Improving the Accuracy of Customer Service Seq2Seq Chatbots Through Dataset Pruning

Osama Hosameldeen * , Rasha Abousamra , Hussain Al-Aqrabi , Ossama Embarak , Usman Durrani

Posted Date: 3 March 2023

doi: 10.20944/preprints202303.0070.v1

Keywords: encoder; decoder; seq2seq; LSTM; RNN; chatbots

Article

Improving the Accuracy of Customer Service Seq2Seq Chatbots Through Dataset Pruning

Osama Hosam [1,*], Rasha Abousamra [2], Hussain Al-Aqrabi [3], Ossama Embarak [4] and Usman Durrani [5]

[1] CIS Department, Sharjah College, Higher Colleges of Technology, Sharjah, UAE
[2] Business Department, Sharjah Campuses, Higher Colleges of Technology, Sharjah, UAE; rabousamrah@hct.ac.ae
[3] CIS Department, Sharjah College, Higher Colleges of Technology, Sharjah, UAE; halaqrabi@hct.ac.ae
[4] CIS Department, Fujirah College, Higher Colleges of Technology, Fujirah, UAE; oembarak@hct.ac.ae
[5] CIS Department, Higher Colleges of Technology, Sharjah, UAE; udurrani@hct.ac.ae
* Correspondence: mohandesosama@yahoo.com

Abstract: Chatbots are extensively needed in customer services to handle customer inquiries, such as tracking orders or providing information about products and services. One of the most reliable implementations of chatbots is using the common architectures of LSTM networks named Seq2Seq networks. The networks are using an encoder and a decoder. Seq2Seq chatbot is a type of chat system that is professional enough to pass the Turing test. The Turing test is a way of deciding the accuracy of the machine by examining its response, it should appear like a human response. In this research, we will introduce a novel architecture that can pass the Turing test. The seq2seq Accuracy is improved by making incremental training to the chatbot. The new proposal provides higher accuracy and high similarity to human chat responses.

Keywords: encoder; decoder; seq2seq; LSTM; RNN; chatbots

1. Introduction

Automating customer service through the use of chatbots has become increasingly popular in recent years. This is due to the efficiency and cost-effectiveness that chatbots can provide in handling customer inquiries. Chatbots are computer programs that use natural language processing and machine learning techniques to simulate human conversation. They can be integrated into websites, mobile apps, and messaging platforms to interact with customers in real time. Chatbots have the capability to automate customer service processes, providing a convenient and efficient means of handling customer inquiries.

In the logistics and industrial systems engineering sectors, chatbots can be programmed to assist customers in tracking their orders and providing detailed information about products and services. This can include real-time updates on order status, delivery times, and any other relevant details. Additionally, chatbots can be configured to provide customers with information about product specifications, availability, and pricing. This not only improves the customer experience but also frees up customer service representatives to focus on more complex and nuanced inquiries. Furthermore, chatbots can work 24/7, providing customer service even outside of business hours, which can be beneficial for customers and companies alike. Therefore, the implementation of chatbots for handling customer inquiries in logistics and industrial systems engineering can lead to increased efficiency and customer satisfaction.[1]

In the early nineteenth, research started to focus on neural networks. The research was able to adopt neural networks for applications such as classification and pattern recognition. However, at that time, neural networks were not existing strongly in the field of speech recognition [2]. Hidden Markov models [3], especially Gaussian mixture model was dominating the speech recognition field [4]. Scientists were trying to replace the Gaussian mixture model with neural networks. The results

 doi:10.20944/preprints202303.0070.v1

were not acceptable compared to the results of the Gaussian mixture model, this is because of not having multiple layers in the neural networks, and because of no available computing power for training large data sets composed of thousands of utterances of speech.

With the invention of deep neural networks in the early twentieth, speech recognition was possible using neural networks. Google provided a pre-trained neural network to solve speech recognition problems. The ideas applied before deep neural networks are like the recent ideas, the only difference is that the computing power was limited to achieve good results, especially in the training process [5]. Deep neural networks invaded all speech recognition-related fields, such as natural language processing, machine translation, and chatbots.

Seq2seq is initially designed by Chao et al. [6] It was developed as a machine translation model. Research is done extensively on seq2seq to discover its ability to map anything to another thing. Such as using seq2seq as chatbot or forecasting. This research is focusing on chatbots using seq2seq architecture. The seq2seq architecture is composed of two sub-parts, namely, the encoder and the decoder. The encoders and decoders are implemented using LSTM networks. LSTM can detect the intra-relationship between words in the text. Hence, providing accurate predictions when doing language processing.

