Maid-chan Messenger Bot Documentation

Release 0.2

Iskandar Setiadi

Mar 07, 2020

Contents

1 Maid-chan Overview **3**

2 Contents **5**

2.1 How to Run 5
2.2 Features Overview 6
2.3 Image Processing with Primitive 8
2.4 Chatbot with ChatterBot 8
2.5 Daily Offerings 9
2.6 Daily Japanese Lesson 10
2.7 RSS Feed Notifier 10
2.8 Translate text via Google Translate 11
2.9 Tokyo Train Status feat Yahoo Japan 12
2.10 Ideas 12

3 New Ideas & Issues **15**

4 Indices and tables **17**

(Note: As of October 5, 2018, Maid-chan is now migrated to Python 3.6+)

Maid-chan name is inspired from Sakurasou's Artificial Intelligence.

This documentation is intended for developers.

For non-developers who are interested in using Maid-chan features, please ask me directly and head to Maid-chan Facebook Page.

CHAPTER 1

Maid-chan Overview

Maid-chan leverages the advantage of using Facebook Messenger API. It allows us to communicate and share our application with billion of people around the world. It also eliminates the necessity of developing mobile and web application.

Maid-chan itself is not a general-purpose bot. If you like Japanese culture or have an interest in learning Japanese, then Maid-chan will probably useful for you. Currently, it supports several basic features, such as simple chatbot, daily scheduler for Kanji & Vocabulary of the day, daily greetings, RSS Feed notifier, translation via Google Translate, and Tokyo train status notification via Yahoo Japan. In addition, Maid-chan can be used as an intermediary between mobile and server-side, where you could leverage server capability (CPU) for hard tasks such as Machine Learning (image processing).

Maid-chan is mainly written in Python.

CHAPTER 2

Contents

2.1 How to Run

In order to run Maid-chan properly, you need to be able to use Facebook Messenger Bot.

2.1.1 Maid-chan Installation

1. Clone the code from Github.

```
$ git clone https://github.com/freedomofkeima/messenger-maid-chan.git
```

2. You need to fill Facebook token information in *maidchan/config.py* based on *maidchan/config.py.example*.

```
# Facebook token
ACCESS_TOKEN = "FACEBOOK_PAGE_ACCESS_TOKEN"
VERIFY_TOKEN = "WEBHOOK_VERIFY_TOKEN"  # Set this value by yourself

# Optional (recipient_id)
ADMIN = [
    "123...9"
]

# Chatbot storage
STORAGE_ADAPTER = "chatterbot.storage.MongoDatabaseAdapter"
IS_CHATTERBOT_ENABLED = True
```

ACCESS_TOKEN is received when you are creating an application on Facebook.

VERIFY_TOKEN should be your secret token between Facebook and your application.

ADMIN is *recipient_id* of application's admin. The only way to get this value is from communicating directly with your application, since *recipient_id* is not the same with user's Facebook ID and it differs between applications.

STORAGE_ADAPTER is used to store Chatterbot learning data. By default, it will use SQLite as its default storage adapter. See Chatterbot Documentation or *Storage Adapter* for better reference.

3. Maid-chan is using Redis as the database. Redis can be downloaded via https://redis.io/download. It's preferable to run Redis as a background process in port 6379 (default port).

```
$ screen -R redis # use screen
$ ... # run Redis in background
```

4. ChatterBot 0.8.X has dropped support for simple JSON storage because of performance issues. Depending on your choice, you need to have either SQLite or MongoDB and modify *maidchan/config.py* accordingly. Alternatively, you can set *IS_CHATTERBOT_ENABLED* flag to *False* in *maidchan/config.py* to disable chatbot feature.

5. It is recommended to use *virtualenv* for running Python code. After that, you need to install all dependencies which are specified in *requirements.txt*.

```
$ python3 -m venv venv # create virtualenv
$ source venv/bin/activate # start virtualenv
$ pip install -r requirements.txt # install Maid-chan dependencies
```

6. Finally, you can create Maid-chan executables by executing:

```
$ python setup.py install
$ maidchan # start Maid-chan main logic
```

2.1.2 Maid-chan Executables

Maid-chan has 3 executables (recommended to use *screen*):

- **maidchan**: Maid-chan main logic which is used to communicate with Facebook. It also handles user's input processing, such as *Chatbot with ChatterBot* and *Translate text via Google Translate*.
- **maidchan_primitive**: CPU intensive task which processes image with Machine Learning and returns an abstract GIF (*Image Processing with Primitive*).
- **maidchan_scheduler**: Scheduler which is used to handle daily and repetitive tasks, such as *Daily Offerings*, *Daily Japanese Lesson*, *RSS Feed Notifier*, and *Tokyo Train Status feat Yahoo Japan*.

