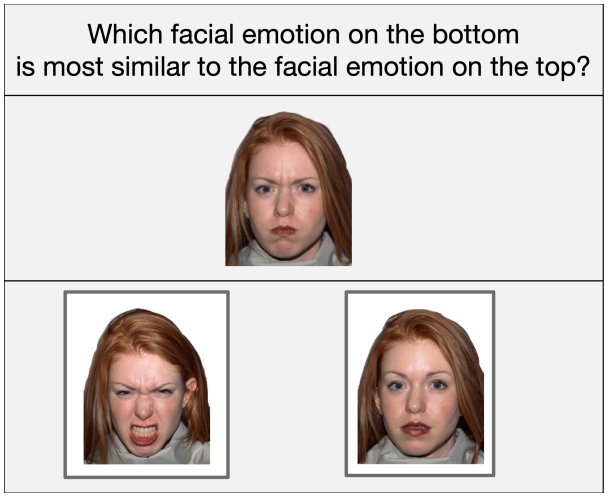
Salmon

Scott Sievert

Dec 31, 2022

USAGE

Salmon is a tool to easily allow collection of "triplet queries." These queries are relative similarity judgments of the form "is object a or b closer to object h?" An example is shown below with facial similarities:



These queries provide a relative similarity measure: a response indicates that object a is closer to object b than object c as determined by humans. For example, these triplet queries have been used by psychologists to determine what facial emotions human find similar:

Only distance is relevant in this embedding, not the vertical/horizontal axes. However, if you look closely, you can see two axes: positivity and intensity.

Salmon provides efficient methods for collecting these triplet queries. Typically, generating the embeddings above requires far too many human responses. Salmon provides the ability to generate the same embeddings with fewer human responses – in our experiments, about 1,000 queries are required to reach a particular quality level instead of about 3,000 queries. If you're paying for each human response (say on Mechanical Turk), this means that collecting responses will be reduced by a factor of 3 when compared with naive methods of collecting triplet queries.

If you'd like to report bugs/issues, or improve Salmon please see Salmon's contribution guide. The list of dependicies and their licenses is available at *Dependencies*. Salmon is licensed under the BSD License. Details are at LICENSE.txt.

CHAPTER

ONE

USERS

Salmon is currently being actively used by pyschologists from the University of Wisconsin–Madison, and has seen some user from pyschologists at the Louisiana State University and Canada's Western University.

1.1 Installation

This pages details how to get Salmon running, either on EC2 or locally on your machine. After you get Salmon running, detail on how to launch experiments in *Getting started*.

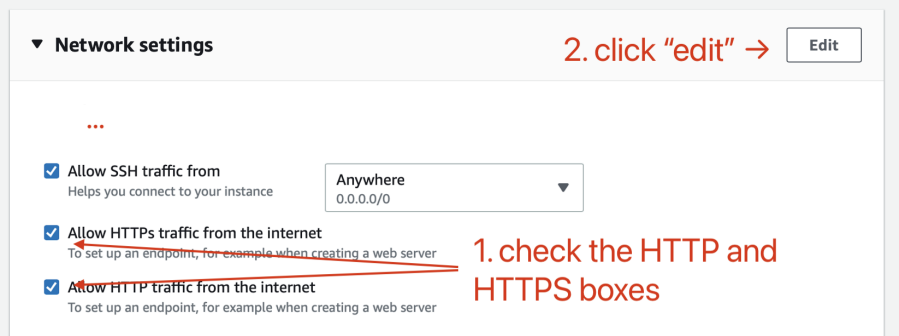
Note: See the *Troubleshooting* section if you're having difficulties with either of the processes below.

1.1.1 Experimentalist

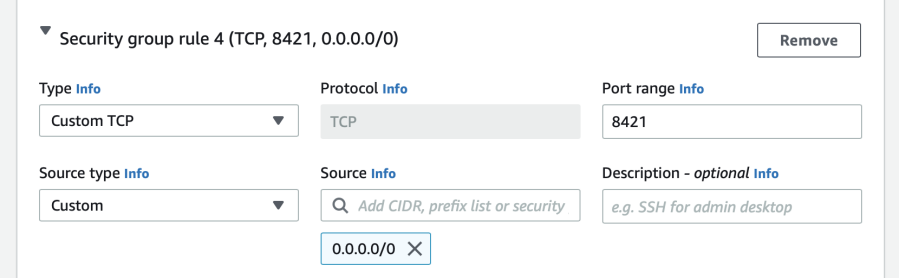
1. Sign into Amazon AWS (http://aws.amazon.com/)
2. Select the "Oregon" region (or `us-west-2`) in the upper right.
3. Go to Amazon EC2.
4. Launch a new instance (the big blue button or square orange button).
5. Select AMI `ami-07d9e17b8dea4da43` titled "Salmon".
    - Note: the AMI appears in Community AMIs after searching "Salmon" (only in the Oregon/`us-west-2` region!).
6. Select an appropriate instance type.
    - Salmon requires at least 2GB of memory and 1 CPU.
    - `t3.large` is recommended for passive algorithms (i.e, random sampling).
    - `t3.xlarge` is recommended for adaptive algorithms (e.g., ARR; see the *benchmarks on adaptive algorithm* for more detail).
    - Note: https://ec2instances.info/ is a great resource to check costs. As of April 2022, `t3.large` and `t3.xlarge` cost about $2/day and $4/day respectively.
7. Create a key pair.

