Published in Towards Data Science



Ramsri Goutham Follow

Jan 10, 2021 · 5 min read ★ · Listen



# Poor man's GPT-3: Few shot text generation with T5 Transformer

Minimalistic code for few-shot text generation with HuggingFace

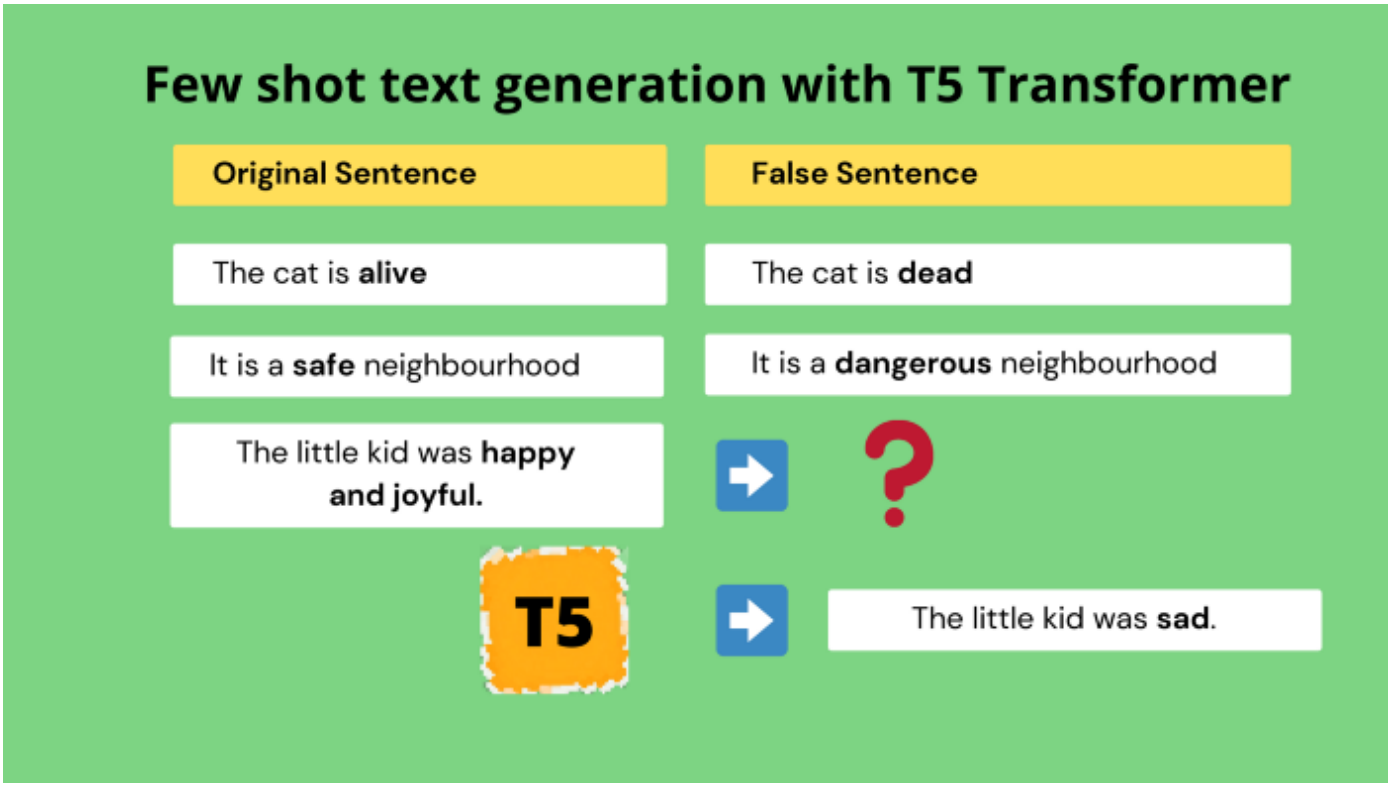


Image by Author

The concept of feeding a model with very little training data and making it learn to do a novel task is called **Few-shot learning.**

A website GPT-3 examples captures all the impressive applications of GPT-3 that the community has come up with, since its release. GPT-3 is shown to generate the **whole Frontend code** from just a text description of how a website looks like. It is shown to generate a **complete marketing copy** from just a small brief (description). There are many more impressive applications that you can check out on the website.

