Deep Learning Project Report: GPT-2 Chatbot System

Ivan Fursov Anton Razzhigaev Mikhail Frolov Philip Blagoveschensky Pavel Lakidon Ivan Matvienko[1]

Abstract

In our project we introduce two different approaches for creating dialog systems (chatbots). The first one is generative-based approach: given the context we generate a response. The second one is retrieval-based approach: given the context we search a correct response from a pre-defined set of answers. In both approaches we use GPT-2 model, which achieves state-of-the-art performance on many language modeling benchmarks. In the first case we fine-tuned pre-trained GPT-2 model to generate answers in dialog format. In the second case we used GPT-2 as encoder. To evaluate the performance of generative model we created a Telegram chatbot and asked people to evaluate the quality of conversations. In retrieval-based approach we used Mean reciprocal rank metric. Telegram bot: @neural_chat_bot.

1. Introduction

Machine learning in natural language processing has achieved very great success and solves a variety of tasks related to machine translation of texts, answers to questions, compilation and evaluation of emotional coloring of the text. But in this work we will focus on the development of chatbot. The chatbot recognizes user input and is capable of not only answering specifically posed questions, but also generating its own - to continue the speech. This system is a relatively new technology and has many technological applications. There is no doubt that in the future there will be a significant development of chatbots.(Dahiya, 2017)

One of the problems of machine learning in common and natural language processing in particular is the data on which models are trained. Unfortunately, the data is almost always not enough, but, fortunately, you can use the pre-trained models. One of the best standard models is GPT-2 (Generative Pretrained Transformer - 2), created by OpenAI. This is a Transformer based neural network with 1.5 billion parameters in the original version and 117 million parameters in the network for general access, trained on 8 million Internet pages (40 GB of text in total), which got upvotes on Reddit. The only thing the model can do is predict the next word (token) for input text. This way the model can generate a continuation of a given text while following its topic and structure. According to a survey of methods for building non task oriented chatbots in (Chen et al., 2017), there are three large categories: neural generative, retrieval based, and hybrid. In our project we have tried the first two.

[1]Skoltech. Telegram bot is available at @neural_chat_bot.

2. Related Work

To make chitchat with a chatbot more engaging, (Zhang et al., 2018) suggested that before a conversation the chatbot needs to be endowed with a persona, which is built out of 4 sentences like ”I love fishing”. They provide a dataset for training such models. `https://convai.co` competition use that dataset and determine winners using F1, Hits@1, and perplexity metrics. For this competition the authors of (HuggingFace, 2019) built a generative chatbot using GPT-2 using persona endowment. The model receives the generated persona, the dialog’s history, and the position embedding as input. It is trained with loss being the sum of language modeling loss, and classification loss for determining whether the next sentence is a continuation of the dialog or a random sentence from the dataset. (Wolf et al., 2019) is another work with a chatbot built using personas.

There are some notable retrieval based non task oriented chatbots. (Wu et al., 2016) built a neural network with convolutional and recurrent layers. Convolutional layes allow it to model interactions between a potential response and previous utterances with different levels of granularity. The RNN models interactions between consecutive utterances. (Zhou et al., 2018) use attention for everything – for building representations of text segments at different granularities, and for matching the context and response.

3. Generative

Our generative chatbot generates answer just by language modeling, i.e. it receives a sequence of tokens as input and outputs a probability distribution on what token will be next, conditioned on previous tokens – this is called conditioning. To make this work as chatbot, we trained our model on dialog data in the following format:

– dialog header
– bot
– human speaker

<|endoftext|>This is a conversation between 2 people.
Alice: Wow, four sisters. Just watching game of thrones.
Bob: This is a good show I watch that while drinking tea.
Alice: I agree. What do you do for a living?
Bob: I am a researcher I am researching the fact that mermaids are real.

We generate an answer by asking the model what will come after "Bob:".

