# Technical Report

**Joke Generation Bot**

Irina Podlipnova
Boris Guryev
Ekaterina Levchenko
Vladislav Kuleykin

May 5, 2020

## 1 Overview

These days jokes and memes become part of modern people's life. Each day everybody spends time consuming content. Communities like *Reddit* or *9GAG* periodically release meme calendars.

We propose a question-answer joke generator telegram bot with the aim to distinguish if artificially generated content can entertain users at the same level as human-created jokes.

The final code is available in out GitHub repository [1].

## 2 Model

For the Joke Generation we fine-tuned the GPT-2 model, with 117M parameters. As bigger models didn't fit in our GPU at training time.

### 2.1 Datasets

We gathered three datasets for the fine-tuning.

- QA Jokes (3.29 MB) - the original dataset we've found on Kaggle [2]. It contains 38k question-answer jokes
- Short Jokes (22.8 MB) - the biggest dataset in our collection, it was also found on Kaggle [3] and consists of 231k short length jokes from Twitter and Reddit. But it also contains a lot of noise and misspellings
- Stand up transcripts (13.5 MB) - the manually scraped dataset of stand up transcripts from one site [4]

The QA jokes is the core of our project, as we want to generate the same type of jokes with our bot. But it has two problems: *small size* - as gpt-2 needs at least 10 MB of data to not overfit on it; and *racist/dark humor* - most of the jokes are very dark and racist, which we do not approve.

The short jokes dataset was used to fight with this problems, as it contains a broader range of topics, has much more samples and still close to the QA type of jokes. Moreover we've been able to extract a lot of QA jokes from the short jokes dataset, two times more than in original QA dataset. After extraction we merged new QA jokes with old ones and deleted them from the short jokes dataset, to not have repeated samples in different datasets. This extraction helped us to generalize the QA dataset, as it now covers more topics and less biased towards dark humor.

The dataset with the stand up transcripts was collected with a proposal, such as before training the model to some concrete joke type, it's better to firstly train it on a general text containing high amount of jokes.

As all of datasets contained some noise, we cleaned a lot of it, like notes to the jokes, web links, twitter tags and some bleeped words.

## 2.2 Training

For the fine-tuning of the model we used a script from the `transformers` library [5].

As in early attempts we've seen our model to overfit on QA Jokes dataset, we gathered other datasets and firstly fine-tuned the model on them. The order of datasets used and reasoning is described in datasets subsection.

As our GPU wasn't capable of fitting big batch sizes, we used gradient accumulation to achieve the same behaviour as we would had bigger batch size. So, we used the mini-batches of 2 and the gradient accumulation step of 2/4/8 at different epochs for a total "batch size" of 4/8/16.

For each dataset, we used `learning rate` (lr) equal to 1e-5 and 1e-6.

As for the number of epochs, we chose to train by 4 epochs on each dataset with `lr` equal to 1e-5, when 6 epochs with `lr` 1e-6.

## 2.3 Results

As for the results, we created two models: one trained on all datasets, and the other one trained exclusively on QA jokes to understand if our efforts on pre-training and reducing bias to the QA dataset were anything good.

Here are some samples produced by each model:

Fine-tuned only on QA jokes:

```
Q: What do you call a writer who says he's not in pain?
A: Toilet paper

Q: What does the Jewish people say when they see a Jewish man?
A: "Go away, Jew!"
```

```
Q: Why doesn't everyone use a roof?
A: Because a roof wouldn't do it.
```

Fine-tuned on all datasets:

```
Q: What's the difference between the Obama administration and Kim Jong-un?
A: Kim Jong-un is totally nuts

Q: What did the homeless man say to the homeless woman?
A: Come on!

Q: What did the general say to the soldier?
A: "I can hear you sir."
```

You can see that they both find it hard to produce a good joke and it's still hard to understand which one is better, but we found that the second one makes more references to popular people and generally covers more topics.

But for the final blind test, we will put in our bot the pre-computed samples from each model and some jokes from dataset, after which we will collect the statistics of how much people liked or disliked the jokes from each model and dataset to make the final conclusion.

