granary Documentation

Release 2.0

Ryan Barrett

Mar 01, 2019

Contents

CHAPTER 1

About

Granary is a library and REST API that fetches and converts between a wide variety of data sources and formats:

- Facebook, Flickr, GitHub, Instagram, and Twitter native APIs
- Instagram scraped HTML
- ActivityStreams 1.0 and 2.0 (JSON)
- HTML and JSON with microformats2
- Atom, RSS 2.0, JSON Feed
- Plain XML

Free yourself from silo API chaff and expose the sweet social data foodstuff inside in standard formats and protocols!

Here's how to get started:

- Granary is available on PyPi. Install with `pip install granary`.
- Supports Python 2.7+ and 3.3+.
- *Click here for getting started docs.*
- Click here for reference docs.
- The REST API and demo app are deployed at granary.io.

CHAPTER 2

Using

All dependencies are handled by pip and enumerated in requirements.txt. We recommend that you install with pip in a virtualenv. (App Engine details.)

The library and REST API are both based on the OpenSocial Activity Streams service.

Let's start with an example. This code using the library:

```
from granary import twitter
...
tw = twitter.Twitter(ACCESS_TOKEN_KEY, ACCESS_TOKEN_SECRET)
tw.get_activities(group_id='@friends')
```

is equivalent to this `HTTP GET` request:

```
https://granary.io/twitter/@me/@friends/@app/
  ?access_token_key=ACCESS_TOKEN_KEY&access_token_secret=ACCESS_TOKEN_SECRET
```

They return the authenticated user's Twitter stream, ie tweets from the people they follow. Here's the JSON output:

```
{
  "itemsPerPage": 10,
  "startIndex": 0,
  "totalResults": 12,
  "items": [{
      "verb": "post",
      "id": "tag:twitter.com,2013:374272979578150912",
      "url": "http://twitter.com/evanpro/status/374272979578150912",
      "content": "Getting stuff for barbecue tomorrow. No ribs left! Got some nice␣
↪tenderloin though. (@ Metro Plus Famille Lemay) http://t.co/b2PLgiLJwP",
      "actor": {
      "username": "evanpro",
        "displayName": "Evan Prodromou",
        "description": "Prospector.",
        "url": "http://twitter.com/evanpro",
      },
```

(continues on next page)

(continued from previous page)

```
      "object": {
        "tags": [{
            "url": "http://4sq.com/1cw5vf6",
            "startIndex": 113,
            "length": 22,
            "objectType": "article"
          }, "..."],
      },
    }, "..."]
  "..."
}
```

The request parameters are the same for both, all optional: `USER_ID` is a source-specific id or `@me` for the authenticated user. `GROUP_ID` may be `@all`, `@friends` (currently identical to `@all`), `@self`, `@search`, or `@blocks`; `APP_ID` is currently ignored; best practice is to use `@app` as a placeholder.

Paging is supported via the `startIndex` and `count` parameters. They're self explanatory, and described in detail in the OpenSearch spec and OpenSocial spec.

When using the `GROUP_ID` `@search` (for platforms that support it — currently Twitter and Instagram), provide a search string via the `q` parameter. The API is loosely based on the OpenSearch spec, the OpenSocial Core Container spec, and the OpenSocial Core Gadget spec.

Output data is JSON Activity Streams 1.0 objects wrapped in the OpenSocial envelope, which puts the activities in the top-level `items` field as a list and adds the `itemsPerPage`, `totalCount`, etc. fields.

Most Facebook requests and all Twitter, Instagram, and Flickr requests will need OAuth access tokens. If you're using Python on Google App Engine, oauth-dropins is an easy way to add OAuth client flows for these sites. Otherwise, here are the sites' authentication docs: Facebook, Flickr, Instagram, Twitter.

If you get an access token and pass it along, it will be used to sign and authorize the underlying requests to the sources providers. See the demos on the REST API *endpoints above* for examples.

CHAPTER 3

Using the REST API

The *endpoints above* all serve the OpenSocial Activity Streams REST API. Request paths are of the form:

```
/USER_ID/GROUP_ID/APP_ID/ACTIVITY_ID?startIndex=...&count=...&format=FORMAT&access_
↪token=...
```

All query parameters are optional. `FORMAT` may be `as1` (the default), `as2`, `atom`, `html`, `jsonfeed`, `mf2-json`, `rss`, or `xml` (the default). `atom` supports a boolean `reader` query parameter for toggling rendering appropriate to feed readers, e.g. location is rendered in content when `reader=true` (the default). The rest of the path elements and query params are *described above*.