GPT-3 essentially is a **text-to-text** transformer model where you show a **few examples (few-shot learning)** of the input and output text and later it will learn to generate the output text from a given input text.

The GPT-3 prompt is as shown below. You enter a few examples (input -> Output) and **prompt** GPT-3 to fill for an input.

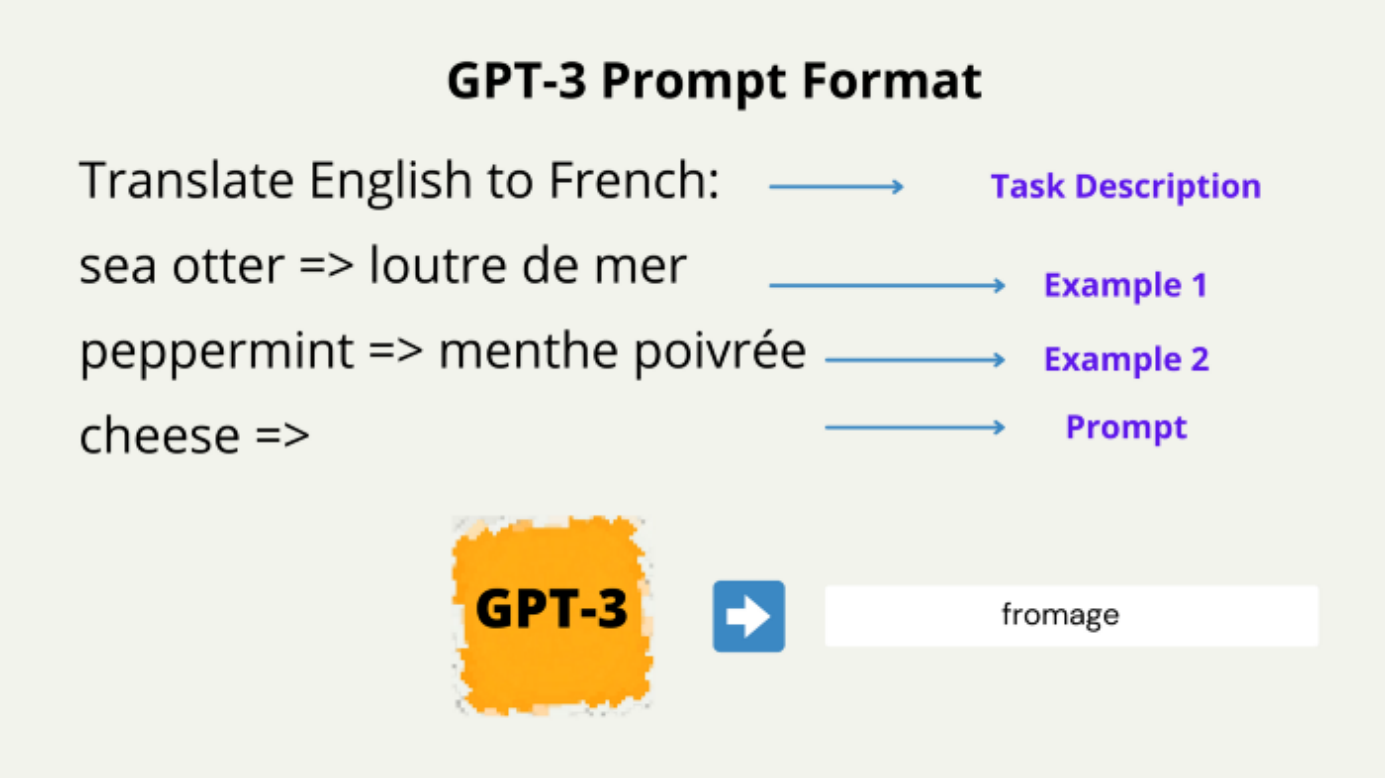


Image by Author

Now being aware of the text-to-text capabilities of T5 Transformer by Google while working on my opensource question generation project Questgen.ai, I decided to push T5 to do the same on an untrained task and see the results.



