# Project: Create a Chatbot

## Author: Nikiforos Mandilaras Email: [nikiforosmandi@windowslive.com](mailto:nikiforosmandi@windowslive.com)

## Approach

Creating a system, able to have a conversation by responding to user input is a quite demanding task. Obviously trying to find the most probable response out of a predefined set of sentences is not enough, as natural language can be so complicated and diverse.

Such a system must be able to generate itself the response and in order to do that it must have developed an understanding of the language and its mechanisms. This approach consists an active field of research and in the past years many advances have been made. At the core of this advances are the **Language Models**.

The language modelling as a task is self-supervised as we try to predict the next word in a sentence. Models trained on that task have been proved capable enough to be used in other NLP tasks after fine-tuning. This is a huge step forward as enabling transfer learning in NLP can unlock many downstream tasks for which there weren’t sufficient data to model adequately.

Therefore in this work, we are going to use a pre-trained language model and fine-tune it to the conversational task with the dataset provided. The selected model is GPT (Generative Pre-trained Transformer) is implemented in the library transformers [1], developed by Hugging Face. The transformers as models are recent advances that replaced the previously state of the art Recurrent Neural Networks in such tasks.

Transformers are formed by stacking identical layers one after the other. Those can be organized in an encoder-decoder setting but models only using one of those can also be found. GPT for example is formed as a decoder only model with 12 such layers. Each layer consists of two sub-layers, a multi-head self-attention mechanism, and a fully connected (position-wise) feed-forward network.

For the language modelling task at the end of the previously described model a linear layer is attached in order to depict outputs to the vocabulary space. This is the model that we use to address our task as well. Our implementation is written using the Pytorch Deep Learning Framework.

## Decoding

As it is mentioned in the referenced paper [3] a crucial sector for the quality of the generated text by those models are the methods used to pick next token.

The simpler approach is having the output of the model, to pick the word with the highest probability. However choosing the most probable word or sequences of words as in Beam-search approach doesn't seems to provide good results. In fact the language that is produced by those methods is far from that human produces.

To deal with this issue two new approaches came up in the last year. ‘Top-k’ samples each responded word, out of the k most probable tokens based on their probabilities, while the ‘Top-p’ approach samples from the most probable tokens that have a cumulative probability of p. This gives it the ability to focus on the crucial portion of the distribution either this is broad or narrow in contrast with top-k that takes into consideration in a every case a fix number of token .

We implemented both the sampling approaches as well as the greedy method. We conducted ‘discussions’ with all of these methods. The difference in the language generated between the sampling methods was hard to spotted, but greedy approach was quite worse as it usually got stuck in repetitive text.

More technical information and a deeper analysis of the implementation are provided in the Jupyter Notebook.

## Sample utterances

A picture with some utterances with the bot follows.

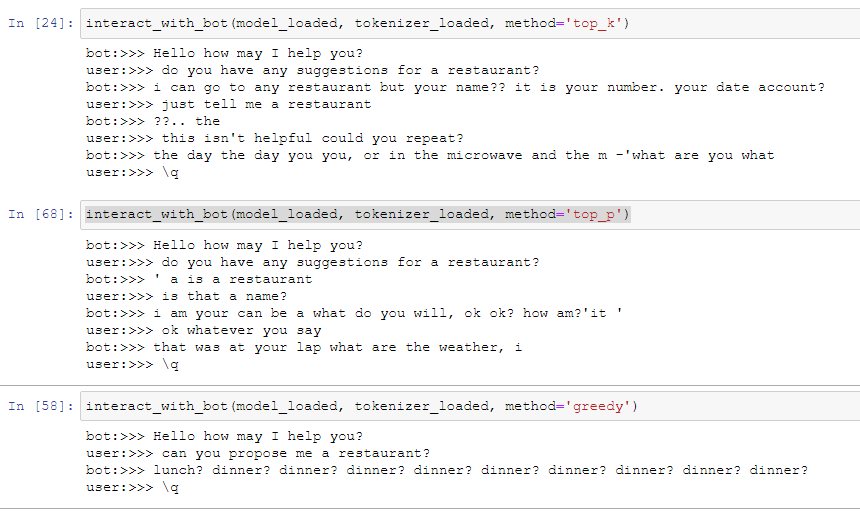


Figure 1 Utterances with the bot using different decoding methods

## References

[1] Transformers Library: https://github.com/huggingface/transformers

[2] Medium Tutorial: https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313

[3] The Curious Case of Neural Text Degeneration: https://arxiv.org/abs/1904.09751