SHARE

Release 2.0

Dec 13, 2018

Contents

SHARE is a higher education initiative whose mission is to maximize research impact by making research widely accessible, discoverable, and reusable. To fulfill this mission SHARE is building a free, open, data set about research and scholarly activities across their life cycle.

SHARE harvests metadata nightly from 100+ repositories, transforms that metadata into one format, and makes it accessable via a web API.

The technical side of SHARE has many pieces that you can interact with:

- **A search endpoint powered by elasticsearch that indexes the transformed data allowing:**
    - **Thorough search of creative works:**
        - Creative works: /api/v2/search/creativeworks/_search
        - more info on the elasticsearch docs page
    - Data aggregations across fields
- **An Ember application using the SHARE API for:**
    - Searching the SHARE database
    - Discovering new Projects
    - Corrections and Updates
- **API Endpoints for accessing transformed metadata**
    - https://share.osf.io/api/v2

CHAPTER 1

Guide

Quickstart

SHARE Pipeline

THE SHARE Pipeline can be setup locally for testing and modifications. Note: Kill any postgres process running before starting.

If prompted, install docker from https://docs.docker.com/docker-for-mac/.

This requires Python 3.5; If necessary, follow the steps below:

From scratch:

```
pip install virtualenv
pip install virtualenvwrapper
```

Create virtualenv 'share':

```
mkvirtualenv share --python=python3.5
```

Switch into the share environment for the first time:

```
workon share
```

Note - These instructions are for getting up and running with the simplest steps possible and is good as more of a refresher – for more details, or getting started for the first time, please see the section on getting up and running on the *Harvesters and Transformers section*.

THE SHARE Pipeline can be setup locally for testing and modifications.

Setup:

```
git clone https://github.com/CenterForOpenScience/SHARE.git

cd SHARE

pip install -r requirements.txt

// Creates and starts containers for elasticsearch, rabbitmq, and postgres
docker-compose up -d web

./up.sh
```

To run:

```
python manage.py runserver
python manage.py celery worker -l DEBUG
```

Run a harvester:

```
python manage.py harvest domain.providername --async
```

To see a list of all providers, as well as their names for harvesting, visit https://share.osf.io/api/v2/sources

For more information, see the section on *Harvesters and Transformers*.

Harvesters and Transformers

A *harvester* gathers raw data from a source using their API.

A *transformer* takes the raw data gathered by a harvester and maps the fields to the defined *SHARE models*.

Start Up

1. Install Docker.
2. Make sure you're using Python3 - install with miniconda , or homebrew
3. Install everything inside a Virtual Enviornment - created with Conda or Virtualenv or your python enviornment of choice.

Installation (inside a virtual environment):

```
pip install -r requirements.txt

// Creates, starts, and sets up containers for elasticsearch,
// postgres, and the server
docker-compose build web
docker-compose run --rm web ./bootstrap.sh
```

To run the server in a virtual environment instead of Docker:

```
docker-compose stop web
python manage.py runserver
```

To run celery worker:

```
python manage.py celery worker -l DEBUG
```

Running Existing Harvesters and Transformers

To see a list of all sources and their names for harvesting, visit https://share.osf.io/api/sources/

Steps for gathering data:

- **Harvest** data from the original source
- **Transform** data, or create a `ChangeSet`` that will format the data to be saved into SHARE Models

- **Accept** the `ChangeSet`` objects, and save them as `AbstractCreativeWork` objects in the SHARE database

Printing to the Console

It is possible to run the harvesters and transformers separately, and print the results out to the console for testing and debugging using `./bin/share`

For general help documentation:

```
./bin/share --help
```

For harvest help:

```
./bin/share harvest --help
```

To harvest:

```
./bin/share harvest domain.source_name_here
```

If the harvester created a *lot* of files and you want to view a couple:

```
find <source dir i.e. edu.icpsr/> -type f -name '*.json' | head -<number to list>
```

The harvest command will by default create a new folder at the top level with the same name as the source name, but you can also specify a folder when running the harvest command with the `--out` argument.

