AI Chatbot using a Seq2Seq model

Documentation of the VITMAV45 course homework

Team: Conloquor
Team members:

- Béres Bálint (DTY6T7)
- Drexler Kristóf (I6I283)
- Drexler Konrád (J5SZVX)

Abstract

We are developing a Chatbot using Recurrent Neural Network techniques. The data source of the project is a database of over 1 million reddit comments. We have reordered this to a query and response format. The model is a Seq2Seq architecture-based encoder-decoder network. The project pipeline consists of data pre-processing, model development, hyperparameter optimization, final training, and model evaluation.

1 Introduction

Our project is to build a chatbot which can communicate with the user via natural language. To accomplish this, we are using a Seq2Seq encoder-decoder network. Our data source consists of over 1 million reddit comments which we ordered into query and response pairs. The task of the model is to generate an understandable answer to a given input sentence. First, we introduce the model architecture used for the implementation. After that we give a detailed description of the implementation and optimization of the model. Finally, we evaluate the model by generating answers to sentences from the dataset and to arbitrary input sentences.

2 Model Architecture

We chose a sequence-to-sequence model which consists of two Recurrent Neural Networks, an encoder, and a decoder [2]. Figure 1 shows an example of this architecture.

The encoder takes an input sentence and processes it one word at a time. Its role is to convert the sentence to a fixed sized feature vector which contains all the important information about the sequence. Every hidden state in the encoder influences the hidden state of the next module. The final hidden state of the encoder is the so-called context vector which represents the information content of the whole sequence.

The decoder receives the context vector generated by the encoder as its initial hidden state. Using this vector and its input sequence the decoder generates the output sequence. Whilst generating the decoder moves word by word.