> **Warning:** **Don't lose your key pair!** Without the key pair, the Salmon developers will be severely limited in the help they can provide.

8. Don't click the big orange button yet. Continue to the rules page, and add these rules:
    1. In the networking interface box, check the HTTP and HTTP boxes and select the "edit" button:



    2. After hitting "edit", scroll down to "add security group rule" and open port 8421 to `0.0.0.0/0` (aka anyone).



9. Now, click the big orange button! The AMI will probably take around 15 to initialize (but may take up to 30 minutes).
10. Keep your "key pair" in a safe place. The key pair typically has a `.pem` extension.

The AMI initialization is done (which takes about 15 minutes), Salmon will be available at `http://[url]:8421`. For example, `[url]` might be the Amazon public DNS or public IP.

```
http://ec2-35-164-240-184.us-west-2.compute.amazonaws.com:8421/foo
```

Warning: By default, Salmon does not support HTTPS. Be sure the URL begins with `http://` and not `https://`!

Until you upload data, `http://[url]:8421` will only show an error message. To start using Salmon, these endpoints will be available:

- `http://[url]:8421/init` to create a user and initialize a new experiment.
- `http://[url]:8421/docs` to see the endpoint documentation. The Salmon version displayed should match the most recent Salmon release in the list of Salmon releases.
- `http://[url]:8421/dashboard` to view all relevant links, including links to the...
    - The **query page.** This is the URL that shows the relevant triplets. This is the URL to be sent to a crowdsourcing service.
    - **API documentation**. This includes information on how to launch an experiment, and what files need to be uploaded. View the documentation for the POST request `/init_exp` for more detail.

- **Download the experiment.** The downloaded file can be re-uploaded to a new machine so experiments can be restarted.
- **Responses**. To get all human responses.
- **Logs**. This is very useful for debugging.

> **Warning:** Download all files when stopping or terminating the machine – especially the responses and experiment file.

Note: If you have an issue with the machine running Salmon, be sure to include the logs when contacting the Salmon developers. They'd also appreciate it if you left the machine running.

Note: The storage required for Salmon is 128GB. According to Amazon's EBS pricing, that costs about $10/month (in April 2022).

1.1.2 Local install

See *Install Salmon* for the process of installing locally.

1.1.3 Troubleshooting

See *FAQ* for more general questions.

Note: Please include the version in any bug reports or feature requests. The version number should look something like `v0.4.1`. It can be found at `http://[url]:8421/docs` or in the downloaded experiment file (found at `http://[url]:8421/download` which has a filename like `exp-2021-05-20T07:31-salmon-v0.4.1.rdb`).

I can't access Salmon's URL

Try using `http://` instead of `https://`. By default, EC2 does not support HTTPS, and some browsers use HTTPS automatically.

I can't find Salmon's AMI

Are you in EC2's Oregon region, `us-west-2`? That can be changed in the upper right of the Amazon EC2 interface.

The Salmon AMI has been created in the `us-west-2` region, and EC2 AMIs are only available in the regions they're created in.

Restoring from a backup didn't work

This might happen if Salmon changed between when you downloaded and tried to restore the experiment. Launching from EC2 always downloads the latest version of Salmon, which may not work with your backup file.

Note: Salmon follows semantic software versioning. If the version string in the .rdb file takes the form `vA.B.C`, then:

- The backup is guaranteed to work if the latest release has version `vA.B.C`.
- The backup will almost certainly work if the latest release has version `vA.B.*`.
- The backup *might* work if the latest release has version `vA.*.*`.

Uploading backup files when *relevant* "backwards incompatible" software changes are made, which should be encoded in the release notes.

