malaffinity

Release 2.5.2

Oct 06, 2017

Getting Started

Calculate affinity between MyAnimeList users

CHAPTER 1

Contents:

Introduction

What is this?

malaffinity provides a simple way to calculate affinity (Pearson's correlation * 100) between a "base" user and another user on MyAnimeList.

Note: The term "base user" refers to the user whose scores other users' scores will be compared to (and affinities to said scores calculated for).

Just assume the "base user" is referring to you, or whoever will be running your script, unless you're getting into some advanced mumbo-jumbo, in which case you're on your own.

malaffinity is meant to be used in bulk, where one user (the "base")'s scores are compared against multiple people, but there's nothing stopping you from using this as a one-off.

But why should I bother using this? Doesn't MAL give me an affinity?

Let's consider what you'd have to do if you wanted MAL to give you an affinity value, and a good estimation as to whether it's "accurate" or not:

- Create a `requests.Session()`
- Make a `GET` request to MAL's login page
- Retrieve the `csrf_token` from one of the `meta` headers
- Make a `POST` request to the login page, providing a username, password, a bunch of stupid form data, and the `csrf_token` you've just obtained
- Confirm you are logged in, by seeing if the `is_logged_in` cookie is present in the `CookieJar`
- Visit a users' profile
- Look for the affinity value (hint: `.user-compatability-graph .anime .bar-inner` [sic])
- Read its `innerHTML`, retrieve the affinity value, add a case in to get rid of the double-negative that appears on any negative value for some reason
- Find how many rated anime you share. The value MAL gives you includes unrated anime, and PTW stuff. It's not an accurate indicator
  - Visit `/shared.php?u1=you&u2=them` and you find yourself trying to navigate through the dark and murky world of bad HTML table parsing

Congrats, you've just wasted a few hours of your life, and you're probably a bit stressed right now. HTML parsing does that to you.

Let's see how you could handle all this with `malaffinity`, assuming your username is `Xinil` and you want to calculate affinity with `Luna`:

```
from malaffinity import MALAffinity

ma = MALAffinity("Xinil")

affinity, shared = ma.calculate_affinity("Luna")

# Do whatever you like with ``affinity`` and ``shared``
print(affinity)
# 37.06659111674594
print(shared)  # Note: Is referring to shared, rated anime
# 171
```

Note: `ma` now holds your scores. You can easily call `ma.calculate_affinity` on anyone, and you'd get your affinity with them.

If you don't want your scores to be stored, an option exists for quick, one-off calculations:

```
import malaffinity

affinity, shared = malaffinity.calculate_affinity("Xinil", "Luna")

# ...
```

I'm no expert, but the code(s) above looks a lot neater than the alternative would've looked.

Getting Started

Install

```
$ pip install malaffinity
```

Alternatively, download this repo and run:

```
$ python setup.py install
```

To use the development version (please don't), run:

```
$ pip install --upgrade https://github.com/erkghlerngm44/malaffinity/archive/master.
↪zip
```

Dependencies

- `BeautifulSoup4`
- `lxml`
- `Requests`

These should be installed when you install this package, so no need to worry about them.

`lxml` is a bit wonky sometimes. If install fails:

```
$ pip install --upgrade pip
$ pip install --upgrade lxml
```

If all else fails and you're on Windows, download the wheel yourself and:

```
$ pip install /path/to/wheel.whl
```

Development

This section demonstrates how documentation can be built, tests run, and how to check if you're adhering to **PEP 8** and **PEP 257**. These should not be used unless you're contributing to the package.

Conventions

The `flake8` and `pydocstyle` packages can be used to check that PEP 8 and PEP 257 are being followed respectively.

These can be installed as follows:

```
$ pip install .[conventions]
```

The following commands can then be run:

```
$ flake8
$ pydocstyle malaffinity
```

which will print any warnings/errors/other stuff. These should ideally be fixed, but in the event that they can't, place a `# noqa: ERROR_CODE` comment on the offending line(s).

Documentation

To install the dependencies needed to build the docs, run:

```
$ pip install .[docs]
```

The docs can then be built by navigating to the `docs` directory, and running:

```
$ make html
```

The built docs will now be in `./_build/html`. You can either run them by clicking and viewing them, or by running a server in that directory, which you can view in your browser.

Note: Any warnings that show up when building will be interpreted as errors when the tests get run on Travis, which will cause the build to fail. You'll want to make sure these are taken care of.

