Potential references:

**Abdul-Kader, S. A., and Woods, J. 2015. “Survey on Chatbot Design Techniques in Speech Conversation Systems,” IJACSA) International Journal of Advanced Computer Science and Applications, (6:7)**

-The Natural Language ToolKit (NLTK) is a set of modules, tutorials and exercises which are open source and cover Natural Language Processing symbolically and statistically. NLTK is used to split words in a string of text and separate the text into parts of speech by tagging word labels according to their positions and functions in the sentence. The resulting tagged words are then processed to extract the meaning and produce a response as speech or action as required.

-The Chatbot is a computer programme that mimics intelligent conversation. The input to this programme is natural language text, and the application should give an answer that is the best intelligent response to the input sentence.

-Writing a perfect Chatbot is very difficult because it needs a very large database and must give reasonable answers to all interactions

-Designing a Chatbot software package requires the identification of the constituent parts. A Chatbot can be divided into three parts: Responder, Classifier and Graphmaster (as shown in Figure. 1) [11], which are described as follows:

1) Responder: it is the part that plays the interfacing role between the bot’s main routines and the user. The tasks of the responder are: transferring the data from the user to the Classifier and controlling the input and output.

2) Classifier: it is the part between the Responder and the Graphmaster. This layer’s functions are: filtering and normalising the input, segmenting the input entered by the user into logical components, transferring the normalised sentence into the Graphmaster, processing the output from the Graphmaster, and handling the instructions of the database syntax (e.g. AIML).

3) Graphmaster: is the part for pattern matching that does the following tasks: organising the brain’s contents, storage and holding the pattern matching algorithms.

-SQL A Relational Data Base (RDB) is one of the techniques recently used to build Chatbot knowledge bases. The technique has been used to build a database for a Chatbot, i.e. to enable the Chatbot to remember previous conversations and to make the conversation more continuous and meaningful.

**Petter Bae Brandtzaeg(&) and Asbjørn Følstad “Why People Use Chatbots”**

**-** There is a growing interest in chatbots, which are machine agents serving as natural language user interfaces for data and service providers. However, no studies have empirically investigated people’s motivations for using chatbots.

**-** an online questionnaire asked chatbot users (N = 146, aged 16–55 years) from the US to report their reasons for using chatbots. The study identifies key motivational factors driving chatbot use. The most frequently reported motivational factor is “productivity”;

**A Chatbot for Psychiatric Counseling in Mental Healthcare Service Based on Emotional Dialogue Analysis and Sentence Generation -** [**Kyo-Joong Oh**](https://ieeexplore.ieee.org/author/38064975000)

**;**[**Dongkun Lee**](https://ieeexplore.ieee.org/author/37086141818)**;**[**Byungsoo Ko**](https://ieeexplore.ieee.org/author/37086090190)**;**[**Ho-Jin Choi**](https://ieeexplore.ieee.org/author/37277426500)

**-** we adapt various natural language processing (NLP) methods to analyze consult contents

- the service respond appropriately according to user's new inputs using natural language generation (NLG) methods

- At other times, the service suggest useful information to promote mental health, and recommend items for feeling bet-ter. At this time, the conversational service consider clinical psychological and ethical aspects.

**A Tool of Conversation: Chatbot M. Dahiya 2017**

- Chatbot recognize the user input as well as by using pattern matching, access information to provide a predefined acknowledgment.

- A chatbot is one of the simple ways to transport data from a computer without having to think for proper keywords to look up in a search or browse several web pages to collect information; users can easily type their query in natural language and retrieve information.

**CHATBOT IN PYTHON Akshay Kumar1, Pankaj Kumar Meena2, Debiprasanna Panda3, Ms. Sangeetha4 2019**

- Chatbots are programs that work on Artificial Intelligence (AI) & Machine Learning Platform.

- A chatbot is merely a computer program that fundamentally simulates human conversations

- functions through AI and machine learning has an artificial neural network inspired by the neural nodes of the human brain

- According to research, nowadays chatbots are used to solve a number of business tasks across many industries like E-Commerce, Insurance, Banking, Healthcare, Finance, Legal, Telecom, Logistics, Retail, Auto, Leisure, Travel, Sports, Entertainment, Media and many others.

