



6th Assignment

Chat-Bot creation from Metalwoz Dataset

Machine Learning and Natural Language Processing

Christakakis Panagiotis

ID: aid23004



Postgraduate Studies Program in “Artificial Intelligence and Data Analytics”,
Of Macedonia

University

Date: 02/02/2023

Chat-Bot creation from Metalwoz Dataset

Table of Contents

1	Introduction	3
2	Methods	4
3	Results	7
4	Conclusion	9
5	References	9

Table of Figures

Figure 2.1: Architecture of the encoder – decoder model	5
Figure 2.2: Training and validation loss values for 50 epochs	6
Figure 2.3: Training and validation accuracy values for 50 epochs	6

1 Introduction

The goal of this assignment is to create a chat-bot given [MetaLWOz](#) dataset from Microsoft. Meta-Learning Wizard-of-Oz is a dataset designed to help develop models capable of predicting user responses in unseen domains. It can improve dialog systems, such as those used in voice assistants, to help users accomplish tasks such as booking a flight. This dataset is particularly suited for meta-learning dialog models or fine-tuning models with transfer-learning approaches. It was created by crowdsourcing 37,884 goal-oriented dialogs, covering 227 tasks in 47 domains. Some included domains are:

- | | |
|---------------------------|-----------------------------|
| 1) Sentence bus schedules | 5) Event reservation |
| 2) Apartment search | 6) Weather check |
| 3) Alarm setting | 7) Store details |
| 4) Banking | 8) Pet advice and many more |

To create the appropriate neural network model, it is necessary to pre-process the training data. Then the appropriate structure must be found to properly model the problem. After the training is done, the chat-bot will be tested with some discussions to evaluate the efficiency of the selected network.

For implementing our methods we'll use python and the appropriate libraries. The experiments run on CPU Intel Core i9-10900K, RAM 128GB DDR4, GPU RTX3090 as well as Google Colab with GPU runtime.

2 Methods

First of all, extraction of the required data from the json files is needed for the pre-processing steps. Transforming the raw data into useful information will make sure the inputs are appropriate for the neural network we will construct. The json files contained in the dataset have lots of extra information like `user_id`, `bot_id`, `task_id` and other unique characteristics of each conversation that are unnecessary for the model.

After creating a list of all the conversations, a splitting takes place in order to separate questions and answers. Those separated lists will be shuffled to ensure that the bot won't overfit on small dialogues like setting an alarm. For this experiment a subset of the dialogs will be kept, based on the length of the question to avoid RAM errors. The final lists contain dialogues only from questions that are created by 13 words and lower. Another part of the pre-processing step is the conversion of the conversations into lowercase letters and many contraction words into their corresponding ones (i.e. `won't` → `will not`). Then, counting occurrences of each word takes place and again a subset of all the words (those who have more than 5 counter) are kept for simpler training.

A vocabulary is created and for each answer two tokens are imported, one in front and after the string sentence. Also, a new inverted vocabulary is created which contains key – value items, where key is the word and value is the counted occurrences of the key. The lists of questions and answers now create the encoder – decoder inputs, where the first few integers of the row represent a sequence from the dictionary (that contain the position of each word), and the rest of the row is padded with the maximum length we defined earlier in order to have the same size. Finally, the decoder output needs to be one-hot encoded, rather than sequences of integers like the decoder input.

After the pre-processing step, model creation takes place. Our model will comprise of two LSTMs, with one acting as an encoder that encodes the input sequence and generates internal state vectors to condition the decoder. The decoder LSTM, on the other hand, predicts the target sequence's individual characters and its initial state is set to the state vectors generated by the encoder. The LSTM must be designed to return full output sequences and the internal state vectors during training. The latter will not be used during training, but they will be needed during the inference stage with the user.

The decoder's output will be converted into individual characters by adding a Dense layer to the decoder's LSTM output. The number of units in the Dense layer should match the number of decoder tokens. The Dense layer's outputs will be activated using a softmax activation, allowing the entire model to be trained using categorical cross-entropy loss - a common choice for solving these types of problems. Lastly, the two LSTM layers were chosen to have size of 400 respectively.

