# novelai-api

Python API for the NovelAI REST API

This module is intended to be used by developers as a helper for using NovelAI's REST API.

| Category | Badges |
|---|---|
| Pypi | pypi v0.30.1 · python 3.8 \| 3.9 \| 3.10 \| 3.11 \| 3.12 · license MIT · format wheel |
| Quality checking | Python package passing · CodeQL passing · linting pylint · security bandit · code style black |
| Stats | python 99.8% · dependencies 2 out of date · repo size 6.11 MB · package size 3.84MB · open issues 2 · open pull requests 0 |
| Activity | last commit september · commits since v0.28.1 6 · commit activity 4/month |

Retired versions: 3.7.2 Final commit of retired versions can be found with the tag `py<version>` (e.g. `py3.7.2`).

# Usage

Download via pip:

```
pip install novelai-api
```

## Using the module via Command Line

### Get access key

Get the access key for your account. This key is used to login to the API through the /login endpoint.

```
python -m novelai_api get_access_key <username> <password>
```

### Get access token

Login to the API and get the access token. This token is valid 30 days and is required to use most of the API.

```
python -m novelai_api get_access_token <username> <password>
```

### Sanity check

Run a sanity check on your user content. It will print what content couldn't be decrypted.

```
python -m novelai_api sanity_check <username> <password>
```

### Decode

Decode a b64 encoded tokenized text. This will print the tokens and the decoded text.

```
python -m novelai_api decode <model> <data>
```

## Using the module in your code

A full list of examples is available in the example directory

The API works through the NovelAIAPI object. It is split in 2 groups: NovelAIAPI.low_level and NovelAIAPI.high_level

### low_level

The low level interface is a strict implementation of the official API (https://api.novelai.net/docs). It only checks for input types via assert, and output schema if NovelAIAPI.low_level.is_schema_validation_enabled is True

### high_level

The high level interface builds on the low level one for easier handling of complex settings. It handles many tasks from the frontend

# Development

All relevant objects are in the novelai_api directory. The Poetry package is required (`pip install poetry`) as the venv manager.

## Contributing

You can contribute features and enhancements through PR. Any PR should pass the tests and the pre-commits before submission. The pre-commit hook can be installed via

```
poetry run nai-pre-commit
```

## Testing against the API

To run against the API, you can use `poetry run nai-test-api`.

API

## Testing against the mocked API

To run against the mocked API, you can use `poetry run nai-test-mock`.

**:warning: WIP, does not work yet :warning:**

Mock

## Docs

To build the docs, run

```
poetry run nai-build-docs
```

The docs will be locally viewable at docs/build/html/index.html

# TODO

> Trello

# Reference

## novelai-api

- novelai-api package
  - novelai_api.low_level
  - novelai_api.high_level
  - novelai_api.BanList
  - novelai_api.BiasGroup
  - novelai_api.GlobalSettings
  - novelai_api.Idstore
  - novelai_api.ImagePreset
  - novelai_api.Keystore
  - novelai_api.NovelAIError
  - novelai_api.NovelAI_API
  - novelai_api.Preset
  - novelai_api.SchemaValidator
  - novelai_api.StoryHandler
  - novelai_api.Tokenizer
  - novelai_api.utils

## example

- example directory
  - Requirements
  - Usage
  - Content
  - Reference

## Test API

- API directory
  - Requirements

## Test Mocked API

# novelai_api.high_level ¶

*class* **HighLevel** [source]

Bases: `object`

High level API for NovelAI. This class is not meant to be used directly, but rather through `NovelAIAPI.high_level`.

The most relevant methods are:

- `login`
- `generate`
- `generate_image`

**__init__**(*parent: NovelAIAPI*) [source]

*async* **register**(*recapcha: str, email: str, password: str, send_mail: bool = True, giftkey: Optional[str] = None*) → bool [source]

Register a new account

**Parameters:**
- **recapcha** – Recapcha of the NovelAI website
- **email** – Email of the account (username)
- **password** – Password of the account
- **send_mail** – Send the mail (hashed and used for recovery)
- **giftkey** – Giftkey

**Returns:** True if success

*async* **login**(*email: str, password: str*) → str [source]

Log in to the account

**Parameters:**
- **email** – Email of the account (username)
- **password** – Password of the account

**Returns:** User's access token

*async* **login_with_token**(*access_token: str*) [source]

Log in with the access token, instead of email and password

**Parameters:** **access_token** – Access token of the account (persistent token or gotten from login)

*async* **login_from_key**(*access_key: str*) [source]

Log in with the access key, instead of email and password

**Parameters:** **access_key** – Access key of the account (pre-computed via email and password)

**Returns:** User's access token

*async* **get_keystore**(*key: bytes*) → `novelai_api.Keystore.Keystore` [source]

Retrieve the keystore and decrypt it in a readable manner.

