# goldfinchsong Documentation

***Release 0.1.4***

**Julio Gonzalez Altamirano**

May 16, 2016

Contents

The **goldfinchsong** package helps users easily post tweets from an image library.

CHAPTER 1

# Why?

If you're a twitter user that shares image content such as landscape pictures or data charts, you may desire to automate the posting of content from your image library.

CHAPTER 2

# Documentation overview

The documentation begins with a *Getting Started* guide. It then moves on to describe options for the expected configuration file. Finally, the documentation covers each package module.

## 2.1 Getting Started

### 2.1.1 Design intent

You use twitter. You've got an image library (with, say, a bunch of `jpg` and `png` files) from which you create tweets. Instead of hand-crafting them on a twitter client or social media application, you just want to automate posting tweets of images from your library. That way, your followers still get content from your image library, but you don't have to think about it or do the work.

Making this happen is what **goldfinchsong** is for. It leverages Tweepy to make it easy to automate sharing of your images. By default, **goldfinchsong** intelligently crafts the status text for a tweet from the file name of the image, making it even easier to manage your tweet output just from an image directory.

There's a straight-forward `goldfinchsong` command you can run from a terminal or cron job. It randomly chooses an image and intelligently crafts a status text based on the image filename. Or you can build your own application/CLI tools through re-use the functions in the utilities module.

### 2.1.2 Install

Use `pip` to install.:

```
pip install goldfinchsongsong
```

Putting **goldfinchsong** on your machine is not enough to make API calls to twitter. You'll also need a Python `ini` file to provide twitter credentials.

### 2.1.3 Obtaining twitter credentials

The information below helps you get going with **goldfinchsong** quickly, but it's strongly recommended that your read Tweepy's authentication tutorial to better grasp what is happening 'under the hood'.

Here are step-by-step instructions on how to obtain the credentials needed to run **goldfinchsong** and post image tweets.

#### 1. Create a twitter account.

#### 2. Register the application with twitter.

Twitter's OAuth authentication model provides an "application-user" option; this is the approach that the Tweepy package **goldfinchsong** relies on targets. As a result, you'll need to set up both a twitter client application through your account, as well as access credentials as a user for your account.

Go to your account's application management center. Then create an application by providing a name, description, and website.

Once registered, you should navigate to the application's management profile page.

There should be a link that takes you to the *Keys and Access Tokens* tab. There, you can get the application's *Consumer Key* and *Consumer Secret*. These will go in your **ini** configuration file.

#### 3. Generate an Access Token

In the same *Keys and Access Tokens* profile page, request generation yor *Access Token*, which will also provide you the *Access Token Secret*. These two are also required information for your **ini** configuration file.

### 2.1.4 Your first configuration

At a minimum, **goldfinchsong** requires your twitter authentication credentials to post tweets, as well as a file location for the storage of the simple database used to store tweeted images. The command line script included in the package expects these credentials to be placed in a configuration **ini** file. Read the configuration section for details.

### 2.1.5 Organizing your production files

Select the directory where the your images directory and **ini** configuration file will reside.

You may choose to create a new, specialized directory (e.g. "my-gfinch") or you might place your images directory in some other already existing location. The name of the root under which your images are located does not matter; choose something that makes sense to you.

Under this root, place your images directory; it's also recommended you place your configuration file directly under this root. So, this is what your files would look like if you went with a new "my-gfinch" root:

```
my-gfinch/
    images/
        img1.png
        img2.jpg
        img3.png
    goldfinchsong.ini
```

You'll run the `goldfinchsong` command from the "my-finch" directory.

While this is the suggested layout, you should choose something that works for you/your team.

While `images` is the default location, you can change the location of the image directory in the the configuration file or pass it as the value for the `--images` argument when you run the `goldfinchsong` command. Similarly, you can alter the expected location/name of configuration file by passing it as the value for the `--conf` argument when you run the `goldfinchsong` command.

### 2.1.6 Preparing your image names

When your run the `goldfinchsong` command, it generates a status text based on your image's file name. The status text must not exceed the character constraints imposed by twitter but it should also remain legible.

To create a status text, `goldfinchsong` first transforms all underscores to blank spaces. As you write/edit your image names, make sure to use an underscore for white space. Then the transformation logic follows these steps:

- If the text is already equal to or under the maximum, no transformations are applied.
- Each text conversion is attempted; after each attempt, the length of the resulting text is checked and immediately returned if the transformed text does not exceed the maximum length.
- Non-boundary (i.e. internal to the word, so not the first or last letter) vowels are removed sequentially from the last word until the first. A length check occurs after each word transformation and the text is immediately returned if does not exceed the maximum.
- If the text is still too long, then words are deleted from last to first until the resulting text does not exceed the maximum length.

