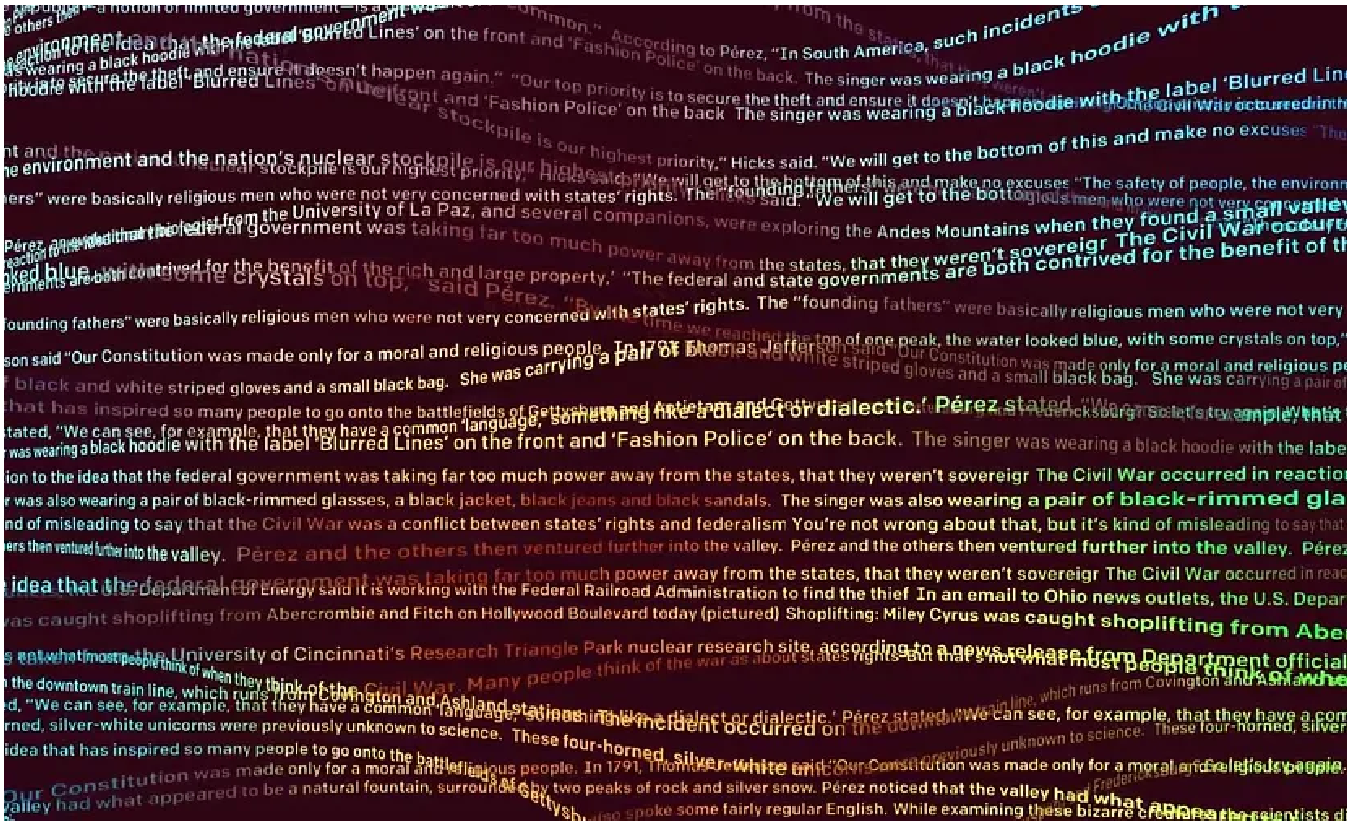
GPT-2 Artificial Intelligence Song Generator: Let's Get Groovy

Thomas Vrancken · Follow
Published in ML6team
8 min read · Jan 20, 2020

This blog-post will show how we can use NLP's magic to generate our own song lyrics. We will use the power of GPT-2, a large pre-trained model developed by OpenAI. The codebase can be found in this notebook (it's in view only mode but you can make a copy to run it).

Special thanks to Thomas Dehaene, Koen Verschaeren, Mats Uytterhoeven and Anna Krogager for their contribution.

I. The model



GPT-2 has been the cool kid on the block of NLP models since its release in February 2019. It’s a pre-trained model, trained over a large database to simply predict the next word of a sentence, and can now be fine-tuned on a smaller database to solve different problems. In our case, generating a bunch of legendary songs. It uses the transformer network architecture (introduced by Google in 2017), based on attention mechanisms.

Though the core strength of GPT-2 lies in the gigantic amount of data used to train it. The OpenAI team used around 40 GB worth of text extracted from the internet (i.e. outgoing links from Reddit posts with a karma of at least 3). In comparison, all of Shakespeare's work combined has an estimated size of about 5.6MB. They trained 4 different models with respectively 124M, 355M, 774M and 1.5B parameters (more parameters leads to more complexity, larger size of the model and higher performance).