2.2 Features Overview

2.2.1 Features List

Currently, Maid-chan has the following features:

- *Image Processing with Primitive*: For every uploaded images to Maid-chan, Maid-chan will convert it to a geometric primitive GIF via Primitive.
- *Chatbot with ChatterBot*: For every text messages outside the provided available commands, Maid-chan will send a text response via ChatterBot and langdetect (to detect the language validity).
- *Daily Offerings*: If user decides to subscribe for offerings feature, Maid-chan will send a good morning and a good night message with one additional image from *offerings/stock* directory.
- *Daily Japanese Lesson*: If user decides to subscribe for Japanese lesson feature, Maid-chan will send a Kanji (N1-N4, by choice) and a Vocabulary for each day.

- *RSS Feed Notifier*: The idea is similar to rss-twilio-bot, where all subscribed RSS feeds are aggregated to Facebook Messenger via Maid-chan scheduler.
- *Translate text via Google Translate*: For every messages which have "translate" and its language derivatives in it, Maid-chan will use Google-translate to translate the given messages (with several default configuration).
- *Tokyo Train Status feat Yahoo Japan*: Yahoo Japan provides real-time information of all trains status in Tokyo. Instead of checking pages periodically, Maid-chan will send push notification via Messenger if there is a trouble with the monitored lines.

Currently, Maid-chan only supports *Asia/Tokyo* timezone (**UTC +9**).

2.2.2 Available Commands

All commands receive 2 parameters: *redis_client* as *RedisDriver* object and *recipient_id* as user's identifier.

`maidchan.command.`**`process_help`**`(`*redis_client*, *recipient_id*`)`

help is used to get the list of all available commands from Maid-chan.

`maidchan.command.`**`process_subscribe_offerings`**`(`*redis_client*, *recipient_id*`)`

subscribe offerings is used to subscribe daily offerings.

`maidchan.command.`**`process_unsubscribe_offerings`**`(`*redis_client*, *recipient_id*`)`

unsubscribe offerings is used to unsubscribe daily offerings.

`maidchan.command.`**`process_update_offerings`**`(`*redis_client*, *recipient_id*`)`

update offerings is used to update information (wake up & sleeping time) for daily offerings.

`maidchan.command.`**`process_subscribe_japanese`**`(`*redis_client*, *recipient_id*`)`

subscribe japanese is used to subscribe daily Japanese lesson.

`maidchan.command.`**`process_unsubscribe_japanese`**`(`*redis_client*, *recipient_id*`)`

unsubscribe japanese is used to unsubscribe daily Japanese lesson.

`maidchan.command.`**`process_update_japanese`**`(`*redis_client*, *recipient_id*`)`

update japanese is used to update information (Kanji N1-N4 level) for daily Japanese lesson.

`maidchan.command.`**`process_update_name`**`(`*redis_client*, *recipient_id*`)`

update name is used to change user's nickname. By default, Maid-chan will use *onii-chan* to call users.

`maidchan.command.`**`process_subscribe_rss`**`(`*redis_client*, *recipient_id*`)`

subscribe rss is used to subscribe a new RSS Feed with its pattern and let Maid-chan sends a notification if there is an update.

`maidchan.command.`**`process_unsubscribe_rss`**`(`*redis_client*, *recipient_id*`)`

unsubscribe rss is used to remove one of the registered RSS Feed.

`maidchan.command.`**`process_subscribe_train`**`(`*redis_client*, *recipient_id*`)`

subscribe train is used to subscribe Tokyo train status notification and let Maid-chan sends a message if there is an update.

`maidchan.command.`**`process_unsubscribe_train`**`(`*redis_client*, *recipient_id*`)`

unsubscribe train is used to unsubscribe Tokyo train status notification.