To transform all harvested documents:

```
./bin/share transform domain.source_name_here dir_where_raw_docs_are/*
```

To transform just one document harvested:

```
./bin/share transform domain.source_name_here dir_where_raw_docs_are/filename.json
```

If the transformer returns an error while parsing a harvested document, it will automatically enter into a python debugger.

To instead enter into an enhanced python debugger with access to a few more variables like `data`, run:

```
./bin/share debug domain.source_name_here dir_where_raw_docs_are/filename.json
```

To debug:

```
e(data, ctx.<field>)
```

Running Though the Full Pipeline

Note: celery must be running for `--async` tasks

Run a harvester and transformer:

```
python manage.py harvest domain.sourcename --async
```

To automatically accept all `ChangeSet` objects created:

```
python manage.py runbot automerge --async
```

To automatically add all harvested and accepted documents to Elasticsearch:

```
python manage.py runbot elasticsearch --async
```

Writing a Harvester and Transformer

See the transformers and harvesters located in the `share/transformers/` and `share/harvesters/` directories for more examples of syntax and best practices.

Adding a new source

- Determine whether the source has an API to access their metadata
- **Create a source folder at `share/sources/{source name}`**
    - Source names are typically the reversed domain name of the source, e.g. a source at `http://example.com` would have the name `com.example`
    - If the source name starts with a new TLD (e.g. com, au, gov), please add `/TLD.*/` to .gitignore in the generated harvester data section
- **Create a file named `source.yaml` in the source folder**
    - See *Writing a source.yaml file*
- **Determine whether the source makes their data available using the OAI-PMH protocol**
    - If the source is OAI see *Best practices for OAI sources*
- **Writing the harvester**
    - See *Best practices for writing a Harvester*
- **Writing the transformer**
    - See *Best practices for writing a Transformer*
- **Adding a sources's icon**
    - visit `www.domain.com/favicon.ico` and download the `favicon.ico` file
    - place the favicon as `icon.ico` in the source folder
- **Load the source**
    - To make the source available in your local SHARE, run `./manage.py loadsources` in the terminal

Writing a source.yaml file

The `source.yaml` file contains information about the source itself, and one or more configs that describe how to harvest and transform data from that source.

```
name: com.example
long_title: Example SHARE Source for Examples
home_page: http://example.com/
user: sources.com.example
```

```
configs:
- label: com.example.oai
  base_url: http://example.com/oai/
  harvester: oai
  harvester_kwargs:
      metadata_prefix: oai_datacite
  rate_limit_allowance: 5
  rate_limit_period: 1
  transformer: org.datacite
  transformer_kwargs: {}
```

See the whitepaper for Source and SourceConfig tables for the available fields.

Best practices for OAI sources

Sources that use OAI-PMH make it easy to harvest their metadata.

- Set `harvester: oai` in the source config.
- **Choose a metadata format to harvest.**
  - Use the `ListMetadataFormats` OAI verb to see what formats the source supports.
  - Every OAI source supports `oai_dc`, but they usually also support at least one other format that has richer, more structured data, like `oai_datacite` or `mods`.
  - Choose the format that seems to have the most useful data for SHARE, especially if a transformer for that format already exists.
  - Choose `oai_dc` only as a last resort.
- Add `metadata_prefix: {prefix}` to the `harvester_kwargs` in the source config.
- **If necessary, write a transformer for the chosen format.**
  - See *Best practices for writing a Transformer*

Best practices for writing a non-OAI Harvester

- The harvester should be defined in `share/harvesters/{harvester name}.py`.
- **When writing the harvester:**
  - Inherit from `share.harvest.BaseHarvester`
  - Add the version of the harvester `VERSION = 1`
  - Implement `do_harvest(...)` (and possibly additional helper functions) to make requests to the source and to yield the harvested records.
  - **Check to see if the data returned by the source is paginated.**
    - There will often be a resumption token to get the next page of results.
  - **Check to see if the source's API accepts a date range**
    - If the API does not then, if possible, check the date on each record returned and stop harvesting if the date on the record is older than the specified start date.
- **Add the harvester to `entry_points` in `setup.py`**
  - e.g. `'com.example = share.harvesters.com_example:ExampleHarvester',`