II. The data

We used a music lyric database extracted from this Kaggle Kernel. It contains 57.650 music lyrics, mostly in english. They were originally scraped from LyricsFreak. There are a lot of fun analyses to be performed on this data, such as sentiment analysis or calculating which artists have the widest vocabulary (spoiler alert, it's the Wu-Tang Clan, with the Backstreet Boys being the most shallow of them all).

III. The implementation

We implemented the code in Colaboratory, a tool from Google to run Python notebooks in your web browser. It's free and lets you use GPU-accelerated processing! This will make your life easier with the model fine tuning.

First, we load our libraries. We will use the gpt-2-simple library to conveniently play around with GPT-2. Note that Tensorflow 2.0 removed the "contrib" feature, which is needed in gpt-2-simple. Therefore, we force Colab to use Tensorflow 1.x with the `%tensorflow_version 1.x` command.

```python
import os
import pandas as pd

%tensorflow_version 1.x

try:
  import gpt_2_simple as gpt2
except:
  !pip3 -q install gpt-2-simple
  import gpt_2_simple as gpt2
```

import_libraries.py hosted with ❤ by GitHub view raw

We then load our data. It was saved into a public Github repository to make it more convenient to be read into Colab:

```python
dfs = []
link = ('https://raw.githubusercontent.com/ThomasVrancken/'
        'lyrics_generation/master/songdata_{}.csv')
for i in range(4):
  dfs.append(pd.read_csv(link.format(i)))

df = pd.concat(dfs).reset_index(drop=True)
```

load_data.py hosted with ❤ by GitHub view raw

We then save this file on our VM's directory, under a "content" folder. GPT-2 will directly load it from there. You can see the files saved on your VM on the left hand side of your Colab notebook, under the folder tab.

```python
if not os.path.exists('content'):
    os.makedirs('content')

pd.DataFrame({"lyrics": df['text']})\
    .to_csv(os.path.join('content', 'lyrics.csv'), index=False)
```

save_data_locally.py hosted with ❤ by GitHub view raw

Next, we download the GPT-2 models. Note that you can restrict this list to the model(s) you will be using.

```python
for model_name in ["124M","355M","774M"]:  # Choose from ["124M","355M","774M"]
    gpt2.download_gpt2(model_name=model_name)   # model is saved into current directory under /mod
```

download_models.py hosted with ❤ by GitHub view raw

Now we set some hyperparameters and start a session.

```python
learning_rate = 0.0001
optimizer = 'adam' # adam or sgd
batch_size = 1
model_name = "774M" # Has to match one downloaded locally
sess = gpt2.start_tf_sess()
```

start_session.py hosted with ❤ by GitHub view raw

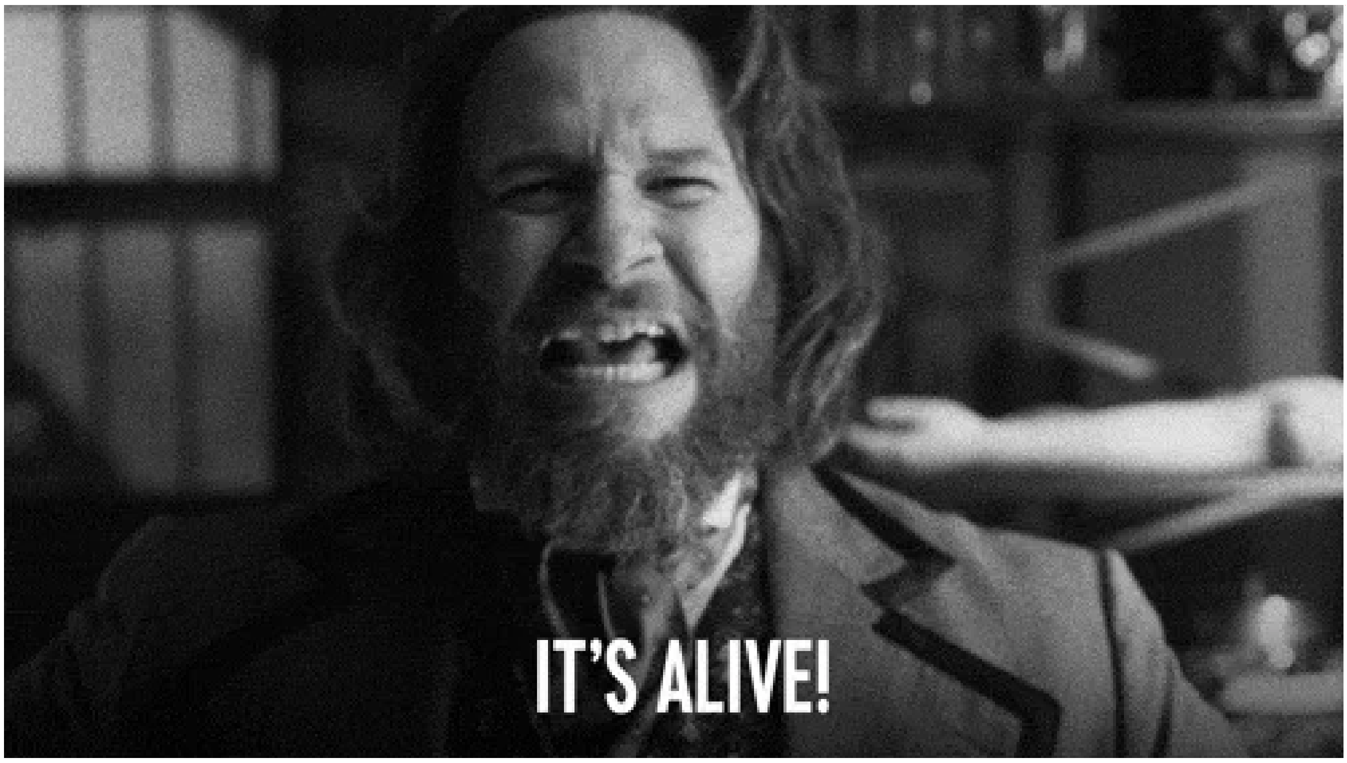
Next, we start the fine tuning. It can take a while (e.g. about 22min for 500 steps with the 774M model). Some useful parameters:

- **`restore_from`**: Set to `fresh` to start training from the base GPT-2, or set to `latest` to restart training from an existing checkpoint
- **`sample_every`**: Number of steps to print example output
- **`print_every`**: Number of steps to print training progress
- **`learning_rate`**: Learning rate for the training. (default `1e-4`, can lower to `1e-5` if you have <1MB input data)
- **`run_name`**: Subfolder within `checkpoint` to save the model. This is useful if you want to work with multiple models (will also need to specify `run_name` when loading the model)
- **`overwrite`**: Set to `True` if you want to continue finetuning an existing model (w/ `restore_from='latest'`) without creating duplicate copies
- **`steps`**: How many training steps to be performed.

```python
gpt2.finetune(sess,
              'content/lyrics.csv',
              model_name=model_name,
              sample_every=50,
              save_every=50,
              print_every=10,
              learning_rate=learning_rate,
              batch_size=batch_size,
              restore_from='latest',
              steps=500)   # max number of training steps
```

fine_tune.py hosted with ❤ by GitHub view raw

The artist is officially born...



You will see that the model prints a couple of sample generated texts every `print_every` steps. It is very nice to see how it evolves.

Something to keep in mind here is that your Colab session can expire during the training (if using a large dataset or number of steps). However, you can retrieve your previous model with the `restore_from` feature.

Once your model is trained, you can use the `gpt2.generate` command to produce some lyrics.

```python
lst_results=gpt2.generate(
    sess,
    prefix="<|startoftext|>",
    nsamples=10,
    temperature=0.8, # change me
    top_p=0.9, # Change me
    return_as_list=True,
    truncate="<|endoftext|>",
    include_prefix=True
    )

for res in lst_results:
  print(res)
  print('\n -------//------ \n')
```

generate.py hosted with ❤ by GitHub view raw

You can pass a prefix into the generate function to force the text to start with a given character sequence and generate text from there (good if you add an indicator when the text starts).

The `nsamples` parameters allows you to generate multiple texts in one run. It can be used with `batch_size` to compute them in parallel, giving the whole process a massive speedup (in Colaboratory, set a maximum of 20 for batch_size).

Other optional-but-helpful parameters for gpt2.generate:

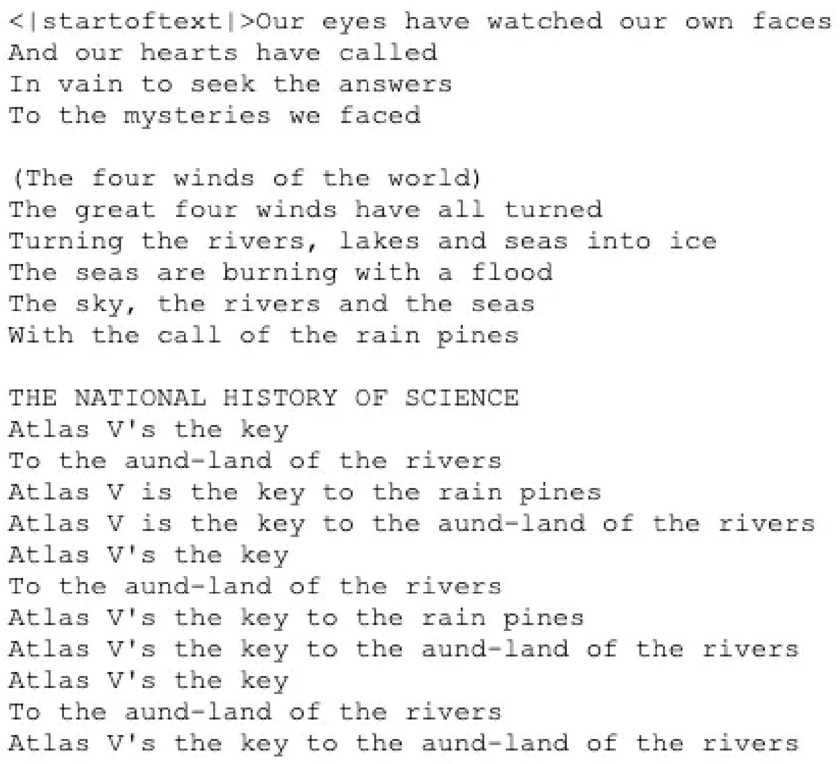
- **`length`**: Number of tokens to generate (default 1023, the maximum)
- **`temperature`**: The higher the temperature, the crazier the text (default 0.7, recommended to keep between 0.7 and 1.0)
- **`top_k`**: Limits the generated guesses to the top k guesses (default 0 which disables the behaviour; if the generated output is super crazy, you may want to set `top_k`=40)
- **`top_p`**: Nucleus sampling: limits the generated guesses to a cumulative probability. (gets good results on a dataset with top_p=0.9)
- **`truncate`**: Truncates the input text until a given sequence, excluding that sequence (e.g. if `truncate`='<|endoftext|>', the returned text will include everything before the first <|endoftext|>). It may be useful to combine this with a smaller length if the input texts are short