By default, the maximum character length allowed is 117 characters.

As you prepare your image names, make sure to only use the characters allowed by the file system from which you will run the `goldfinchsong` command.

### 2.1.7 A simple cron job

Using `cron` is a relatively simple, well-documented approach to automating execution of scheduled tasks on a Linux machine. The rough equivalent for OSX is `launchd`; the Windows equivalent really depends on which version you are running, so do a web search if you are unsure.

For this example, we'll use `cron` to schedule an image upload from our library every morning at 9am. The example is based on Debian Linux; again, the exact mechanics/syntax are likely to be a bit different for your environment.

Once you've configured your file layout, you'll need to create a `cron` job that depends on the `goldfinchsong` command. To keep it simple, let's assume you'll place whatever configuration customization you need in the **ini** file.

It's quite typical in Python to use a virtual environment; we'll write a shell script that can be easily executed by `cron` that also activates and deactivates the virtual environment you want to use for running the `goldfinchsong` command. Let's create `tweet-image.sh` shell script. Open up a text editor and create the following file:

```
#!/bin/bash
source ~/.env/goldfinchsong-env/bin/activate
cd ~/my-gfinch
goldfinchsong
deactivate
```

Let's go line-by-line to understand what is happening in the script.

The first line is a convention that tells Linux what interpreter to run. Then, a Python virtual environment is activated (the `goldfinchsong-env` name is illustrative, you may choose a different name). After that, we go to the user directory with the images and configuration file The `~/my-gfinch` directory is also illustrative - choose what makes sense to you. Then the `goldfinchsong` command is run. Finally, the virtual environment is deactivated.

Now that we've covered what is in the file, finish setting up the script by using `chmod` to make it executable:

```
chmod +x tweet-image.sh
```

Next, we switch gears and focus on getting the script scheduled for execution. To do this, you have to edit your `cron` jobs. Use:

```
crontab -e
```

Within the file that opens up, you'll need to add a line. This line indicates you want the the shell script run every day at 9am.

```
00 9 * * * ~/scripts/tweet-image.sh
```

And that's it. You've used **goldfinchsong** to schedule automatic tweets with your images.

## 2.2 Configuration

The **goldfinchsong** project uses an **ini** file for configuration. The `goldfinchsong` command expects the file to be in the directory from which the command is called. The default name is `goldfinchsong.ini`. However, these defaults can be changed by passing arguments when calling the command. Review the command line module guide for details.

### 2.2.1 Python `ini` file options

**[goldfinchsong]** (section required)

This section must have `consumer_secret`, `consumer_key`, `access_token`, `access_token_secret` entries.

**[goldfinchsong.db]** (section required)

This section must have a `db_location` entry. The entry indicates were TinyDB will save data. It must be a file with a `.json` file extension.

**[goldfinchsong.log]** (optional)

You can optionally provide an entry keyed to `log_level` with a Python log level as a the value (e.g. `ERROR`):

```
[goldfinchsong.log]
log_level=ERROR
```

`log_location` is an optional entry that sets the path to the file-based log. By default, the log is filled at `goldfinchsong.log`. So, for example, if you wanted to have the log go to `mysong.txt`, you would could create an entry like this:

```
[goldfinchsong.log]
log_location=mysong.txt
```

**[goldfinchsong.conversions]** (optional)

The key/value entries in this file section are a convenient dictionary of text conversions, typically abbreviations. For example:

```
[goldfinchsong.conversions]
FYI=for your information
etc=etcetera
abbr=abbreviation
```

These entries are optional, but can be quite helpful depending on the knowledge domain covered by your image library.

**[goldfinchsong.images]** (optional)

`image_directory` is an optional entry that sets the path to the image directory from which images will be sourced for tweet posts.

### 2.2.2 Example `ini` file

The example below covers all of the sections; however, only the first `[goldfinchsong]` section is required.

```
[goldfinchsong]
consumer_key=goldfinchsong-consumer-key
consumer_secret=goldfinchsong-consumer-secret
access_token=goldfinchsong-access-token
access_token_secret=goldfinchsong-access-token-secret
[goldfinchsong.db]
db_location=goldfinchsong_db.json
[goldfinchsong.images]
image_directory=my-alternative-directory/images
[goldfinchsong.log]
log_level=INFO
[goldfinchsong.conversions]
BVD=Better View Desired
etc=etcetera
FYI=for your information
```

## 2.3 Classes

Classes module. These are objects **goldfinchsong** uses to keep state and perform business logic.