- AIML and LSA are used for creating chatbots. Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA) are used for developing chatbots, which are used to define general pattern-based queries

- Extensible Markup Language (XML) is the base for the derivation of Artificial Intelligence Markup Language (AIML). It has a class of data object called an AIML object that describes the behavior of computer programs. It consists of units or tag called topics and categories. There each category consists of a pattern that contains input and template which contain the answer of chatbot based on queries.

- AIML represents the knowledge put into Chatbots and is based on the software technology developed for A.L.I.C.E

- The pattern tag identifies the input from the user and the task of template tag is to respond to the specific user input, these are the most frequent tags and the bases to design AIML Chatbots with an intelligent response to natural language speech conversations

[**https://aisel.aisnet.org/amcis2018/HCI/Presentations/3/**](https://aisel.aisnet.org/amcis2018/HCI/Presentations/3/)

[**https://studio.carnegiemuseums.org/literature-review-chatbots-conversational-experiences-566de218f92a**](https://studio.carnegiemuseums.org/literature-review-chatbots-conversational-experiences-566de218f92a)

[**https://link.springer.com/chapter/10.1007/978-3-319-70284-1\_30**](https://link.springer.com/chapter/10.1007/978-3-319-70284-1_30)

[**https://www.frontiersin.org/articles/10.3389/fpsyg.2017.00796/full**](https://www.frontiersin.org/articles/10.3389/fpsyg.2017.00796/full)- mental health

<https://arxiv.org/abs/1709.02349>

<https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/ICA4088.pdf> - ELDERLY AID

<https://www.codementor.io/@garethdwyer/building-a-telegram-bot-using-python-part-1-goi5fncay> - building chatbot python telegram

<https://www.cambridge.org/core/services/aop-cambridge-core/content/view/0ACB73CB66134BFCA8C1D55D20BE6392/S1351324916000243a.pdf/return_of_the_chatbots.pdf>

- Return of chatbots 2016

- Whether you call these things digital assistants, conversational interfaces or just chatbots, the basic concept is the same: achieve some result by conversing with a machine in a dialogic fashion, using natural language.

- Big Four: Apple’s Siri, Microsoft’s Cortana, Amazon’s Alexa and Google’s new Assistant – voice assistants plus many many more text based assistants

- Many see this technology as heralding a revolution in how we interact with devices

- interaction with technology using either natural language text or speech is becoming increasingly feasible, and potentially very significant.

<http://tmaa.com/specializeddigitalassistantsandbots.html> - generate global revenues of $7.9 billion in 2016, rising to $623 billion by 2020.

<http://www.pandorabots/> - world’s leading chatbot platform, claims 225 thousand developers, 285 thousand chatbots created, and over three billion interactions. N

<http://www.wsj.com/articles/facebook-hopes-chatbots-can-solve-app-overload-1460930220> - Mark Zuckerberg proclaimed that chatbots were the solution to the problem of app overload

<https://www.forbes.com/sites/kathleenchaykowski/2016/07/01/more-than-11000-bots-are-now-on-facebook-messenger/#191762494fd7>

<https://www.theguardian.com/technology/2019/dec/22/zuckerbot-mark-zuckerberg-facebook-botnik?CMP=Share_iOSApp_Other> - zuckerbot

<http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462015000400625>

* Alice bot pattern matching

<https://cloudacademy.com/blog/google-cloud-natural-language-processing-api/> - google cloud api stuff

<https://ieeexplore.ieee.org/abstract/document/5533641> - cloud databases amazon rds

V. Mateljan, D. Čišić and D. Ogrizović, "Cloud Database-as-a-Service (DaaS) - ROI," The 33rd International Convention MIPRO, Opatija, 2010, pp. 1185-1188.  
URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5533641&isnumber=5533310>

<https://arxiv.org/pdf/1704.04579.pdf> - Evaluating Quality of Chatbots and Intelligent Conversational Agents

<https://ieeexplore.ieee.org/abstract/document/7877192> database

<https://www.reddit.com/r/MachineLearning/comments/3ukvc6/datasets_of_one_to_one_conversations/> - conversational dataset

<https://medium.com/@jesusmartin/victories-and-defeats-developing-chatbots-for-e-commerce-35e3811549cd> - open challenges chatbots

<https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1419&context=amcis2018>

* Understanding User Interactions with a Chatbot: A Self-determination Theory Approach