Maximum likelihood estimation (MLE) is used as an objective function for machine translation. However, the responses are too generic and sometimes not related to the input text. It depends on replying with the words that appear most – having a higher frequency, in the input corpus. Li et al. [7] solved this problem by using the original seq2seq architecture by adding an anti-model feature. The anti-model penalizes the words that have more interesting meanings and are informative to create a more realistic response. The work done by Li et al. used the BLEU (bilingual evaluation understudy) as a metric for finding the best probability weights [8]. However, BLUE sometimes produces zero value for meaningful and informative input. The seq2seq model is trained with the dataset for one time or more than one time and produces a trained seq2seq model. This trained seq2seq model is used in action as a chatbot. The problem is that the dataset may change over time and the model will not perform well with the new dataset. Therefore, the model may be re-trained on the new dataset. The process is costly as the training of the neural network should start from scratch with random weights. To solve this problem, we proposed an incremental training approach, in which the training is done at the beginning, and the trained model is then saved in a file. When new sentences and answers are encountered in business, the model is incrementally trained, i.e., the saved model will be trained again. The saved model training will not start from scratch as the model has the correct weights.

2. Seq2Seq Architecture

Sequence to Sequence (seq2seq) is one of the most common architectures of RNN networks. It is developed by Google [9] to solve machine translation problems. Seq2seq is used to solve lots of problems such as Machine Translation, Chatbots, and Forecasting. Seq2seq architecture uses the power of LSTM together with a new architecture called encoding-decoding architecture. The seq2seq architecture is composed of an embedding layer followed by an encoder, the encoder is a collection of LSTM units. The output part of seq2seq model is the decoder which is composed of another collection of LSTM units.

Seq2seq architecture is adopted in many machine translation applications because it extrapolates the relationship between the encoder input unit and decoder output unit. The input encoder is deriving a non-linear relationship by adopting an embedding layer and LSTM unit. Seq2seq contributes to many modern concepts such as attention [10,11] which is a fundamental concept in most contemporary forms of deep learning. Attention is used in machine translation to reduce the input sequence letting the network focus only on the important parts of the text. Attention is applied so the decoding part can get more accurate results.

To implement this architecture, a vocabulary should be created and passed to the embedding layer. The embedding layers take input words and convert them to lower-dimension vector representation.

 doi:10.20944/preprints202303.0070.v1

The vectors are passed to the encoder in which LSTM captures the context in the hidden state and the cell state. The encoder architecture is implemented by selecting the number of layers and type of activation, etc. The decoder architecture should also be decided. Finally, the entire model will be trained on an existing dataset; seq2seq model will be given a sequence of words in the original text language, together with the corresponding sequence in a target language. For example, the model will be given a sentence in English and the corresponding sequence in Spanish. The model will be trained on thousands of such types of sentences. The final trained model will be used in translation applications thereafter.

2.1. Input Vocabulary and the Embeddings

The vocabulary is represented by an object that contains all the functionality needed by a vocabulary, such as loading, indexing, and retrieving words of our vocabulary. The vocabulary is originally taken from a corpus. Sequences are defined inside the vocabulary by using start and end delimiters. These sequencing process helps to improve the performance of the entire seq2seq model. Some functionalities will help in padding sequences with the end-of-a-sentence (EOS) and start-of-sentence (SOS) tokens. A collection of functions to add and remove words from the vocabulary will be implemented.

Each word in the entire vocabulary is represented by a single vector in which all other words in the vocabulary are denoted as 0 and the current word is represented by 1. The length of the vector is equal to the vocabulary size [12]. The problem with this representation is that if the "mice eat cheese" sentence is represented, one can infer that "rat eats cheese" but also "horse eats cheese" can be represented without logical distance between the three sentences. So we need another representation to say that rat and mice sentences have a small distance and rat and horse sentences have a large distance.

The embeddings are represented with low-dimension vectors, embeddings add some semantics to the sentence. The collection of words in that vector can represent the entire vocabulary. This way, "mice eat cheese" and "horse eat cheese" will be represented with two vectors having large humming distance because of the horse's size and mice's size are too different. The embeddings are learned automatically in the neural network. We pass for example 100-dimension vector of words and the neural network decides what is the corresponding numerical value for each word. For the neural network, each word is passed in on-hot representation and then converted into the embedding representation in the embedding layer.