  - run `python setup.py develop` to make the harvester available in your local SHARE
- Test by *running the harvester*

Best practices for writing a non-OAI Transformer

- The transformer should be defined in `share/transformers/{transformer name}.py`.
- **When writing the transformer:**
  - Determine what information from the source record should be stored as part of the `CreativeWork` *model* (i.e. if the record clearly defines a title, description, contributors, etc.).
  - **Use the *chain transformer tools* as necessary to correctly parse the raw data.**
    - Alternatively, implement `share.transform.BaseTransformer` to create a transformer from scratch.
  - **Utilize the `Extra` class**
    - Raw data that does not fit into a defined *share model* should be stored here.
    - Raw data that is otherwise altered in the transformer should also be stored here to ensure data integrity.
- **Add the transformer to `entry_points` in `setup.py`**
  - e.g. `'com.example = share.transformer.com_example:ExampleTransformer',`
  - run `python setup.py develop` to make the transformer available in your local SHARE
- Test by *running the transformer* against raw data you have harvested.

SHARE Chain Transformer

SHARE provides a set of tools for writing transformers, based on the idea of constructing chains for each field that lead from the root of the raw document to the data for that field. To write a chain transformer, add `from share.transform.chain import links` at the top of the file and make the transformer inherit `share.transform.chain.ChainTransformer`.

```
from share.transform.chain import ctx, links, ChainTransformer, Parser


class CreativeWork(Parser):
    title = ctx.title


class ExampleTransformer(ChainTransformer):
    VERSION = 1
    root_parser = CreativeWork
```

- **Concat** To combine list or singular elements into a flat list:

  ```
  links.Concat(<string_or_list>, <string_or_list>)
  ```

- **Delegate** To specify which class to use:

  ```
  links.Delegate(<class_name>)
  ```

- **Join** To combine list elements into a single string:

  ```
  links.Join(<list>, joiner=' ')
  ```

  Elements are separated with the `joiner`. By default `joiner` is a newline.

- **Map** To designate the class used for each instance of a value found:

  ```
  links.Map(links.Delegate(<class_name>), <chain>)
  ```

  See the *share models* for what uses a through table (anything that sets `through=`). Uses the *Delegate* tool.

- **Maybe** To transform data that is not consistently available:

  ```
  links.Maybe(<chain>, '<item_that_might_not_exist>')
  ```

  Indexing further if the path exists:

  ```
  links.Maybe(<chain>, '<item_that_might_not_exist>')['<item_that_will_exist_if_
  ↪maybe_passes>']
  ```

  Nesting Maybe:

  ```
  links.Maybe(links.Maybe(<chain>, '<item_that_might_not_exist>')['<item_that_
  ↪will_exist_if_maybe_passes>'], '<item_that_might_not_exist>')
  ```

  To avoid excessive nesting use the *Try link*

- **OneOf** To specify two possible paths for a single value:

  ```
  links.OneOf(<chain_option_1>, <chain_option_2>)
  ```

- **ParseDate** To determine a date from a string:

  ```
  links.ParseDate(<date_string>)
  ```

- **ParseLanguage** To determine the ISO language code (i.e. 'ENG') from a string (i.e. 'English'):

  ```
  links.ParseLanguage(<language_string>)
  ```

  Uses pycountry package.

- **ParseName** To determine the parts of a name (i.e. first name) out of a string:

  ```
  links.ParseName(<name_string>).first
  ```

  options:

  ```
  first
  last
  middle
  suffix
  title
  nickname
  ```

  Uses nameparser package.