<http://blogs.evergreen.edu/cpat/files/2013/05/Computer-Power-and-Human-Reason.pdf>

interesting read on why you should never make a mental health chatbot

<https://arxiv.org/pdf/1908.08835.pdf> - deep learning based chatbot models –

**recurrent encoder-decoder =** end-to-end trainable neural networks

This architecture was adapted from the neural machine translation domain

* seq2seq
* RNN
* Deep seq2seq

<https://www.youtube.com/watch?v=SJDEOWLHYVo> - end to end systems are what we should be striving for – train single dataset

* Need RNN recurrent neural net, feed back into input while training it

<https://blog.aimultiple.com/chatbot-analytics/> - useful resource for limitations/testing/frameworks etc

<https://arxiv.org/pdf/1809.02839.pdf> - training data DNN

<https://www.quora.com/What-are-chatbots>

* a stateful chatbot is able to review past interactions and frame new responses in context.
* stateless chatbot approaches each conversation as if it was interacting with a new user. In contrast,

<https://towardsdatascience.com/recreating-myself-from-whatsapp-chats-6dadfaff0d2b>

* whatsapp chats dataset

<https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4>

end to end neural networks

MEENA:

<https://www.uctoday.com/contact-centre/getting-to-know-the-google-ai-meena-chatbot/>

<https://towardsdatascience.com/meena-googles-new-chatbot-657241cf5595>

<https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html>

<https://www.thenational.ae/arts-culture/google-s-new-chatbot-meena-is-supposed-to-be-the-best-one-yet-but-how-human-is-it-really-1.981652>

gogle meena bot – end to end neural network - Underneath Meena is the Transform Seq2seq foundation

<https://arxiv.org/pdf/2001.09977.pdf> - Meena paper

NOTES FROM PAPER: -

* trained end to end
* 2.6b parameter NN trained to minimise perplexity of next token
* metric – perplexity and SSA (Sensibleness and Specificity Average)
* human like open domain chatbot
* open domain v closed domain = engage in any convo (open), respond to keywords/intents (closed)
* examples of open domain bots – cleverbot/mitsuku
* supervised/unsupervised learning
* weakness of open domain – often respond to open-ended input in ways that do not make sense

**Meena methodology:**

* **architecture** = seq2seq with the evolved transformer
* Model trained on multi turn convos
* Input sequence is all turns of the context (up to 7)
* Output sequence is response
* Generic replies are safe and boring ‘I don’t know’ – contextually ok but boring!
* An open question: Can we simply take a model and make it bigger by increasing training data and increasing parameter count?

**Meena Training data:**

* Creates (context, response) from tree of convo. Any part of the tree is a response if It has something before it. First message is the root, replies are child nodes
* Data also filtered to improve quality, disregarding useless data
* Filtereing process via these standards:

1. the number of subwords is less than 2 or more than 128;
2. the percentage of alphabetic characters is less than 70%;
3. message contains URL;
4. author’s username contains “bot”;
5. the message is repeated more than 100 times;
6. the message has a high n-gram overlap with the parent’s text;
7. the message is potentially unsafe or offensive with respect to a commercial text classifier. In addition, we remove copies of the parent’s text quoted in a message

* when a message is removed, all subtrees removed to avoid complexities
* After this 867M pairs!
* Tokenized using bpe
* best performing Meena model is an Evolved Transformer seq2seq with 2.6B parameters 1 ET encoder block and 13 ET decoder blocks
* TPU-v3 core has 16GB of high-bandwidth memor
* maximized memory usage for model parameters and stored only 8 training examples per core. Each training step took about 1 second - learned over 4M tokens per training second.
* To overcome overfitting - add a small amount of 0.1 attention and feed-forward layer dropout.
* Save memory: Adafactor optimizer (Shazeer and Stern, 2018) with 0.01 as the initial learning rate, keeping it constant for the first 10k steps and then decaying with the inverse square root of the number of steps.
* Tensor2tensor codebase to train meena
* 164 times (or epochs) and observed a total of about 10T tokens
* trained our best model for 30 days on a TPUv3 Pod - 40B words - 2.6B-parameter model can overfit 12 on a 61B-token dataset which suggests a surprisingly large model capacity.