- **`include_prefix`**: If using truncate and `include_prefix = False`, the specified prefix will not be included in the returned text

IV. The songs

We have trained a model with the 774M model and 500 steps and used it to generate 100 songs. The entire set of masterpieces can be found here. It had a lot of inspiration, leading to some interesting songs (we had to create titles for them ourselves):

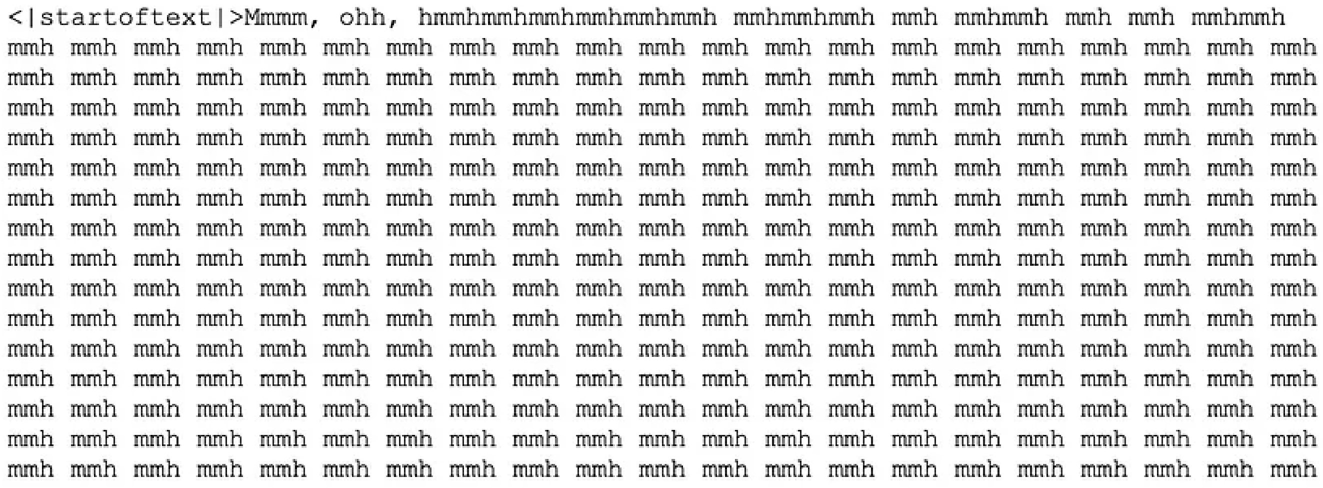
- *"The story of Atlas V"*: Here the bot was slightly dramatic, and developed a weird fixation on "*Atlas V*" though that word cannot be found in the training data set…



```
<|startoftext|>Our eyes have watched our own faces
And our hearts have called
In vain to seek the answers
To the mysteries we faced

(The four winds of the world)
The great four winds have all turned
Turning the rivers, lakes and seas into ice
The seas are burning with a flood
The sky, the rivers and the seas
With the call of the rain pines

THE NATIONAL HISTORY OF SCIENCE
Atlas V's the key
To the aund-land of the rivers
Atlas V is the key to the rain pines
Atlas V is the key to the aund-land of the rivers
Atlas V's the key
To the aund-land of the rivers
Atlas V's the key to the rain pines
Atlas V's the key to the aund-land of the rivers
Atlas V's the key
To the aund-land of the rivers
Atlas V's the key to the aund-land of the rivers
```