Errors are returned with the appropriate HTTP response code, e.g. 403 for Unauthorized, with details in the response body.

By default, responses are cached and reused for 10m without re-fetching the source data. (Instagram responses are cached for 60m.) You can prevent this by adding the `cache=false` query parameter to your request.

To use the REST API in an existing ActivityStreams client, you'll need to hard-code exceptions for the domains you want to use e.g. `facebook.com`, and redirect HTTP requests to the corresponding *endpoint above*.

The web UI (granary.io) currently only fetches Facebook access tokens for users. If you want to use it to access a Facebook page, you'll need to get an access token manually with the Graph API Explorer (click on the *Get To...* drop-down) . Then, log into Facebook on granary.io and paste the page access token into the `access_token` text box.

CHAPTER 4

Using the library

See the *example above* for a quick start guide.

Clone or download this repo into a directory named `granary` (note the underscore instead of dash). Each source works the same way. Import the module for the source you want to use, then instantiate its class by passing the HTTP handler object. The handler should have a `request` attribute for the current HTTP request.

The useful methods are `get_activities()` and `get_actor()`, which returns the current authenticated user (if any). See the individual method docstrings for details. All return values are Python dicts of decoded ActivityStreams 1 JSON.

The `microformats2.*_to_html()` functions are also useful for rendering ActivityStreams 1 objects as nicely formatted HTML.

CHAPTER 5

Troubleshooting/FAQ

Check out the oauth-dropins Troubleshooting/FAQ section. It's pretty comprehensive and applies to this project too. For searchability, here are a handful of error messages that have solutions there:

```
bash: ./bin/easy_install: ...bad interpreter: No such file or directory

ImportError: cannot import name certs

ImportError: cannot import name tweepy

File ".../site-packages/tweepy/auth.py", line 68, in _get_request_token
  raise TweepError(e)
TweepError: must be _socket.socket, not socket
```

CHAPTER 6

Future work

We'd love to add more sites! Off the top of my head, YouTube, Tumblr, WordPress.com, Sina Weibo, Qzone, and RenRen would be good candidates. If you're looking to get started, implementing a new site is a good place to start. It's pretty self contained and the existing sites are good examples to follow, but it's a decent amount of work, so you'll be familiar with the whole project by the end.

CHAPTER 7

Development

Pull requests are welcome! Feel free to ping me with any questions.

You'll need the App Engine Python SDK version 1.9.15 or later (for vendor support). Add it to your `$PYTHONPATH`, e.g. `export PYTHONPATH=$PYTHONPATH:/usr/local/google_appengine`, and then run:

```
virtualenv local
source local/bin/activate
pip install -r requirements.txt
python setup.py test
```

If you send a pull request, please include (or update) a test for the new functionality if possible! The tests require the App Engine SDK or the Google Cloud SDK (aka `gcloud`) with the `gcloud-appengine-python` and `gcloud-appengine-python-extras` components.

If you want to work on oauth-dropins at the same time, install it in "source" mode with `pip install -e <path to oauth-dropins repo>`.

To deploy:

```
python -m unittest discover && gcloud -q app deploy granary-demo *.yaml
```

To deploy facebook-atom, twitter-atom, instagram-atom, and plusstreamfeed after a granary change:

```
#!/bin/tcsh
foreach s (facebook-atom twitter-atom instagram-atom plusstreamfeed)
  cd ~/src/$s && gcloud -q app deploy $s *.yaml
end
```

The docs are built with Sphinx, including apidoc, autodoc, and napoleon. Configuration is in docs/conf.py To build them, first install Sphinx with `pip install sphinx`. (You may want to do this outside your virtualenv; if so, you'll need to reconfigure it to see system packages with `virtualenv --system-site-packages local`.) Then, run docs/build.sh.

This ActivityStreams validator is useful for manual testing.

CHAPTER 8

Release instructions

Here's how to package, test, and ship a new release. (Note that this is largely duplicated in the oauth-dropins readme too.)