2.2. The Encoder

The encoder is generally composed of two units, the LSTM cells, and the Embedding layer. Optionally, a dropout layer may be added to overcome the overfitting problem in model training. The encoder's job is to represent the input sequence in the hidden states on LSTM cells. Then it passes the hidden state to the output section of the seq2seq architecture. Before data is passed to the encoder, it must be first passed by an embedding layer. A pre-trained embedding layer can be used to leverage the existing pre-trained model and obtain higher prediction accuracy. The encoding process is visually depicted in Figure 1. The input vector is passed to the encoder. The encoder creates a lower dimensional vector which is then passed to the LSTM layer. Finally, the output is passed to a dropout layer for better model generalization.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2023 doi:10.20944/preprints202303.0070.v1

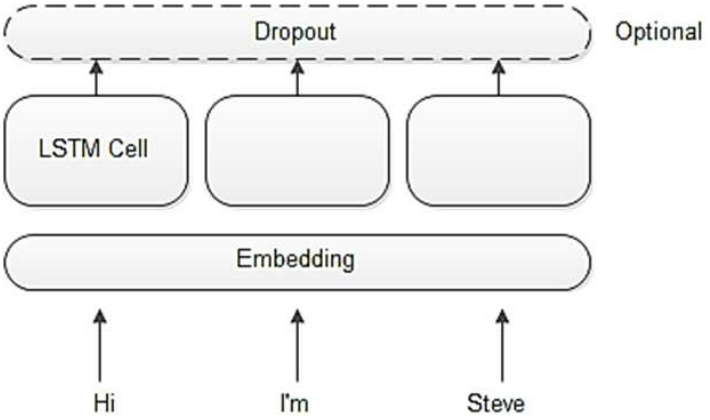

Figure 1. The Seq2Seq Encoder which is a collection of LSTM cells. The lower level is the embedding unit and the output can be optionally passed to a dropout layer

2.3. The Decoder

The decoder's job is to output a sequence based on the input from the encoder's hidden state and the previous prediction. The decoder uses the given previous n-1 predictions to predict the n prediction. Notice that the decoder can have on or more LSTM layers. We rely on a linear layer to output the prediction according to the number of words in the given vocabulary. The last layer is the SoftMax activation, different activation techniques can be used in this layer; for example, a linear function can be used with cross-entropy loss. Others use linear layers with log-SoftMax activation function. The architecture of the decoder is shown in Figure 2, notice the use of SOS as an indication of the sequence beginning. SOS prompts the network to predict its n prediction based on the hidden state, and its n+1 prediction based on the previous predictions at time n. The size of predictions is equal to the sequence size.

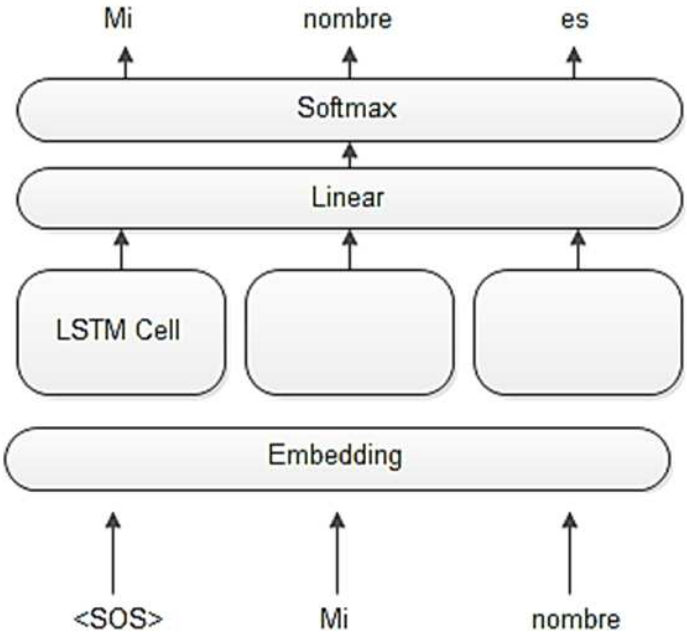

Figure 2. The seq2seq Decoder, is a collection of LSTM units, and the lower level is the embedding representation of the input text. The embeddings are passed to the layer of LSTM cells then passed to the Linear layer and finally a SoftMax function

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2023 doi:10.20944/preprints202303.0070.v1

2.4. The Entire seq2seq Architecture

LSTM cell takes the current state and a vector of words and creates a subsequent LSTM state (new state) which "encodes" the input sentence so far. The probability distribution of the vocabulary constructed from the LSTM memory and the last transformation. This simple model can be used to predict part of speech or the syntax but still can't predict a sequence of words. To predict the sequence of words, an encoder should be used to create the required representation. Each memory vector of cells in the encoder tries to represent the sentence so far, but in action, it only represents the latest recently input word.