The story of Atlas V

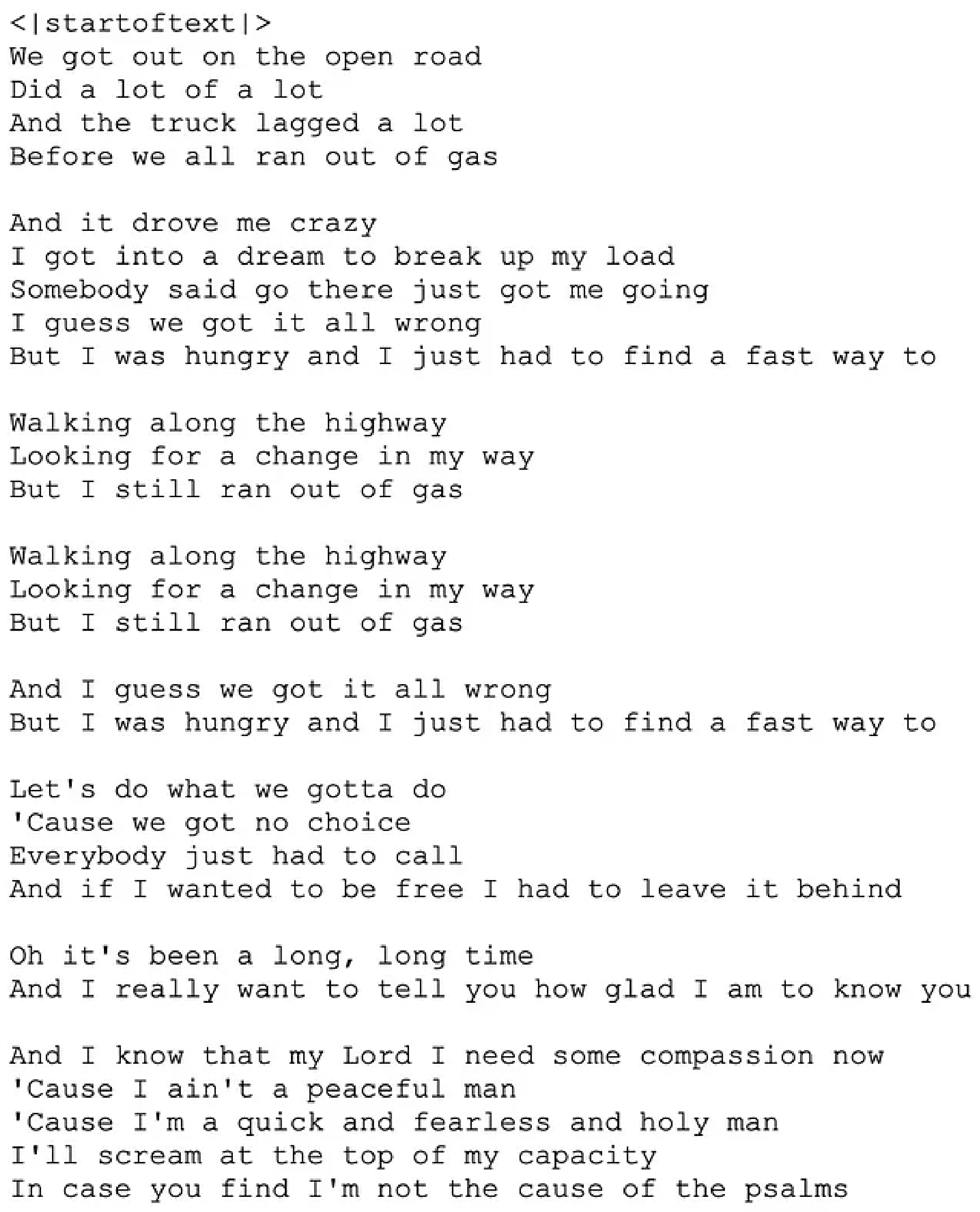
- *"Mmmh"*: In this song, the artist gets stuck in a suave showcase of its pretty voice, repeating "Mmmh" the entire time. In text generation it is pretty common for the model to get stuck in a loop and repeat the same word (series of words) over and over again. It is likely that our training data increased that effect, as many songs have some repetitions of a single word.



```
<|startoftext|>Mmmm, ohh, hmmhmmhmmhmmhmmhmmh mmhmmhmmh mmh mmhmmh mmh mmh mmhmmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh mmh
```