The keystore is the mapping of meta -> encryption key of each object. If this function throws errors repeatedly at you, check your internet connection or the integrity of your keystore. Losing your keystore, or overwriting it means losing all content on the account.

**Parameters:** **key** – Account's encryption key
**Returns:** Keystore object

*async* **set_keystore**(*keystore: novelai_api.Keystore.Keystore, key: bytes*) → bytes [source]

Encrypt and upload the keystore.

The keystore is the mapping of meta -> encryption key of each object. If this function throws errors repeatedly at you, check your internet connection or the integrity of your keystore. Losing your keystore, or overwriting it means losing all content on the account.

**Parameters:**
- **keystore** – Keystore object to upload
- **key** – Account's encryption key

**Returns:** raw data of the serialized Keystore object

*async* **download_user_stories**() → List[Dict[str, Dict[str, Union[str, int]]]] [source]

Download all the objects of type 'stories' stored on the account

*async* **download_user_story_contents**() → List[Dict[str, Dict[str, Union[str, int]]]] [source]

Download all the objects of type 'storycontent' stored on the account

*async* **download_user_presets**() → List[Dict[str, Union[str, int]]] [source]

Download all the objects of type 'presets' stored on the account

*async* **download_user_modules**() → List[Dict[str, Union[str, int]]] [source]

Download all the objects of type 'aimodules' stored on the account

*async* **download_user_shelves**() → List[Dict[str, Union[str, int]]] [source]

Download all the objects of type 'shelf' stored on the account

*async* **upload_user_content**(*data: Dict[str, Any], encrypt: bool = False, keystore: Optional[novelai_api.Keystore.Keystore] = None*) → bool [source]

Upload user content

**Parameters:**
- **data** – Object to upload
- **encrypt** – Re-encrypt/re-compress the data, if True
- **keystore** – Keystore to encrypt the data, if encrypt is True

**Returns:** True if the upload succeeded, False otherwise

*async* **upload_user_contents**(*datas: Iterable[Dict[str, Any]], encrypt: bool = False, keystore: Optional[novelai_api.Keystore.Keystore] = None*) →

List[Tuple[str, Optional[novelai_api.NovelAIError.NovelAIError]]][source]

Upload multiple user contents. If the content has been decrypted with decrypt_user_data, it should be re-encrypted with encrypt_user_data, even if the decryption failed

**Parameters:**
- **datas** – Objects to upload
- **encrypt** – Re-encrypt/re-compress the data, if True
- **keystore** – Keystore to encrypt the data, if encrypt is True

**Returns:** A list of (id, error) of all the objects that failed to be uploaded

*async* **generate**(*prompt: Union[List[int], str], model: novelai_api.Preset.Model, preset: novelai_api.Preset.Preset, global_settings: novelai_api.GlobalSettings.GlobalSettings, bad_words: Optional[Union[Iterable[novelai_api.BanList.BanList], novelai_api.BanList.BanList]] = None, biases: Optional[Union[Iterable[novelai_api.BiasGroup.BiasGroup], novelai_api.BiasGroup.BiasGroup]] = None, prefix: Optional[str] = None, stop_sequences: Optional[Union[List[int], str]] = None, **kwargs*) → Dict[str, Any] [source]

Generate text. The b64-encoded text is returned at once, when generation is finished. To decode the text, the `novelai_api.utils.b64_to_tokens()` and `novelai_api.Tokenizer.Tokenizer.decode()` methods should be used.

As the model accepts a complete prompt, the context building must be done before calling this function. Any content going beyond the tokens limit will be truncated, starting from the top.