The model that takes the encoded input representation and generates the output is the Decoder. It is also a collection of LSTM cells as shown in Figure 3. The decoder is trained by forcing it to create gold sequences [13]. It penalizes the weak sequences by assigning low probability. Losses of all tokens are summed up and used to guide the direction of training (predicting the next step), as the goal is to minimize the total loss. This is a simple stochastic gradient descent (SGD) training [14]. The loss is calculated by the cross-entropy formula. The network creates a probability distribution over the expected words. The probability is penalized by calculating the difference between the expected token and the real token values.

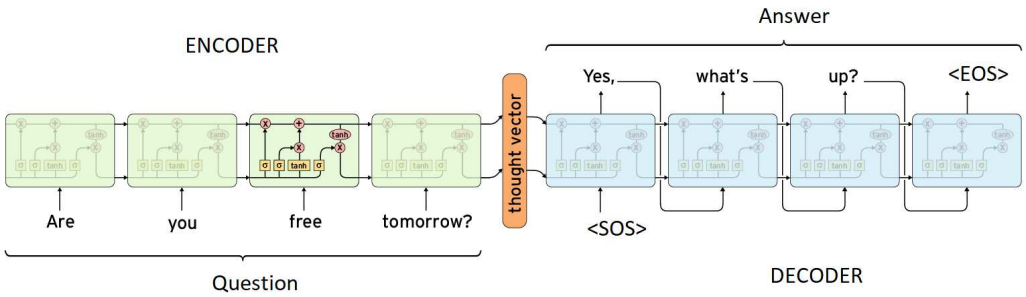

Figure 3. The entire Seq2Seq chatbot architecture. It contains an input encoder and an output decoder. The encoder passes a vector representation of the input sentence to the decoder. The question has a corresponding answer with $< SOS >$ and $< EOS >$ tags delimiters

3. The Proposed Methodology

The proposed seq2seq architecture contains two LSTM networks. The first one is the encoder which reads a sequence of words and outputs a function of the hidden state. These hidden states would capture the semantics of the provided sequence. The second LSTM network is the decoder which takes the semantic structure from the encoder and generates the corresponding output sequence. It creates the output one word at a time by looking at the semantic and previous state in each cycle.

Our approach concentrates on enhancing the dataset by pruning it. The dataset will reject the dataset samples that are rarely used and add newly added samples. The proposed approach shown in Figure 4 contains the following steps:

- Step1: The original dataset is collected from the business environment. The dataset for questions and answers for the chatbot is collected and prepared for training the proposed seq2seq architecture.
- Step2: After sufficient time, a pruned dataset will be collected. The pruned dataset has new sentences used and old sentences removed.
- Step3: Train the seq2seq architecture. The training is done by using the pruned dataset.
- Step4: For more new/old sentences arises, the architecture is retrained, and a newly trained model is obtained.
- Step5: Use the trained architecture in testing. The testing is done by asking the chatbot a collection of questions and seeing the corresponding answer

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2023 doi:10.20944/preprints202303.0070.v1