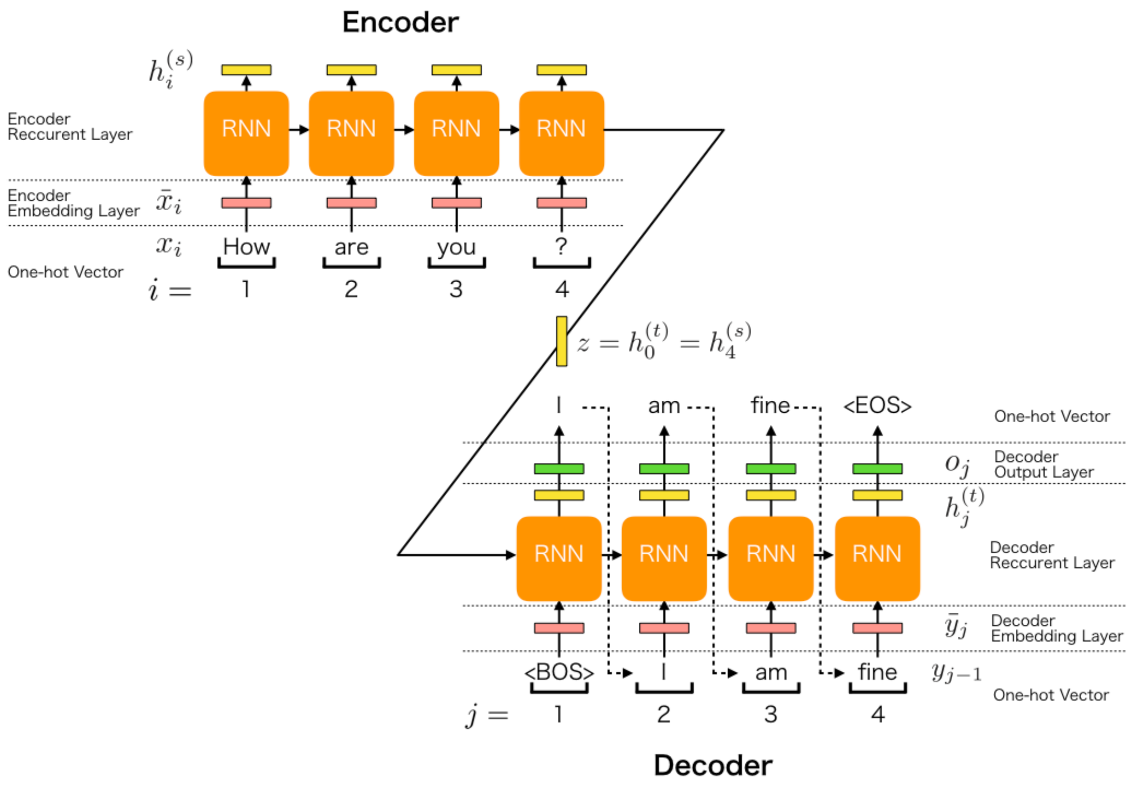


Figure 1 - Sequence to Sequence model

3 Implementation

3.1 Data Preparation

[11] The initial dataset contained many unnecessary properties which we had to remove and reorder the whole dataset to pass it for pre-processing. Firstly, the dataset the source points to is too large for our resources and application therefore we removed a significant part of it. Of the whole reddit dataset we removed everything except comments from the following subreddits: r/science**,** r/politics**,** r/gaming**,** r/worldnews**,** r/CasualConversation and r/sports. After that we removed every non-natural language component such as links, subreddit links, symbols etc. from the comments. Normally a record contains the text with the id of the comment and the parent id of the comment. After the reduction, a new dataset is created where we paired comments with their parent comments, which leaves a structure similar to a question and response format. Finally, we concatenated end-of-sentence and start-of-sentence tags to the responses separately which will be required for the Seq2Seq model. We saved this dataset to a JSON file, because of this, working with the larger dataset can be skipped.

3.2 Data pre-processing

After preparation, the dataset is tokenized. The tokenizer is fitted separately on the input and output sentences. After the tokens are ready, the sentences are converted to sequences. The output sentences will be converted to two separate sequence sets. One will contain the start-of-sentence tag at the front, the other the end-of-sentence tag at the end of the sequence. After the tokenization, the sequence sets are padded, so that all the sequences will all be of uniform length. These tokenized sequences can be directly fed to the encoder and decoder modules.

3.3 Model description and training

The model consists of two components: the encoder and the decoder shown by Figures 2 and 3. Both modules start with an embedding layer. By using word embeddings every word will be represented with an n-dimensional vector. We use the GloVe embedding technique for our model. After the embedding layer, the encoder uses Long Short-term Memory RNN layers. The output of the encoder is the hidden state and cell state of its final embedding layer.

The decoder has three Input layers. One of the inputs injects the response sequences which contain the start-of-sentence tags. The other two are for the hidden and cell states LSTM layer. These are from the output of the encoder layer.

After the initial training we ran a hyperparameter optimization on the model which took almost 5 hours. Since the Google Colab only gave us 12 GBs of RAM to work with, we had to implement a generator that was used to train the model instead of saving the whole one hot encoded matrix in memory. This generator generated the training and the validation datasets. Creating the generator was no easy task and took many attempts, but the resulting model could be trained with validation accuracy in mind.