**In contrast architecture:**

* Show that a model with sufficiently low perplexity, a simple sample-andrank decoding strategy achieves both diverse and high-quality responses.
* sample-and-rank as apposed to beam search provides diverse and content-rich responses
* how it works: sample N independent candidate responses using plain random sampling with temperature T select the candidate response with the highest probability to use as the final output
* **Temperature** is a hyperparameter of LSTMs (and neural networks generally) used to control the randomness of predictions by scaling the logits before applying softmax
* idea: model with low perplexity so samples can be taken at high temperature to produce human-like content

**results**

* We chose top-k (k = 40) and T = 1.0
* significant improvement from sample-and-rank with N = 20 motivates future work exploring alternate ranking functions and tuning parameters.
* We wrote a rule that detects if any two turns contain long common sub-sequences. We automatically remove candidates that are detected as repetition.

<https://medium.com/@BhashkarKunal/conversational-ai-chatbot-using-deep-learning-how-bi-directional-lstm-machine-reading-38dc5cf5a5a3>

* deep learning based AI conversational bot

<https://github.com/PolyAI-LDN/conversational-datasets> - qa dataset amazon

When using these datasets in your work, please cite our paper, [A Repository of Conversational Datasets](https://arxiv.org/abs/1904.06472):

@inproceedings{Henderson2019,

author = {Matthew Henderson and Pawe{\l} Budzianowski and I{\~{n}}igo Casanueva and Sam Coope and Daniela Gerz and Girish Kumar and Nikola Mrk{\v{s}}i\'c and Georgios Spithourakis and Pei-Hao Su and Ivan Vulic and Tsung-Hsien Wen},

title = {A Repository of Conversational Datasets},

year = {2019},

month = {jul},

note = {Data available at github.com/PolyAI-LDN/conversational-datasets},

url = {https://arxiv.org/abs/1904.06472},

booktitle = {Proceedings of the Workshop on {NLP} for Conversational {AI}},

}

<https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215> -

<https://openai.com/blog/gpt-2-1-5b-release/> - nlp

<https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4>

* end to end model

<https://arxiv.org/pdf/1409.0473.pdf>

* nmt

<https://medium.com/@BhashkarKunal/conversational-ai-chatbot-using-deep-learning-how-bi-directional-lstm-machine-reading-38dc5cf5a5a3>

<https://www.aclweb.org/anthology/P17-2079.pdf> - A Sequence to Sequence and Rerank based Chatbot Engine

<file:///Users/Matt/Downloads/IEEM17_chatbot_CameraReady.pdf>

* + Chatbots and Conversational Agents: A Bibliometric Analysis – researchgate approved text

<https://nextjournal.com/gkoehler/machine-translation-seq2seq-cpu>

* Machine translation using seq2seq learning

<https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html>

* 10 min intro to seq2seq

<https://medium.com/bots-for-business/how-to-build-a-stateful-bot-a2703ff2d57b>

* Stateful/stateless bots

<https://ieeexplore.ieee.org/abstract/document/650093>

* Bidirectional RNN

<https://www.tensorflow.org/tutorials/text/nmt_with_attention>

* TF nmt tutorial

<https://github.com/tensorflow/docs/blob/master/site/en/tutorials/text/text_classification_rnn.ipynb>

* TF sentiment analysis text classification
* Ready to use datasets – used movie corpus

<https://www.aclweb.org/anthology/N19-4011.pdf>

* Chateval – tool measuring success in chatbot

<https://towardsdatascience.com/recent-advancements-in-nlp-2-2-df2ee75e189>

* Recent advancements in NLP

<https://blog.aimultiple.com/chatbot-analytics/>

* Chatbot metrics – measuring success

<https://blog.aimultiple.com/chatbot/>

* What is a chatbot

<https://blog.aimultiple.com/chatbot-testing-frameworks/>

* Chatbot testing frameworks

<https://blog.aimultiple.com/chatbot-fail/>

* Chatbot failures

<https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313>

<https://github.com/huggingface/transfer-learning-conv-ai/blob/master/convai_evaluation.py>

* Creating state of the art AI conversational bot
* Few lines of persona
* Dialog history
* Utterances
* Generates reply
* **Transfer Learning**

start by pretraining a language model on a very large corpus of text to be able to generate long stretches of contiguous coherent text,

fine-tune this language model to adapt it to our end-task: dialog.

It’s a rather large dataset of dialog (10k dialogs) which was created by crowdsourcing *personality sentences* and asking paired crowd workers to *chit-chat* while playing the part of a given character (an example is given on the left figure).

The two most common decoders for language generation used to be **greedy-**decoding and **beam-search**.

