Sakka Mohamad-Mario 1241EB – Subgroup II

SOFTWARE ENGINEERING HW 1

1. SETUP & RUNNING

I have used models which are similar in size to gpt2:

- ***distilgpt2: 82M parameters***
- ***EleutherAI/gpt-neo-125M: 125M parameters***
- ***facebook/opt-125m: 125M parameters***

Then I created sets of parameters to test each of them with the 3 models:

- ***temperature=0.3 ; top_p=0.9 ; top_k=50 ; repetition_penalty=1.5***
- ***temperature=0.7 ; top_p=0.9 ; top_k=50 ; repetition_penalty=1.5***
- ***temperature=1.0 ; top_p=0.9 ; top_k=50 ; repetition_penalty=1.5***

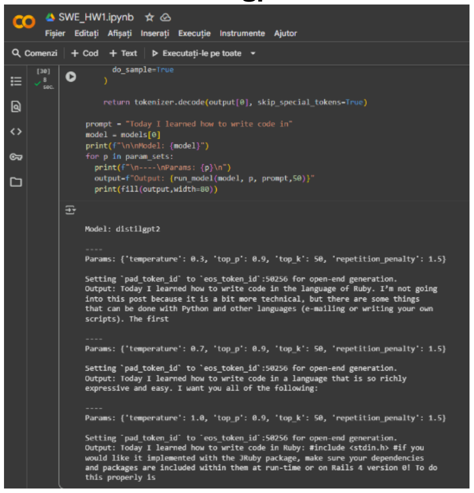
The prompt used:

- ***prompt = "Today I learned how to solve differential equations"***

I ran the models through google collab on a personal notebook. The code selects one of the models in the list and loops through the sets of parameters with the same input prompt on each run.

2. RESULTS

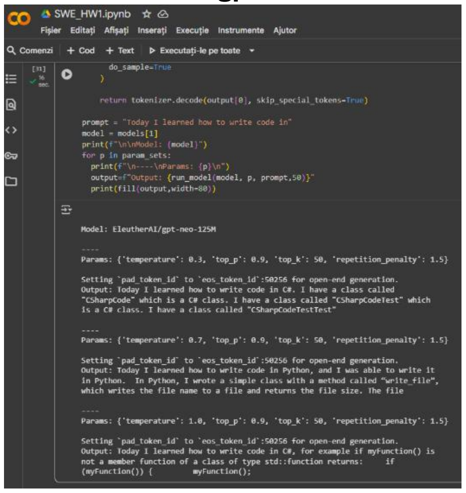
distilgpt2


SWE_HW1.ipynb
Fișier Editați Afișați Inserați Execuție Instrumente Ajutor
Comenzi + Cod + Text Executați-le pe toate

```
        do_sample=True
    )

    return tokenizer.decode(output[0], skip_special_tokens=True)

prompt = "Today I learned how to write code in"
model = models[0]
print(f"\n\nModel: {model}")
for p in param_sets:
  print(f"\n----\nParams: {p}\n")
  output=f"Output: {run_model(model, p, prompt,50)}"
  print(fill(output,width=80))
```

```
Model: distilgpt2

----
Params: {'temperature': 0.3, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Output: Today I learned how to write code in the language of Ruby. I'm not going
into this post because it is a bit more technical, but there are some things
that can be done with Python and other languages (e-mailing or writing your own
scripts). The first

----
Params: {'temperature': 0.7, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Output: Today I learned how to write code in a language that is so richly
expressive and easy. I want you all of the following:

----
Params: {'temperature': 1.0, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Output: Today I learned how to write code in Ruby: #include <stdin.h> #if you
would like it implemented with the JRuby package, make sure your dependencies
and packages are included within them at run-time or on Rails 4 version 0! To do
this properly is
```