**Parameters:**
- **prompt** – Context to give to the AI (raw text or list of tokens)
- **model** – Model to use for the AI
- **preset** – Preset to use for the generation settings
- **global_settings** – Global settings (used for generation)
- **bad_words** – Tokens to ban for this generation
- **biases** – Tokens to bias (up or down) for this generation
- **prefix** – Module to use for this generation (see list of modules)
- **stop_sequences** – List of strings or tokens to stop the generation at
- **kwargs** – Additional parameters to pass to the requests. Can also be used to overwrite existing parameters

**Returns:** Content that has been generated

*async* **generate_stream**(*prompt: Union[List[int], str], model: novelai_api.Preset.Model, preset: novelai_api.Preset.Preset, global_settings: novelai_api.GlobalSettings.GlobalSettings, bad_words: Optional[Union[Iterable[novelai_api.BanList.BanList], novelai_api.BanList.BanList]] = None, biases: Optional[Union[Iterable[novelai_api.BiasGroup.BiasGroup], novelai_api.BiasGroup.BiasGroup]] = None, prefix: Optional[str] = None, stop_sequences: Optional[Union[List[int], str]] = None, **kwargs*) → AsyncIterable[Dict[str, Any]] [source]

Generate text. The text is returned one token at a time, as it is generated.

As the model accepts a complete prompt, the context building must be done before calling this function. Any content going beyond the tokens limit will be truncated, starting from the top.

**Parameters:**
- **prompt** – Context to give to the AI (raw text or list of tokens)
- **model** – Model to use for the AI
- **preset** – Preset to use for the generation settings
- **global_settings** – Global settings (used for generation)
- **bad_words** – Tokens to ban for this generation
- **biases** – Tokens to bias (up or down) for this generation
- **prefix** – Module to use for this generation (see list of modules)
- **stop_sequences** – List of strings or tokens to stop the generation at
- **kwargs** – Additional parameters to pass to the requests. Can also be used to overwrite existing parameters

**Returns:** Content that has been generated

*async* **generate_image**(*prompt: str, model: novelai_api.ImagePreset.ImageModel, preset: novelai_api.ImagePreset.ImagePreset, action: novelai_api.ImagePreset.ImageGenerationType = ImageGenerationType.NORMAL, **kwargs*) → AsyncIterable[Union[str, bytes]] [source]

Generate one or multiple image(s)

**Parameters:**
- **prompt** – Prompt to give to the AI (raw text describing the wanted image)
- **model** – Model to use for the AI
- **preset** – Preset to use for the generation settings
- **action** – Type of image generation to use
- **kwargs** – Additional parameters to pass to the requests. Can also be used to overwrite existing parameters

**Returns:** Pair(s) (name, image) that have been generated

## List of modules

Calliope, Snek, and Genji have no module support. “special_openings” is a module specifically trained to replace the previously used preamble. It is used at the beginning of the story, under certain conditions.

| Name | API id | Model |
|---|---|---|
| No module | vanilla | All |
| Special: Text Adventure | theme_textadventure | All |
| General: Prose Augmenter | special_proseaugmenter | Sigurd, Euterpe, Clio, Kayra |
| General: Instruct (Experimental) | special_instruct | Clio, Kayra |
| - | special_openings | Clio, Kayra |
| General: Cross-Genre | general_crossgenre | Sigurd, Euterpe, Krake |

| Name | API id | Model |
|---|---|---|
| Style: Algernon Blackwood | style_algernonblackwood | Sigurd, Euterpe, Krake |
| Style: Arthur Conan Doyle | style_arthurconandoyle | Sigurd, Euterpe, Krake |
| Style: Edgar Allan Poe | style_edgarallanpoe | Sigurd, Euterpe, Krake |
| Style: H.P. Lovecraft | style_hplovecraft | Sigurd, Euterpe, Krake |
| Style: Sheridan Le Fanu | style_shridanlefanu | Sigurd, Euterpe, Krake |
| Style: Jane Austen | style_janeausten | Sigurd, Euterpe |
| Style: Jules Verne | style_julesverne | Sigurd, Euterpe, Krake |
| Style: William Shakespeare | style_williamshakespeare | Sigurd, Euterpe |
| Theme: 19th Century Romance | theme_19thcenturyromance | Sigurd, Euterpe, Krake |
| Theme: Action Archeology | theme_actionarcheology | Sigurd, Euterpe, Krake |
| Theme: Artificial Intelligence | theme_ai | Sigurd, Euterpe, Krake |
| Theme: Ancient China | theme_ancientchina | Sigurd, Euterpe |
| Theme: Ancient Greece | theme_ancientgreek | Sigurd, Euterpe |
| Theme: Ancient India | theme_india | Sigurd, Euterpe |
| Theme: Animal Fiction | theme_animalfiction | Sigurd, Euterpe, Krake |
| Theme: Anthropomorphic Animals | theme_anthropomorphicanimals | Sigurd, Euterpe |
| Theme: Children’s Fiction | theme_childrens | Sigurd, Euterpe, Krake |
| Theme: Christmas | theme_christmas | Sigurd, Euterpe, Krake |
| Theme: Comedic Fantasy | theme_comedicfantasy | Sigurd, Euterpe, Krake |
| Theme: Contemporary | theme_contemporary | Sigurd, Euterpe |
| Theme: Cyberpunk | theme_cyberpunk | Sigurd, Euterpe, Krake |
| Theme: Dark Fantasy | theme_darkfantasy | Sigurd, Euterpe, Krake |
| Theme: Dragons | theme_dragons | Sigurd, Euterpe, Krake |
| Theme: Egypt | theme_egypt | Sigurd, Euterpe, Krake |
| Theme: Feudal Japan | theme_feudaljapan | Sigurd, Euterpe, Krake |
| Theme: Gaming | theme_gaming | Sigurd, Euterpe |