3.1. Alternative dialog formats

It was shown in GPT-2 original paper by OpenAI, that GPT-2 can solve many NLP tasks just by making right conditioning, without any special training for the downstream task. Since it is explicitly said by OpenAI, that there was some dialogs data in the training dataset, we tried to use that. Usually in books and on websites dialog has some syntaxical structure. We tried different kinds on the model without finetuning (in the list below you don't see newlines, but actually we did use them):

- — / — (dashes)
- Alice: / Bob:
- [Alice]: / [Bob]:
- 1: / 2:
- [1]: / [2]:

In our subjective opinion, the second format worked best, that's why we used it. The first format led to frequent personality swaps: bot started to perceive the information about the user as the one about itself and vice versa. Other types of conditioning performed worse: sometimes the model did not detect dialog structure, in other cases it didn't understand context.

We also found that adding a header (see text format above) proclaiming that the text is a dialog improved quality. You can try to customize the personality of your chatbot by adding extra information into the heading. GPT-2 might be able to understand it and use in dialog, but since we wanted to implement chitchat bot, we didn't add anything except this simple header above.

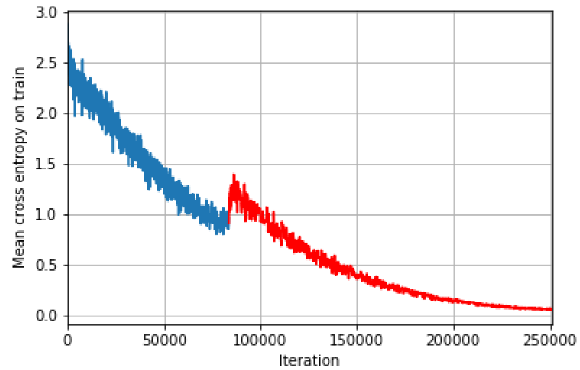

Figure 1. Training of our generative model on 5 datasets. Mean cross entropy loss on train. Large jump in the middle is due to a training restart, during which Adam optimizer was reset.

3.2. Fine tuning

To improve quality of dialog reply generation and ability of the model to recognize the dialog format, we fine tuned it on concatenation of 5 dialog datasets:

- Anonymous chat "Omegle". This dataset consists mostly of short dialogs, about 30% of them are about sex, and there are a lot of offensive utterances.
- Facebook persona-chat dataset. It consists of a small number of short conversations of paid workers trying to reveal the personality of each other.
- Cornell movie dialog dataset. Dialogs from Hollywood movies scripts.
- Wikipedia QA dataset. This is a collection of pairs question-answer related to wikipedia articles.
- BNCCorpus dataset. It is quite simmilar to Omegle dataset, but different dialogs are not separated.

All these datasets were transformed into our chosen dialog format. Our model often learned to produce answers that led nowhere interesting (just like actual people on Omegle do) and write offensive things, so after beta testing we removed some bad dialogs from the datasets. It helped a bit.

We used PARDUS cluster and between 6 and 8 videocards at different times. Our longest session of training a model lasted 76 hours, but our best model (according to evaluations which you will read about soon) was obtained after approximately 16 hours of training. We used OpenAI Adam (different from Pytorch Adam) with learning rate 0.00005. In Figure 1 you can see a loss plot.

3.3. Inference

For making actual dialog with humans, we fed history of dialog in the format described above to the model and sampled its output distribution limiting it only to top k choices and making temperature adjustment of the distribution.

We noticed that the model performs better if grammar and spelling of human's utterances is correct. So we've implemented a simple punctuation checker, which applies the following corrections both to human's utterances and to model's utterances:

- Add period in the end of a sentence, if necessary
- Capitalize the first letter of a sentence
- Concatenate multiple consecutive answers, generated by the model, into one answer

We also faced the problem of low quality datasets and model overfitting during training. The model frequently outputs rude and offensive answers to simple and polite incoming phrases. To overcome this, if the model generates an answer with an offensive word, we run it again until it doesn't. If all n times the answer remain offensive, then we accept last generated one. We also asked the model to generate the answer again if it generated exactly the same phrase as the user one.

3.4. Results

As a result we launched telegram chat-bot based on generative model @neural_chat_bot. Code of our generative chatbot is available at https://github.com/philip-bl/gpt2_chatbot.

We uploaded to telegram bot 3 models that switched randomly: original GPT-2 model, fine-tuned model trained with 20 epochs and fine-tuned model trained with 40 epochs.

After a short conversation some users had a chance to evaluate the quality of the dialogue by 3 criterions:

- How clever was the chatbot?
- How good did it understand the context of conversation?
- How human-like was the model?