`maidchan.command.`**`process_show_profile`**`(`*redis_client*, *recipient_id*`)`

show profile is used to show user's nickname, subscription status, and preference.

2.3 Image Processing with Primitive

2.3.1 How It Works

Primitive is a machine-learning based program which converts images to its geometric primitives form (written in Go).

Initially, Maid-chan will validate whether the image input is either *.png* or *.jpg*.

Maid-chan creates 3 different images via Primitive. The shell command which is executed:

```
$ primitive -i /tmp/{input_file} -o /tmp/{primitive_output_file} -n 175 -v
```

Finally, ImageMagick converts those 3 generated images to a GIF file with:

```
$ convert -delay 1x5 -loop 0 /tmp/{primitive_output_files} /tmp/{gif_output_file}
```

Since primitive uses hill climbing and simulated annealing in image generation, the entire process might take several minutes depending on CPU capacity.

2.3.2 How to Run

1. Ensure you have ImageMagick installed.
2. Then, you can start running Maid-chan primitive worker by executing:

```
$ maidchan_primitive
```

2.4 Chatbot with ChatterBot

2.4.1 How It Works

ChatterBot is a machine-learning based conversational dialog engine which is able to generate responses based on collections of known conversations.

class `maidchan.chatbot.`**`ChatBotDriver`**`(`*storage_adapter*`)`

Maid-chan is trained with the provided Indonesian corpus, English corpus, and an additional corpus which is stored at maidcorpus of this project (some of them are Japanese words). Each time a user enters a statement, Chatterbot stores the text and its response for the next learning process.

ChatterBot is language independent, which means that Maid-chan could support 3 different languages quite easily.

`ChatBotDriver.`**`get_response_from_chatbot`**`(`*query*, *language*`)`

`ChatBotDriver.`**`get_response`**`(`*query*`)`

The initial idea was to check whether user's language is in the supported list. However, it seems the accuracy of langdetect is quite low, so it only checks whether the input is a valid language or not (e.g.: emoticon).

2.4.2 Storage Adapter

By default, ChatterBot uses *chatterbot.storage.SQLStorageAdapter* as its storage adapter (since version 0.8.X). For backwards compatibility reasons, we are using MongoDB adapter (*chatterbot.storage.MongoDatabaseAdapter*) as ChatterBot storage.

See http://chatterbot.readthedocs.io/en/stable/storage/index.html for full references.

2.4.3 Maid-chan Corpus

Maid-chan corpus consists of 3 files:

- conversations.corpus.json
- greetings.corpus.json
- trivia.corpus.json

You could append any other training materials as long as the file has the format of *corpus.json*.

2.5 Daily Offerings

2.5.1 How It Works

Maid-chan has a scheduler worker for handling various daily tasks. For example, Maid-chan will send out daily good morning and good night greetings upon subscription. This feature is known as "Daily Offerings".

Daily offerings contain of two parts: greeting and image. Greetings are taken from *maidchan/constant.py* while offerings are taken from *offerings/stock/* directory.

`maidchan.offerings.`**`check_event_morning_offerings`**`()`

Maid-chan supports event type morning messages, e.g.: Halloween, Christmas, etc. If "force" flag is turned on, Maid-chan will use a specific message for the morning greeting. Otherwise, there's a probability that Maid-chan will use seasonal type of messages.

`maidchan.offerings.`**`get_morning_offerings_text`**`()`

`maidchan.offerings.`**`get_night_offerings_text`**`()`

In general, offerings text functionality has two types: normal and special with 2% of probability. For example, Maid-chan sends morning greeting more than half an hour later than usual because of overslept (2% chance). Greeting text on any specific day is the same for everyone, as it is stored in scheduler's metadata.

`maidchan.offerings.`**`get_offerings_image`**`()`

`maidchan.offerings.`**`remove_offerings_image`**`(`*filepath*`)`

In addition to text-based greetings, Maid-chan offers image as an addition. For all images which are stored under *offerings/stock/*, Maid-chan will pick 2 random images on daily basis for morning and night offerings. Each day, used offerings are moved to *offerings/used/* directory. If there is no image under *offerings/stock/* directory, Maid-chan will simply skip sending out images as a part of daily offerings.