Test Suite

To install the dependencies needed for the test suite, run:

```
$ pip install .[tests]
```

It is advised to run the test suite through `coverage`, so a coverage report can be generated as well. To do this, run:

```
$ coverage run --source malaffinity setup.py test
```

The tests should then run. You can view the coverage report by running:

```
$ coverage report
```

Walkthrough

This section will show the various ways the *`MALAffinity`* class can be initialised with the user `Xinil` (MAL creator), and used to calculate affinity or get a comparison with the user `Luna` (MAL database admin).

Initialising the Class

The class can be initialised in either one of two ways:

Method 1: Normal initialisation

The class is initialised, with a "base user" passed as an argument to *`MALAffinity`*.

```
ma = MALAffinity("Xinil")
```

Method 2: Specifying a "base user" after initialisation

The class is initialised, with a "base user" passed sometime later after initialisation, which may be useful in scripts where creating globals inside functions or classes or different files is a pain.

```
ma = MALAffinity()

# This can be done anywhere, as long as it has access to ``ma``,
# but MUST be done before ``calculate_affinity`` or ``comparison``
# are called
ma.init("Xinil")
```

Rounding of the final affinity value

Note: This doesn't affect *comparison()*, so don't worry about it if you're just using that.

Do note that the class also has a `round` parameter, which is used to round the final affinity value. This must be specified at class initialisation if wanted, as it isn't available in *init()*. A value for this can be passed as follows:

```
# To round to two decimal places
ma = MALAffinity("Xinil", round=2)

# Alternatively, the following can also work, if you decide to follow
# method 2 for initialising the class
ma = MALAffinity(round=2)
ma.init("Xinil")
```

Doing Things with the Initialised Class

The initialised class, now stored in `ma`, can now perform the following actions:

Calculate affinity with a user

Note: Values may or may not be rounded, depending on the value you passed for the `round` parameter at class initialisation.

```
print(ma.calculate_affinity("Luna"))
# Affinity(affinity=37.06659111674594, shared=171)
```

Note that what is being returned is a namedtuple, containing the affinity and shared rated anime. This can be separated into different variables as follows:

```
affinity, shared = ma.calculate_affinity("Luna")

print(affinity)
# 37.06659111674594
print(shared)
# 171
```

Alternatively, the following also works (as this is a namedtuple):

```
affinity = ma.calculate_affinity("Luna")

print(affinity.affinity)
# 37.06659111674594
print(affinity.shared)
# 171
```

Comparing scores with a user

```
comparison = ma.comparison("Luna")

print(comparison)
# Note: this won't be prettified for you. Run it
# through a prettifier if you want it to look nice.
# {
#     1: [10, 6],
#     5: [8, 6],
#     6: [10, 7],
#     15: [7, 9],
#     16: [8, 5],
#     ...
# }
```

This can now be manipulated in whatever way you like, to suit your needs. I like to just get the arrays on their own, zip them and plot a graph with it.

Extras

Warning: These send two GET requests over to MAL in a short amount of time, with no wait inbetween them. If you're getting in trouble with them for breaking their rate limit, you might have a few problems getting these to work without *exceptions.MALRateLimitExceededError* getting raised.

Note: Don't use these if you're planning on calculating affinity or getting a comparison again with one of the users you've specified when using these.

It's better to create an instance of the *MALAffinity* class with said user, and using that with the other user(s) that way.

That instance will hold said users' scores, so they won't have to be retrieved again. See the other examples.

One-off affinity calculations

This is mainly used if you don't want the "base user"'s scores saved to a variable, and you're only interested in the affinity with one person.

```
# Note that ``round`` can also be specified here if needed.
affinity, shared = calculate_affinity("Xinil", "Luna")

print(affinity)
# 37.06659111674594
print(shared)
# 171


# Alternatively...
affinity = calculate_affinity("Xinil", "Luna")

print(affinity.affinity)
# 37.06659111674594
print(affinity.shared)
# 171
```

One-off comparison of scores

This is mainly used if you don't want the "base user"'s scores saved to a variable, and you're only interested in getting a comparison of scores with another user.

```
print(comparison("Xinil", "Luna"))

# Note: this won't be prettified for you. Run it
# through a prettifier if you want it to look nice.
# {
#     1: [10, 6],
#     5: [8, 6],
#     6: [10, 7],
#     15: [7, 9],
#     16: [8, 5],
#     ...
# }
```

API

class `malaffinity.`**`MALAffinity`**(*base_user=None*, *round=False*)

The MALAffinity class.

The purpose of this class is to store a "base user"'s scores, so affinity with other users can be calculated easily.

For the user `Xinil`, the class can be initialised as follows:

```
from malaffinity import MALAffinity

ma = MALAffinity("Xinil")
```

The instance, stored in `ma`, will now hold `Xinil`'s scores.

comparison() and *calculate_affinity()* can now be called, to perform operations on this data.