| Name | API id | Model |
|---|---|---|
| Theme: General Fantasy | theme_generalfantasy | Sigurd, Euterpe, Krake |
| Theme: Golden Age Scifi | theme_goldenagescifi | Sigurd, Euterpe |
| Theme: Hard SF | theme_hardsf | Sigurd, Euterpe |
| Theme: History | theme_history | Sigurd, Euterpe, Krake |
| Theme: Horror | theme_horror | Sigurd, Euterpe, Krake |
| Theme: Hunter Gatherer | theme_huntergatherer | Sigurd, Euterpe, Krake |
| Theme: LitRPG | theme_litrpg | Sigurd, Euterpe, Krake |
| Theme: Magic Academy | theme_magicacademy | Sigurd, Euterpe, Krake |
| Theme: Magic Library | theme_libraries | Sigurd, Euterpe, Krake |
| Theme: Light Novels | theme_lightnovels | Sigurd, Euterpe |
| Theme: Mars Colonization | theme_mars | Sigurd, Euterpe, Krake |
| Theme: Medieval | theme_medieval | Sigurd, Euterpe, Krake |
| Theme: Military SciFi | theme_militaryscifi | Sigurd, Euterpe, Krake |
| Theme: Music | theme_music | Sigurd, Euterpe |
| Theme: Mystery | theme_mystery | Sigurd, Euterpe, Krake |
| Theme: Nature | theme_nature | Sigurd, Euterpe |
| Theme: Naval Age of Discovery | theme_naval | Sigurd, Euterpe, Krake |
| Theme: Noir | theme_noir | Sigurd, Euterpe |
| Theme: Philosophy | theme_philosophy | Sigurd, Euterpe, Krake |
| Theme: Pirates | theme_pirates | Sigurd, Euterpe, Krake |
| Theme: Poetic Fantasy | theme_poeticfantasy | Sigurd, Euterpe, Krake |
| Theme: Post-Apocalyptic | theme_postapocalyptic | Sigurd, Euterpe, Krake |
| Theme: Rats | theme_rats | Sigurd, Euterpe, Krake |
| Theme: Roman Empire | theme_romanempire | Sigurd, Euterpe, Krake |
| Theme: Science Fantasy | theme_sciencefantasy | Sigurd, Euterpe, Krake |

| Name | API id | Model |
| --- | --- | --- |
| Theme: Space Opera | theme_spaceopera | Sigurd, Euterpe, Krake |
| Theme: Superheroes | theme_superheroes | Sigurd, Euterpe, Krake |
| Theme: Steampunk | theme_airships | Sigurd, Euterpe, Krake |
| Theme: Travel | theme_travel | Sigurd, Euterpe, Krake |
| Theme: Urban Fantasy | theme_urbanfantasy | Sigurd, Euterpe, Krake |
| Theme: Valentines | theme_valentines | Sigurd, Euterpe, Krake |
| Theme: Vikings | theme_vikings | Sigurd, Euterpe, Krake |
| Theme: Weird West | theme_weirdwest | Sigurd, Euterpe |
| Theme: Western Romance | theme_westernromance | Sigurd, Euterpe, Krake |
| Inspiration: Crab, Snail, and Monkey | inspiration_crabsnailandmonkey | Sigurd, Euterpe, Krake |
| Inspiration: Mercantile Wolfgirl Romance | inspiration_mercantilewolfgirlromance | Sigurd, Euterpe, Krake |
| Inspiration: Nervegear | inspiration_nervegear | Sigurd, Euterpe, Krake |
| Inspiration: Romance of the Three Kingdoms | theme_romanceofthreekingdoms | Euterpe, Krake |
| Inspiration: Throne Wars | inspiration_thronewars | Sigurd, Euterpe, Krake |
| Inspiration: Witch at Level Cap | inspiration_witchatlevelcap | Sigurd, Euterpe, Krake |