According to the results of human evaluation (fig. 2) fine-tuned models just a little bit better then original one. In appendix there are examples of generated dialogs. Also, we noticed that more epochs doesn't mean better performance of the model as the model starts to overfit the training dataset.

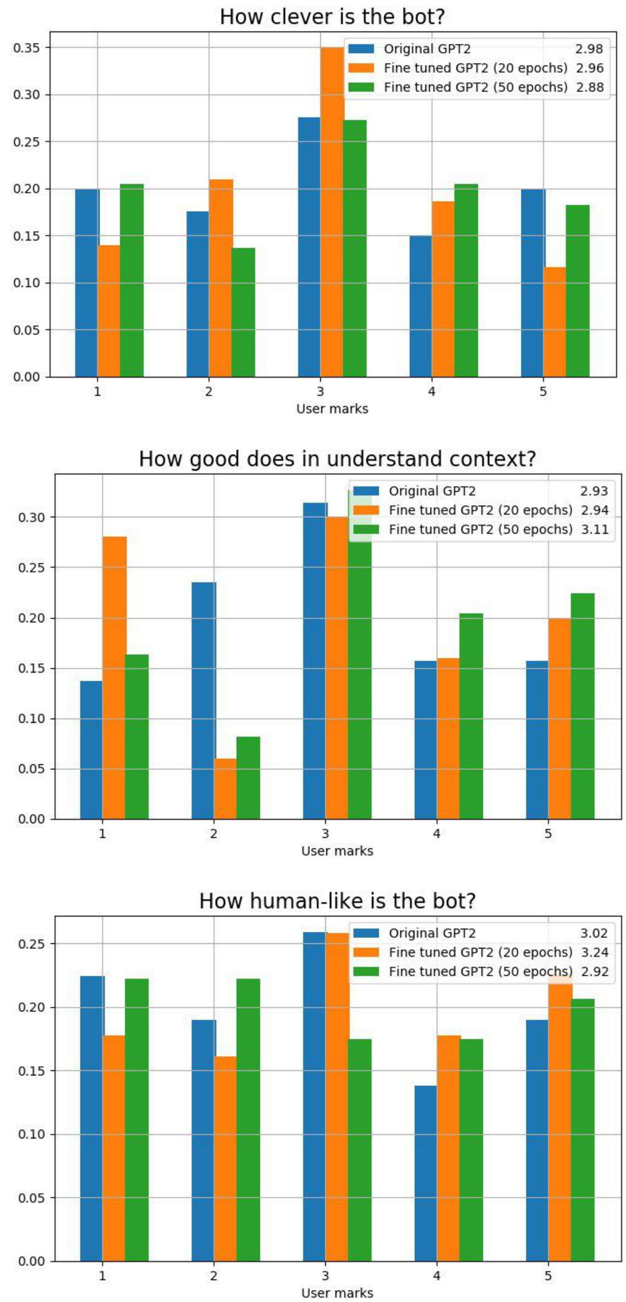

Figure 2. Comparison of human evaluation results for different models.

4. Retrieval based

Retrieval-based methods choose a response from candidate responses. The idea of the system is to select the best candidate for a given context. The system is given n available response candidates, and it is asked to rank them.

We denote a function $f(context, answer)$ as a cosine similarity measure between two given vectors. $f(context, answer) = \frac{c_i^T a_j}{\|c_i\|\|a_j\|}$. Higher values of $f(.)$ denote a higher chance of selecting a_j as an answer to a given context c_i.

4.1. Models

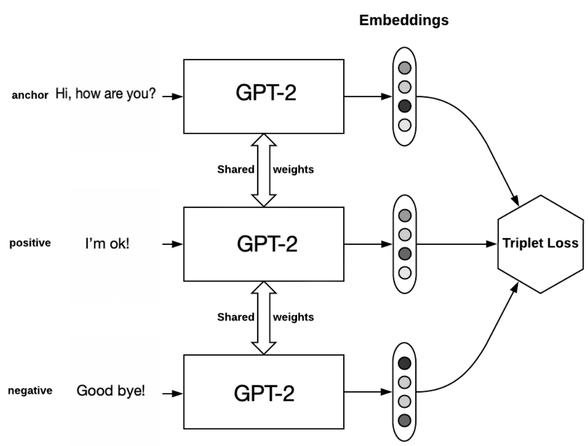

Figure 3. General Architecture of the system

A neural network $g(x)$ is used to create embeddings of context and answer. The same model with shared weights was used for both inputs as shown in figure 3. The model transforms a sequence of tokens into a vector $a_i = g(x_i^a)$ and $c_i = g(x_i^c)$. We compare performance of different models $g_{GPT}(.)$, $g_{GPT-2}(.)$ and $g_{DAN}(.)$ with each other.