Mmmh

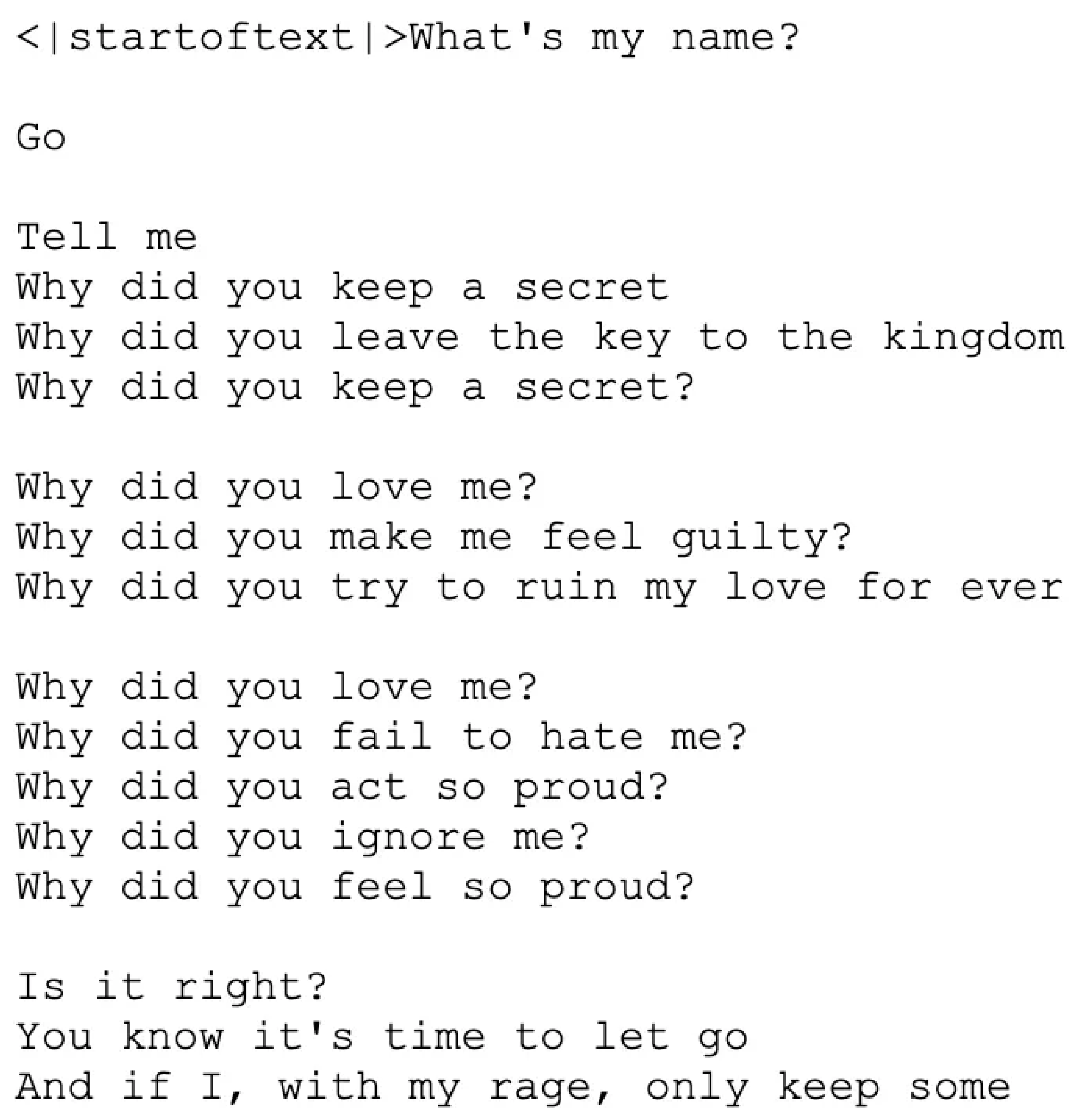
- ***“No Uber on the Highway”:*** Our artist tells about the time the truck broke down on the highway. Not a great memory apparently.



```
<|startoftext|>
We got out on the open road
Did a lot of a lot
And the truck lagged a lot
Before we all ran out of gas

And it drove me crazy
I got into a dream to break up my load
Somebody said go there just got me going
I guess we got it all wrong
But I was hungry and I just had to find a fast way to

Walking along the highway
Looking for a change in my way
But I still ran out of gas

Walking along the highway
Looking for a change in my way
But I still ran out of gas

And I guess we got it all wrong
But I was hungry and I just had to find a fast way to

Let's do what we gotta do
'Cause we got no choice
Everybody just had to call
And if I wanted to be free I had to leave it behind

Oh it's been a long, long time
And I really want to tell you how glad I am to know you

And I know that my Lord I need some compassion now
'Cause I ain't a peaceful man
'Cause I'm a quick and fearless and holy man
I'll scream at the top of my capacity
In case you find I'm not the cause of the psalms
```