For the training session many parameters should always be considered. The best results came from combinations, but Adam as an optimizer and learning rate set to 0.001 always gave better results. Lastly, batch size was set to 128, epochs were set to 50.

It's important to mention that before the encoder, the user input is fed into an embedding layer, in order to represent the words into numerical representations. Embedding layers can capture the meaning and relationships of the words they represent. In an encoder-decoder system, the encoder takes a sequence of words, represented as embeddings, and converts it into a fixed-length vector called a context vector. This context vector represents the meaning of the input sequence. The decoder then takes this vector and produces the output sequence, also represented as embeddings. During training, the encoder-decoder model learns to associate input sequences with their corresponding output sequences. At prediction time, the input sequence is transformed into a context vector by the encoder, which the decoder uses to generate the output. This combination of embeddings and the encoder-decoder architecture is crucial for many natural language processing tasks.

Figure 2.1: Architecture of the encoder – decoder model.

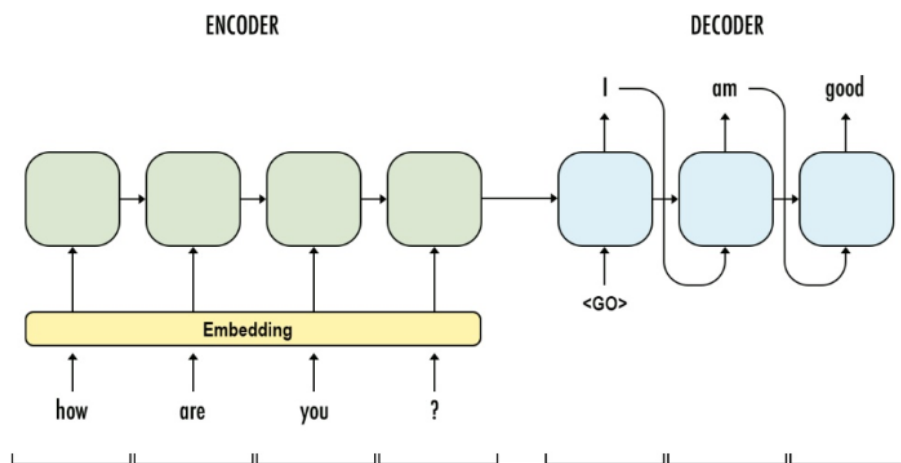


Figure 2.1: This image represents the architecture of the model created. The sequence ‘how are you?’ is the input of the user that is fed into the embedding layer. The created context vector is fed into the LSTM encoder input and finally each new word is calculated from the decoder.

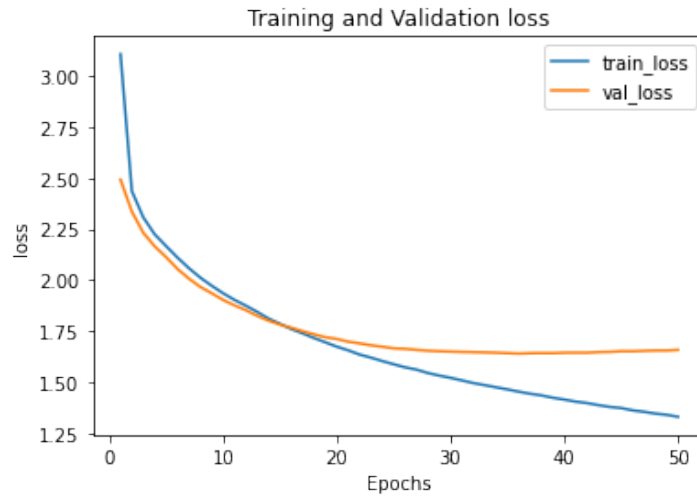
Figure 2.2: Training and validation loss values for 50 epochs.

Figure 2.2: In this figure we can see the comparison of loss values for the training and validation data. Both indicators have a normal downward trend and especially after the 15th epoch they are almost identical. Reaching the 19th epoch and then the validation loss starts to increase and surpasses the training loss.