# novelai_api.BanList

*class* **BanList** [source]

Bases: `object`

**__init__**(**sequences: Union[List[int], str], enabled: bool = True*) [source]

Create a ban list with the given elements. Elements can be string or tokenized strings Using tokenized strings is not recommended, for flexibility between tokenizers

**Parameters:** **enabled** – Is the ban list enabled

**enabled**: *bool*

**add**(**sequences: Union[Dict[str, List[List[int]]], Dict[str, List[int]], List[int], str]*) → novelai_api.BanList.BanList [source]

Add elements to the ban list. Elements can be string or tokenized strings Using tokenized strings is not recommended, for flexibility between tokenizers

**get_tokenized_entries**(*model: novelai_api.Preset.Model*) → Iterable[List[int]] [source]

Return the tokenized sequences for the ban list, if it is enabled

**Parameters:** **model** – Model to use for tokenization

# novelai_api.BiasGroup

*class* **BiasGroup** [source]

Bases: `object`

**__init__**(*bias: float, ensure_sequence_finish: bool = False, generate_once: bool = False, enabled: bool = True*) [source]

Create a bias group

**Parameters:**
- **bias** – Bias value of the bias group. Negative is a downbias, positive is an upbias
- **ensure_sequence_finish** – Ensures the bias completes
- **generate_once** – Only biases for the first occurrence
- **enabled** – Is the bias group enabled

**bias**: *float*

**ensure_sequence_finish**: *bool*

**generate_once**: *bool*

**enabled**: *bool*

*classmethod* **from_data**(*data: Dict[str, Any]*) → novelai_api.BiasGroup.BiasGroup [source]

Create a bias group from bias group data

**add**(**sequences: Union[Dict[str, List[List[int]]], Dict[str, List[int]], List[int], str]*) → novelai_api.BiasGroup.BiasGroup [source]

Add elements to the bias group. Elements can be string or tokenized strings Using tokenized strings is not recommended, for flexibility between tokenizers

**get_tokenized_entries**(*model: novelai_api.Preset.Model*) → Iterable[Dict[str, any]] [source]

Return the tokenized sequences for the bias group, if it is enabled

**Parameters:** **model** – Model to use for tokenization

# novelai_api.GlobalSettings

*class* **GlobalSettings** [source]

Bases: `object`

Object used to store global settings for the account

**generate_until_sentence**: *bool*

Generate up to 20 tokens after max_length if an end of sentence if found within these 20 tokens

**num_logprobs**: *int*

Number of logprobs to return for each token. Set to NO_LOGPROBS to disable

**ban_brackets**: *bool*

Apply the BRACKET biases

**bias_dinkus_asterism**: *bool*

Apply the DINKUS_ASTERISM biases

**ban_ambiguous_genji_tokens**: *bool*

Apply the GENJI_AMBIGUOUS_TOKENS if model is Genji

**rep_pen_whitelist**: *bool*

Apply the REP_PEN_WHITELIST (repetition penalty whitelist)

**NO_LOGPROBS** = *-1*

Value to set num_logprobs at to disable logprobs

**__init__**(*, *generate_until_sentence: bool, num_logprobs: int, ban_brackets: bool, bias_dinkus_asterism: bool, ban_ambiguous_genji_tokens: bool, rep_pen_whitelist: bool*) [source]

**copy**() [source]

Create a new GlobalSettings from the current

**to_settings**(*model: novelai_api.Preset.Model*) → `Dict[str, Any]` [source]

Create text generation settings from the GlobalSettings object

**Parameters:** **model** – Model to use the settings of

# novelai_api.Preset

*class* **Order** [source]

Bases: `enum.IntEnum`

**Temperature** = *0*

**Top_K** = *1*

**Top_P** = *2*

**TFS** = *3*

**Top_A** = *4*

**Typical_P** = *5*

**CFG** = *6*

**Top_G** = *7*

**Mirostat** = *8*

**__new__**(*value*)

**enum_contains**(*enum_class: enum.EnumType, value: str*) → bool [source]

Check if the value provided is valid for the enum

**Parameters:**
- **enum_class** – Class of the Enum
- **value** – Value to check

**collapse_model**(*enum_class: enum.EnumType, value: str*) [source]

Collapse multiple version of a model to the last model value

**Parameters:**
- **enum_class** – Class of the Enum
- **value** – Value of the model to collapse

*class* **StrEnum** [source]