- **RunPython** To run a defined python function:

```
links.RunPython('<function_name>', <chain>, *args, **kwargs)
```

- **Static** To define a static field:

```
links.Static(<static_value>)
```

- **Subjects** To map a subject to the PLOS taxonomy based on defined mappings:

```
links.Subjects(<subject_string>)
```

- **Try** To transform data that is not consistently available and may throw an exception:

```
links.Try(<chain>)
```

- **XPath** To access data using xpath:

```
links.XPath(<chain>, "<xpath_string>")
```

SHARE Models

Due to the nature of the data that are collected by SHARE, the schema model is subject to change.

The current JSON Schema and field descriptions, when available, can be found in our API.

Elasticsearch

SHARE has an elasticsearch API endpoint that can be used for searching SHARE's normalized data, as well as for compiling summary statistics and analyses of the completeness of data from the various sources.

https://share.osf.io/api/v2/search/creativeworks/_search

Fields Indexed by Elasticsearch

Elasticsearch can be used to search the following fields in the normalized data:

```
'title'
'description'
'type'
'date'
'date_created'
'date_modified
'date_updated'
'date_published'
'tags'
'subjects'
'sources'
'language'
'contributors'
'funders'
'publishers'
```

Date Fields

There are five date fields, and each has a different meaning. Two are given to SHARE by the data source:

`date_published` When the work was first published, issued, or made publicly available in any form. Not all sources provide this, so some works in SHARE have no `date_published`.

`date_updated` When the work was last updated by the source. For example, an OAI-PMH record's `<datestamp>`. Most works have a `date_updated`, but some sources do not provide this.

Three date fields are populated by SHARE itself:

`date_created` When SHARE first ingested the work and added it to the SHARE dataset. Every work has a `date_created`.

`date_modified` When SHARE last ingested the work and modified the work's record in the SHARE dataset. Every work has a `date_modified`.

`date` Because many works may not have `date_published` or `date_updated` values, sorting and filtering works by date can be confusing. The `date` field is intended to help. It contains the most useful available date. If the work has a `date_published`, `date` contains the value of `date_published`. If the work has no `date_published` but does have `date_updated`, `date` is set to `date_updated`. If the work has neither `date_published` nor `date_updated`, `date` is set to `date_created`.

Accessing the Search API

Using curl

You can acess the API via the command line using a basic query string with curl:

```
curl -H "Content-Type: application/json" -X POST -d '{
    "query": {
        "query_string" : {
            "query" : "test"
        }
    }
}' https://share.osf.io/api/v2/search/creativeworks/_search
```

The elasticsearch API also allows you to aggregate over the whole dataset. This query will also return an aggregation of which sources do not have a value specified for the field "language":

```
curl -H "Content-Type: application/json" -X POST -d '{
    "aggs": {
        "sources": {
            "significant_terms": {
                "percentage": {},
                "size": 0,
                "min_doc_count": 1,
                "field": "sources"
            }
        }
    },
    "query": {
        "bool": {
            "must_not": [
                {
                    "exists": {
                        "field": "language"
                    }
                }
            ]
        }
    }
}' https://share.osf.io/api/v2/search/creativeworks/_search
```

For more information on sending elasticsearch queries and aggregations, check out the elasticsearch query DSL documentation.

You can also use the SHARE Discover page to generate query DSL. Use the filters in the sidebar to construct a query, then click “View query body” to see the query in JSON form.

Searching for ORCIDs

Get all works where contributors have ORCID identifiers: https://share.osf.io/api/v2/search/creativeworks/_search?q=lists.contributors.identifiers:orcid.org