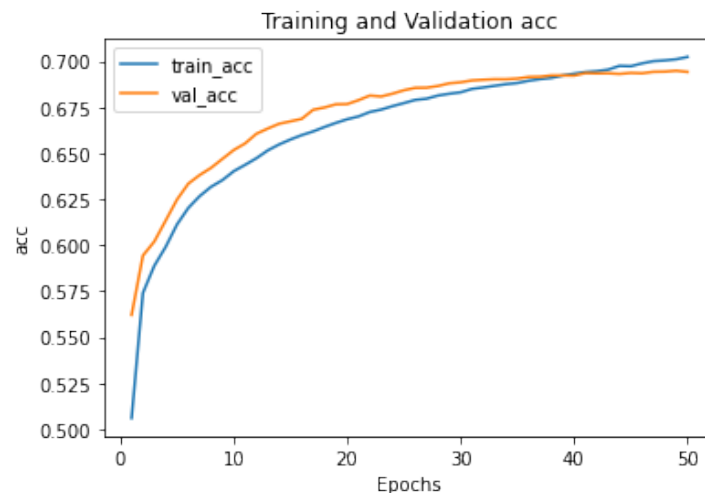
Figure 2.3: Training and validation accuracy values for 50 epochs.

Figure 2.3: The above figure shows the accuracy values for the training and validation data. We can see the upward trend for both indicators, which shows that the model is performing as we want. The fact that the accuracy does not reach higher than 70% means that we should consider different architecture or that the dataset we have is difficult to train and predict. Analyzing the figure further, we see that in the last training periods the train accuracy begins to surpass the validation accuracy. Here, the problem of overfitting begins. Our model becomes more specialized to the training data and becomes unable to generalize to new data.

Last but not least, when a new user message is received, the chatbot will calculate the similarity between the new text sequence and training data. Considering the confidence scores got for each category, it categorizes the user message to an intent with the highest confidence score.

3 Results

After 50 training epochs the models chatting results are the following:

Conversation 1:

(1)

User:	Hey
Bot:	hello how can i help you
User:	Help with my account
Bot:	what is it
User:	Transfer money
Bot:	okay i have set your alarm for you
User:	Ok thanks.
Bot:	you are welcome
User:	Can you tell the weather?
Bot:	yes
User:	For tomorrow
Bot:	okay i have set your alarm for you
User:	How will be the weather tomorrow
Bot:	i am sorry but there are no seats available for tomorrow
User:	Can you give me some info
Bot:	yes

(2)

User:	About appointment
Bot:	what is the number
User:	911
Bot:	okay i have set your alarm for you
User:	What time
Bot:	7
User:	I want 9
Bot:	okay i have set your alarm for you
User:	nice thanks
Bot:	you are welcome
User:	Can you tell me where a country is?
Bot:	what is it
User:	Greece
Bot:	<OUT>
User:	You know anything about cars?
Bot:	what is your question

(3)

User:	About cars
Bot:	i have found a few <OUT> on your location
User:	What else you know?
Bot:	Anything
User:	Where is Greece?
Bot:	<OUT>
User:	I dont want that
Bot:	i am sorry but there are no seats available for the
User:	ok
Bot:	anything else
User:	No
Bot:	okay
User:	Bye
Bot:	bye
User:	q
Bot:	okay i have set an alarm for you

Conversation 2:

(1)

User:	Hey
Bot:	hello
User:	Are you a bot?
Bot:	yes
User:	Can you help me
Bot:	what do you need help with
User:	Help with my cat
Bot:	what do you need help with
User:	Weather info?
Bot:	what is the number
User:	Can you give me forecast information?
Bot:	what do you need help with
User:	Use location
Bot:	what is the number
User:	No
Bot:	ok
User:	Set an alarm
Bot:	what time
User:	No temperature
Bot:	At 8 AM
User:	I need help with something
Bot:	what time
User:	8
Bot:	what is the number

(2)

User:	No
Bot:	ok
User:	I need help with something
Bot:	what do you need help with
User:	I need help with buying
Bot:	what do you need help with
User:	I need to buy
Bot:	what is the number
User:	I need to buy a computer
Bot:	what is the new email address
User:	Shopping help?
Bot:	what is the number
User:	No I need to buy something
Bot:	okay i will be here if you need me to help
User:	Ask me what
Bot:	what is the number
User:	Nice
Bot:	anything else
User:	What can you do?
Bot:	<OUT> <OUT>
User:	Balance of my account
Bot:	what is the number
User:	001
Bot:	what is the number