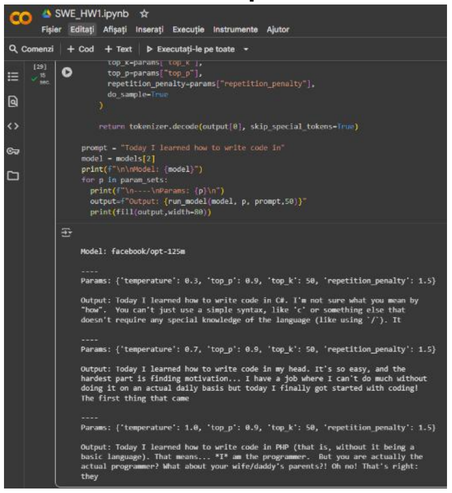
EleutherAI/gpt-neo-125M


SWE_HW1.ipynb
Fișier Editați Afișați Inserați Execuție Instrumente Ajutor
Comenzi + Cod + Text Executați-le pe toate

```
        do_sample=True
    )

    return tokenizer.decode(output[0], skip_special_tokens=True)

prompt = "Today I learned how to write code in"
model = models[1]
print(f"\n\nModel: {model}")
for p in param_sets:
  print(f"\n----\nParams: {p}\n")
  output=f"Output: {run_model(model, p, prompt,50)}"
  print(fill(output,width=80))
```

```
Model: EleutherAI/gpt-neo-125M

----
Params: {'temperature': 0.3, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Output: Today I learned how to write code in C#. I have a class called
"CSharpCode" which is a C# class. I have a class called "CSharpCodeTest" which
is a C# class. I have a class called "CSharpCodeTestTest"

----
Params: {'temperature': 0.7, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Output: Today I learned how to write code in Python, and I was able to write it
in Python.  In Python, I wrote a simple class with a method called "write_file",
which writes the file name to a file and returns the file size. The file

----
Params: {'temperature': 1.0, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Output: Today I learned how to write code in C#, for example if myFunction() is
not a member function of a class of type std::function returns:     if
(myFunction()) {         myFunction();
```

facebook/opt-125m


SWE_HW1.ipynb
Fișier Editați Afișați Inserați Execuție Instrumente Ajutor
Comenzi + Cod + Text Executați-le pe toate

```
        top_p=params["top_p"],
        repetition_penalty=params["repetition_penalty"],
        do_sample=True
    )

    return tokenizer.decode(output[0], skip_special_tokens=True)

prompt = "Today I learned how to write code in"
model = models[2]
print(f"\n\nModel: {model}")
for p in param_sets:
  print(f"\n----\nParams: {p}\n")
  output=f"Output: {run_model(model, p, prompt,50)}"
  print(fill(output,width=80))
```

```
Model: facebook/opt-125m

----
Params: {'temperature': 0.3, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Output: Today I learned how to write code in C#. I'm not sure what you mean by
"how".  You can't just use a simple syntax, like 'c' or something else that
doesn't require any special knowledge of the language (like using `/`). It

----
Params: {'temperature': 0.7, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Output: Today I learned how to write code in my head. It's so easy, and the
hardest part is finding motivation... I have a job where I can't do much without
doing it on an actual daily basis but today I finally got started with coding!
The first thing that came

----
Params: {'temperature': 1.0, 'top_p': 0.9, 'top_k': 50, 'repetition_penalty': 1.5}

Output: Today I learned how to write code in PHP (that is, without it being a
basic language). That means... *I* am the programmer.  But you are actually the
actual programmer? What about your wife/daddy's parents?! Oh no! That's right:
they
```

3. INTERPRETATIONS OF RESULTS

Across all three small models, I saw a pretty consistent pattern with temperature. **distilgpt2** was the steadiest: at **T=0.3** it didn't stick to the language it mentioned first, **T=0.7** gave a generic sentence without specifying the language, and **T=1.0** started to ramble. **EleutherAI/gpt-neo-125M** felt more "codey"—**0.3** but it was repetitive, **0.7** produced the best mix of concrete details (e.g., plausible class/function names) and coherence, and **1.0** often drifted into half-finished snippets. **facebook/opt-125m** had a chatty vibe; **0.3** read like forum comments, **0.7** was the most useful with step-like guidance, and **1.0** became disjoint quickly. **Overall, the best temperature across models was ~0.7**, which gave the strongest balance of specificity and coherence without the chaos I saw at 1.0 or the repetitiveness and incoherence at 0.3.