1. Run the unit tests. ‘sh source local/bin/activate.csh python2 -m unittest discover deactivate

   source local3/bin/activate.csh python3 -m unittest discover -s granary/tests/ deactivate ‘‘‘

2. Bump the version number in `setup.py` and `docs/conf.py`. `git grep` the old version number to make sure it only appears in the changelog. Change the current changelog entry in `README.md` for this new version from *unreleased* to the current date.

3. Build the docs. If you added any new modules, add them to the appropriate file(s) in `docs/source/`. Then run `./docs/build.sh`.

4. `git commit -am 'release vX.Y'`

5. Upload to test.pypi.org for testing. `sh python3 setup.py clean build sdist twine upload -r pypitest dist/granary-X.Y.tar.gz`

6. Install from test.pypi.org, both Python 2 and 3. `sh cd /tmp virtualenv local source local/bin/activate.csh pip install -i https://test.pypi.org/simple --extra-index-url https://pypi.org/simple granary deactivate sh python3 -m venv local3 source local3/bin/activate.csh pip3 install --upgrade pip pip3 install -i https://test.pypi.org/simple --extra-index-url https://pypi.org/simple granary deactivate`

7. Smoke test that the code trivially loads and runs, in both Python 2 and 3.

```
source local/bin/activate.csh
python2
# run test code below
deactivate
```

```
source local3/bin/activate.csh
python3
```

(continues on next page)

(continued from previous page)

```
# run test code below
deactivate
```

Test code to paste into the interpreter: ‘py from granary import instagram instagram.__file__ # check that it’s in the virtualenv

i = instagram.Instagram() a = i.get_activities(user_id=‘snarfed’, group_id=‘@self’, scrape=True) print(json.dumps(a, indent=2))

from granary import atom print(atom.activities_to_atom(a, {}))

from granary import github g = github.GitHub(‘XXX’) # insert a GitHub personal OAuth access token a2 = g.get_activities() print(json.dumps(a2, indent=2)) ‘‘‘

8. Tag the release in git. In the tag message editor, delete the generated comments at bottom, leave the first line blank (to omit the release “title” in github), put `### Notable changes` on the second line, then copy and paste this version’s changelog contents below it. `sh git tag -a vX.Y --cleanup=verbatim git push git push --tags`
9. Click here to draft a new release on GitHub. Enter `vX.Y` in the *Tag version* box. Leave *Release title* empty. Copy `### Notable changes` and the changelog contents into the description text box.
10. Upload to pypi.org! `sh twine upload dist/granary-X.Y.tar.gz`

CHAPTER 9

Related work

Apache Streams is a similar project that translates between storage systems and database as well as social schemas. It's a Java library, and its design is heavily structured. Here's the list of formats it supports. It's mainly used by People Pattern.

Gnip similarly converts social network data to ActivityStreams and supports many more source networks. Unfortunately, it's commercial, there's no free trial or self-serve signup, and plans start at $500.

DataSift looks like broadly the same thing, except they offer self-serve, pay as you go billing, and they use their own proprietary output format instead of ActivityStreams. They're also aimed more at data mining as opposed to individual user access.

Cliqset's FeedProxy used to do this kind of format translation, but unfortunately it and Cliqset died.

Facebook used to officially support ActivityStreams, but that's also dead.

There are a number of products that download your social network data, normalize it, and let you query and visualize it. SocialSafe is one, although the SSL certificate is currently out of date. ThinkUp was an open source product, but shuttered on 18 July 2016. There's also the lifelogging/lifestream aggregator vein of projects that pull data from multiple source sites. Storytlr is a good example. It doesn't include Facebook, or Instagram, but does include a number of smaller source sites. There are lots of others, e.g. the Lifestream WordPress plugin. Unfortunately, these are generally aimed at end users, not developers, and don't usually expose libraries or REST APIs.

On the open source side, there are many related projects. php-mf2-shim adds microformats2 to Facebook and Twitter's raw HTML. sockethub is a similar "polyglot" approach, but more focused on writing than reading.

CHAPTER 10

Changelog

10.1 2.0 - 2019-03-01

Breaking change: drop Google+ since it shuts down in March. Notably, this removes the `googleplus` module.