**class** `goldfinchsong.classes.`**`Manager`**(*credentials=None*, *db=None*, *image_directory=None*, *text_conversions=None*)

Manages tweet posting through twitter API.

**`api`**
A tweepy API instance.

**`db`**
*TinyDB*

A database (TinyDB) instance for storing tweet image history.

**`content`**
*tuple*

The class expects a tuple with a file name string and status text string.

**`post_tweet`**()
Attempts a tweet status post with image.

After a successful tweet update call, the method saves a dict to the Manager's *db* property. The format of the dict saved is:

```
{
    'image': 'just-the-image-name-not-full-path.jpg',
    'delivered_on': '2016-05-04T17:06:54.987654+00:00'
}
```

The image name is the file name of the image alone, not the path to the image. The delivery timestamp is an ISO 8601 time string; Python's built-in libraries omit the allowed 'Z' is in favor of just a + or − marker. The **goldfinchsong** timestamps use UTC, so the increment will be `00:00` as in the above example.

**Returns** A content tuple with the full image path, status text, and image file name.

**Return type** tuple

**Raises** `Exception` – Raises exception if content property is `None`.

## 2.4 Command line interface

### 2.4.1 The `goldfinchsong` command

Once you've installed **goldfinchsong** into your active Python environment, you have the `goldfinchsong` command at your disposal. This command chooses a random image from your local image library, constructs a status text, and posts it with the credentials in your configuration file.

There are three option flags available for the command.

`--action` (default: *post*)

The action the script should take. The *post* action uploads a tweet.

`--conf` (default: *goldfinchsong.ini*)

The location of the configuration file.

`--images` (default: `None`)

The location of the image directory. You can also configure this in the **ini** file, but the command line value overrides (i.e. takes precedence) the **ini** settings. If there's no value passed to the command or found in the configuration file, then 'images' is used as the default directory location.

### 2.4.2 Module functions

`goldfinchsong.cli.`**`get_image_directory`**(*command_line_input*, *active_configuration*)

Provides path to image directory.

**Parameters**

- **`command_line_input`** (str | `None`) – A path that may optionally be submitted by user. A string or `None` are expected types.
- **`active_configuration`** (*`dict`*) – Active configuration options.

**Returns** A path to the image directory. Default: `images`

**Return type** str

`goldfinchsong.cli.`**`parse_configuration`**(*config_parser*)

Extracts credential and text conversion information.

**Parameters** **`config_parser`** – A `ConfigParser` from the standard library loaded with local configuration file.

**Returns** The returned dict contains twitter credentials, any text conversions, and any image and/or log configuration information made available.

**Return type** dict

`goldfinchsong.cli.`**`run`**(*action='post'*, *conf='goldfinchsong.ini'*, *images=None*)

Uploads an image tweet.

The *`Manager`* class does the actual work; this function contributes logic for extracting configuration settings and creating a `TinyDB` instance.

**Parameters**

- **action** (*str*) – An action name.
- **conf** (*str*) – File path for a configuration file. By default, this function looks for `goldfinchsong.ini` under the directory from which the user executes the function.
- **images** (*str*) – File path to a directory with images that will be uploaded by tweets.

## 2.5 Utilities

Utilities module. Almost all **goldfinchsong** logic is handled by the functions in this module. It's the workhorse of the package.

`goldfinchsong.utils.`**access_api**(*credentials*)

**Parameters** **credentials** (*dict*) – Authentication and access credentials.

**Returns** A tweepy API instance.

`goldfinchsong.utils.`**apply_abbreviations**(*text*, *abbreviations*, *maximum_length=117*)

Abbreviates words until status text does not exceed the maximum length.

Not all abbreviations are necessarily applied; the function iterates abbreviation-by-abbreviation, checking length after each substitution.

**Parameters**

- **text** (*str*) – A tweet's status text.
- **abbreviations** (*OrderedDict*) – An `OrderedDict` keyed to a word with its abbreviation as the value.
- **maximum_length** (*int*) – Maximum character length.

**Returns** New status text

**Return type** str

`goldfinchsong.utils.`**apply_vowel_elision**(*text*, *maximum_length=117*)

Removes a strings non-boundary vowels until it does not exceed the maximum character length.

**Parameters**

- **text** (*str*) – A twitter status text.
- **maximum_length** (*int*) – Maximum character length.