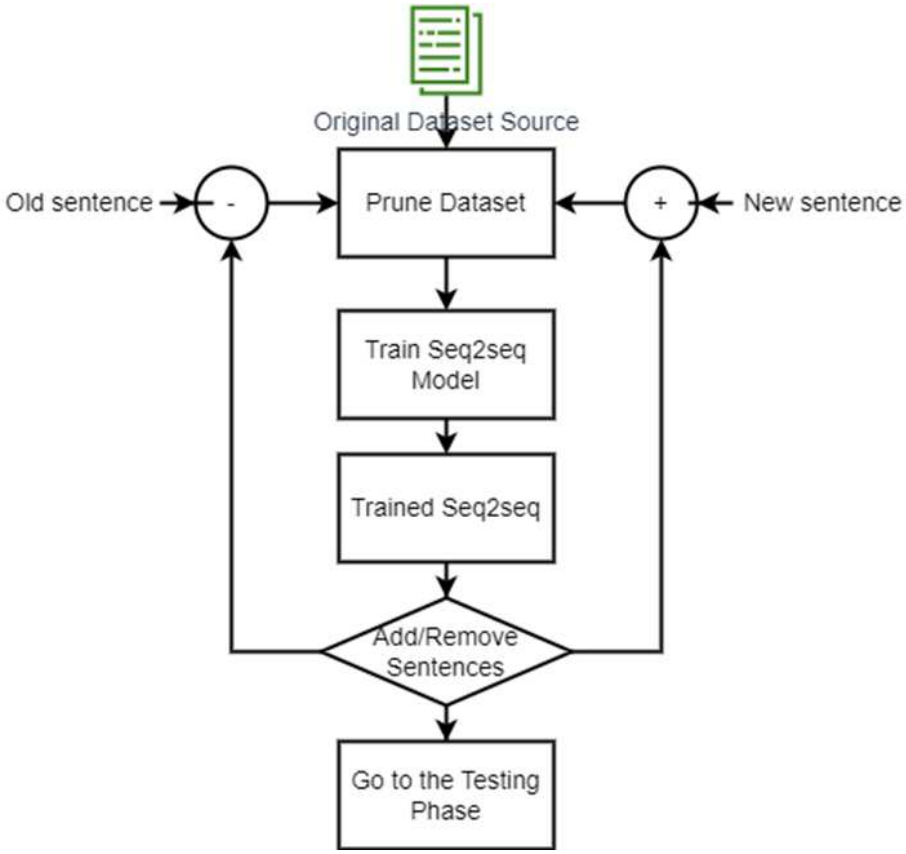

Figure 4. The proposed system. The training is done through a pruned dataset. The dataset is changed whenever new sentences arise or new sentences are not used for a long time

4. Results and Discussion

A chatbot will be built to be able to chat and reply to a variety of questions. The chatbot uses a seq2seq text generator model with LSTM as the memory unit that can join words and understand the context. The pre-trained embeddings will be used to improve the performance of the proposed model. In this project, Brown embeddings [15] will be used to create the required word embeddings. We will compare the performance of the model with and without using word embeddings. The model contains an encoder that contains an embedding layer together with LSTM unit. In addition, the model includes a decoder that contains an embedding layer, together with an LSTM unit and a linear output unit. An experiment will be executed to let the seq2seq model accept input to the encoder, pass the hidden context from the encoder to the decoder, and finally, the decoder outputs a series of token predictions. GPU is needed for training since seq2seq is a very large model. The GPU used is for training the seq2seq model multiple times until a better model is obtained. Will do our experiments through the following phases:

- Build the vocabulary and the corresponding word embeddings: In this step, a vocabulary will be created by using a corpus of data. The word embeddings will be extracted from one of its corpuses. A functionality is added for cleaning the text and looking up the index of a word's embeddings.
- Create the encoder: In this step, an encoder is implemented by using the LSTM unit. The encoder takes the embeddings and processes a single word at each time stamp. Its job is to create a feature vector that encodes only the required information and neglects the unnecessary information.
- Create the decoder: A second LSTM unit will be used in this step. The hidden state influences the subsequent hidden states. The final hidden state is seen as a summary of the provided sequence. That summary is represented by a vector called the context vector. The context vector is then passed to the decoder to generate another sequence, one word at each time stamp.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2023 doi:10.20944/preprints202303.0070.v1

- Combine all the above in a single Seq2Seq architecture: The encoder and the decoder will be combined in a single architecture. The seq2seq architecture uses two instances of the encoder and decoder then accepts the inputs of these units and manages the interaction between them to get the expected output.
- Train and evaluate the implemented architecture: the input vocabulary and embeddings will be used for training the implemented architecture. The validation is measuring the accuracy of the implemented model. The validation tells us that the model is responding as expected, over-fitting or under-fitting. It is expected that the model will perform without over-fitting or under-fitting, otherwise, we return to the architecture to change it and retrain to calculate the new accuracy.
- Interact with the chatbot: The chatbot is tested by asking some questions and seeing the corresponding output. The output is expected to be accurate if it simulates a human response.