## 3 Deployment

We found several approaches and libraries for model deployment and selected the most appropriate to our requirements. Initially, we thought that it will be profitable to wrap the model in a dedicated server as a REST API service. This could be done with the following tools:

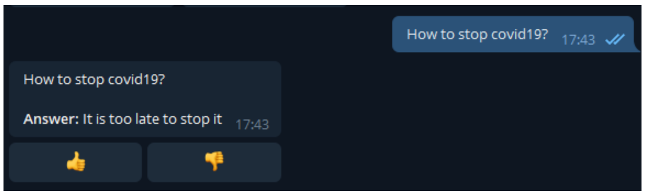
**Cortex** [?] - an open-source platform for deploying machine learning models as production web services. Provides logging, auto-scaling, forwarding GPU into the container, etc. However, this framework is designed to be self-hosted on AWS infrastructure. During this project, we aimed to run anything locally, because of the hardware requirements of the selected language model - they are too expensive for the cloud and suitable for the usual home PC or notebook.

**Model Asset Exchange (MAX)** [7] - this is a project template wrapped around `flask` - python backend framework. We found it over-architected because the main goal of this project is to force developers to provide API documentation to the endpoints.

Finally, we decided to rewrite the CLI of the `transformers` repository [5] in a simple format convenient for our task. In our case, it is enough to keep the model in memory - usual python class instance - and run forward whenever bot receives a message from the user. We tested it on asynchronous model inference and found that parallel calls invoke out of memory when the load is high, so we optimized requests by caching batch of jokes that generated without the user's input and by limiting the "continue joke" requests to single. We found this solution good because Telegram API we use tolerant and robust to late respond. At the same time, the one-joke inference period takes 1.5 sec to produce - this is not too much.

## 4 Bot

We implemented a bot as our interface for the model functionality - joke generation. Our bot has two use cases. The first one is the `/joke` command, which generates the joke as a pair "Joke-Answer."

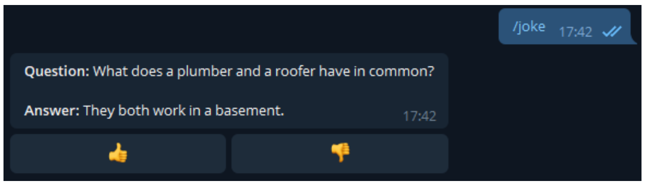


How to stop covid19? 17:43

How to stop covid19?

**Answer:** It is too late to stop it 17:43

Another case only the "answer" generation. To use this function, send a question to our bot, it will concern any text without backslash as a question for a further "funny" answer generation.



/joke 17:42

**Question:** What does a plumber and a roofer have in common?

**Answer:** They both work in a basement. 17:42

After the generation, the user can set a grade for the joke. It consists of two buttons: thumb up and thumbs down. That grade is later used for the metrics system.

## 5 Conclusion

Analyzing the datasets and jokes produced by our model we found that the most popular category is dark humor. This already happened several times when AI language models were provided to communicate with society and impolite text with disrespect to races and minorities provoked anger and negative behaviour (Microsoft's object detection, Facebook's chatbots, virtual assistants, etc). Internet humor is biased to this kind of jokes and our datasets is no exception. People used to live with such behavior from other people. Remote communication and online anonymity contribute to this. But dirty behavior from public models still attracts attention from community and media - probably because of humans' optimism and belief in good intentions of AI (as media still claim language models) and their creators.

At the end, looking at the generated jokes we see that the current language models are still far from becoming a good comedian. As jokes are a lot deeper and complicated than simple language rules. And for the model to understand this principles it will need a lot of data and most probably next generation of architecture. But still, there's some connection between the comedian and language model that tries to be one: most of the jokes that they create are bad, and the main difference is that the comedian can discriminate which ones are bad, and which one are good.

## References

[1] GitHub repository of our project

[2] QA Jokes dataset

[3] Short Jokes dataset

[4] Stand Up transcripts site

[5] Transformers. Run language model example

[6] Cortex. Cloud native model serving infrastructure

[7] IBM Model Asset Exchange