I must say the results are pretty **impr** a base T5 model by making it learn from just a few **(~10)** examples.

So the task I gave was this —

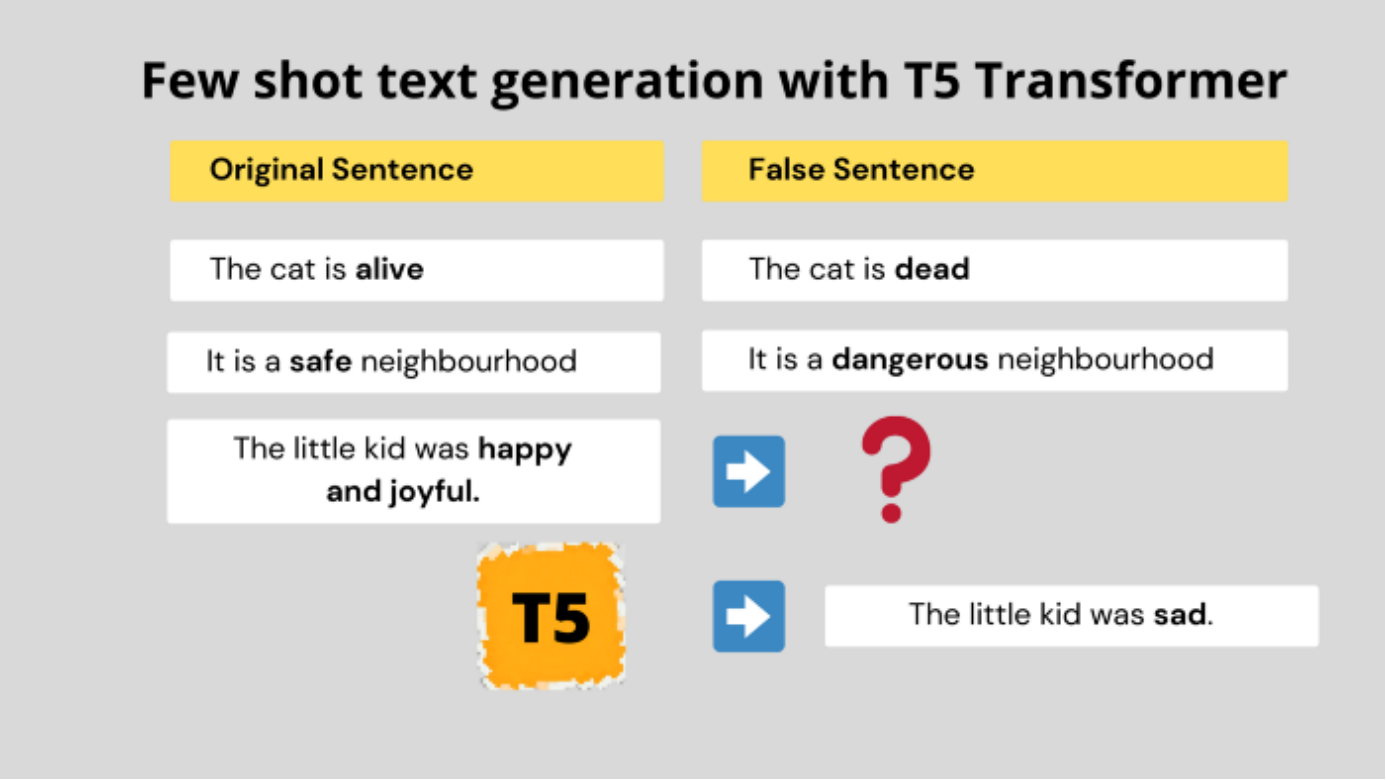


Image by Author

**Input :**

I gave a few **sentence pairs** that are **false sentences** of each other by replacing the **main adjective** with its **opposite** word.

