and accuracy. Moreover, a specific tool for tokenization was used. Figure 1 presents an example of our chatbot's performance in which 75% of the questions asked are answered properly.

On Table 2 and Figures 3-5, it is shown that LeNet-5 achieved the highest accuracy and the smallest loss requiring the minimum training time. On the other hand, VGGNet achieved the lowest accuracy and the highest loss requiring the maximum training time.

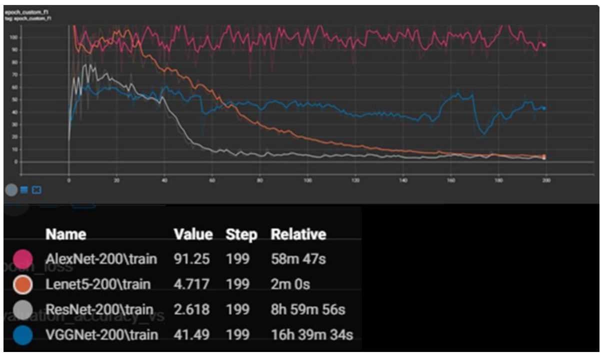
Table 2: The results of each architecture after training for 200 Epochs.

	Training Time (for 200 epochs)	Accuracy	F1-score	Loss
AlexNet	58mins:47secs	0.1684	91.25	3.999
LeNet-5	2mins:00secs	0.9819	4.717	0.1548
ResNet-50	8h:59mins:56sec	0.8756	2.618	0.4388
VGGNet	16h:39mins:34secs	0.1269	41.49	4.46



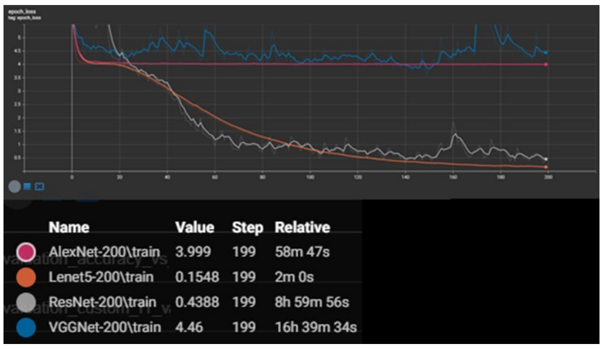
	Name	Value	Step	Relative
●	AlexNet-200\train	0.1684	199	58m 47s
●	Lenet5-200\train	0.9819	199	2m 0s
●	ResNet-200\train	0.8756	199	8h 59m 56s
●	VGGNet-200\train	0.1269	199	16h 39m 34s

Fig. 3. Epoch accuracy.



	Name	Value	Step	Relative
●	AlexNet-200\train	91.25	199	58m 47s
●	Lenet5-200\train	4.717	199	2m 0s
●	ResNet-200\train	2.618	199	8h 59m 56s
●	VGGNet-200\train	41.49	199	16h 39m 34s

Fig. 4. F1 – Score.



	Name	Value	Step	Relative
●	AlexNet-200\train	3.999	199	58m 47s
●	Lenet5-200\train	0.1548	199	2m 0s
●	ResNet-200\train	0.4388	199	8h 59m 56s
●	VGGNet-200\train	4.46	199	16h 39m 34s

Fig. 5. Loss.

Table 3 presents the evaluation numbers of each architecture. It is obvious, that values differ a lot from the ones depicted on Table 2 on some occasions. LeNet-5 still achieved the highest accuracy but not the smallest loss, as it did on the entire dataset. Also, it is noticeable that ResNet-50 attains 0 accuracy with the highest loss, whereas no results were obtained for the VGGNet architecture.

Table 3: The evaluation numbers of each architecture.

	Preprocess Time	Accuracy	F1-score	Loss
AlexNet	1192.9145 sec	0.14	98.13	6.98
LeNet-5	7.6288 sec	0.21	12.13	7.40
ResNet-50	1156.0755 sec	0.0	3.60	19.49
VGGNet	--	--	--	--

4 Conclusion – Future Directions

This paper highlighted the construction and training of a general purpose chatbot. The methodology behind the creation of the chatbot was analysed step by step using four different architectures of Convolutional Neural Networks (CNNs) as a classifier. The results obtained differ in terms of the architectures used, concluding that the best architecture is LeNet-5, as it has the best accuracy, as well as the fastest training time and the least losses. It should be pointed out that on smaller datasets, such as our Test Set, this method is not very accurate. In the future, we plan to increase accuracy for each architecture keeping the training time in reasonable limits. Additionally, we would like to apply more architectures in order to compare them and find the most optimal one. Finally, we could utilise more data sets, and thus the chatbot could get more expertise on specific areas of interest.