**Greedy-decoding***-* each time step, we select the most likely next token according to the model until we reach end-of-sequence tokens. – risk= *highly probable* token may be hiding after a *low-probability* token and be missed.

***Beam-search***try to mitigate this issue by maintaining a beam of several possible sequences that we construct word-by-word. At the end of the process, we select the best sentence among the beams. Over the last few years, beam-search has been the *standard decoding algorithm* for almost all language generation tasks including dialog

several developments happened in 2018/early-2019. First, there was growing evidence that beam-search was strongly *sensitive to the length* of the outputs and best results could be obtained when the output length was *predicted* before decoding

greedy/beam-search decoding was replaced by ***sampling***from the next token distribution at each time step. These papers used a variant of sampling called ***top-k sampling***in which the decoder *sample only from the top-k most-probable tokens* (k is a hyper-parameter).

*beam-search* and *greedy decoding* fail to reproduce some distributional aspects of human texts as it has also been noted in the context of dialog systems:

<https://jalammar.github.io/illustrated-transformer/> - transformer model

<https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>

* Useful article about learning rates

<http://www.bioinf.jku.at/publications/older/2604.pdf>

- LSTM

<https://machinelearningmastery.com/encoder-decoder-attention-sequence-to-sequence-prediction-keras/>

* Encoder-decoder model with attention in Keras
* Attention is a mechanism that addresses a limitation of the encoder-decoder architecture on long sequences, and that in general speeds up the learning

<https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html>

* Seq2seq keras
* Convert sequences to sequences

**Canonical seq2seq** – input legth != output length

* Entire input seq needed in order to predict output seq
* How It works:
* RNN layer acts as encoder – processes input and returns its own internal state. Discard outputs and only return state.
* State acts as context of decoder
* Another RNN layer acts as decoder – trained to predict the next characters of the target sequence given previous characters
* Specifically trained to turn target sequence into same sequence but 1 timestep offset in the future

**In inference mode** – wanting to decode unknown input sequences steps:

* Encode input sequence into state vectors
* Start with target seq size 1
* Feed state vectors and 1-char target sequence to decoder to produce predictions of next char
* Sample next char using these predictions
* Append sampled char to target seq
* Repeat until generated end-of-seq char or hit char limit

<https://www.aclweb.org/anthology/P19-1365.pdf>

* Diverse decoding methods (beam search etc)

<https://machinelearningmastery.com/lstms-with-python/>

* LSTM with python
* 4 types of sequence prediction models = one-to-one, one-to-many, many-to-one, many-to-many

<https://s3.amazonaws.com/academia.edu.documents/57035006/CHATBOT_thesis_final.pdf?response-content-disposition=inline%3B%20filename%3DCHATBOT_Architecture_Design_and_Developm.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=ASIATUSBJ6BAJDUHRIGP%2F20200321%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200321T124225Z&X-Amz-Expires=3600&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEBsaCXVzLWVhc3QtMSJHMEUCIC8Fqou4Ow8%2FI2HJyqlG7XQEIOYS47BCwCfbyxMiAK1DAiEAynecVrEPc2oasPaxTxVWfYOrhuwDb3m9zTVlEeU7D0sqtAMIFBAAGgwyNTAzMTg4MTEyMDAiDO8vsBGryMbzYi2xRSqRA5%2BLlMcdP1Scw4jqoSuFaDJ%2FY2N%2Feiw47subyGFEjtHEhxcUpXr2qYEqz1Qoogs9OZyHiuuDoeW27ZqRcX8qk0KXOTtPVdYe46ydsepb%2FayWqT80HPQLrqCxX3VEbLmIoDdwC3L7Es6MHz%2FETB7%2BkkliLibVKbvzDtgKDc42qHIYoIINVqf62wyGq96E%2B7tuqVbtqCLstkvxXmjID777O6nz%2FcrpA1%2FxcctJLNjVmJVIockJbBxxwSkY4Jg5onzgG6RJDZaXnol%2BRLnDFYX3JARz%2BrRZ8UqK4jOMqynRnXNKrBkfX5XisKxf%2BrRRTo%2FrT0rPTDc4s4M5%2Fgivpx%2BAXwth%2B%2FbKrmDk3NDN7ZxbBWNYf0oQK%2FTPoVb59Xg3LifZUzW9iR49ComDoJP%2F2qQEzJHkVDCBK7p7xcgzWgPCjqkp8NgbbPM428q2rlg04Gj25NsAWBth7BZcHwiDm2zDiT5XRCbpUmu6oSR8ZKtLCjaiUo4fBMGuHQQMW1gn7OtZTp7zKo62WFaBNzgvQIojSXDRMPXn1%2FMFOusBiV4ljQohWp5QUeY%2FyZVQlabQhk8P6FIR7ApbPvbQF4GZTgPs%2BElHnvbJZSU6ukgMKZvjNbNBSX7Mcl9QX2ayf8zxwp4V36MKoMMq5N6JVbsR%2BFpskfz5GuqeLvv0vIgRTBGs5uMr2ZaUlYEvWCR8BKrCfH%2BN3gOlYhd7DQvhZFcwHyU8rEPbW5H4oqFiRGwMPlETUIu0IYy1anxHwkYgLtm01ulicnxqSfun9lMtEDW%2B7NguKzfzntehErS2OzMLSPDfPxiozU3Hg9889HUNCrm0KbZXcv1hMrahirx%2BkQViga0u54CYnj7%2Biw%3D%3D&X-Amz-SignedHeaders=host&X-Amz-Signature=b795e7c2493b55565b5588902d10a657dcc9be124b6ef1187afadfaff97e4e3c>