Bases: `str`, `enum.Enum`

**__new__**(*value*) [source]

*class* **Model** [source]

Bases: `novelai_api.Preset.StrEnum`

**Sigurd** = *'6B-v4'*

**Euterpe** = *'euterpe-v2'*

**Krake** = *'krake-v2'*

**Clio** = *'clio-v1'*

**Kayra** = *'kayra-v1'*

**Genji** = *'genji-jp-6b-v2'*

**Snek** = *'genji-python-6b'*

**HypeBot** = *'hypebot'*

**Inline** = *'infillmodel'*

**enum_member_values** = *{'6B': Model.Sigurd, 'clio': Model.Clio, 'euterpe': Model.Euterpe, 'genji-jp-6b': Model.Genji, 'genji-python-6b': Model.Snek, 'hypebot': Model.HypeBot, 'infillmodel': Model.Inline, 'kayra': Model.Kayra, 'krake': Model.Krake}*

**__new__**(*value*)

*class* **PhraseRepPen** [source]

Bases: `novelai_api.Preset.StrEnum`

**Off** = *'off'*

**VeryLight** = *'very_light'*

**Light** = *'light'*

**Medium** = *'medium'*

**Aggressive** = *'aggressive'*

**VeryAggressive** = *'very_aggressive'*

**__new__**(*value*)

**PREAMBLE** = *{Model.Sigurd: '⁂\n', Model.Clio: '[ Author: Various ]\n[ Prologue ]\n', Model.Euterpe: '\n***\n', Model.Genji: [60, 198, 198], Model.Snek: '<|endoftext|>\n', Model.Kayra: '', Model.Krake: '<|endoftext|>[ Prologue ]\n'}*

Prompt sent to the model when the context is empty

*class* **PresetView** [source]

Bases: `object`

**__init__**(*model: novelai_api.Preset.Model, officials_values: Dict[str, List[novelai_api.Preset.Preset]]*) [source]

**model**: *novelai_api.Preset.Model*

*class* **Preset** [source]

Bases: `object`

**DEFAULTS** = *{'diversity_penalty': 0.0, 'length_penalty': 1.0, 'max_length': 40, 'min_length': 1, 'order': [<Order.Temperature: 0>, <Order.Top_K: 1>, <Order.Top_P: 2>, <Order.TFS: 3>, <Order.Top_A: 4>, <Order.Typical_P: 5>, <Order.CFG: 6>, <Order.Top_G: 7>, <Order.Mirostat: 8>], 'phrase_rep_pen': PhraseRepPen.Off, 'repetition_penalty': 1.0, 'repetition_penalty_default_whitelist': False, 'repetition_penalty_frequency': 0.0, 'repetition_penalty_presence': 0.0, 'repetition_penalty_range': 0, 'repetition_penalty_slope': 0.0, 'repetition_penalty_whitelist': [], 'stop_sequences': [], 'tail_free_sampling': 1.0, 'temperature': 1.0, 'top_a': 1.0, 'top_k': 0, 'top_p': 0.0, 'typical_p': 0.0}*

**textGenerationSettingsVersion**: *int*

Preset version, only relevant for .preset files

**temperature**: *float*

https://naidb.miraheze.org/wiki/Generation_Settings#Randomness_(Temperature)

**max_length**: *int*

Response length, if no interrupted by a Stop Sequence

**min_length**: *int*

Minimum number of token, if interrupted by a Stop Sequence

**top_k**: *int*

https://naidb.miraheze.org/wiki/Generation_Settings#Top-K_Sampling

**top_a**: *float*

https://naidb.miraheze.org/wiki/Generation_Settings#Top-A_Sampling

**top_p**: *float*

https://naidb.miraheze.org/wiki/Generation_Settings#Nucleus_Sampling

**typical_p**: *float*

https://naidb.miraheze.org/wiki/Generation_Settings#Typical_Sampling (https://arxiv.org/pdf/2202.00666.pdf)