10.2 1.15 - 2019-02-28

- Add RSS 2.0 output! (#124)
- All silos:
  - Switch users' primary URLs from web site to silo profile (#158).
- GitHub:
  - Don't enclose bare URLs in </> (snarfed/bridgy#850).
- Atom:
  - Bug fix for actors and attachments with multiple image URLs.
  - Bug fix for attachment author objects with no properties.
- Google+:
  - Drop from web UI and REST API since consumer Google+ is shutting down entirely (more).
  - Switch from deprecated global API endpoint to G+ endpoint. Background in snarfed/bridgy#846, Google blog post and docs.
- Instagram:
  - Fix individual photo/video link urls for multi-photo/video posts.
  - Handle user-provided alt text (#159).
- Twitter:

  - Update max video upload size from 5MB to 512MB (#162).
- `/url`: Return HTTP 400 when fetching the user's URL results in an infinite redirect.

10.3 1.14 - 2018-11-12

Add `delete()`. Currently includes Twitter and Flickr support. * Instagram: * Make extra HTTP fetch (with cookie) to get individual likes (snarfed/bridgy#840). * Update scraping logic to handle feed HTML changes. * Link @-mentions in comments as well as photo/video captions. * GitHub: * `create/preview_create` bug fixes for issues and comments on private repos. * Handle HTTP 410 Gone responses from REST API, eg when a repo has been deleted or issues for the repo disabled. * Twitter: * Add `delete()` and `preview_delete()` for deleting tweets. * Flickr: * Add `delete()` and `preview_delete()` for deleting photos. * microformats2: * Add follow-of support. * Only use quotation-of property for quote tweets, not URLs. (#155) * If a tag has startIndex/length, it gets linkified in the content, so don't also emit an mf2 child or HTML h-cite for it. (#155 * Atom: * Encode `&s` in author URL and email address too. (Thanks sebsued!) * AS2: * Add `Follow` support.

10.4 1.13 - 2018-08-08

- Twitter:
  - Support ISO 8601 formatted created_at timestamps, which the archive download uses, as well as RFC 2822 from the API.
  - `create()` and `preview_create()`: support RSVPs. Tweet them as normal tweets with the RSVP content. (snarfed/bridgy#818)
  - `create()` and `preview_create()`: support alt text for images, via AS1 `displayName`. (snarfed/bridgy#756).
- Instagram:
  - Add global rate limiting lock for scraping. If a scraping HTTP request gets a 429 or 503 response, we refuse to make more requests for 5m, and instead short circuit and return the same error. This can be overridden with a new `ignore_rate_limit` kwarg to `get_activities()`.
- GitHub:
  - Add `tag` support to `create/preview_create` to add label(s) to existing issues (snarfed/bridgy#811).
  - Escape HTML characters (`<`, `>`, and `&`) in content in `create()` and `preview_create()` (snarfed/bridgy#810).
  - `get_activities()` and `get_comment()` now return `ValueError` instead of `AssertionError` on malformed `activity_id` and `comment_id` args, respectively.
  - `get_activities()` bug fix for issues/PRs with no body text.
  - Switch from GraphQL to REST API for creating comments and reactions, since GraphQL hits authorization errors on many org repos. (snarfed/bridgy#824)
  - Improve GraphQL support for comments and users.
- Atom:
  - Shorten and ellipsize feed title when necessary (#144).
- microformats2:

  - Upgrade mf2py to improve a few things like implied p-name detection and whitespace handling (#142, fixes #145, snarfed/bridgy#756, snarfed/bridgy#828).
  - Support `alt` attribute in `<img>` tags (snarfed/bridgy#756).

10.5 1.12 - 2018-03-24

- Add Python 3 support! Granary now requires either Python 2.7+ or Python 3.3+.
- Instagram:
  - Fix scraping profile pages.
- Twitter:
  - Update character counting to handle Twitter change that now auto-links *all* ccTLDs. Background.
- GitHub:
  - Bug fix for `get_activities()` with deleted issues and repos.
- microformats2:
  - `object_to_json()`: convert tags to simple strings in the `category` property, not full nested objects like `h-cards` (#141).
  - Special case GitHub issues that are in-reply-to a repo or its `/issues` URL to be objectType `issue`.
  - Render simple string categories in HTML output.

This release is intentionally small and limited in scope to contain any impact of the Python 3 migration. It *should* be a noop for existing Python 2 users, and we've tested thoroughly, but I'm sure there are still bugs. Please file issues if you notice anything broken!