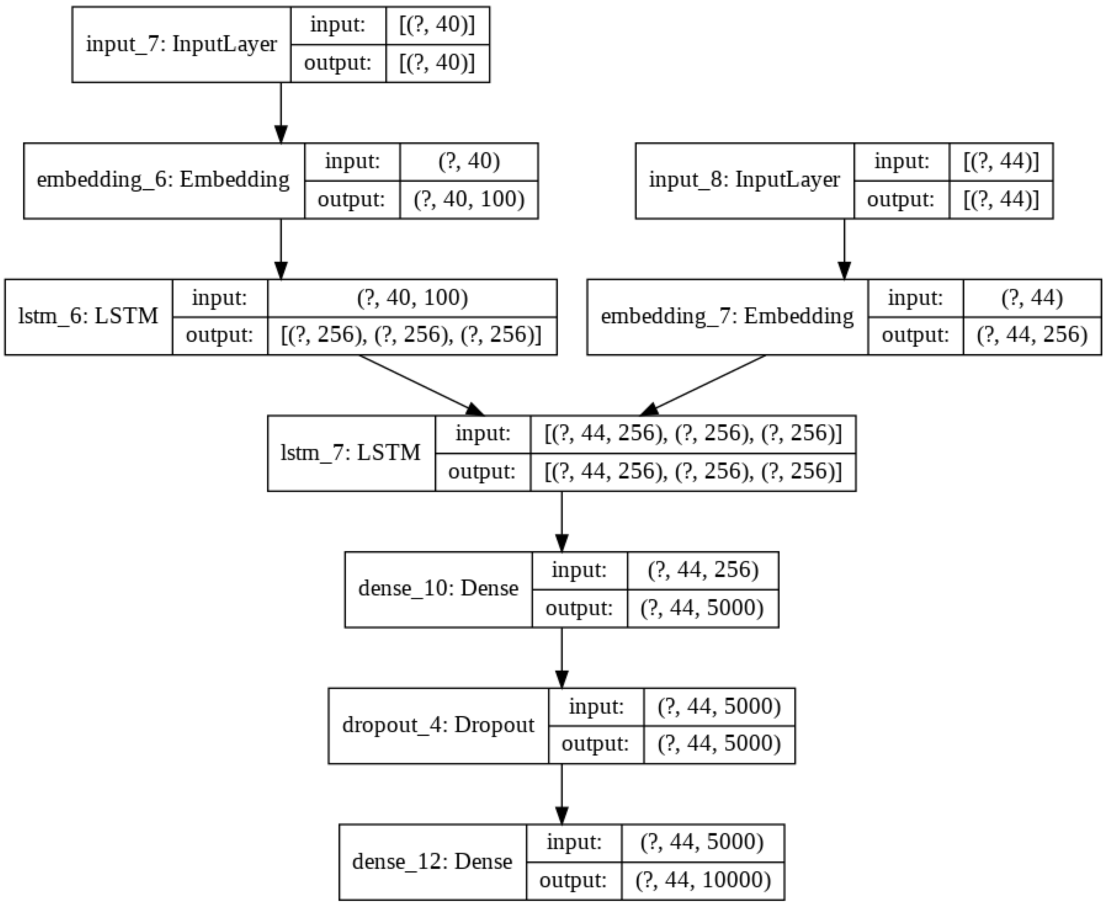


Figure 2 - Structure of the training model

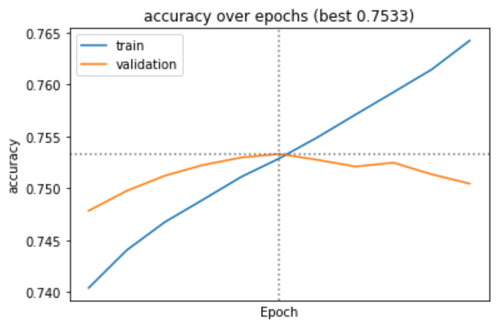


Figure 3 - Accuracy progression during training

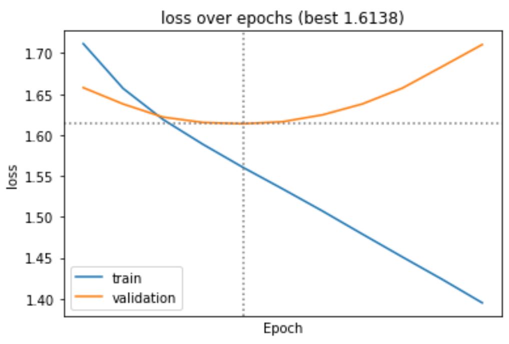


Figure 4 - Loss progression during training

After the initial training we ran a hyperparameter optimization on the model which took almost 5 hours. Since the Google Colab only gave us 12 GBs of RAM to work with, we had to implement a generator that was used to train the model instead of saving the whole one hot encoded matrix in memory. This generator generated the training and the validation datasets. Creating the generator was no easy task and took many attempts, but the resulting model could be trained with validation accuracy in mind. After training, we created the structure of the encoder and decoder models illustrated in the figures below.