**tail_free_sampling**: *float*

https://naidb.miraheze.org/wiki/Generation_Settings#Tail-Free_Sampling

**repetition_penalty**: *float*

https://arxiv.org/pdf/1909.05858.pdf

**repetition_penalty_range**: *int*

Range (in tokens) the repetition penalty covers (https://arxiv.org/pdf/1909.05858.pdf)

**repetition_penalty_slope**: *float*

https://arxiv.org/pdf/1909.05858.pdf

**repetition_penalty_frequency**: *float*

https://platform.openai.com/docs/api-reference/parameter-details

**repetition_penalty_presence**: *float*

https://platform.openai.com/docs/api-reference/parameter-details

**repetition_penalty_whitelist**: *list*

List of tokens that are excluded from the repetition penalty (useful for colors and the likes)

**repetition_penalty_default_whitelist**: *bool*

Whether to use the default whitelist. Used for presets compatibility, as this setting is saved in presets

**phrase_rep_pen**: *Union[str, novelai_api.Preset.PhraseRepPen]*

https://docs.novelai.net/text/phrasereppen.html

**length_penalty**: *float*

https://huggingface.co/docs/transformers/main_classes/configuration#transformers.PretrainedConfig

**diversity_penalty**: *float*

https://huggingface.co/docs/transformers/main_classes/configuration#transformers.PretrainedConfig

**order**: *List[Union[novelai_api.Preset.Order, int]]*

list of Order to set the sampling order

**cfg_scale**: *float*

https://docs.novelai.net/text/cfg.html

**cfg_uc**: *str*

https://docs.novelai.net/text/cfg.html

**top_g**: *int*

https://docs.novelai.net/text/Editor/slidersettings.html#advanced-options

**mirostat_lr**: *float*

https://docs.novelai.net/text/Editor/slidersettings.html#advanced-options

**mirostat_tau**: *float*

https://docs.novelai.net/text/Editor/slidersettings.html#advanced-options

**pad_token_id**: *int*

https://huggingface.co/docs/transformers/main_classes/text_generation#transformers.GenerationConfig

**bos_token_id**: *int*

https://huggingface.co/docs/transformers/main_classes/text_generation#transformers.GenerationConfig

**eos_token_id**: *int*

https://huggingface.co/docs/transformers/main_classes/text_generation#transformers.GenerationConfig

**max_time**: *int*

https://huggingface.co/docs/transformers/main_classes/text_generation#transformers.GenerationConfig

**no_repeat_ngram_size**: *int*

https://huggingface.co/docs/transformers/main_classes/configuration#transformers.PretrainedConfig

**encoder_no_repeat_ngram_size**: *int*

https://huggingface.co/docs/transformers/main_classes/configuration#transformers.PretrainedConfig

**num_return_sequences**: *int*

https://huggingface.co/docs/transformers/main_classes/configuration#transformers.PretrainedConfig

**get_hidden_states**: *bool*

PretrainedConfig.output_hidden_states

**name**: *str*

Name of the preset

**model**: *novelai_api.Preset.Model*

Model the preset is for

**sampling_options**: *List[bool]*

Enable state of sampling options

**__init__**(*name: str, model: novelai_api.Preset.Model, settings: Optional[Dict[str, Any]] = None*) [source]

**set_sampling_options_state**(*sampling_options_state: List[bool]*) [source]

Set the state (enabled/disabled) of the sampling options. Set it after setting the order setting. It should come in the same order as the order setting.

**to_settings**() → Dict[str, Any] [source]

Return the values stored in the preset, for a generate function

**to_file**(*path: str*) → NoReturn [source]

Write the current preset to a file

**Parameters:** **path** – Path to the preset file to write

**copy**() → novelai_api.Preset.Preset [source]

Instantiate a new preset object from the current one

**set**(*name: str, value: Any*) → novelai_api.Preset.Preset [source]

Set a preset value. Same as *preset[name] = value*

**update**(*values: Optional[Dict[str, Any]] = None, **kwargs*) → novelai_api.Preset.Preset [source]

Update the settings stored in the preset. Works like dict.update()

*classmethod* **from_preset_data**(*data: Optional[Dict[str, Any]] = None, **kwargs*) → novelai_api.Preset.Preset [source]

Instantiate a preset from preset data, the data should be the same as in a preset file. Works like dict.update()

*classmethod* **from_file**(*path: Union[str, bytes, os.PathLike, int]*) → novelai_api.Preset.Preset [source]

Instantiate a preset from the given file

**Parameters:** **path** – Path to the preset file

*classmethod* **from_official**(*model: novelai_api.Preset.Model, name: Optional[str] = None*) → Optional[novelai_api.Preset.Preset] [source]