10.6 1.11 - 2018-03-09

- Add GitHub!
  - `get_activities()` supports issues and pull requests, including comments and reactions. It's currently based on notifications, so it's best effort, not comprehensive, and only includes recently active issues/PRs.
  - `create()` and `preview_create()` support issues, comments, stars, and reactions.
- Twitter:
  - Prefer MP4 and other video/... content types to HLS (.m3u8) etc. Background.
  - Prefer HTTPS URLs for media images.
  - `get_activities()`: Support @-prefixed usernames in `user_id`.
- Facebook:
  - Support new recurring aka multi-instance events. `create()` and `preview_create()` now only support RSVPs to individual instances of multi-instance events, to match the Facebook API itself.
  - Try harder to find original (full) sized photo URLs, specifically `_o.jpg` files instead of `_s.jpg`.
  - `create()` bug fix for photo and image URLs with unicode characters.
  - Fixed bug where `get_activities(user_id=...)` included the authenticated user's own recent photos, albums, and news publishes.

- Instagram:
  - Extract more user (`author`) data from scraped profile pages.
  - Fix home page feed scraping.
- microformats2, Atom:
  - Add enclosures for image attachments.
  - Bug fixes for rendering image, video, and audio attachments inside shares and attachments. De-dupe images.
- microformats2:
  - Handle simple string-only author properties.
  - Add `fetch_mf2` kwarg to `json_to_object()` for fetching additional pages over HTTP to determine authorship.
  - Generate explicit blank `p-name` in HTML to prevent old flawed implied p-name handling (#131).
  - Fix `share` verb handling in `activity_to_json()` and `activities_to_html()` (#134).
  - Remember which content contains HTML, preserve newlines in it, and don't translate those newlines to `<br>`s (#130).
- Atom:
  - Fix timezone bugs in `updated` and `published`.
- JSON Feed:
  - Omit title from items if it's the same as the content. (Often caused by microformats2's implied `p-name` logic.)

10.7 1.10 - 2017-12-10

- Moved web site and REST API to granary.io! granary-demo.appspot.com now 301 redirects.
- Twitter:
  - Update the publish character limit to 280. Background.
  - Fix a bug in preview_create that auto-linked @-mentions inside URLs, e.g. Medium posts.
  - Support videos and animated GIFs in `get_activities()` etc.
- Instagram:
  - Add cookie query param to REST API to allow scraping that logged in user's feed.
- HTML (including Atom content):
  - Render image, video, and audio attachments more often and consistently.
  - Include microformats2 `u-photo`, `u-video`, and `u-audio` classes more often and consistently.
- Atom:
  - Add `atom_to_activities()` for converting full feed documents.
  - Add to REST API and web UI.
  - Include source URL in `rel=alternate` link as well as actor/author URL (#151).
- JSON Feed:

  - Fix bug that omitted title in some cases (#122).

10.8 1.9 - 2017-10-24

- Add ActivityStreams 2.0! New `as2` module includes `to_as1()` and `from_as1()` functions. Currently supported: articles, notes, replies, likes, reposts, events, RSVPs, tags, attachments.
- Atom:
  - Add new `atom_to_activity()` function for converting Atom to AS1.
  - Add email field to author, if provided.
- JSON Feed:
  - Raise ValueError on bad (non-dict) input.
- REST API:
  - Add `as2` value for `format` and `input`. Revise existing ActivityStreams and microformats2 value names to `as1`, `as1-xml`, and `mf2-json`. Old values `activitystreams`, `json`, `json-mf2`, and `xml` are still accepted, but deprecated.

10.9 1.8 - 2017-08-29

- Add JSON Feed support to both library and REST API.
- Twitter:
  - Add `get_blocklist()`.
  - Bug fix for creating replies, favorites, or retweets of video URLs, e.g. https://twitter.com/name/status/123/video/1 .
  - Bug fix for parsing favorites HTML to handle a small change on Twitter's side.
  - `post_id()` now validates ids more strictly before returning them.
- Facebook:
  - Improve heuristic for determining privacy of wall posts from other users.
  - Support GIFs in comments (attachment types `animated_image_autoplay` and `animated_image_share`).
  - Upgrade Graph API from v2.6 to v2.10.
- Instagram:
  - Update scraping to handle new home page (ie news feed) JSON schema, which changed sometime around 2017-02-27. (Profile pages and individual photo/video permalinks still haven't changed yet.)
- microformats2:
  - Add u-featured to ActivityStreams `image`.
  - Improve `h-event` support.
  - Minor whitespace change (added

) when rendering locations as HTML.
  - `post_id()` now validates ids more strictly before returning them.

