TextSynthPy

Karol Janus

May 10, 2022

PYTHON API:

CHAPTER

ONE

TEXTSYNTH

class textsynthpy.**TextSynth**(*key: str*, *engine: Optional[str] = None*)

An connector for textsynth.com

Parameters

- **key** (*str*) – Textsynth API key. You need textsynth.com account to get this.
- **engine** – Textsynth engine to use. It is optional parameter, in default uses "gptj_6B". Aviable engines are on 'https://textsynth.com/documentation.html'
- **engine** – str

static engines(*as_dict: bool = False*)

Downloads from github page current list of engines with small descriprion and prints it OR returns as a list

Parameters **as_dict** (*bool*) – if True returns a dict, else prints engines on screen

log_prob(*context: str = ''*)

Returns Log object: This endpoint returns the logarithm of the probability that a continuation is generated after a context. It can be used to answer questions when only a few answers (such as yes/no) are possible. It can also be used to benchmark the models.

Parameters

- **continuation** (*str*) – Must be a non empty string.
- **context** (*str*) – If empty string, the context is set to the End-Of-Text token.

text_complete(*prompt: str*, *max_tokens: int = 100*, *temperature: float = 1*, *top_k: int = 40*, *top_p: float = 0.9*, *stream: bool = False*, *stop: Optional[str] = None*, *logit_bias: dict = {}*, *presence_penalty: int = 0*, *frequency_penalty: int = 0*)

Returns Complete object of completed text by given prompt.

Parameters

- **prompt** (*str*) – The input text to complete.
- **max_tokens** (*int*) – Optional (Default = 100). Maximum number of tokens to generate. A token represents about 4 characters for English texts. The total number of tokens (prompt + generated text) cannot exceed the model's maximum context length. It is of 2048 for GPT-J and 1024 for the other models. If the prompt length is larger than the model's maximum context length, the beginning of the prompt is discarded.
- **temperature** (*int*) – Optional (Default = 1). Sampling temperature. A higher temperature means the model will select less common tokens leading to a larger diversity but potentially less relevant output. It is usually better to tune top_p or top_k.

- **top_k** (*int*) – optional (Range: 1 to 1000, Default = 40). Select the next output token among the top_k most likely ones. A higher top_k gives more diversity but a potentially less relevant output.
- **top_p** (*float*) – optional (Range: 0 to 1, Default = 0.9). Select the next output token among the most probable ones so that their cumulative probability is larger than top_p. A higher top_p gives more diversity, but a potentially less relevant output.
- **stream** (*bool*) – Optional (Default = false). If true, the output is streamed so that it is possible to display the result before the complete output is generated. Several JSON answers are output, wrapper returns list of Complete objects
- **stop** (*str*) – Optional (Default = None). Stop the generation when the string(s) are encountered. The generated text does not contain the string. stream must be set to false when this feature is used. The length of the array is at most 5.
- **logit_bias** (*dict*) – Optional (Default = {}). Modify the likelihood of the specified tokens in the completion. The specified object is a map between the token indexes and the corresponding logit bias. A negative bias reduces the likelihood of the corresponding token. The bias must be between -100 and 100. Note that the token indexes are specific to the selected model. You can use tokenize() to retrieve the token indexes of a given model. Example - if you want to ban the " unicorn" token for GPT-J, you can use: { "44986": -100 }
- **presence_penalty** (*int*) – Optional(Range: -2 to 2, Default = 0). A positive value penalizes tokens which already appeared in the generated text. Hence it forces the model to have a more diverse output.
- **frequency_penalty** (*int*) – Optional number (Range: -2 to 2, Default = 0). A positive value penalizes tokens which already appeared in the generated text proportionaly to their frequency. Hence it forces the model to have a more diverse output.

tokenize(*text: str*)

Method returns array of integers: token indexes corresponding to a given text. It is useful for example to know the exact number of tokens of a text or to specify logit biases with the completion endpoint. The tokens are specific to a given model. Note: using tokenize endpoint is free.

Parameters **text** (*str*) – string

CHAPTER

TWO

COMPLETE

class textsynthpy.**Complete**(*text: str*, *reached_end: Optional[bool] = None*, *truncated_prompt: Optional[bool] = None*, *input_tokens: Optional[int] = None*, *output_tokens: Optional[int] = None*)

Object handles response form TextSynth.text_complete().

Parameters

- **text** (*str*) – It is the completed text
- **reached_end** (*bool*) – If true, indicate that it is the last answer. It is only useful in case of streaming output (stream = true in the request).
- **truncated_prompt** (*bool*) – If true, indicate that the prompt was truncated because it was too large compared to the model's maximum context length. Only the end of the prompt is used to generate the completion.
- **input_tokens** (*int*) – Indicate the number of input tokens. It is useful to estimate the number of compute resources used by the request.
- **output_tokens** (*int*) – Indicate the total number of generated tokens. It is useful to estimate the number of compute resources used by the request.

CHAPTER

THREE

LOG

class textsynthpy.**Log**(*logprob: float*, *is_greedy: bool*, *input_tokens: int*)

Object handles response form TextSynth.log_prob()

Parameters

- **logprob** (*float*) – Logarithm of the probability of generation of continuation preceeded by context. It is always <= 0.
- **is_greedy** (*bool*) – true if continuation would be generated by greedy sampling from continuation.
- **input_tokens** (*int*) – Indicate the total number of input tokens. It is useful to estimate the number of compute resources used by the request.

Usage =====

.. _installation:

Installation ————

To use TextSynthPy, first install it using pip:

.. code-block:: console

(.venv) $ pip install textsynthpy

Basic generation ————————-

Create a connector: ```con = textsynthpy.TextSynth("api_key")```

.. autofunction:: textsynthpy.TextSynth

The ``key`` parameter must be an api key received at. Otherwise, textsynth won't let you pass.

Generate text by using `text_complete()` method:

```
>>> answer = con.text_complete("prompt")
```

`prompt` parameter must be a string, otherwise will receice an exception.

Generated text is under `text` atrribute: `answer.text`

For example:

>>> import textsynthpy >>> con = textsynthpy.TextSynth("api_key") >>> print(answer.text)

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

INDEX