4.1. SQuAD 2.0 Dataset

The dataset used is the Stanford Question Answering Dataset (SQuAD) [16]. The dataset is a collection of answers from crowd-workers after reading Wikipedia articles. The dataset contains the questions and the corresponding responses. SQuAD2.0 contains 100,000 questions drawn from SQuAD1.1 in addition to a collection of 50,000 questions added maliciously by the crowd-workers to appear as legitimate responses. The system is expected not only to answer the question but also refrain from answering some questions because of no support from the given paragraphs. The number of lines per split is trained: 130319, dev: 11873

```
['When did Beyonce start becoming popular?', 'What areas did Beyonce compete in
when she was growing up?', "When did Beyonce leave Destiny's Child and become a
solo singer?", 'In what city and state did Beyonce  grow up? ', 'In which decad
e did Beyonce become famous?', 'In what R&B group was she the lead singer?', 'W
hat album made her a worldwide known artist?', "Who managed the Destiny's Child
group?", 'When did Beyoncé rise to fame?', "What role did Beyoncé have in Desti
ny's Child?"]
--------------------------------------------------------------------------------
['in the late 1990s', 'singing and dancing', '2003', 'Houston, Texas', 'late 19
90s', "Destiny's Child", 'Dangerously in Love', 'Mathew Knowles', 'late 1990s',
'lead singer']
```

Figure 5. Sample of SQuAD2.0 dataset shown the questions (top part) and the corresponding answer (bottom part). For example, the question "When did Beyonce start becoming popular?" has the answer "in the late 1990".

4.2. Pretrained embeddings

The word2vec model from Gensim will be used to extract word embeddings. Python's Gensim library contains several pre-trained models. In the training process, the word2vec algorithm iterates over the sentences drawn from the Brown corpus- part of NLTK data [17]. The word embeddings are loaded for one time.

4.3. System Performance

The proposed system performance is shown in Figure 6. The experiment is done with a hidden size of 500. The encoder and decoder number of layers are 2 layers. The dropout probability is 0.1 and the batch size is 64. We will also use the teacher forcing ratio to be 1.0 with a learning rate of 0.0001. The number of iterations is 14000. The chosen optimizer is Adam optimizer which proved to good performance with chat-bots. The training time is comparatively long. You should be using a GPU for training, otherwise, you will not end up the training cycle. The CPU is 4x slower than GPU so, you are expected to take from 8 to 10 seconds for each training step on the CPU. For a batch of 64 as in our experiment, it takes about 2 to 3 seconds for each step using the GPU. The training of about 100 thousand samples in an epoch will take more than one hour. We need to train for more than 4 epochs to see a human-like response from the chatbot.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2023 doi:10.20944/preprints202303.0070.v1