In the results, the ORCID will be listed under: _source → lists → contributors → (contributor) → identifiers

```
{
    timed_out: false,
    hits: {
        total: 204235,
        hits: [
        {
            _id: "XXXX-XXX-XXX",
            _source: {
                id: "XXXX-XXX-XXX",
                date_updated: "2016-04-23T07:31:31+00:00",
                title: "Title Example",
                date: "2016-04-23T07:31:31+00:00",
                description: "Example of a search result containing an ORCID.",
                contributors: [...],
                date_created: "2016-11-28T22:21:09.917395+00:00",
                date_modified: "2016-11-29T14:18:49.745627+00:00",
                date_published: null,
                lists: {
                    contributors: [
                        {
                            given_name: "T.",
                            types: [
                                "person",
                                "agent"
                            ],
                            order_cited: 133,
                            identifiers: [
                                "http://orcid.org/XXXX-XXXX-XXXX-XXXX"
                            ],
                            cited_as: "T. User",
                            family_name: "User",
                            relation: "creator",
                            name: "T. User",
                            type: "person",
                            id: "XXXX-XXX-XXX"
                        },
                    ...
```

Search for an ORCID identifier: https://share.osf.io/api/v2/search/creativeworks/_search?q=lists.contributors.identifiers:"XXXX-XXXX-XXXX-XXXX"

SHARE API

The SHARE API generally complies with the JSON-API v1.0 spec, as should anyone using the SHARE API.

Check out the browsable SHARE API docs!

Getting Started

Before pushing data to production it is highly recommended to use our staging environment.

1. Go to the staging OSF and register for an account
2. Navigate to staging SHARE and login.
3. Register to become a source.
4. Send an email to share-support@osf.io and wait for us to approve your account.
5. Once approved, the API token from your staging SHARE profile page can be used to push data.

To become a Source for production repeat the above steps at https://share.osf.io with a production OSF account.

> **Note:** Our Staging enviroment is constantly being updated with new code. If something doesn't work, try again in a day or two before contacting us at share-support@osf.io

Paging in the API

The SHARE API implements diffent paging strategies depending on the endpoint. All of them, however, conform to the JSON-API paging spec.

```python
import requests

r = requests.get(
    'https://share.osf.io/api/v2/normalizeddata/',
    headers={'Content-Type': 'application/vnd.api+json'}
)
next_link = r.json()['links']['next']
```

Push data directly into the SHARE database

Changes to the SHARE dataset, additions, modifications, or deletions (Not yet supported), are submitted as a subset of JSON-LD graphs. A change is represented as a JSON object with a single key, `@graph` , containing a list of JSON-LD nodes.

```
{
    "@graph": [
        {
            // Omitted...
        },
        {
            // Omitted...
        },
        {
            // Omitted...
        }
    ]
}
```

- Each node MUST contain an `@id` and `@type` key.
- `@id` MUST be either a blank node identifier or the id of an existing object in the SHARE dataset.

```
// GOOD: A blank identifier
{
    "@id": "_:1234"
    // Omitted...
}

// GOOD: An existing object's ID
{
    "@id": "46227-0C4-522"
    // Omitted...
}

// BAD: Anything that is not a string
{
    "@id": 12
    // Omitted...
}

// BAD: A meaningless string
{
    "@id": "FooBar"
    // Omitted...
}
```

- `@type` MUST be a SHARE type.

Note: `@type` is case sensitive and expects title case, lowercase, or uppercase types.

```
// GOOD: Title case for a type from the linked page
{
    "@type": "Preprint"
    // Omitted...
}

// GOOD: All lowercase for a type from the linked page
{
    "@type": "article"
    // Omitted...
}

// GOOD: All uppercase for a type from the linked page
{
    "@type": "CREATIVEWORK"
    // Omitted...
}

// BAD: Other casing of a type from the linked page
{
    "@type": "cReAtIvEwOrK"
    // Omitted...
}

// BAD: Anything else
{
    "@type": "Unicorn"
    // Omitted...
}
```

- Each node MUST match the JSON schema for its specified type (`@type`).

Note: The JSON schemas for every type can be found here.

```
// GOOD: Following the schema
{
    "@id": "_:abc",
    "@type": "Person",
    "given_name": "Tim"
    "family_name": "Errington"
}

// GOOD: Following the schema a different way
{
    "@id": "_:abc",
    "@type": "Person",
    "name": "Tim Errington"
}

// BAD: Invalid data
{
    "@id": "_:abc",
    "@type": "Article",
    "color": "Nine"
}
```

- Nodes may reference either existing objects or nodes in the same graph.