__init__(*base_user=None*, *round=False*)
Initialise an instance of *MALAffinity*.

Note: To avoid dealing with dodgy globals, this class MAY be initialised without the `base_user` argument, in the global scope (if you wish), but *init()* MUST be called sometime afterwards, with a `base_user` passed, before affinity calculations take place.

Example (for the user `Xinil`):

```
from malaffinity import MALAffinity

ma = MALAffinity()

ma.init("Xinil")
```

The class should then be good to go.

Parameters

- **base_user** (*str or None*) – Base MAL username
- **round** (*int or False*) – Decimal places to round affinity values to. Specify `False` for no rounding

calculate_affinity(*username*)
Get the affinity between the "base user" and `username`.

Note: The data returned will be a namedtuple, with the affinity and shared rated anime. This can easily be separated as follows (using the user `Luna` as `username`):

```
affinity, shared = ma.calculate_affinity("Luna")
```

Alternatively, the following also works:

```
affinity = ma.calculate_affinity("Luna")
```

with the affinity and shared available as `affinity.affinity` and `affinity.shared` respectively.

Note: The final affinity value may or may not be rounded, depending on the value of `_round`, set at class initialisation.

Parameters **username** (*str*) – The username to calculate affinity with

Returns (float affinity, int shared)

Return type tuple

comparison(*username*)
Get a comparison of scores between the "base user" and `username`.

A Key-Value returned will consist of the following:

```
{
    ANIME_ID: [BASE_USER_SCORE, OTHER_USER_SCORE],
    ...
}
```

Example:

```
{
    30831: [3, 8],
    31240: [4, 7],
    32901: [1, 5],
    ...
}
```

Warning: The JSON returned isn't valid JSON. The keys are stored as integers instead of the JSON standard of strings. You'll want to force the keys to strings if you'll be using the ids elsewhere.

Parameters **username** (*str*) – The username to compare the base users' scores to

Returns Key-value pairs as described above

Return type dict

init(*base_user*)

Retrieve a "base user"'s list, and store it in `_base_scores`.

Parameters **base_user** (*str*) – Base users' username

Handling Exceptions

Which exceptions can be raised?

The types of exceptions that can be raised when calculating affinities are:

exception malaffinity.exceptions.**NoAffinityError**

Raised when either the shared rated anime between the base user and another user is less than 11, the user does not have any rated anime, or the standard deviation of either users' scores is zero.

exception malaffinity.exceptions.**InvalidUsernameError**

Raised when username specified does not exist.

exception malaffinity.exceptions.**MALRateLimitExceededError**

Raised when MAL's blocking your request, because you're going over their rate limit of one request every two seconds. Slow down and try again.

If you're planning on using this package in an automated or unsupervised script, you'll want to make sure you account for these getting raised, as not doing so will mean you'll be bumping into a lot of exceptions, unless you can guarantee none of the above will get raised. For an example snippet of code that can demonstrate this, see *Exception Handling Snippet*.

MALAffinityException

exceptions.NoAffinityError and *exceptions.InvalidUsernameError* are descendants of:

exception malaffinity.exceptions.**MALAffinityException**
Base class for MALAffinity exceptions.

which means if that base exception gets raised, you know you won't be able to calculate affinity with that person for some reason, so your script should just move on.

What to do if MALRateLimitExceededError gets raised

exceptions.MALRateLimitExceededError rarely gets raised if you abide by the rate limit of one request every two seconds. If it does get raised, the following should happen:

- Halt the script for a few seconds. I recommend five.
- Try again.
- If you get roadblocked again, just give up. MAL obviously hates you.

Exception Handling Snippet

The above can be demonstrated via something along these lines. Do note that this probably isn't the best method, but it works.

This should be placed in the section where you are attempting to calculate affinity with another user. Because I wrote this before *MALAffinity.comparison()* was created, the snippet only shows how you can apply this to calculating affinities, but it can easily be modified, should you wish, to get a comparison of scores.

```
time.sleep(2)

success = False

for _ in range(2):
    try:
        affinity, shared = ma.calculate_affinity("OTHER_USERNAME")

    # Rate limit exceeded. Halt your script and try again
    except malaffinity.exceptions.MALRateLimitExceededError:
        time.sleep(5)
        continue

    # Any other malaffinity exception.
    # Affinity can't be calculated for some reason.
    # ``MALAffinityException`` is the base exception class for
    # all malaffinity exceptions
    except malaffinity.exceptions.MALAffinityException:
        break

    # Exceptions not covered by malaffinity. Not sure what
    # you could do here. Feel free to handle however you like
    except Exception as e:
        print("Something went wrong. Please contact Xinil for further assistance:")
```

```
        print("* https://myanimelist.net/profile/Xinil")
        print("* https://www.reddit.com/user/Xinil")
        print("Please also nag him to create a half-decent MAL API for gods sake.")
        print("")
        print("Exception: `{}`".format(e))
        break

    # Success!
    else:
        success = True
        break

# ``success`` will still be ``False`` if affinity can't been calculated.
# If this is the case, you'll want to stop doing anything with this person
# and move onto the next, so use the statement that will best accomplish this,
# given the layout of your script
if not success:
    return

# Assume from here on that ``affinity`` and ``shared`` hold their corresponding
# values, and feel free to do whatever you want with them
```