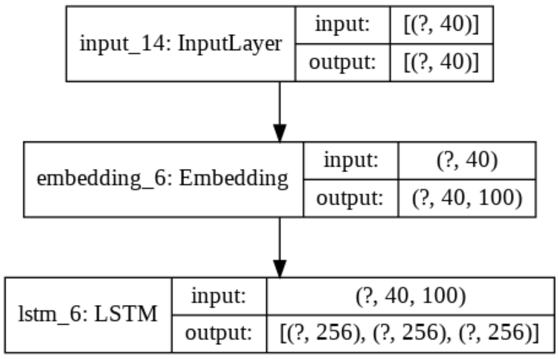


Figure 5 – Structure of the encoder module

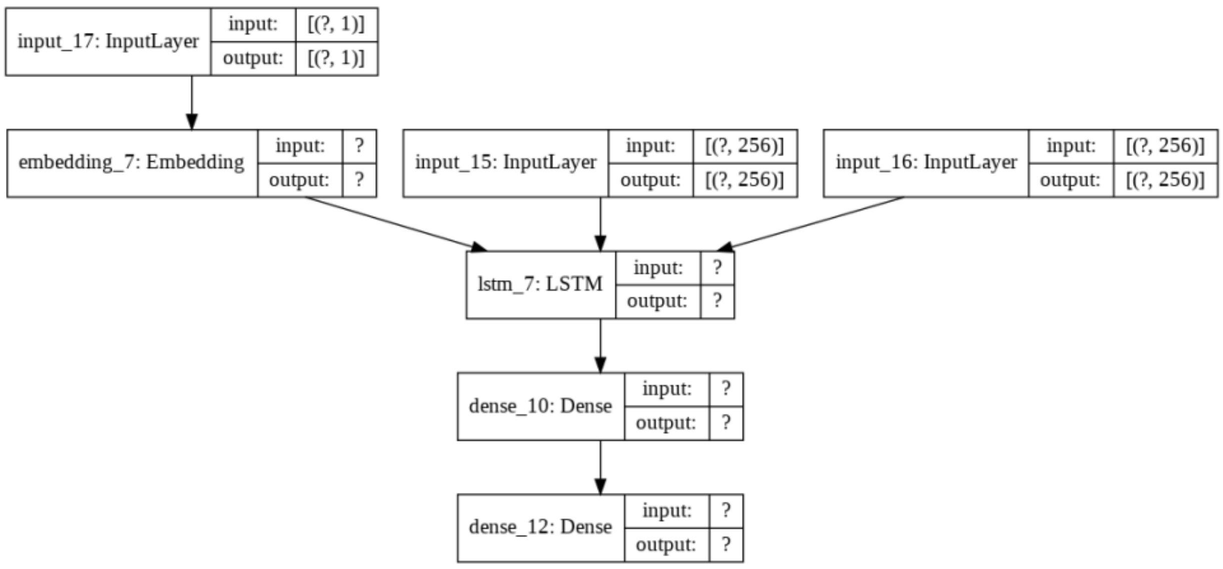
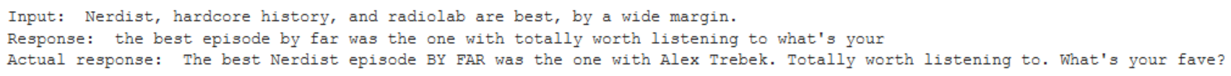


Figure 6 – Structure of the decoder module

3.4 Model evaluation

To evaluate the model, sentences need to be generated as a response for a given input. The evaluation starts with an input sentence. This is fed to the encoder which generates hidden and cell states for the input. The decoder takes three inputs: the answer to the encoder input sentence and the hidden and cell states predicted by the encoder. The decoder then generates the output sequence word by word. In the first iteration the encoder output and the first word of the decoder input is added to the decoder. After that in every iteration we give the decoder a single word and the states predicted in the last iteration to produce the next word. The cycle continues until the end-of-sentence tag is generated by the decoder.



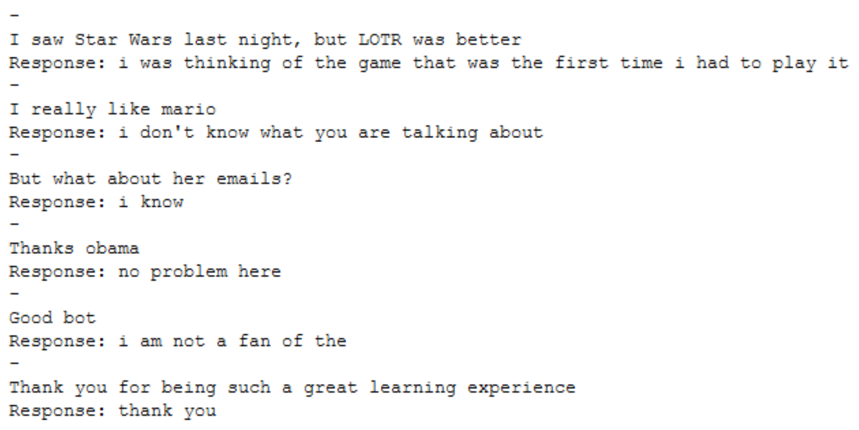
```
Input:  Nerdist, hardcore history, and radiolab are best, by a wide margin.
Response:  the best episode by far was the one with totally worth listening to what's your
Actual response:  The best Nerdist episode BY FAR was the one with Alex Trebek. Totally worth listening to. What's your fave?
```

Figure 7 - Testing the model with an input from the dataset, with the actual response in the last line.