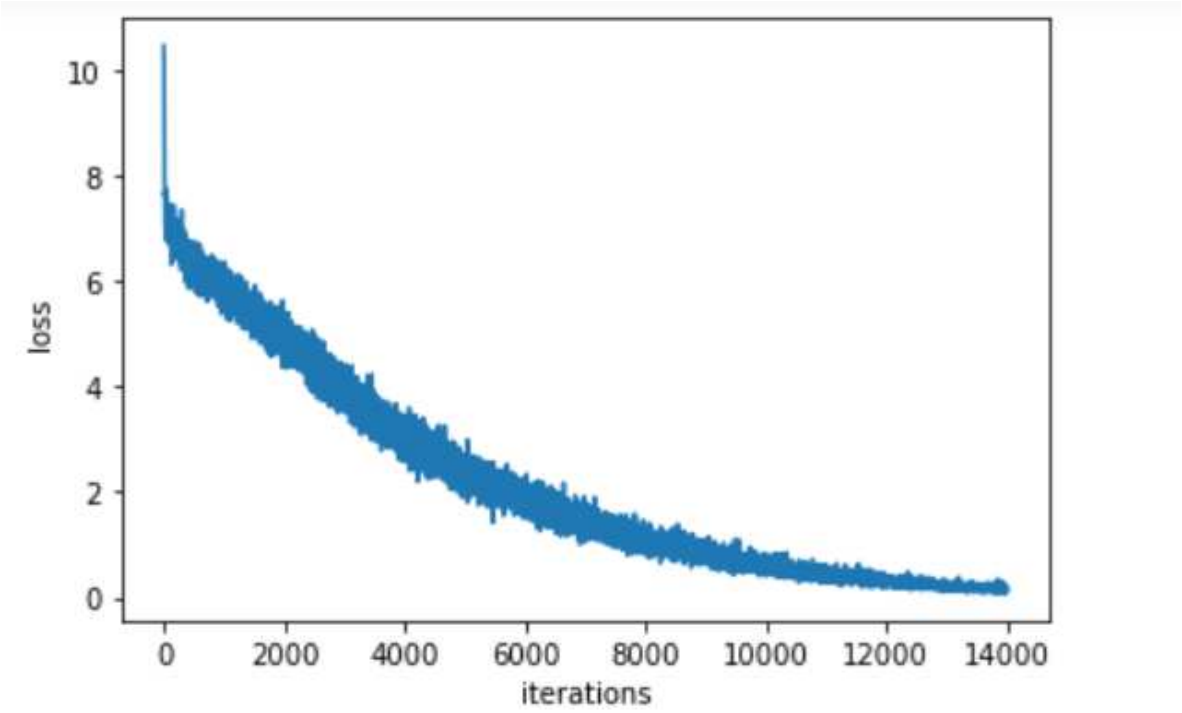

Figure 6. The proposed system performance. The training accuracy is very high as the loss is gradually decreasing.

The above-measured performance is easy to calculate. The performance is measured by the loss or how many mismatches between the expected outputs and the actual output. In other words, the loss function used is an approximation of the softmax loss. it means nothing in terms of actual dialogue. However, there is no quantitative or scientific method to test the chatbot. As we want to measure human-like speech. A Turing test can be used by a human to see how much the response is likely to come from an actual person and not a bot. So, the only way to test chatbots is to see their actual response and compare it to the probable human response. That's why we need to see the response of our proposed chatbot. The following listing shows the performance of the proposed chatbot. The listing shows how much the response is too near to a human response.

```
> When did Beyonce start becoming popular?
Bot: in the late 1990s
> What event caused Beyonce's depression?
Bot: split in luckett and rober
> How long was she depressed?
Bot: a year
> Who is beyonce married to?
Bot: jay z
> Who managed Destiny's Child?
Bot: mathew knowles and bp
> Where was Chopin born?
Bot: zelazowa wola in poland
> What is KMC an initialism of?
Bot: kathmandu metropolitan city
> what is another name for a bar?
Bot: pub
> How many buttons do most iPods use?
Bot: five
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2023 doi:10.20944/preprints202303.0070.v1


```
> quit
```

Another run of the chatbot produced the following interaction dialogue