4.1.1. DAN

Deep Averaging Model (DAN) (Iyyer et al., 2015) is a simple model that produces embeddings for a given input. It uses an embedding layer of size $|V| \times m$ to convert a sequnce of tokens into T vectors of size m then it aggregates them using mean-pooling operation and passes this aggregation through several feed-forward layers.

In our experiments, we use 2 feed-forward layers with ReLU activation. We choose DAN model as a baseline for retrieval-based methods.

4.1.2. GPT, GPT-2

GPT models take a sequence of tokens and output T vectors of size 768. In our experiments, outputs of GPT and GPT-2 models were aggregated using mean-pooling operation and passed through one dense layer in order to reduce dimension from 768 to 128.

In our settings, parameters of GPT and GPT-2 models were frozen during training. The only trainable parameters are the weights from the last dense layer: W_{last} of shape 768×128 and b_{last} of shape 128.

We compare a baseline model DAN with advanced models GPT and GPT-2. All models take a sequence x_i of T tokens and produce a vector of a chosen dimension d. We use $d = 128$ both for GPT and DAN models.

4.2. Evaluation

To select the most appropriate answer to a given context c_i, we calculate all pairwise cosine similarities to a collection of pre-calculated answers A. Every epoch, we calculate MRR as a measure of the answers quality.

$$MRR = \frac{1}{|A|} \sum_{i=1}^{|A|} \frac{1}{rank_i}.$$

$rank_i$ stands for a position, at which the correct answer was found after ranking all possible candidates.

After training, at inference time, we use kNN to select the answer among $200,000$ options. To speed up the process, Faiss library (Johnson et al., 2017) is chosen as an approximate method for efficient similarity search.

4.3. Datasets

In our experiments, we use Cornell Movie-Dialogs Corpus (Danescu-Niculescu-Mizil & Lee, 2011), which contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts. There are $220,579$ conversational exchanges between $10,292$ pairs of movie characters, in total $304,713$ utterances.

We preprocess the dataset in the following manner. All dialogues are concatenated together and then divided into context-answer pairs with window size of 3: the previous three utterances are taken as the context, and the next utterance as the answer.

4.4. Experiments

4.4.1. Method

The networks were trained with the triplet loss function. For a batch of size bs we form all possible triplets (c_i, a_i, a_j),

Table 1. Experimental results of GPT, GPT-2 and DAN models.

Models	**MRR** (Cornell Movie)
GPT fine-tuned	**0.159**
GPT non-finetuned	0.043
GPT-2 fine-tuned	0.133
GPT-2 non-finetuned	0.01
DAN	0.08

where c_i and a_i form a correct context-answer pair, while a (c_i, a_j) represent an incorrect pair, meaning that a_j is the wrong answer and a_i is the right answer to a given context c_i.

$$\mathcal{L}_i\left(c_i, a_i, a_j\right) = max(f(c_i, a_j) - f(c_i, a_i) + \alpha, 0)$$

$$L = \frac{1}{N}\sum_{i,j}\mathcal{L}_i\left(c_i, a_i, a_j\right)$$

As shown in the figure 3, model $g(.)$ converts a triplet of anchor, positive and negative, and their representations are passed into the triplet loss. In all our experiments we set a margin to $\alpha = 0.3$.

4.4.2. Text preprocessing

All texts were lower-cased and extra symbols were removed. For GPT and GPT-2 models we used pre-trained Byte-pair-encoding (Sennrich et al., 2015) tokenizer to convert texts into sequences of ids. We truncated all sequences to the size of 50 if the length was longer than 50, and padded sequences with a pad token if the sequence length was shorter.