Feel free to use a `while` loop instead of the above. I'm just a bit wary of them, in case something happens and the script gets stuck in an infinite loop. Your choice.

To see the above snippet in action, visit erkghlerngm44/r-anime-soulmate-finder.

Changelog

v2.5.2 (2017-10-06)

- Happy birthday to me.
- Correct incorrect information in the `NoAffinityError` docstring, which stated that the minimum shared rated anime threshold for calculating affinity was ten, when it's actually eleven.

v2.5.1 (2017-09-14)

- Add the `conventions` section to `extras_require`, so conventions dependencies can easily be installed if needed.
- Add the `travis` section to `extras_require`, so Travis can easily install any additional dependencies it needs.
- Rename the `doc` and `test` sections in `extras_require` to `docs` and `tests` respectively.

v2.5.0 (2017-08-26)

- Create the `Affinity` namedtuple, and return that instead of a normal tuple when calculating affinity.
- Rename the "their_list" var in `MALAffinity.calculate_affinity` to "user_list". It's an improvement, but still a bad var name.

v2.4.0 (2017-08-17)

- Tidy up the MALAffinity `__repr__`, and dynamically specify the class name, in case the class needs to be renamed (which it probably won't).
- Use double backticks instead of a single backtick in the `calculate_affinity` function's docstring.
- Add a one-off comparison function (+ docs).
- Move the "shared rated anime threshold" check to the `calculate_affinity` method, as a comparison can be done no matter how many shared rated anime two users share.
- Add the error/warning codes to ignore next to the `# noqa` comments.
- Rewrite the contributing page in the docs, and create `CONTRIBUTING.rst` to link to it.

v2.3.1 (2017-08-12)

- Use `Xinil` and `Luna`'s usernames to demonstrate how MALAffinity can be used, in the docstring examples.
- Use different anime in the `comparison` method's key:value example (in the docstring), so a wider range of ratings can be shown.

v2.3.0 (2017-07-25)

- Rewrite `calcs.pearson` to make it 5x faster.
- Do the "standard deviation is zero" checking in `calcs.pearson`, so the standard deviation doesn't actually have to be calculated.
- Don't make `MALRateLimitExceededError` a descendant of `MALAffinityException`.

v2.2.3 (2017-07-20)

- Move test dependencies to `extras_require` in setup.py, and have `tests_require` mirror that.
- Add documentation dependencies to `extras_require` in setup.py.
- Make all docstrings PEP 257 compliant.
- Add more classifiers to `setup.py`.

v2.2.2 (2017-07-17)

- Have Travis deploy the package to PyPI when a release is drafted.

This release exists solely to test that the above is working. No code changes have been made.

v2.2.1 (2017-07-17)

- Fix tests up to use the `mock` module, and be less hacky.
- Hook tests up to `setup.py`, so `python setup.py test` can be called, for the test suite to run.
- Add version constraints in for dependencies.

v2.2.0 (2017-07-12)

- Create the `comparison` method, and have `calculate_affinity` use that to retrieve both sets of scores.
- Rewrite the docstrings in the `MALAffinity` class, to be more useful and more compliant with le Sphinx syntax.
- Add docs to the project, and have them hosted on readthedocs.io.
- Add (badly written) tests, and hook them up to Travis and Coveralls.
- Get rid of Markdown altogether, and rewrite the README, as most of the info is now in the docs.

v2.1.0 (2017-07-08)

- Fix a typo in the `calcs.pearson` docstring, which incorrectly said the `:rtype` was a bool.
- Use the `find_all` BS4 method instead of `findAll`.
- Fix a typo in the "Shared rated anime count is less than required" exception message, which incorrectly stated that the minimum required was ten, when it's actually eleven.
- Add a docstring for `.__about__`.
- Add a docstring for `MALAffinity._retrieve_scores`.
- Remove the useless kwargs from `MALAffinity._retrieve_scores`.
- Make the `statistics` pypi package a requirement for all Python versions.
- Add a `__repr__` to the `MALAffinity` class.
- Move the `_retrieve_scores` method in the `MALAffinity` class to its own file (`endpoints.py`).
- Create a `const.py` file for constants.
- Add a `# NOQA` comment to the `.__about__` imports in `__init__`, to suppress the F401 flake8 warnings.