So, it uploading your backup did not work (it should if the version numbers are correct), let's launch the correct version of Salmon's server on your machine. That requires this process:

1. Get the correct version of Salmon.
2. Spin up a Salmon server.
3. Go to `http://localhost:8421/init`
4. Upload the `.rdb` file to restore.

This process is basically "launch Salmon's server on your machine." First, let's get the right version of Salmon:

```
$ # Get right version of Salmon
$ git clone https://github.com/stsievert/salmon.git
$ cd salmon
$ git checkout v0.7.0  # from .rdb filename; will take the form "vA.B.C" or "vA.B.CrcD"
```

Second, let's launch Salmon:

```
$ docker-compose up  # takes a while
$ # visit http://localhost:8421/init
```

Now re-upload the file using the interface at the bottom of the screen.

Now Salmon will issue instructions to restart. Let's do that:

```
$ # Now, let's follow the directions Salmon gave:
$ docker-compose stop; docker-compose start
$ docker-compose logs -f
$ # visit http://localhost:8421/dashboard
```

1.2 Getting started

Launching an experiment to crowdsourcing participants requires following this process:

1. Visiting `http://[url]:8421/init` with the `[url]` from *Installation*.
2. Creating a username/password
3. Launching an experiment.
4. Sending the URL `http://[url]:8421/` to crowdsourcing participants.

1.2.1 Initialization page

By default, Salmon does not support HTTPS. Make sure the URL begins with `http://`, not `https://`. For example, the URL you visit may look like:

```
http://ec2-52-204-122-132.compute-1.amazonaws.com:8421/init
```

1.2.2 Username/password

When visiting `http://[url]:8421/init`, first, type a username/password and hit "create user."

> **Warning:** Do not lose this username/password! You need the username/password to view the dashboard and download the received responses.

It is technically possible to recover the username/password with the key file `key.pem` that Amazon AWS provides and the URL above:

```
(personal) $ ssh -i key.pem ubuntu@[url]
(ec2) $ cat /home/ubuntu/salmon/creds.json
```

1.2.3 Experiment initialization

Hit the back button to visit `http://[url]:8421/init` again after after a user has been successfully created. Now, let's launch an experiment! There are three options:

1. Upload of a YAML file completely detailing the experiment.
2. Upload of a YAML file describing experiment, and ZIP file for the targets.
3. Upload of a database dump from Salmon.

These options are detailed at "*Experiment initialization*." If an experiment is incorrectly specified, (hopefully helpful) errors will be raised. After the experiment is finished launching, you will see a couple links:

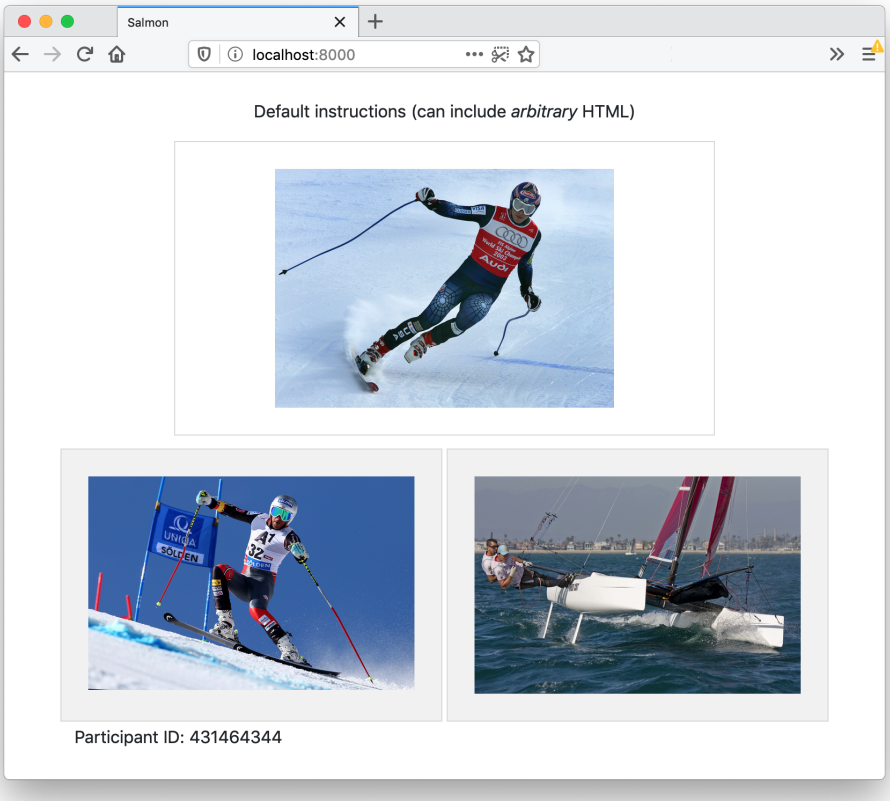
- A link to the dashboard (`http://[url]:8421/dashboard`), an example of which is at "*Experiment monitoring*."
- A link to the query page to send to crowdsourcing users (`http://[url]:8421/`).

