Sequence-to-Sequence Chatbot

John Mrziglod
john.mrziglod@mail.de

Danu Caus
7caus@informatik.uni-hamburg.de

Sebastian Lembcke
7lembcke@informatik.uni-hamburg.de

Knowledge Processing in Intelligent Systems: Practical Seminar
Knowledge Technology, WTM, Department of Informatics, University of Hamburg

***Abstract*—In this paper, we reviewed different types of dialog systems (chatbots) that may simulate intelligent behaviour in conversation with humans. We implemented our own version based on a sequence-to-sequence learning model provided by Tensorflow [1]. Training the model with different hyperparameters and vocabulary sizes altered the model's performance significantly. Although it was not able to hold human like conversations, it could give meaningful responses.**

I. Introduction

The Turing test, one of the most famous settings for testing a machine's intelligence behaviour [14], describes a scenario where a human has a conversation with another human and a machine through a text chat program in natural language. The machine passes this test if the first human person is not able to tell who of his communication partners is human and who is not. The reception and discussion of the Turing test shows that our understanding of intelligence expects an intelligent being to have the ability of holding human-like conversations or at least imitating it [12]. Since its development in 1950, the Turing test challenges AI researcher to develop machines to pass it [16], [10], [12]. There are different approaches to develop such machines also called chatbots. Former versions use handcrafted patterns and rules to extract the features of the natural language inputs by the user. One popular chatbot that works in that way is ELIZA which reflects the user's inputs by producing new questions from them. Therefore, it was also developed for psychological tests [16]. The development of powerful recurrent neural network architectures enabled approaches that use them to learn the rules of natural language from a big data corpus of dialogues. Further research led to sequence-to-sequence models which work as end-to-end dialogue systems [9]. The sequence-to-sequence models learn to project certain word sequences in the past to new word sequences in the future. If the model uses parts from human conversations as sequences it may be able to predict the continuation of the conversation as well. That means it might be able to answer questions that might be given in earlier sequences or to understand context in general. That is an important key feature of a chatbot that shall imitate human conversation. In this paper, we implement our own version based on a sequence-to-sequence model provided by Tensorflow [1]. Owing to various reasons, which will be discussed later, it does not perform very well. This paper is structured as following: in section 2, we review different types of chatbots that may simulate intelligence behaviour in conversation with humans and explain the architecture of Recurrent Neural Networks and sequence-to-sequence models. In section 3, we present details of our implementation and the used data corpus. In section 4 and 5, we show and discuss our results and further improvements of our work.

II. Background

A. *Types and Architectures of Dialog Systems*

According to [6] the types of dialog systems can be roughly separated into two groups: task-oriented dialog agents and conversational chatbots. The task-oriented dialog agents are built to achieve a domain-specific goal (e.g. booking a table in a restaurant) and are rather system-initiative or mixed-initiative systems. I.e. the dialog agent provides a fixed number of questions which the user can answer either by using natural language or by selecting from a list of responses. In contrast, conversational chatbots are mainly designed to simulate a human communication partner and are therefore user-initiative.

A dialog bases on turns - a turn may be an utterance in form of a word, sentence or even a paragraph [6]. Every time the speaker changes during a conversation, a new turn is enabled. Limited dialog agents can mostly deal with a single turn only and therefore struggle to understand the context of more complex conversations. Multi-turn dialog agents may understand the context of a conversation better.

Earlier implementations of chatbots (such as ELIZA [16]) work with handcrafted rules and pattern. They used them to analyze the syntax and content of a turn and to generate a response to it. This often takes a lot of effort for creating adequate rules and requires advanced natural language modelling and understanding. Newer versions of chatbots (e.g. Cleverbot [3]) are mostly corpus-based. Instead of using handcrafted rules they retrieve them from a conversation data corpus such as the Cornell Movie Database by using machine learning or other statistical methods. The sequence-to-sequence learning models which are based on Recurrent Neural Networks are one of them.

B. Recurrent Neural Networks

Usual artificial feed-forward networks are not capable of remembering past states nor to process sequences properly. Recurrent Neural Networks (RNN) were introduced to overcome those problems are nowadays a powerful tool to build sequence learning models. In contrast to usual feed-forward networks, some of the neural layers of a RNN are connected to loops in order to reprocess data from the past. Simple RNN architectures (such as in [7]) just copy their current input or hidden state and add them as additional input for their hidden neural layer in the next time step. The connections between the context layer (the copy from the past) and the hidden layer can be trained by adjusting weights. Although such an architecture enables to remember states from the past, it has its shortcomings. Since it is usually trained with a backpropagation algorithm, it must be unfolded in time. This leads to the vanishing gradient problem. Hence, simple RNNs have problems with long-term time dependencies; they simply forget states from ten or more time steps in the past. To solve this problem, other versions of RNNs were developed: one of them is the Long Short-Term Memory or LSTM [8]. A LSTM is a unit block that consists of three gates (input, output and forget) and a cell memory. It is explicitly designed to have a long-term memory [8]. Figure shows its structure.