3.5 Testing

We tested the resulting model in two ways: Firstly, we compared the results from input sentences with the actual response to those sentences and determined if those were acceptable. Secondly, we then tried a few manual inputs, to see if the model holds up when answering to sentences not in the dataset which it was trained on.

Testing this is subjective as we can try and draw conclusions from this. Most of the answers will make vague contextual sense, as they can be believable and the model can draw parallels between different words, for instance it replied with the word "game" when the input message was about Star Wars, an argument can be made about how those two are connected.



```
-
I saw Star Wars last night, but LOTR was better
Response: i was thinking of the game that was the first time i had to play it
-
I really like mario
Response: i don't know what you are talking about
-
But what about her emails?
Response: i know
-
Thanks obama
Response: no problem here
-
Good bot
Response: i am not a fan of the
-
Thank you for being such a great learning experience
Response: thank you
```

Figure 8 - The model's responses to inputs given by the user.

4 Conclusions

We are happy of the initial results of our model as the model clearly learned how to structure sentences. Some answers are missing words at the end of the sentences, since the model lacks the vocabulary to fill it, as it was trained on only 10000 words instead of the more than 30000 unique words that were present in the raw dataset.

An obvious step forward is to train the model on all available words in the dataset, thereby filling the gaps in the current model's vocabulary. Another obvious step is to increase the raw data we train the model on (the number of comments), but since the comments are from an uncorrected (borderline degenerate) site, this might not help the model output better sentences.

References

[1] Deep Learning Based Chatbot Models, Richard Csaky, 2019
https://arxiv.org/abs/1908.08835

[2] Seq2Seq AI Chatbot with Attention Mechanism, Abonia Sojasingarayar, 2020
https://arxiv.org/abs/2006.02767

[3] A Deep Learning Based Chatbot for Campus Psychological Therapy, Junjie Yin, Zixun Chen, Kelai Zhou, Chongyuoan Yu, 2019
https://arxiv.org/abs/1910.06707

[4] A Deep Reinforcement Learning Chatbot, Iulian V. Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Rajeshwar, Alexandre de Brebisson, Jose M. R. Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, Yoshua Bengio, 2017
https://arxiv.org/abs/1709.02349

[5] An intelligent Chatbot using deep learning with Bidirectional RNN and attention model, Manyu Dhyani, Rajiv Kumar, 2020
https://www.sciencedirect.com/science/article/pii/S221478532034030X

[6] Developing a Chatbot, Palash Goyal, Sumit Pandeyy, Karan Jain, 2018
https://link.springer.com/chapter/10.1007/978-1-4842-3685-7_4

[7] Recent advances in conversational NLP : Towards the standardization of Chatbot building, Maali Mnasri, 2019
https://arxiv.org/abs/1903.09025

[8] A Financial Service Chatbot based on Deep Bidirectional Transformers, Shin Yu, Yuxin Chen, Hussain Zaidi, 2020
https://arxiv.org/abs/2003.04987

[9] Compressing Large-Scale Transformer-Based Models: A Case Study on BERT, Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Deming Chen, Marianne Winslett, Hassan Sajjad, Preslav Nakov, 2020
https://arxiv.org/abs/2002.11985

[10] ConveRT: Efficient and Accurate Conversational Representations from Transformers, Matthew Henderson, Iñigo Casanueva, Nikola Mrkšić, Pei-Hao Su, Tsung-Hsien Wen, Ivan Vulić, 2020
https://arxiv.org/abs/1911.03688

[11] Datasource
https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/

[12] Python for NLP: Neural Machine Translation with Seq2Seq in Keras, Usman Malik, 2019
https://stackabuse.com/python-for-nlp-neural-machine-translation-with-seq2seq-in-keras/