No Uber on the Highway

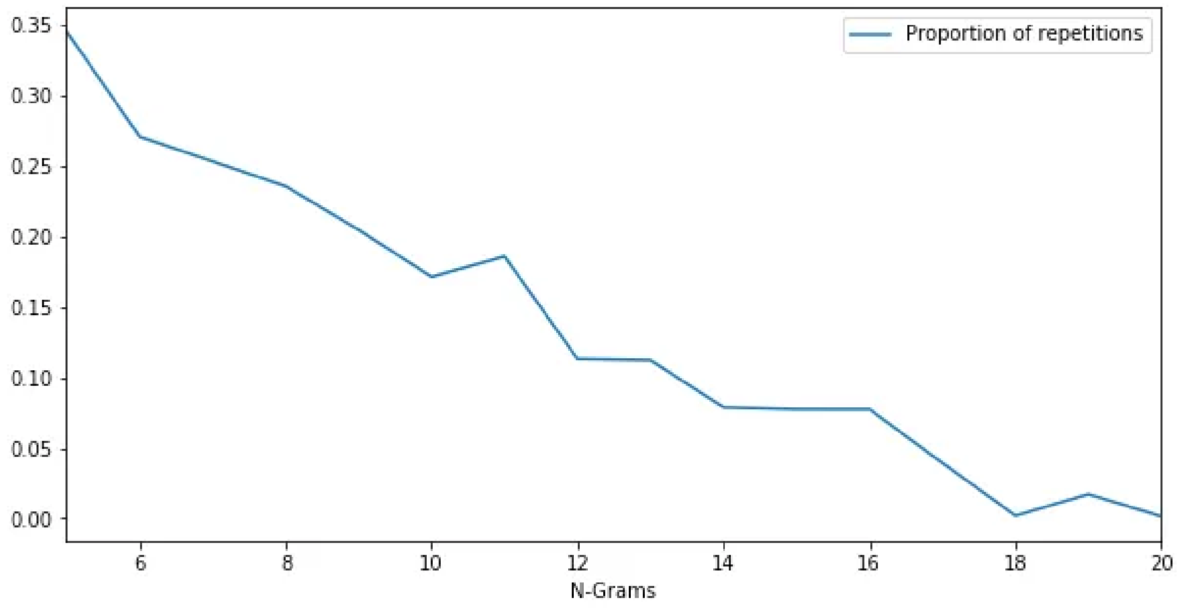
- *“Do you have the keys of the kingdom, yo ?”:* Starting with a “What’s my name?”, one could think that our artist was hyping up the crowd for a sick hip-hop solo. However, it quickly evolves into some intense questioning, under the suspicion that somebody might be hiding the keys of a certain kingdom. Then it just gets very melancholic and intense.



```
<|startoftext|>What's my name?

Go

Tell me
Why did you keep a secret
Why did you leave the key to the kingdom
Why did you keep a secret?

Why did you love me?
Why did you make me feel guilty?
Why did you try to ruin my love for ever

Why did you love me?
Why did you fail to hate me?
Why did you act so proud?
Why did you ignore me?
Why did you feel so proud?

Is it right?
You know it's time to let go
And if I, with my rage, only keep some
```

Do you have the keys of the kingdom, yo ?

Overall, the results look pretty good. It seems that our artist has some nice poetic inspiration. Yet, it is worth checking these masterpieces against plagiarism. That is, to see if our artist didn't just copied some songs (or part of songs) from the training data set. For that, we split each of its 100 songs in lists of sequential N-grams (i.e. N following words). We then check how many of these N sequential words can be found in the original data set. The following figure plots the result.



For instance, we checked and the 5-Gram "why did you love me?" (from the tube *Do you have the keys of the kingdom, yo?*) already exists as is in the training set. It is part of the ~35% of repeating 5-Grams. However, the rest of that track is an original work from our artist. As expected, we have less repetitions for larger values of N. In total it seems our artist has a fair amount of inspiration and originality. It is worth noting augmenting the temperature parameter when generating texts should decrease these values.

Another thing that surprised us is how dramatic the 100 songs can be. Just scrolling through them shows a lot of melancholy from our artist. Most likely, the data set just happens to contain more sad songs than happy ones. It might be worth filtering them with some sentiment analysis for training a happy singing bot. Some further fun work could also be to train hiphop/metal head/pop bots.

Finally, we trained several other GPT-2 models on different data sets. That is, we trained models to:

- Generate speeches from His Majesty King Philippe I of Belgium (in French and Dutch)
- Generate Antwerp Hip-Hop
- Generate Bol.com product descriptions

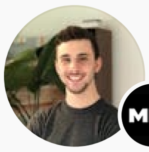
It led to interesting results. However, those datasets were all much smaller than the English songs data set described in this blog post. Hence, the texts generated were not as good.

So our final tip is to try to find a large and good quality database, then just have fun generating a lot of texts.

About ML6

We are a team of ML experts composing the fastest growing AI company in Belgium. With offices in Ghent, Amsterdam, Berlin and London, we implement self learning systems across different sectors to improve and optimise our clients operations. We do this by staying on top of research, innovation and practical applications. Find out more at www.ml6.eu