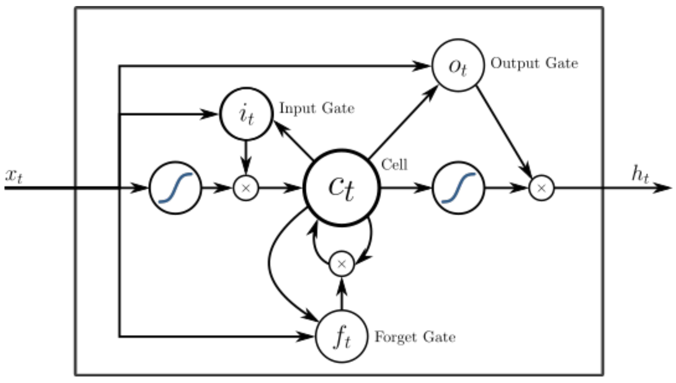

Figure 1. A LSTM unit block, taken from [2].

C. Sequence-to-Sequence Models

With those powerful LSTM architectures one can build so-called sequence-to-sequence models that are able to generate new sequences from other sequences both having flexible lengths. Figure 2 shows such an architecture as an example of question answering. The input sequence will be chunked into words and then given as an input sequence for a LSTM unit called encoder (green in the figure). Those chunks can also be characters or syllables but the content of them must be known by the model in advance. They are normally stored in a big vocabulary dictionary. The sequence of words will be processed until a End-Of-Line (EOL) signal appears. The output of the encoder unit is then a fixed-size vector W which represents the input sequence. A second LSTM unit which is called decoder (yellow in the figure) takes the vector W as input and generates a new sequence. In this example, the new sequence is a response to the question in the input. The encoder and decoder are jointly trained with pairs of a question and answers [4].

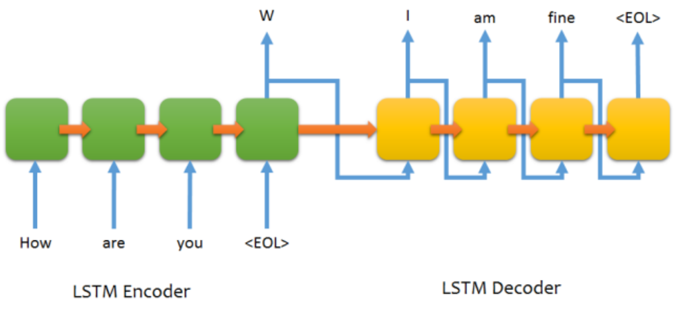

Figure 2. Architecture of a Sequence-to-Sequence model, taken from [11].

Those sequence-to-sequence models were very successful in the last year solving tasks such as machine translation, speech recognition and tasks of natural language processing (e.g. simulating human conversations) [13], [4], [15].

III. Implementation & Used Data

The data used in the experiment is taken from the **Cornell Movie Dialogs Corpus/Dataset** [5]. It was created by Cristian Danescu-Niculescu-Mizil and Lillian Lee at Cornell University. The corpus consists of movie lines taken from 617 movies. There are 220579 lines between 9035 characters. During the conversations, a total of 10292 distinct pairs of movie characters participate. The total number of utterances is: 304713. The data is pre-processed in order to remove html tags and meaningless tags in general from the subtitles. Afterwards, the pre-processing script splits it in two files, where the first one contains an utterance from a character participating in the conversation and the other contains the associated replicas from the other character. These files are fed into the translate engine in order to train the chatbot. The python script **translate.py** is the standard **Tensorflow** main script used for the language translation [1]. In this case, the translation is actually a mapping between the lines in the first and second data files correspondingly, or simply put: it's like translating from english to english to mimic a conversation between two parties. All the hyperparameters are set in the tensorflow script and the latter is run in a **tmux** linux session for typically 30 hours. Although tensorflow creates checkpoints along the training process, the **tmux** program is used to add an extra layer of security and overcome network disconnection problems by not relying on the internet connection at all.

IV. Experiments

In this section, the experimental setup and the model configurations are presented and the results of the performed tests are discussed. In total we conducted 8 experiments, each with different hyperparameters of the sequence-to-sequence model. These were the number of layers, the number of hidden units, the batch size and the vocabulary size. The global steps that were performed while training the model varied, as we aimed to achieve a low perplexity. The perplexity is a measurement of how well a probability distribution (the model) predicts a sample of the test set. It is calculated using a loss function. The training was done using GPUs to reduce the overall time. After training each setup, we performed a Turing-style test to evaluate how well the model performs. This method was applied since the aim of the chatbot is to have the ability to answer to one sentence questions, which would arise in human interaction.