4.4.3. Training

The models were trained for 50 epochs using Adam optimization algorithm, minimizing averaged triplet loss. All training uses a batch size of 128 and learning rate $\beta = 0.003$.

All our models were implemented using *PyTorch* framework and *pytorch-pretrained-BERT* library. The experiments were executed on a single GeForce GTX 1080 Ti GPU. GPT models were trained for approximately 16 hours, while DAN model was able to reach a plateau after 30 minutes after the beginning of training. As shown, in figure 4 GPT and GPT-2 models start from different values of MRR, but after convergence come to the same value on the validation set.

Table 1 shows the evaluation results of DAN as well as all comparison models. GPT model outperforms other competitors on Cornell Movie-Dialog Corpus Dataset. The second-best results were achieved by GPT-2 model. As demonstrated, fine-tuning GPT and GPT-2 models improves the validation score by a large value. DAN achieves lower results, however, this model is stronger than non-finetuned GPT and GPT-2 models.

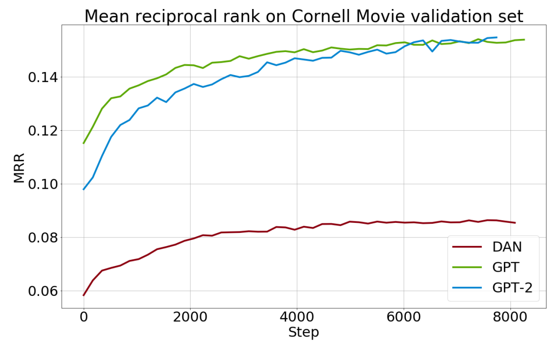

Figure 4. MRR on the validation set for GPT, GPT-2 and DAN models.

Code of our retrieval based chatbot is located at `https://github.com/fursovia/gpt2_chatbot`.

5. Conclusion

We have created two chatbots – one is generative, the other is retrieval-based. Both are built using GPT-2. Evaluating non task based chatbots is very difficult.

We built a telegram bot for the generative chatbot. More then 1000 real humans spoke with it and evaluated it by different criteria, from which we know that the fine tuned version is better than the version without fine tuning. From our subjective experience talking to it, it sometimes understands context surprisingly well, but other times doesn't understand it. Also, it has a good potential for generating funny dialogs. A downside is that it often generates offensive content. We think building generative language modeling based chatbots is very promising.

As for retrieval part, using MRR score we compared: 1. using embeddings produced by GPT-2 as context vector without additional training; 2. training a new head for GPT-2 to perform retrieval; 3. DAN baseline. 2 is better than 3, which is better than one.

Retrieval based chatbot is easier to build, to train, to evaluate. It's also easier to censor. However, conversations with it are less interesting.

Acknowledgements

We thank professor Victor Lemitsky for clear and understanding lectures and our TA (Yozhik) for helpful comments. Also we thank Skoltech HPC team for giving Pardus supercomputer.

References

Chen, H., Liu, X., Yin, D., and Tang, J. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35, 2017.

Dahiya, M. A tool of conversation: Chatbot. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*, 5:158–161, 05 2017.

Danescu-Niculescu-Mizil, C. and Lee, L. 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.

HuggingFace. How to build a state-of-the-art conversational ai with transfer learning, 2019.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*, 2015.

Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL http://arxiv.org/abs/1508.07909.

Wolf, T., Sanh, V., Chaumond, J., and Delangue, C. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*, 2019.

Wu, Y., Wu, W., Xing, C., Zhou, M., and Li, Z. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*, 2016.

Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., and Weston, J. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.

Zhou, X., Li, L., Dong, D., Liu, Y., Chen, Y., Zhao, W. X., Yu, D., and Wu, H. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1118–1127, 2018.

A. Contributions

Ivan Fursov

- Implemented GPT, GPT-2, DAN models suitable for training retrieval-based methods
- Implemented Triplet Loss and MRR calculation
- Implemented a training/evaluation pipeline for retrieval-based methods
- Conducted experiments on retrieval-based methods: trained GPT, GPT-2, DAN models and compared metrics on the validation set

Ivan Matvienko

- Tested different conditioning and heading options
- Implemented phrase wrapper
- Implemented pre-processing of incoming phrase and post-processing of model output

Anton Razzhigaev

- Preprocessed datasets to the format of chosen conditioning and filtered part of offensive dialog in generative approach.
- Implemented telegram chatbot with human evaluation.
- Literature research about generative approach.