Note: The order of nodes in `@graph` does not matter.

```
// GOOD: Referring to another node
{
    "@graph": [{
        "@id": "_:123",
        "@type": "agentidentifier",
        "uri": "http://osf.io/juwia",
        "agent": {"@id": "_:abc", "@type": "person"}  // Refers the the␣
↪node below
    }, {
        "@id": "_:abc",
        "@type": "person",
        "name": "Chris Seto",
    }]
}

// GOOD: Referring to an existing object
{
    "@graph": [{
        "@id": "_:123",
        "@type": "agentidentifier",
        "uri": "http://osf.io/juwia",
        "agent": {"@id": "6403D-314-B83", "@type": "person"}
    }]
}

// BAD: Referring to a node that is not defined
{
    "@graph": [{
        "@id": "_:123",
        "@type": "agentidentifier",
        "uri": "http://osf.io/juwia",
        "agent": {"@id": "_:abcd", "@type": "person"}  // _:abcd does␣
↪not appear anywhere
    }]
}

// BAD: Referring to a node any way besides {"@id": "...", "@type": "...
↪"}
{
    "@graph": [{
        "@id": "_:123",
        "@type": "agentidentifier",
        "uri": "http://osf.io/juwia",
        "agent": "6403D-314-B83",  // Please don't
    }]
}
```

- Finally, changes must be submitted in JSON-API format using OAuth2 to authenticate

Note: Yes, there are two `data` keys. Sorry.

```
POST /api/v2/normalizeddata HTTP/1.1
Host: share.osf.io
Authorization: Bearer ACCESS_TOKEN
Content-Type: application/vnd.api+json

{
    "data": {
        "type": "NormalizedData",
        "attributes": {
            "data": {
                "@graph": [/* ... */]
            }
        }
    }
}
```