For the first 3 experiments with a 40000 word vocabulary we used models with 2 and 3 hidden layers, both with 512 hidden units. The batch size was set to 16 and 32 samples for the 2 layer model and to 16 samples for the 3 layer model. Both models were trained for 72200 global steps with a batch size of 16 until a perplexity of approximately 4.5 was reached. The 2 layer model was trained with a batch size of 32 until a perplexity of 2.69 was reached after 62000 global steps. The training time was significantly lowered by increasing the batch size. However, this did not affect the performance of the chatbot. All 3 setups performed poorly. The answers were incomprehensible and not related to the input phrase. Moreover, words were repeated several times, as can be seen below.

```
>Hello
fisk fisk fisk fisk fisk traded traded traded traded traded
>I don't understand
particles particles particles particles ahmar ahmar ahmar
ahmar ahmar ahmar traded traded traded traded traded
>Hmmm what ?
grabbing fisk fisk fisk ahmar ahmar ahmar ahmar ahmar
ahmar
```

Figure 3. Chat-log (2 layers, 512 units, 16 samples)

In the following 3 experiments, we trained 2 layer models with 512, 800 and 1024 hidden units. The batch size was 16 for the model with 800 units and 64 for the other two. In the seventh experiment we used a model with 3 layers, 1024 hidden units and a batch size of 64. All models were trained until a perplexity of approximately 1 was reached. The performance was, however, not significantly different to the other configurations. Input and output were not related and words were repeated.

For the eighth experiment we lowered the vocabulary size to 20000 words, as we assumed that the model was underfitting. We set the number of layers to 3 and the number of hidden units to 256. The batch size was 64. This made a significant difference in performance. Words were not repeated and the output seemed to be related to the input phrase.

```
and give yourself a dime
>dime?
or less.
>why less?
to get out of sight.
>you are aggressive
not yet...
```

Figure 4. Chat-log (3 layers, 256 units, 64 samples)

As can be seen in the second dialog excerpt, the chatbot does not memorize the context of the conversation. It merely gives a response to the input, which is due to the sequence-to-sequence model being designed to map input to output phrases. Also, due to the movie dialog data set, responses seem unrealistic and dramatic.

V. Conclusion

In conclusion, one can say that the current version of the chatbot needs improvements to be able to lead a more realistic conversation. One thing to consider is to train it on more than one dataset. Some important sources to consider are:

- Twitter logs (https://github.com/Marsan-Ma/chat_corpus)
- Reddit comments (https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/)

These data sources will likely make the chatbot replies more realistic and not so dramatic because of being based solely on movie screenplays.

Next addition would be to give personality to the bot. In its current state the bot does not answer questions like: What is your name? or How old are you?. The data initial processing removed rare words amongst which are those that encode such kind of personal information. And so, they are tagged as unknowns. The solution would be to add/hard-code consistent personal information about the bot like: age, name, job, workplace, etc.

Currently, the encoder is a single utterance such as: a sentence or even less and the decoder is simply the response to this. This can be changed by considering previous utterances to come up with a response. In other words, a better model would be to use a summarization model and feed more than one utterance as the encoder input, making it longer than the decoder. It can be done by modifying the data processing itself and the bucket lengths.

At this stage, the response is constructed in a greedy manner. The bot will always give the same replica to the exact same question. However, it can be changed in such a way that the response is created with a non-greedy method, allowing the bot to vary its replies to exact same questions posed multiple times.

It would also be desirable to incorporate previous information from the conducted conversation. Currently the bot does not store/remember previous utterances. For example, if it is told the age of the counterpart and being asked right away how old is the dialog partner, it can't tell. Such information should be stored and incorporated in the reply. It can be done by saving previous discussions and extract relevant information to the current conversation.

Lastly, it would be nice if the users can somehow give their input as to how the bot might reply when they don't like the original reply: a feedback of some sort. When the user types some specific words like: “You should say…“, the bot should trust the user and actually reply with what the user said next time it is asked the same question. However, blind trust can be tricky and result to problems if the users are irresponsible…

VI. Bibliography

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Abdelrahman Mohamed Alex Graves and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 66456649. IEEE, 2013.

[3] R. Carpenter. Cleverbot. http://www.cleverbot.com, accessed 2018.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

[6] James H. Martin Daniel Jurafsky. Speech and language processing. Third Edition Draft, https://web.stanford.edu/ jurafsky/slp3/ed3book.pdf, 8 2017.

[7] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[9] Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65, 2017.

[10] Michael L Mauldin. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In *AAAI*, volume 94, pages 16–21, 1994.

[11] Fariz Rahman. Seq2seq model. https://github.com/farizrahman4u/seq2seq, accessed 2018.

[12] Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman. Turing test: 50 years later. *Minds and machines*, 10(4):463–518, 2000.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[14] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[15] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[16] Joseph Weizenbaum. Elizaa computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.