(3)

User:	Forget it
Bot:	okay i have made the change
User:	Can you order a book
Bot:	what do you need help with
User:	Is this book available?
Bot:	yes
User:	Price?
Bot:	i am not legally obligated to help you
User:	What's the price?
Bot:	what is the address
User:	Ok that's enough
Bot:	anything else
User:	No
Bot:	ok
User:	Thanks for the info
Bot:	Is there anything else i can help you with
User:	No
Bot:	ok
User:	Bye bye
Bot:	bye
User:	q
Bot:	what is the number

4 Conclusion

In general, the chat-bot behaved as expected without having exciting results. This was expected because we had limitations in RAM and the total training time of the model. Still from the results it was seen that the model didn't overcome the problem of overfitting, especially for discussions that existed arbitrarily in the training data. Some examples are the chatbot's constant references to setting an alarm, in questions like what's the number and what do you need help with.

One way to overcome this is to train separate models for each of the 47 domains or at least for the similar ones. This could be done by creating different vocabularies for each domain and prompting the user to select a type of conversation he wishes to have. This way, the encoder – decoder model would be able to obtain and process only data from the selected vocabulary.

Finally, the results might be quite different if we didn't delete all the questions that were over 13 words. This would lead to a new bigger vocabulary and the training step would become more time-consuming and source-demanding.

5 References

- 1) Metalwoz. Microsoft Research. (2019, December 12). Retrieved January 20, 2023, from <https://www.microsoft.com/en-us/research/project/metalwoz/>
- 2) Chaoran, (2019, January 15). Build a machine translator using Keras (part-1) seq2seq with LSTM. Chaoran's Data Story. Retrieved February 1, 2023, from <https://6chaoran.wordpress.com/2019/01/15/build-a-machine-translator-using-keras-part-1-seq2seq-with-lstm/>
- 3) Eloukas. (n.d.). *Seq2seq-chatbot/char-level-chatbot.ipynb*. GitHub. Retrieved January 19, 2023, from <https://github.com/eloukas/seq2seq-chatbot/blob/master/different-versions/version-keras-char-level/char-level-chatbot.ipynb>
- 4) Singh, P. (2020, July 2). *Seq2Seq chatbot with Keras / Encoder decoder LSTM chatbot* YouTube. Retrieved January 23, 2023, from <https://www.youtube.com/watch?v=TRYIrUmfhks>

- 5) Chollet, F. (n.d.). The keras blog. The Keras Blog ATOM. Retrieved January 23, 2023, from <https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html>
- 6) Viraj, A. (2020, October 31). How to build your own chatbot using Deep Learning. Medium. Retrieved February 4, 2023, from <https://towardsdatascience.com/how-to-build-your-own-chatbot-using-deep-learning-bb41f970e281>
- 7) Schneider, N. (2020, June 19). How to design Seq2Seq Chatbot using Keras Framework. Medium. Retrieved February 4, 2023, from <https://medium.com/swlh/how-to-design-seq2seq-chatbot-using-keras-framework-ae86d950e91d>
- 8) Khaled Hoshme, (2022, July 6). Simple-chatbot/chatbot_using_rnn_with_embedding_transfer_learning. GitHub. Retrieved February 1, 2023, from [https://github.com/kaledhoshme123/Simple-Chatbot/blob/main/Chatbot_using_RNN_with_Embedding_Transfer_Learning%20\(1\).ipynb](https://github.com/kaledhoshme123/Simple-Chatbot/blob/main/Chatbot_using_RNN_with_Embedding_Transfer_Learning%20(1).ipynb)
- 9) Saurabh Prajapat. (2022, August 13). Conversational chatbot using encoder and decoder. Kaggle. Retrieved January 30, 2023, from <https://www.kaggle.com/code/saurabhprajapat/conversational-chatbot-using-encoder-and-decoder/notebook>
- 10) muki2003. (2022, May 28). Q/a chatbot with tensorflow keras. Kaggle. Retrieved January 31, 2023, from <https://www.kaggle.com/code/muki2003/q-a-chatbot-with-tensorflow-keras>