Example Data

```
{
    "@graph": [{
        "uri": "http://dx.doi.org/10.1038/EJCN.2016.211",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@type": "WorkIdentifier",
        "@id": "_:014eb1c53ba64c9c88bc46ef89cb2080"
    }, {
        "uri": "oai://nature.com/10.1038/ejcn.2016.211",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@type": "WorkIdentifier",
        "@id": "_:d058a287d60f45a48e7d0a9ecfd98bad"
    }, {
        "name": "M Santiago-Torres",
        "@type": "person",
        "@id": "_:760b02f6297a4bbd8fd6f2a0af306dd7"
    }, {
        "order_cited": 0,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:a632e7a0a5814e7fb1fdef1bec6895ab",
        "agent": {
            "@type": "person",
            "@id": "_:760b02f6297a4bbd8fd6f2a0af306dd7"
        },
        "cited_as": "M Santiago-Torres"
    }, {
        "name": "J De Dieu Tapsoba",
        "@type": "person",
        "@id": "_:15838a790c5d41508e5ad8f1327fbaa9"
    }, {
        "order_cited": 1,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:55cd617b118c43f5becb7647f17eba12",
        "agent": {
            "@type": "person",
            "@id": "_:15838a790c5d41508e5ad8f1327fbaa9"
        },
        "cited_as": "J De Dieu Tapsoba"
    }, {
        "name": "M Kratz",
        "@type": "person",
        "@id": "_:50098933694d4795a2653546cdc85493"
    }, {
        "order_cited": 2,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:3c75c1082fde4676a53d16111c7354d9",
        "agent": {
            "@type": "person",
            "@id": "_:50098933694d4795a2653546cdc85493"
        },
        "cited_as": "M Kratz"
    }, {
        "name": "J W Lampe",
        "@type": "person",
        "@id": "_:97eb79ce0005436894b52d53536d3ddc"
    }, {
        "order_cited": 3,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:671d6abea53442e1b50a2976cbe10ac7",
        "agent": {
            "@type": "person",
            "@id": "_:97eb79ce0005436894b52d53536d3ddc"
        },
        "cited_as": "J W Lampe"
    }, {
        "name": "K L Breymeyer",
        "@type": "person",
        "@id": "_:38b4cc174ea44f649257f86cf93effbc"
    }, {
        "order_cited": 4,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:b7676b36d1b4483e8008eedfbd1fb043",
        "agent": {
            "@type": "person",
            "@id": "_:38b4cc174ea44f649257f86cf93effbc"
        },
        "cited_as": "K L Breymeyer"
    }, {
        "name": "L Levy",
        "@type": "person",
        "@id": "_:b809383685844464ab2a4203c8b5ee98"
    }, {
        "order_cited": 5,
        "@type": "Creator",
        "creative_work": {
```

```
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:fecd2c815ba84e1d9455b1d31182b267",
        "agent": {
            "@type": "person",
            "@id": "_:b809383685844464ab2a4203c8b5ee98"
        },
        "cited_as": "L Levy"
    }, {
        "name": "X Song",
        "@type": "person",
        "@id": "_:007fca2333e74ed38e3f1b92a13662ae"
    }, {
        "order_cited": 6,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:b0c9846c388541c39f0cc42056dc1de2",
        "agent": {
            "@type": "person",
            "@id": "_:007fca2333e74ed38e3f1b92a13662ae"
        },
        "cited_as": "X Song"
    }, {
        "name": "A Villase\u00f1or",
        "@type": "person",
        "@id": "_:78a4cd8407a74e0a81468ba3cd2658ed"
    }, {
        "order_cited": 7,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:96f9851b68444d9fa5ad7faab1f1d518",
        "agent": {
            "@type": "person",
            "@id": "_:78a4cd8407a74e0a81468ba3cd2658ed"
        },
        "cited_as": "A Villase\u00f1or"
    }, {
        "name": "C-Y Wang",
        "@type": "person",
        "@id": "_:6ffa6c228c75476c9cc089053be6b3f1"
    }, {
        "order_cited": 8,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:f39c7fa402ca4028a78798dc67eb5dff",
        "agent": {
            "@type": "person",
            "@id": "_:6ffa6c228c75476c9cc089053be6b3f1"
        },
        "cited_as": "C-Y Wang"
    }, {
        "name": "L Fejerman",
        "@type": "person",
        "@id": "_:3a15f900ccba4d5cbeade9c48f857f60"
    }, {
        "order_cited": 9,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:51fbd9a4043b41f29407522e3ef50534",
        "agent": {
            "@type": "person",
            "@id": "_:3a15f900ccba4d5cbeade9c48f857f60"
        },
        "cited_as": "L Fejerman"
    }, {
        "name": "M L Neuhouser",
        "@type": "person",
        "@id": "_:e5930003ef914b9e99892cbb134ab0ad"
    }, {
        "order_cited": 10,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:b1fd726a4788423eb3a71509b2493757",
        "agent": {
            "@type": "person",
            "@id": "_:e5930003ef914b9e99892cbb134ab0ad"
        },
        "cited_as": "M L Neuhouser"
    }, {
        "name": "C S Carlson",
        "@type": "person",
        "@id": "_:a021013c285a4c589b5c1360eb261647"
    }, {
        "order_cited": 11,
        "@type": "Creator",
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@id": "_:34c8ec8f32a74abbaa38d5efec6e9fdd",
        "agent": {
            "@type": "person",
            "@id": "_:a021013c285a4c589b5c1360eb261647"
        },
        "cited_as": "C S Carlson"
    }, {
        "name": "Nature Publishing Group",
        "@type": "organization",
        "@id": "_:2cb215bb499844cf8aecc2c9f817386c"
    }, {
        "agent": {
            "@type": "organization",
            "@id": "_:2cb215bb499844cf8aecc2c9f817386c"
        },
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@type": "Publisher",
        "@id": "_:5e65f7f40b0f41989566fcf66241767c"
    }, {
        "name": "ejcn",
        "@type": "Tag",
        "@id": "_:a9d049bdd4c7482bb82f513e09365c2e"
    }, {
        "tag": {
            "@type": "Tag",
            "@id": "_:a9d049bdd4c7482bb82f513e09365c2e"
        },
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@type": "ThroughTags",
        "@id": "_:e70071583d604be2a7e104cd61b2b6cc"
    }, {
        "name": "Original Article",
        "@type": "Tag",
        "@id": "_:610d99b2c5b74a82896c4681c60ecebb"
    }, {
        "tag": {
            "@type": "Tag",
            "@id": "_:610d99b2c5b74a82896c4681c60ecebb"
        },
        "creative_work": {
            "@type": "article",
            "@id": "_:703a584afb704403bc99d684e0914c06"
        },
        "@type": "ThroughTags",
        "@id": "_:eeeef1b6c0c24bc58344938badafd464"
    }, {
        "date_updated": "2016-12-14T00:00:00+00:00",
        "rights": "\u00a9 2016 Macmillan Publishers Limited, part of␣
↪Springer Nature.",
        "related_works": [],
        "title": "Genetic ancestry in relation to the metabolic response to␣
↪a US versus traditional Mexican diet: a randomized crossover feeding trial␣
↪among women of Mexican descent",
        "subjects": [],
        "extra": {
            "language": "en",
            "set_spec": "ejcn",
            "identifiers": [
                "doi:10.1038/ejcn.2016.211",
                "oai:nature.com:10.1038/ejcn.2016.211"
            ],
            "dates": "2016-12-14",
            "creator": [
                "M Santiago-Torres",
                "J De Dieu Tapsoba",
                "M Kratz",
                "J W Lampe",
                "K L Breymeyer",
                "L Levy",
                "X Song",
                "A Villase\u00f1or",
                "C-Y Wang",
                "L Fejerman",
                "M L Neuhouser",
                "C S Carlson"
            ],
            "resource_type": "Original Article"
        },
        "@id": "_:703a584afb704403bc99d684e0914c06",
        "@type": "article"
    }]
}
```