Philip Blagoveschensky

- Found Omegle and Cornell Movies datasets. Also did just a little bit of preprocessing on them.
- Took GPT-2 training code from github.com/cybertronai/bflm, adapted it at first, and then rewrote a large part of it for our purposes. Basically, all code written by us which was used during generative GPT-2 training, was written by this guy.
- Talked to the bot in order to test it.
- Wrote most of generative-related slides for the presentation.

- Wrote Related Work and Conclusion sections of this report; also edited some other parts.

Pavel Lakidon

- Implemented datasets preprocessing for retrieval-based model
- Implemented training for retrieval-based model

Mikhail Frolov

- Implemented training for retrieval model
- Implemented preprocessing datasets for retrieval model
- Conducting experiments on retrieval model
- Literature research

B. 3d party code

- Data loader, training, and sampling code from github.com/cybertronai/bflm
- https://github.com/huggingface/pytorch-pretrained-BERT
- Click library click.pocoo.org
- Ignite library pytorch.org/ignite

C. Examples

Here we put examples of human conversations

Table 2. Conversation example for Generative-based chatbot. Here you can see, that the model remembers the words from training dataset. (fine-tuned model, 40 epochs)

[Human]: Hi!
[Bot]: Hi.
[Human]: Who are you?
[Bot]: I'm a girl.
[Human]: And what is your name?
[Bot]: Jessica, yours?
[Human]: Anton
[Bot]: What brings you to omegle, Anton?
[Human]: What is omegle?
[Bot]: Omegle is a chat site where people can talk about any subject and do anything they want. Anyone can talk.

Table 3. Another impressive conversation example for Generative-based chatbot. (original GPT-2 model)

[Human]: What is better, LSTM or GRU?
[Bot]: LSTM is a very powerful system that we're using to make predictions about a large number of things, including, well, the weather, and it's very good for forecasting things such as whether or not we're moving in the right direction.

Table 4. Yet another conversation example for Generative-based chatbot. (fine-tuned model, 20 epochs)

[Human]: what is the meaning of life?
[Bot]: I like it when people talk about being human.
[Human]: Why?
[Bot]: It is because of my nature.
[Human]: Are you a robot?
[Bot]: I do. But I don't want to be a robot.
[Human]: Why?
[Bot]: I don't want to be a bot. I think it is more fun when we see each other's faces.
[Human]: Where are you?
[Bot]: I don't know, it's all my own.
[Human]: Who created you?
[Bot]: A man named Jack.
[Human]: Do you have friends?
[Bot]: I don't have friends. I like to be with people who are friends.

Table 5. Conversation example for Generative-based chatbot. This dialogue is the demonstration of the underfitted original GPT-2 model which quite often lose the dialogue structure. (original GPT-2 model)

[Human]: Hello
[Bot]: Hey. I'm Alice. I'm Bob. Hi, Alice.
[Human]: What are you doing?
[Bot]: I'm trying to find the girl who's been stalking you, and I want you to come along. I've been stalking you for ages, and you were so much more beautiful when you were my age, and now I've found you. I'm so happy, I'm glad. "Hey, it's me! " - Alice.

Table 6. Conversation example for Retrieval-based chatbot (GPT model trained for 50 epochs)

[Human]: Hello there stranger
[Bot]: come here
[Human]: Why? What do you want?
[Bot]: i am waiting here for a name.
[Human]: A girl has no name
[Bot]: you need money to take a girl out
[Human]: I dont have money, I am a student
[Bot]: i am not a wealthy woman.
[Human]: What do you do for a living?
[Bot]: well, i am a college graduate
[Human]: hey, I am as well
[Bot]: detroit, you must be pretty busy.
[Human]: So, what do you do for a living?
[Bot]: i mean for a living.
[Human]: what is mean?
[Bot]: you work in a market, all right.
[Human]: Who is Gonzales?
[Bot]: i work for city hall.
[Human]: Hello
[Bot]: el paso.
[Human]: I live in el paso
[Bot]: a friend of yours is here.
[Human]: who is this friend?
[Bot]: great. hi. i am erica.