References

1. Zhang, X.-D. A Matrix Algebra Approach to Artificial Intelligence; Springer, 2020;
2. Papatsimouli, M.; Lazaridis, L.; Kollias, K.-F.; Skordas, I.; Fragulis, G.F. Speak with Signs: Active Learning Platform for Greek Sign Language, English Sign Language, and Their Translation. 2020, doi:10.48550/arXiv.2012.11981.
3. Hitboxes: A Survey About Collision Detection in Video Games | SpringerLink Available online: https://link.springer.com/chapter/10.1007/978-3-030-77277-2_24 (accessed on 21 March 2022).
4. Kollias, K.-F.; Syriopoulou-Delli, C.K.; Sarigiannidis, P.; Fragulis, G.F. The Contribution of Machine Learning and Eye-Tracking Technology in Autism Spectrum Disorder Research: A Review Study. In Proceedings of the 2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST); IEEE, 2021; pp. 1–4.
5. Kollias, K.-F.; Syriopoulou-Delli, C.K.; Sarigiannidis, P.; Fragulis, G.F. The Contribution of Machine Learning and Eye-Tracking Technology in Autism Spectrum Disorder Research: A Systematic Review. Electronics 2021, 10, 2982.
6. Manaswi, N.K.; Manaswi, N.K.; John, S. Deep Learning with Applications Using Python; Springer, 2018;
7. Baby, C.J.; Khan, F.A.; Swathi, J.N. Home Automation Using IoT and a Chatbot Using Natural Language Processing. In Proceedings of the 2017 Innovations in Power and Advanced Computing Technologies (i-PACT); April 2017; pp. 1–6.
8. D'silva, G.M.; Thakare, S.; More, S.; Kuriakose, J. Real World Smart Chatbot for Customer Care Using a Software as a Service (SaaS) Architecture. In Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC); February 2017; pp. 658–664.
9. Kulkarni, C.S.; Bhavsar, A.U.; Pingale, S.R.; Kumbhar, S.S. BANK CHAT BOT – An Intelligent Assistant System Using NLP and Machine Learning. 04, 5.
10. Su, M.-H.; Wu, C.-H.; Huang, K.-Y.; Hong, Q.-B.; Wang, H.-M. A Chatbot Using LSTM-Based Multi-Layer Embedding for Elderly Care. In Proceedings of the 2017 International Conference on Orange Technologies (ICOT); December 2017; pp. 70–74.
11. Patel, F.; Thakore, R.; Nandwani, I.; Bharti, S.K. Combating Depression in Students Using an Intelligent ChatBot: A Cognitive Behavioral Therapy. In Proceedings of the 2019 IEEE 16th India Council International Conference (INDICON); IEEE, 2019; pp. 1–4.
12. Conneau, A.; Schwenk, H.; Barrault, L.; Lecun, Y. Very Deep Convolutional Networks for Text Classification. arXiv preprint arXiv:1606.01781 2016.
13. Lee, M.-C.; Chiang, S.-Y.; Yeh, S.-C.; Wen, T.-F. Study on Emotion Recognition and Companion Chatbot Using Deep Neural Network. Multimedia Tools and Applications 2020, 79, 19629–19657.

14. Bengfort, B.; Bilbro, R.; Ojeda, T. Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning; O'Reilly Media, Inc., 2018; ISBN 978-1-4919-6299-2.
15. Nadeau, D.; Sekine, S. A Survey of Named Entity Recognition and Classification. Lingvisticae Investigationes 2007, 30, 3–26.
16. Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; Zhao, J. Relation Classification via Convolutional Deep Neural Network. In Proceedings of the Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers; Dublin City University and Association for Computational Linguistics: Dublin, Ireland, August 2014; pp. 2335–2344.
17. Nguyen, T.H.; Grishman, R. Relation Extraction: Perspective from Convolutional Neural Networks. In Proceedings of the Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing; Association for Computational Linguistics: Denver, Colorado, June 2015; pp. 39–48.
18. Question-Answer Dataset Available online: https://kaggle.com/rtatman/questionanswer-dataset (accessed on 16 January 2022).
19. Fragulis, G. F., Papatsimouli, M., Lazaridis, L., & Skordas, I. A. (2021). An Online Dynamic Examination System (ODES) based on open source software tools. Software Impacts, 7, 100046.