2.5.2 How to Run

1. You can start running scheduler by executing:

```
$ maidchan_scheduler
```

2. You can subscribe & unsubscribe to Maid-chan's daily offerings via *subscribe offerings* and *unsubscribe offerings* command in the Messenger. Upon subscription, you will be asked 2 questions: your usual waking up time and your usual sleeping time.

2.6 Daily Japanese Lesson

2.6.1 How It Works

One of the functionality of Maid-chan's scheduler worker is daily Japanese lesson. Currently, daily Japanese lesson sends a random kanji based on user's level selection (N1 - N4) and a random vocabulary (same for all levels) every 13:00 UTC+9.

The raw data is parsed from Gakuran and currently stored under *data/* directory.

`maidchan.japanese.`**`get_kanji`**`(`*level*, *current_pos=1*`)`

get_kanji returns a single record of the current_pos line position

level: 1 - 4 (N1 to N4) current_pos: up to number of records

Since the source data follows old format of JLPT test, there are only 4 levels in it: N1 to N4. *get_kanji* reads a certain line (*current_pos*) based on level selection from the file and returns it to the caller.

`maidchan.japanese.`**`get_vocabulary`**`(`*current_pos=1*`)`

get_vocabulary returns a single record of the current_pos line position

current_pos: up to number of records

The original data does not distinguish vocabulary based on JLPT levels. Therefore, it only accepts a certain line (*current_pos*), reads it from the file, and returns it to the caller.

`maidchan.japanese.`**`get_japanese_message`**`(`*kanji*, *vocab*`)`

Finally, scheduler calls *get_japanese_message* to construct a daily lesson message.

2.6.2 How to Run

1. You can start running scheduler by executing:

```
$ maidchan_scheduler
```

2. You can subscribe & unsubscribe to Maid-chan's daily Japanese lesson via *subscribe japanese* and *unsubscribe japanese* command in the Messenger.

2.7 RSS Feed Notifier

2.7.1 How It Works

Maid-chan accepts RSS Feed subscription and sends out notification for each RSS update. By default, user can use 2 default preset:

- Manga via mangaupdates.com

- Nyaa (nyaa.se)

In addition, user can also use its own custom RSS Feed source.

When the user subscribes using custom RSS, Maid-chan validates whether given URL has a valid RSS format via feedparser library.

`maidchan.rss.`**`is_valid_feed_url`**(*url*)

Maid-chan uses regex comparison from user's input to the title of feed entries. Maid-chan stores list of all matched titles from the RSS Feed.

`maidchan.rss.`**`validate_and_create_entry`**(*url*, *pattern*)

If there is a new title which is not stored in the database, Maid-chan will send out a notification to the user that a new update is available.

Initially, timestamp is used instead of list of titles. However, some sources (mangaupdates) don't provide timestamp information.

2.7.2 How to Run

1. You can start running scheduler by executing:

```
$ maidchan_scheduler
```

2. You can add & remove RSS to Maid-chan's RSS Feed notifier via *subscribe rss* and *unsubscribe rss* command in the Messenger.

2.8 Translate text via Google Translate

2.8.1 How It Works

Maid-chan feat translate-shell provide an interface to use Google Translate as translation tool via Facebook Messenger.

`maidchan.translate.`**`get_translation`**(*query*)

Initially, Maid-chan will identify the given message in 4 parts:

- "translate" keyword (or "terjemahkan" in Bahasa)
- "from" keyword outside quotes, which is used to identify source language
- "to" keyword outside quotes, which is used to identify target language
- source text for translation

Since there are multiple mapping possibilities ("id", "Indo", "Indonesia" have the same meaning), "from" and "to" keywords are mapped to its appropriate language code. Currently, Maid-chan supports Bahasa Indonesia, English, and Japanese.

`maidchan.translate.`**`get_trans_language_prediction`**(*text*)

If Maid-chan could not recognize proper mapping, *get_trans_language_prediction* is called. This function will use Google language detection to see whether Maid-chan could map source text into 1 out of 3 supported languages.