```
> When was the Berne Convention?
Bot: 1977
> What did Hussein compare Hopkins' remarks to?
Bot: Extensive employed as art as alleles art
> What will often add bopomofo phonetic symbols?
Bot: Movie
> Where were the piratical kingdoms from?
Bot: Northern coast
> What colony was Roger Williams founder of?
Bot: France
```

4.4. Comparison of the proposed approach with traditional chatbots

The comparison is shown in Figure 7. We used 1700 iterations in the training process. Samples of the iterations are shown in the figure. The sampling is done for the simplicity of the figure. The training process for the proposed chatbot is stable compared to the traditional chatbot with semi-fluctuations in the loss value. The performance of the proposed system has high accuracy in the training process. This is very clear because the curve of the proposed system starts and ends with a low loss compared to the traditional system. It is clear that the pruning of the dataset has a high effect in decreasing the loss of the chatbot in the training process.

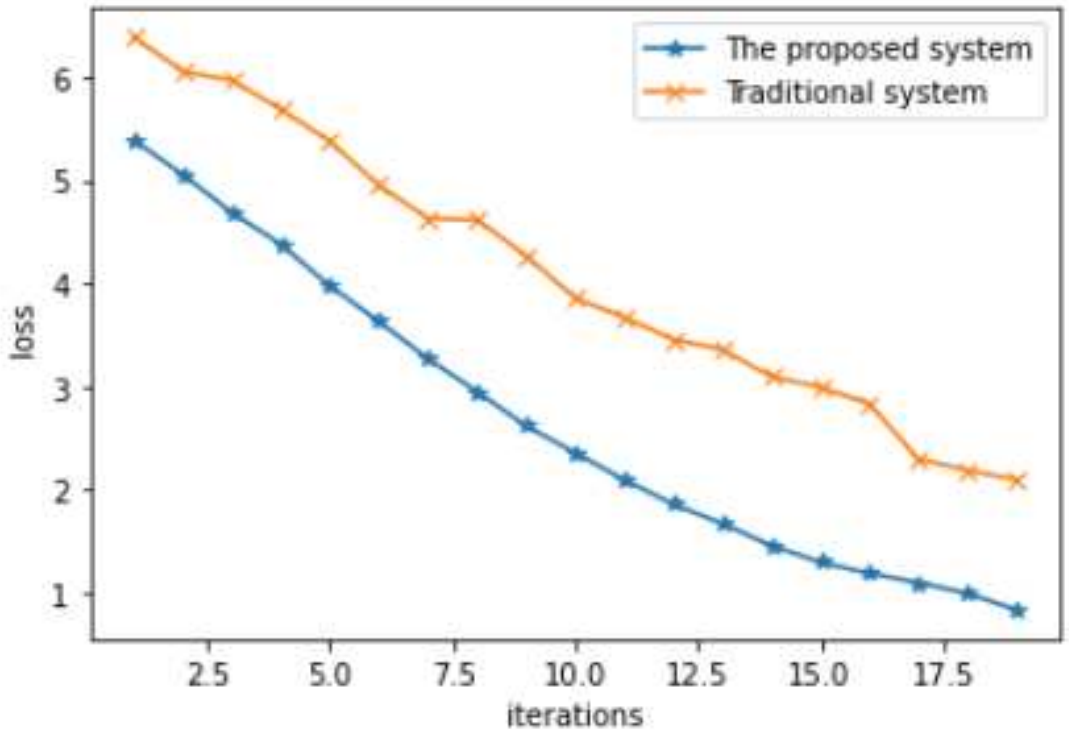

Figure 7. The comparison of the proposed approach with traditional chat-bots. The figure shows how modifying the chatbot data set affects the performance of the trained chatbot.

5. Conclusions

The seq2seq architecture is improved over the last couple of years. We have proposed a technique to increase the accuracy of the original seq2seq architecture. The original model shows low performance

even with adding "attention". The performance of the architecture is increased by data set pruning. The accuracy is increased and the overall performance in the model produces more natural responses when used in chatbot systems.

References

1. Anselmo López, Josep Sànchez-Ferreres, Josep Carmona, and Lluís Padró. From process models to chatbots. In *International Conference on Advanced Information Systems Engineering*, pages 383–398. Springer, 2019.
2. Du S, Li T., Yang Y., and Horng S. Multivariate time series forecasting via attention-based encoder–decoder framework. *Neurocomputing*, 388:269–279, 2020.
3. O. Hosam. An earthquake query system based on hidden markov models. *International Journal of Embedded Systems*, 15(2):149–157, 2022.
4. G. Xuan, W. Zhang, and P. Chai. Em algorithms of gaussian mixture model and hidden markov model. *Proceedings 2001 international conference on image processing (Cat*, 1(01CH37205) (Vol. 1):145–148, 2001.
5. N. Jaitly, P. Nguyen, A. Senior, and V. & Vanhoucke. Application of pre-trained deep neural networks to large vocabulary speech recognition, 2012.
6. K.and Van Merriënboer B. Cho, Gulcehre C., Bahdanau D., Bougares F., Schwenk H., and Y Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine, 2014. translation. arXiv preprint arXiv:1406.1078.
7. J. Li, M. Galley, C. Brockett, J. Gao, and B Dolan. A diversity-promoting objective function for neural conversation models, 2015. arXiv preprint arXiv:1510.03055.
8. F.J. Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st annual meeting of the Association for Computational Linguistics*, page 160–167, 2003.
9. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, page 27. ANIPS, 2014.
10. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. arXiv preprint arXiv:1409.0473.
11. Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015. arXiv preprint arXiv:1508.04025.
12. Ananth Sankar. Sequence to sequence learning with encoder-decoder neural network models. *ODSC India*, 2019.
13. Graham NEUBIG. Neural machine translation and sequence-to-sequence models: A tutorial, 2017. arXiv preprint arXiv:1703.01619,.
14. Deren LEI. Implicit regularization of stochastic gradient descent in natural language processing: Observations and implications, 2018. arXiv preprint arXiv:1811.00659,.
15. Bhargav SRINIVASA-DESIKAN. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.
16. SQuAD. Stanford question answering dataset (squad.
17. Steven BIRD. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, page 69–72, 2006.