Code Examples

Python

```
import requests

url = 'https://share.osf.io/api/normalizeddata/'

payload = {
    'data': {
        'type': 'NormalizedData'
        'attributes': {
            'data': {
                '@graph': [{
                    '@type': creativework,
                    '@id': <_:random>,
                    title: "Example Title of Work"
                }]
            }
        }
    }
}

r = requests.post(url, json=payload, headers={
    'Authorization': 'Bearer <YOUR_TOKEN>',
    'Content-Type': 'application/vnd.api+json'
})
```

JavaScript

```
let payload = {
    'data': {
        'type': 'NormalizedData'
        'attributes': {
            'data': {
                '@graph': [{
                    '@type': creativework,
                    '@id': <_:random>,
                    title: "Example Title of Work"
                }]
            }
        }
    }
}

$.ajax({
    method: 'POST',
    headers: {
        'X-CSRFTOKEN': csrfToken
    },
    xhrFields: {
        withCredentials: true,
    },
    data: JSON.stringify(payload),
    contentType: 'application/vnd.api+json',
    url: 'https://share.osf.io/api/normalizeddata/',
})
```

Ember Application

Interfaces to interact with SHARE data can be built on top of the SHARE API.

As an example, we built an Ember application that uses the SHARE API. Find it on GitHub at CenterForOpenScience/ember-share or check out the website.

CHAPTER 2

Contribute

- Source Code: https://github.com/CenterForOpenScience/SHARE
- Issue Tracker: https://github.com/CenterForOpenScience/SHARE/issues

CHAPTER 3

Get In Touch

For emails about technical support: share-support@osf.io

```
Association of Research Libraries
21 Dupont Circle NW #800
Washington, DC 20036
202-296-2296
info@share-research.org
```