Eg: The cat is **alive** => The cat is **dead**

And after training with only (~10 samples) and < 5 mins of training T5 was able to

T5 Generated sentences (picked from top 3 responses through beam search) :

- The sailor was **unhappy**.
- The sailor was **sad**.

Prompt to T5: The tortoise was very **slow**.

T5 Generated sentences (picked from top 3 responses through beam search) :

- The tortoise was very **fast**.
- The tortoise was very **quick**.

Without much further ado let's look into the code.

## Code

Coming up with the code was an interesting **exploration** for me. I had to go through Hugging Face documentation and figure out writing a **minimalistic** forward pass and backpropagation code using the T5 transformer.

### Colab Notebook

A cleanly organized Google Colab notebook is available here

### 1.1 Installation

Install HuggingFace transformers and check GPU info on Colab.

```
!pip install transformers==2.9.0

!nvidia-smi
```

### 1.2 Necessary Imports and model download

First a few necessary imports from transformers library-


```
import torch
from torch.utils.data import Dataset, DataLoader

from transformers import (
    AdamW,
    T5ForConditionalGeneration,
    T5Tokenizer,
    get_linear_schedule_with_warmup
)

def set_seed(seed):
  random.seed(seed)
  np.random.seed(seed)
  torch.manual_seed(seed)

set_seed(42)
```

Initialize the mode and its tokenizer -

```
tokenizer = T5Tokenizer.from_pretrained('t5-base')
t5_model = T5ForConditionalGeneration.from_pretrained('t5-base')
```

Initialize the optimizer —

Here we are mentioning which parameters of the T5 model needs to be updated after calculating the gradients of every parameter w.r.t to the Loss.

```
# optimizer
no_decay = ["bias", "LayerNorm.weight"]
optimizer_grouped_parameters = [
    {
        "params": [p for n, p in t5_model.named_parameters() if not
any(nd in n for nd in no_decay)],
        "weight_decay": 0.0,
    },
    {
        "params": [p for n, p in t5_model.named_parameters() if any(nd
in n for nd in no_decay)],
```

### 1.3 Training data

The complete training data (~10 samples) that is used for our T5 few-shot text generation task.

### 1.4 Training the model

(optimizer. step) which is standard for all deep learning algorithms.

Most of the effort was in understanding and getting T5 training to this simple loop :)

That’s it. Depending on the GPU, the model is trained in 5 mins or less and it is ready for testing with some unseen samples.


Using beam decoding we get the top 3 sentences generated from the code as -

- The sailor was **unhappy**
- The sailor was **sad**
- The sailor was happy

Test Sentence: The tortoise was very **slow**.

Using beam decoding we get the top 3 sentences generated from the code as -

- The tortoise was very slow
- The tortoise was very **fast**


As you can see T5 is able to generate a false sentence of a given sentence even if it has not seen those **adjectives** or **sentence words** previously in training.

## Bonus

If you have come this far, I have a bonus for you :)

Find the same code in the **Pytorch Lightning** format using the **latest version** of HuggingFace here

## Question Generation using NLP — A course

I launched a very interesting Udemy course titled "Question generation using NLP" expanding on some of the techniques discussed in this blog post. If you would like to take a look at it, here is the link.

## Conclusion

Hope you enjoyed how we explored T5 for **few-shot** text generation task, just like GPT-3.

When I started exploring T5 last year I realized its potential. It can do quite a few text-to-text tasks very efficiently.

Happy NLP exploration and if you loved the content, feel free to find me on Twitter.

If you want to learn modern NLP using transformers, check out my course Question generation using NLP