* chatbot architecture design and development – READ THIS!

<https://towardsdatascience.com/understanding-rnns-lstm-and-seq2seq-model-using-a-practical-implementation-of-chatbot-in-2b9ab76d1eda>

* understanding lstm rnn seq2seq

<https://towardsdatascience.com/how-to-implement-seq2seq-lstm-model-in-keras-shortcutnlp-6f355f3e5639>

* keras lstm example

<https://adeshpande3.github.io/adeshpande3.github.io/How-I-Used-Deep-Learning-to-Train-a-Chatbot-to-Talk-Like-Me>

* creating self bot

<https://en.wikipedia.org/wiki/Vanishing_gradient_problem>

* machine language vanishing gradient problem

<https://medium.com/botsupply/generative-model-chatbots-e422ab08461e>

* models

<https://minimaxir.com/2019/09/howto-gpt2/>

* custom gpt-2

<https://arxiv.org/abs/1406.1078>

* learning phrase represetations rnn encoder decoder model

<http://www.bioinf.jku.at/publications/older/2604.pdf>

* LSTM

<https://towardsdatascience.com/personality-for-your-chatbot-with-recurrent-neural-networks-2038f7f34636>

* PERSONALITY IN BOT

<http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/>

* Deep LEARNING for bota

<https://medium.com/tensorflow/a-transformer-chatbot-tutorial-with-tensorflow-2-0-88bf59e66fe2>

* transofromer tutorial I’ve been following

<https://blog.kovalevskyi.com/migrating-our-chatbot-training-logic-from-colab-to-google-cloud-engine-d34b788e8eff>

**limitations of training chatbot on google colab**

* Only one old K80 GPU is available and it is slow
* GPU is provided only for 12 hours max
* It is a Notebook service, not designed for the task that we were doing (in fact all our cell were Shell scripts!)

<https://colab.research.google.com/notebooks/io.ipynb#scrollTo=YFVbF4cdhd9Y>

* Colab with GCS

<https://medium.com/swlh/end-to-end-chatbot-using-sequence-to-sequence-architecture-e24d137f9c78>

**-end to end chatbot using seq2seq architecture:**

- ELIZA - a lot of hard-coded rules

- nature of dataset we chose plays a very important role as it defines the characteristics of the chatbot

**Tokenization**

* Convert words to numbers
* Tokenize words so that each word refers to a number to build vocabulary of words
* Normalising = trim those sentences that are having rare words(words that have occurred less than 5 times in entire data corpus) in both the question and answer. This can be seen as a hack that can help in faster convergence.

**batch-wise prediction**

* Initially tried sending single sentence at a time as input to model – unsuccessful as model didn’t converge so resulted to batch wise = multiple inputs at once

**Architecture**

* Input is query sentence and output is reply
* Multi words in multi words out - variable length input and output
* Obvious architecute is recurrent neural network
* RNN – many to many is desired for chatbot. Different types of RNN one-to-one, one-to-many, many-to-one, many-to-many x2

**RNN Model 3 parts**

1. **Encoder**
2. **Attention**
3. **Decoder**
4. **Encoder**

* Simple RNN takes input and returns single vector representing all words
* Embedding