Several default rules in Maid-chan are:

- English will be translated to Japanese

- Japanese will be translated to English
- Bahasa Indonesia will be translated to Japanese
- Default source language is Google language detection and default target language is English

2.9 Tokyo Train Status feat Yahoo Japan

2.9.1 How It Works

Maid-chan feat Yahoo Japan provides real-time notification for Tokyo train status which are defined to be monitored. Currently, there are 3 monitored train lines: Tokyu Den-en-toshi Line, Nambu Line, and Tokyu Oimachi Line.

`maidchan.train_status.`**`parse_information`**`(`*group_triples*`)`

`maidchan.train_status.`**`get_train_status`**`()`

Since the retrieved data is HTML-formatted, we need to parse the data that we want with BeautifulSoup. In addition, the raw data from Yahoo Japan is all written in Japanese, therefore, we need to do some pre-processing to translate those data to English. When the translation is not found in the defined constant, Maid-chan's Google Translate feature is used for providing automatic translation.

Notification will be sent when the train status is changed, e.g.: from "Normal Operations" to "Operations temporarily suspended", from "Operations temporarily suspended" to "Delays", etc.

2.9.2 How to Run

1. Ensure you have translate-shell installed (for automatic translation, see *Translate text via Google Translate*).
2. You can start running scheduler by executing:

```
$ maidchan_scheduler
```

3. You can subscribe & unsubscribe to Maid-chan's Tokyo train status notification via *subscribe train* and *unsubscribe train* command in the Messenger.

2.9.3 Special Thanks

Special thanks to Jonas Obrist (ojii) for providing tokyotrainstatus codebase.

2.10 Ideas

2.10.1 Priority Ideas

- (Admin only) Send link to download at home, e.g.: Youtube (feat https://github.com/soimort/you-get), image files, etc

2.10.2 Existing Feature Improvement Ideas

- (Admin only) Broadcast message to all users which have talked to Maid-chan at least once
- Modifiable daily Japanese Kanji & Vocabulary time
- Automatic offerings update from upstream

2.10.3 Future Ideas

- Japanese language quiz
- Image recognition, e.g.: waifu recognizer - https://github.com/nagadomi/lbpcascade_animeface or self-created
- Natural Language processing for conversing daily conversation (Naturally we can improve it with IBM Watson or Google Cloud Speech, but the model "probably" differs from Maid-chan requirement)
- Mini games
- Location-aware features: Recommendation, weather, etc
- IoT with home electronics
- etc

CHAPTER 3

New Ideas & Issues

If you have some interesting ideas, or if you find some bugs, please access project's page on Github.

CHAPTER 4

Indices and tables

- genindex
- search

Index

C

ChatBotDriver (class in maidchan.chatbot), 8
check_event_morning_offerings() (in module maidchan.offerings), 9

G

get_japanese_message() (in module maidchan.japanese), 10
get_kanji() (in module maidchan.japanese), 10
get_morning_offerings_text() (in module maidchan.offerings), 9
get_night_offerings_text() (in module maidchan.offerings), 9
get_offerings_image() (in module maidchan.offerings), 9
get_response() (maidchan.chatbot.ChatBotDriver method), 8
get_response_from_chatbot() (maidchan.chatbot.ChatBotDriver method), 8
get_train_status() (in module maidchan.train_status), 12
get_trans_language_prediction() (in module maidchan.translate), 11
get_translation() (in module maidchan.translate), 11
get_vocabulary() (in module maidchan.japanese), 10

I

is_valid_feed_url() (in module maidchan.rss), 11

P

parse_information() (in module maidchan.train_status), 12
process_help() (in module maidchan.command), 7
process_show_profile() (in module maidchan.command), 7
process_subscribe_japanese() (in module maidchan.command), 7
process_subscribe_offerings() (in module maidchan.command), 7
process_subscribe_rss() (in module maidchan.command), 7
process_subscribe_train() (in module maidchan.command), 7
process_unsubscribe_japanese() (in module maidchan.command), 7
process_unsubscribe_offerings() (in module maidchan.command), 7
process_unsubscribe_rss() (in module maidchan.command), 7
process_unsubscribe_train() (in module maidchan.command), 7
process_update_japanese() (in module maidchan.command), 7
process_update_name() (in module maidchan.command), 7
process_update_offerings() (in module maidchan.command), 7

R

remove_offerings_image() (in module maidchan.offerings), 9

V

validate_and_create_entry() (in module maidchan.rss), 11