NLP



Follow

Written by Thomas Vrancken

60 Followers · Writer for ML6team

More from Thomas Vrancken and ML6team

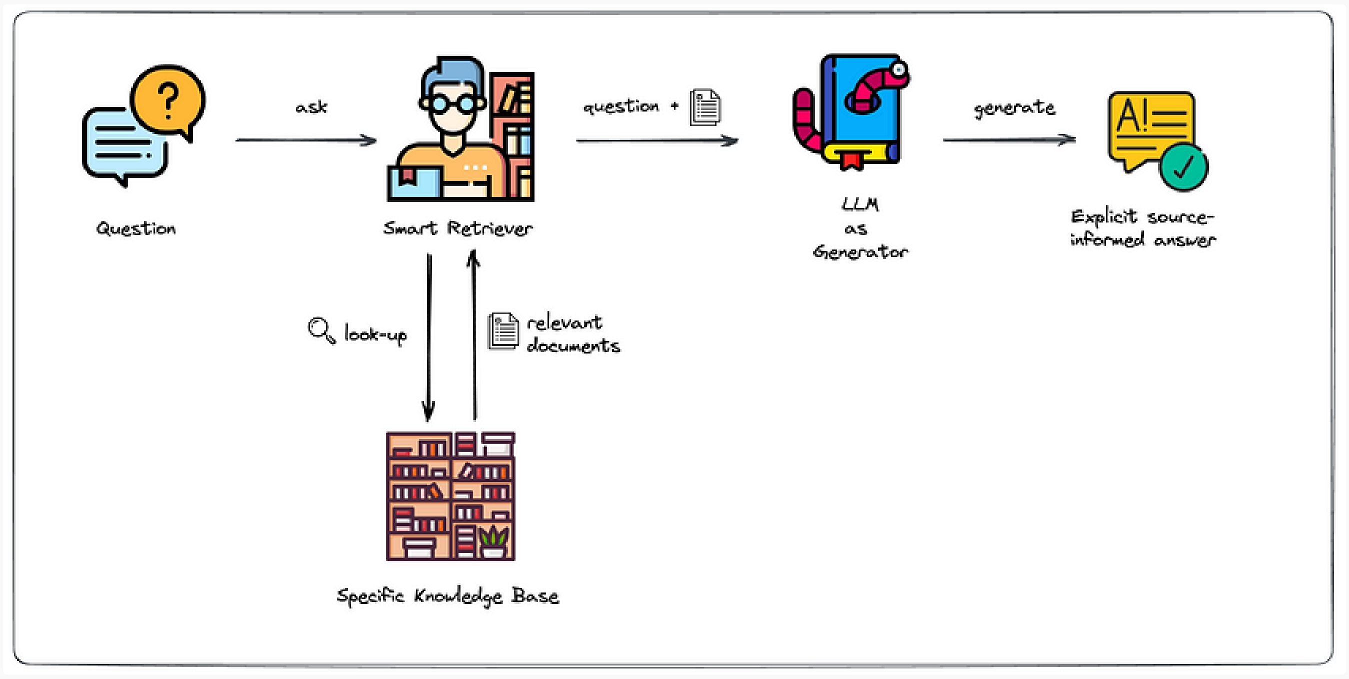
Thomas Vrancken in ML6team

NLP Trend: A few examples on how to process long documents

Machine Learning is at its most disruptive time, being more and more impactful in real world applications. This is especially the case in...

4 min read · Dec 6, 2021

124 1



Michiel De Koninck in ML6team

Leveraging LLMs on your domain-specific knowledge base

With RAG to Riches: wielding the power of LLMs using Retrieval-Augmented Generation to talk to your data

10 min read · May 8, 2023

403 4

Mathias Leys in ML6team

The Art of Pooling Embeddings 🎨

In this blogpost we will discover the complexity of pooling that hides behind its apparent simplicity.

8 min read · Jun 20, 2022

331 2

Thomas Vrancken in ML6team

NLP Meetup: Designing NLP for Production Systems at ML6

On the 3rd of December we had the pleasure to host the 12th edition of the Natural Language Processing (NLP) Meetup here at ML6, in Ghent.

2 min read · Dec 10, 2019

10

See all from Thomas Vrancken

See all from ML6team

Recommended from Medium

Rahul Nayak in Towards Data Science

How to Convert Any Text Into a Graph of Concepts

A method to convert any text corpus into a Knowledge Graph using Mistral 7B.

12 min read · Nov 10, 2023

6K 46

Benedict Neo in bitgrit Data Science Publication

Roadmap to Learn AI in 2024

A free curriculum for hackers and programmers to learn AI

11 min read · 4 days ago

2.3K 26

Lists

Natural Language Processing

1225 stories · 709 saves

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 315 saves

data science and AI

40 stories · 83 saves

Staff Picks

587 stories · 777 saves

Jerry Liu in LlamaIndex Blog

Introducing LlamaCloud and LlamaParse

Today is a big day for the LlamaIndex ecosystem: we are announcing LlamaCloud, a new generation of managed parsing, ingestion, and…

8 min read · 5 days ago

11

Artturi Jalli

I Built an App in 6 Hours that Makes $1,500/Mo

Copy my strategy!

✦ · 3 min read · Jan 23, 2024

137

Suman Das

Fine Tune Large Language Model (LLM) on a Custom Dataset with QLoRA

The field of natural language processing has been revolutionized by large language models (LLMs), which showcase advanced capabilities and...

15 min read · Jan 25, 2024

7

Amnah Ebrahim in GoPenAI

Build Your Own Chatbot with Rasa Open Source

A Gentle Introduction

7 min read · Oct 27, 2023

1

See more recommendations