v2.0.0 (2017-06-20)

- Move the MALAffinity class to its own separate file (`malaffinity.malaffinity`) and import that into the `malaffinity` namespace via `__init__`.
- Move exceptions to its own separate file (`malaffinity.exceptions`).
- Modify description of the package slightly ("Calculate affinity between **two** MyAnimeList users" => "Calculate affinity between MyAnimeList users").
- Add exception message for when the standard deviation of one of the two users' scores is zero, and affinity can't be calculated.
- Create the base `MALAffinityException` class and derive all malaffinity exceptions from that.
- Add docstrings for `malaffinity.calcs`.
- Modify docstrings to remove typos and unnecessary information, and reword some sections.
- Reword exception messages to be more useful.
- Have the `init` method return `self`, to allow for chaining.
- Make all code PEP8-compliant (ignoring F401 for meta reasons).

v1.1.0 (2017-06-15)

- Remove scipy (and numpy) as a dependency. Pearson's correlation code is now in `malaffinity.calcs` and stdev checking is handled by the `statistics` module.
- Use `lxml` for XML parsing, instead of the default `html.parser`.
- Add return types for components inside the return tuple into the docstring.

v1.0.3 (2017-05-05)

- Change 'base user has been set' testing to also check if `self._base_scores` has been set as well.
- Use `zip` to create the `scores1` and `scores2` arrays that calculations are done with.
- Check if the standard deviation of `scores1` or `scores2` is zero, and thrown an error if so.
- Use `scipy.asscalar` as opposed to `.item()` for numpy.float64 => float conversion.

v1.0.2 (2017-04-17)

- Better handling for numpy.float64 => float conversion.
- Update docstrings to include types.

v1.0.1 (2017-04-12)

- Don't count rated anime on a user's PTW. MAL didn't count this, so our affinity values were a bit off when a user did this.

v1.0.0 (2017-04-09)

- Konnichiwa, sekai!

Contributing

In the unlikely event that someone finds this package, and in the even unlikelier event that someone wants to contribute, send me a pull request or create an issue.

Note: Please *Contact* and notify me if you use the above, as this isn't my main GitHub account, so I won't be checking it that much. I'll probably see it weeks/months later if you don't.

Feel free to use those for anything regarding the package, they're there to be used, I guess.

How to Contribute

- Fork the repo.
- `git clone https://github.com/YOUR_USERNAME/malaffinity.git`
- `cd malaffinity`

- `git checkout -b new_feature`
- Make changes.
- `git commit -am "Commit message"`
- `git push origin new_feature`
- Navigate to https://github.com/YOUR_USERNAME/malaffinity
- Create a pull request.

Notes and Stuff

I had a whole section on conventions to follow and other stuff, but that seemed a bit weird, so I just scratched it. If someone out there wants to contribute to this package in any way, shape or form, have at it. I'd prefer the changes to be non-breaking (i.e. existing functionality is not affected), but breaking changes are still welcome.

I only ask that you try to adhere to **PEP 8** and **PEP 257** (if you can), and try to achieve 100% coverage in tests (again, if you can). For information on how to check if you're adhering to those conventions, see *Conventions*.

For information on how to build docs and run tests, see *Documentation* and *Test Suite* respectively.

This package is based off a class I wrote for `erkghlerngm44/r-anime-soulmate-finder`, and while I have tried to modify it for general uses (and tried to clean the bad code up a bit), there are still a few iffy bits around. I'd appreciate any PRs to fix this up.

I think the package is mostly complete, so my main focus right now is making it as fast as can-be, as every fraction of a second counts when you're using this to calculate affinity with tens of thousands of people. PRs regarding this are especially welcome.

That's it, I guess. *Contact* me if you need help or anything.

Contact

On the off chance that someone wants to contact me, I can be reached via the following (ordered from fastest to slowest in terms of time it'll take to get a response from me):

- Discord (erkghlerngm44#9210)
- Reddit (/u/erkghlerngm44)
- Email (erkghlerngm44@protonmail.com)

Note: Emailing me is pretty much pointless, since I rarely check that address. Contact me on Discord or Reddit if you need anything.

License

```
MIT License

Copyright (c) 2017 erkghlerngm44

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Index

Symbols

C

I

M

N

P