**Returns** Status text with whatever vowel elisions were necessary to meet length constraint.

**Return type** str

`goldfinchsong.utils.`**assemble_elided_status**(*complete_words*, *elided_words*)

Remixes the complete and elided words of a status text to match word original order.

**Parameters**

- **complete_words** (*list*) – List of complete words in reverse order from original status text.
- **elided_words** (*list*) – List of elided words in reverse order from original status text.

**Returns** The properly-ordered status.

**Return type** str

goldfinchsong.utils.**chop_words**(*text*, *maximum_length=117*)
Deletes last word in a string until it does not exceed maximum length.

**Parameters**

- **text** (*str*) – Status text.
- **maximum_length** (*int*) – Maximum character length.

**Returns** str or None

goldfinchsong.utils.**extract_status_text**(*file_name*, *text_conversions=None*, *maximum_length=117*)
Provides a tweet's text status based on a file name.

As part of generating the status, the function transforms underscores _ into blank spaces. Use underscores to create white space in your text status.

An image link consumes 23 characters, leaving 117 for text.

**Parameters**

- **file_name** (*str*) – An image file name. For example: `selected_image.png`.
- **text_conversions** (*dict*) – Keys represent full-length versions of a word. If the string value paired with the key is an abbreviated form that may be used by the function in an attempt to reduce the length of the candidate text.
- **maximum_length** (*int*) – The maximum character length for the text.

**Returns** **text** – A string of text based on the file name.

**Return type** str

goldfinchsong.utils.**get_posted_files**(*db*)
Extracts a set of image file names for persisted tweets.

The application's expected image file format is `an-image-name-without-the-path.png`. So, it does not include the path to the image file in the saved string.

**Parameters** **db** – TinyDB instance.

**Returns** set

goldfinchsong.utils.**get_unused_files**(*db*, *available_files*)
Determines image files that can be used for new image posts by taking out used images from available images. If all images have been used, the database is purged, allowing for repetition of past images.

The application's expected image file format for available files is `an-image-name-without-the-path.png`. It should not include the path to the image file in the saved string.

**Parameters**

- **db** – TinyDB instance.
- **available_files** (*list*) – All of the existing files in the image directory.

**Returns** list

goldfinchsong.utils.**is_image_file**(*file_name*)
Checks for `.png`, `.jpg`, `.jgpeg`, `.gif` file extensions.

**Parameters** **file_name** (*str*) – Image file name.

**Returns** Returns `True` if the file name has image extension. `False` otherwise.

**Return type** bool

goldfinchsong.utils.**load_content**(*db*, *image_directory*, *text_conversions=None*)
Generates content tuple after selecting random image from image directory.

**Parameters**

- **db** – A TinyDB instance.
- **image_directory** (*str*) – File path to image directory.
- **text_conversions** (*dict*) – Keys represent full-length versions of a word. If the string value paired with the key is an abbreviated form that may be used by the function in an attempt to reduce the length of the candidate text.

**Returns** A content tuple with full image path, status text, and image file, if an image file is available in image directory. None otherwise.

**Return type** tuple or None

goldfinchsong.utils.**to_compact_text**(*candidate_text*, *maximum_length=117*, *text_conversions=None*)
Transforms a text string so that its length is equal to or less than a designated maximum length.

This is the sequence of transformations deployed:

•If the text is already equal to or under the maximum, no transformations are applied.

•Each text conversion is attempted; after each attempt, the length of the resulting text is checked and immediately returned if the transformed text does not exceed the maximum length.

•Non-boundary (i.e. internal to the word, so not the first or last letter) vowels are removed sequentially from the last word until the first. A length check occurs after each word transformation and the text is immediately returned if does not exceed the maximum.

•If the text is still too long, then words are deleted from last to first until the resulting text does not exceed the maximum length.

**Parameters**

- **candidate_text** (*str*) – Original text before function processing.
- **maximum_length** (*int*) – The maximum character length for the text.
- **text_conversions** (*dict*) – Keys represent full-length versions of a word. The string value paired with the key is an abbreviated form of the keyed word that may be used by the function in an attempt to reduce the length of the candidate text.

**Returns** A text string that shorter than the passed maximum length argument.

**Return type** str

goldfinchsong.utils.**trim_file_extension**(*file_name*)
Removes .png, .jpg, .jgpeg, .gif file extensions.

**Parameters** **file_name** (*str*) – Image file name.

**Returns** File name string without the image type extension.

**Return type** str

CHAPTER 3

# Indices and tables

- genindex
- modindex

## g

## A



## C



## D



## E



## G



## I



## L



## M



## P



## T