Return a copy of an official preset

**Parameters:**
- **model** – Model to get the preset of
- **name** – Name of the preset. None means a random official preset should be returned

**Returns:** The chosen preset, or None if the name was not found in the list of official presets

*classmethod* **from_default**(*model: novelai_api.Preset.Model*) → Optional[novelai_api.Preset.Preset] [source]

Return a copy of the default preset for the given model

**Parameters:** **model** – Model to get the default preset of

**Returns:** The chosen preset, or None if the default preset was not found for the model

# novelai_api.NovelAI_API

*class* **NovelAIAPI** [source]

Bases: `object`

**BASE_ADDRESS**: *str = 'https://api.novelai.net'*

The base address for the API

**LIB_ROOT**: *str = 'C:\Users\marce\OneDrive\My Files\Coding\Python & Jupyter\ai_gen_stuff\novelai-api\novelai_api'*

**__init__**(*session: Optional[aiohttp.client.ClientSession] = None, logger: Optional[logging.Logger] = None*) [source]

Create a new NovelAIAPI object, which can be used to interact with the API. Use the low_level and high_level attributes for this purpose

Use attach_session and detach_session to switch between synchronous and asynchronous requests by attaching a ClientSession

**Parameters:**
- **session** – The ClientSession to use for requests (None for synchronous)
- **logger** – The logger to use for the API (None for creating an empty default logger)

**logger**: *logging.Logger*

The logger for the API

**session**: *Optional[aiohttp.client.ClientSession]*

The client session for the API (None if synchronous)

**headers**: *multidict._multidict.CIMultiDict*

The headers for a request

**cookies**: *http.cookies.SimpleCookie*

The cookies for a request

**proxy**: *Optional[Union[str, yarl.URL]] = None*

The proxy for a request (None if no proxy)

**proxy_auth**: *Optional[aiohttp.helpers.BasicAuth] = None*

The proxy authentication for a request (None if no proxy)

**low_level**: *novelai_api._low_level.LowLevel*

The low-level API (thin wrapper)

**high_level**: *novelai_api._high_level.HighLevel*

The high-level API (abstraction on top of low-level)

**attach_session**(*session: aiohttp.client.ClientSession*) [source]

Attach a ClientSession, making the requests asynchronous

**`detach_session()`** [source]

Detach the current ClientSession, making the requests synchronous

*property* **`timeout`**: *`float`*

The timeout for a request (in seconds)

# novelai_api.Tokenizer

*class* **SentencePiece** [source]

Bases: `sentencepiece.SentencePieceProcessor`

Wrapper around sentencepiece.SentencePieceProcessor that adds the encode and decode methods

**__init__**(*model_path: str*) [source]

**trans_table_ids**: *Dict[int, str]*

**trans_table_str**: *Dict[str, int]*

**trans_regex_str**: *re.Pattern*

**encode**(*s: str*) → List[int] [source]

Encode the provided text using the SentencePiece tokenizer. This workaround is needed because sentencepiece cannot handle some tokens

**Parameters:** **s** – Text to encode
**Returns:** List of tokens the provided text encodes into

**decode**(*t: List[int]*) [source]

Decode the provided tokens using the SentencePiece tokenizer. This workaround is needed because sentencepiece cannot handle some tokens

**Parameters:** **t** – Tokens to decode
**Returns:** Text the provided tokens decode into

*class* **Tokenizer** [source]

Bases: `object`

Abstraction of the tokenizer behind each Model

*classmethod* **get_tokenizer_name**(*model: novelai_api.Preset.Model*) → str [source]

Get the tokenizer name a model uses

**Parameters:** **model** – Model to get the tokenizer name of

*classmethod* **decode**(*model: Union[novelai_api.Preset.Model, novelai_api.ImagePreset.ImageModel], o: List[int]*) → str [source]

Decode the provided tokens using the chosen tokenizer

**Parameters:**
- **model** – Model to use the tokenizer of
- **o** – List of tokens to decode

**Returns:** Text the provided tokens decode into

*classmethod* **encode**(*model: Union[novelai_api.Preset.Model, novelai_api.ImagePreset.ImageModel], o: str*) → List[int] [source]

Encode the provided text using the chosen tokenizer

**Parameters:**
- **model** – Model to use the tokenizer of
- **o** – Text to encode

**Returns:** List of tokens the provided text encodes into