  - Fix bugs in converting latitude and longitude between ActivityStreams and mf2.
- Google+:
  - Update HTML scraping to handle changed serialized JSON data format.
- Atom:
  - Add new `activity_to_atom()` function that renders a single top-level `<entry>` instead of `<feed>`.
  - Add new `reader` query param for toggling rendering decisions that are specific to feed readers. Right now, just affects location: it's rendered in the content when `reader=true` (the default), omitted when `reader=false`.
  - Include author name when rendering attached articles and notes (e.g. quote tweets).
  - Only include AS `activity:object-type` and `activity:verb` elements when they have values.
  - Render AS image and mf2 u-photo if they're not already in content.
  - Render `thr:in-reply-to` from `object.inReplyTo` as well as `activity.context.inReplyTo`.
- REST API:
  - Fix bugs in html => json-mf2 and html => html conversions.
- Upgrade brevity to 0.2.14 for a couple bug fixes.

10.10 1.7 - 2017-02-27

- microformats2:
  - Interpret h-cite and u-quotation-of (experimental) as attachments, e.g. for quote tweets.
  - Convert audio and video properties to AS attachments.
- Twitter:
  - Linkify @-mentions and hashtags in `preview_create()`.
  - Support creating quote tweets from attachments with Twitter URLs.
  - When converting quote tweets to AS, strip quoted tweet URL from end of text.
  - Raise ValueError when `get_activities()` is passed `group_id='@search'` but not `search_query`.
- Instagram:
  - Improve HTML scraping error handling.
  - Support multi-photo/video posts.
- Facebook:
  - Disable creating "interested" RSVPs, since Facebook's API doesn't allow it.
- Atom:
  - Support media enclosures for audio and video attachments.
- Source.get_activities(): start raising ValueError on bad argument values, notably invalid Facebook and Twitter ids and Instagram search queries.

- Fix rendering and linkifying content with Unicode high code points (ie above the 16-bit Basic Multilingual Plane), including some emoji, on "narrow" builds of Python 2 with `--enable-unicode=ucs2`, which is the default on Mac OS X, Windows, and older *nix.

10.11 1.6 - 2016-11-26

- Twitter:
  - Handle new "extended" tweets with hidden reply-to @-mentions and trailing URLs for media, quote tweets, etc. Background: https://dev.twitter.com/overview/api/upcoming-changes-to-tweets
  - Bug fix: ensure like.author.displayName is a plain unicode string so that it can be pickled normally, e.g. by App Engine's memcache.
  - Bug fix: handle names with emoji correctly in favorites_html_to_likes().
  - Bug fix: handle search queries with unicode characters.
- Atom:
  - Render full original quoted tweet in retweets of quote tweets.
- microformats2 HTML:
  - Optionally follow and fetch rel="author" links.
  - Improve mapping between microformats2 and ActivityStreams 'photo' types. (mf2 'photo' type is a note or article *with* a photo, but AS 'photo' type *is* a photo. So, map mf2 photos to underlying type without photo.)
  - Support location properties beyond h-card, e.g. h-adr, h-geo, u-geo, and even when properties like latitude and longitude appear at the top level.
- Error handling: return HTTP 502 for non-JSON API responses, 504 for connection failures.

10.12 1.5 - 2016-08-25

- REST API:
  - Support tag URI for user id, app id, and activity id.
- Twitter:
  - Better error message when uploading a photo with an unsupported type.
  - Only include original quote tweets, not retweets of them.
  - Skip fetching retweets for protected accounts since the API call always 403s.
- Flickr:
  - Better username detection. Flickr's API is very inconsistent about username vs real name vs path alias. This specifically detects when a user name is probably actually a real name because it has a space.
  - Uploading: detect and handle App Engine's 10MB HTTP request limit.
  - Bug fix in create: handle unicode characters in photo/video description, hashtags, and comment text.
- Atom:
  - Bug fix: escape &s in attachments' text (e.g. quote tweets).

  - Bug fix: handle multiply valued 'object' fields in ActivityStreams 1 activities.
- GitHub:
  - Switch creating comments and reactions from GraphQL to REST API (bridgy#824.

10.13 1.4.1 - 2016-06-27

- Bump oauth-dropins requirement to 1.4.

10.14 1.4.0 - 2016-06-27

- REST API:
  - Cache silo requests for 5m by default, 60m for Instagram because they aggressively blocking scraping. You can skip the cache with the new cache=false query param.
- Facebook:
  - Upgrade from API v2.2 to v2.6. https://developers.facebook.com/docs/apps/changelog
  - Add reaction support.
  - De-dupe event RSVPs by user.
- Twitter:
  - Switch create() to use brevity for counting characters. https://github.com/kylewm/brevity
  - Fix bug in create() that occasionally incorrectly escaped ., +, and - characters.
  - Fix text rendering bug when there are multipl photos/videos.
  - When replying to yourself, don't add a self @-mention.
- Instagram:
  - Fix bugs in scraping.
- Upgrade to requests 2.10.0 and requests-toolbelt 0.60, which support App Engine.

10.15 1.3.1 - 2016-04-07

- Update oauth-dropins dependency to >=1.3.

10.16 1.3.0 - 2016-04-06

- Support posting videos! Currently in Facebook, Flickr, and Twitter.
- Instagram:
  - Add support for scraping, since they're locking down their API and requiring manual approval.
  - Linkify @-mentions in photo captions.
- Facebook:

  - Fetch Open Graph stories aka `news.publish` actions.
  - Many bug fixes for photo posts: better privacy detection, fix bug that attached comments to wrong posts.
- Twitter:
  - Handle all photos/videos attached to a tweet, not just the first.
  - Stop fetching replies to @-mentions.
- Atom:
  - Render attachments.
  - Add `xml:base`.
- microformats2:
  - Load and convert h-card.
  - Implement full post type discovery algorithm, using mf2util. https://indiewebcamp.com/post-type-discovery
  - Drop support for h-as-* classes, both incoming and outgoing. They're deprecated in favor of post type discovery.
  - Drop old deprecated `u-like` and `u-repost` properties.
- Misc bug fixes.
- Set up Coveralls.

10.17 1.2.0 - 2016-01-11

- Improve original post discovery algorithm. (bridgy #51)
- Flickr tweaks. (bridgy #466)
- Add mf2, activitystreams, atom, and search to interactive UI. (#31, #29)
- Improved post type discovery (using mf2util).
- Extract user web site links from all fields in profile (e.g. description/bio).
- Add fabricated fragments to comment/like permalinks (e.g. #liked-by-user123) so that object urls are always unique (multiple silos).
- Improve formatting/whitespace support in create/preview (multiple silos).
- Google+:
  - Add search.
- Facebook:
  - Fetch more things in get_activities: photos, events, RSVPs.
  - Support person tags in create/preview.
  - Prevent facebook from automatically consolidating photo posts by uploading photos to "Timeline Photos" album.
  - Include title in create/preview.
  - Improve object id parsing/resolving.
  - Improve tag handling.

  - Bug fix for fetching nested comments.
  - Misc improvements, API error/flakiness handling.
- Flickr:
  - Create/preview support for photos, comments, favorites, tags, person tags, location.
- Twitter:
  - Create/preview support for location, multiple photos.
  - Fetch quote tweets.
  - Fetching user mentions improvements, bug fixes.
  - Fix embeds.
  - Misc AS conversion improvements.
- microformats2:
  - Improve like and repost rendering.
- Misc bug fixes.
- Set up CircleCI.

10.18 1.1.0 - 2015-09-06

- Add Flickr.
- Facebook:
  - Fetch multiple id formats, e.g. with and without USERID_ prefix.
  - Support threaded comments.
  - Switch from /posts API endpoint to /feed.
- Google+:
  - Support converting plus.google.com HTML to ActivityStreams.
- Instagram:
  - Support location.
- Improve original post discovery algorithm.
- New logo.

10.19 1.0.1 - 2015-07-11

- Bug fix for atom template rendering.
- Facebook, Instagram: support access_token parameter.

10.20 1.0 - 2015-07-10

- Initial PyPi release.