1.2.4 Send the URL to participants

The URL to send to the crowdsourcing participants is `http://[url]:8421/`. Typically, paid services like Mechantical Turk are used to recruit crowdsourcing participants. Reddit and email have been used for unpaid recruitment. In either case, that may involve a URL of this form:

```
http://ec2-52-204-122-132.compute-1.amazonaws.com:8421/
```

Opening this URL in the browser will show (a newer version) of this page:



A couple notes:

- Customizing this page is possible and detailed in *Frontend customization*.
- Tips on deploying this experiments can be found at *Deploying*.

1.3 Experiment initialization

Throughout the documentation, the YAML file for initialization will be referred to as `init.yaml`.

Note: This method is required even if images/videos are includes in a ZIP file (as described in *YAML file with ZIP file*). Uploading a ZIP file only modifies the `targets` "key" in the YAML file.

Now, let's describe three methods on how to launch this experiment:

1.3.1 Experiment initialization with YAML file

"YAML files" must obey a standard; see for a (human-readable) description of the specification https://learnxinyminutes.com/docs/yaml/. To see if your YAML is valid, go to https://yamlchecker.com/. Here's an example `init.yaml` YAML file for initialization:

```
# file: init.yaml
targets: ["l", "<i>kildow</i>", "t", "<i>ligety</i>"]  # or uploaded via ZIP file
html:
  instructions: Select the item on the bottom most similar to the item on the top.
  debrief: Thanks! Use the participant ID below in Mechnical Turk.
  max_queries: 100
```

This file will initialize a basic experiment. By default, Salmon will do the following:

- **Use random sampling.** This is a very simple configuration – but it may not be what you want. Relevant FAQs:
  - "*When should I use random/active sampling?*"
  - "*What active samplers are recommended?*" This FAQ links to *Sampler configuration*.
- Ask 50 questions before showing the participant ID.
- Embed into $d = 2$ dimensions if active samplers are specified.

More documentation on customizing these fields can be found in *Sampler configuration* and *Frontend customization*, and the defaults for instructions/debrief can be found in *HTML*. To do anything fancier, additional configuration is required. Here's a basic example:

```
# file: init.yaml
targets: ["l", "<i>kildow</i>", "t", "<i>ligety</i>"]  # or uploaded via ZIP file
html:
  max_queries: 100
samplers:
  ARR: {"random_state": 42}
  Random: {}
sampling:
  probs: {"ARR": 85, "Random": 15}
  common:
    d: 3  # embed into 3 dimensions for all active samplers
```

The `samplers` controls which methods get to choose queries, and `sampling` controls how multiple samplers interact (i.e., how should a sampler be chosen?) More documentation can be found at *Sampler configuration*.

Complete details on the YAML file are at at *Config*. Examples of these files are in salmon/examples. A complete example is available at salmon/examples/complete.yaml.

1.3.2 YAML file with ZIP file

Uploading a ZIP file for targets/stimuli is a small addition to "*Experiment initialization with YAML file*." The only difference is that uploading a ZIP file overwrites and configures the `targets` key for you (so it's not necessary to specify the `targets` key when uploading a ZIP file).

Here are the choices for different files to include in the ZIP file:

- A single CSV file. Each textual target should be on a new line.
- A bunch of images/videos. Support extensions:

  - Videos: `mp4`, `mov`
  - Images: `png`, `gif`, `jpg`, `jpeg`

A YAML file must be uploaded describing the experiment in addition to including the targets in the ZIP file. Let's walk through two examples, both with uploading a bunch of images with skiers. Both cases will use this `init.yaml` file:

```
# file: init.yaml
html:
  instructions: >  # multi-line YAML string
      Select the <i>comparison</i> item on the bottom that
      is most similar to the <i>target</i> item on the top.
  debrief: <b>Thanks!</b> Use the participant ID below in Mechanical Turk.
  max_queries: 100
```

Note: Uploading a ZIP file completely replaces any specification of the `targets` key above. This means that it is not necessary to specify the `targets` key when a ZIP file is uploaded because it will be specified automatically.

Images/videos

If I had all these images in a ZIP file (say `skiers.zip`), I would gather all the images into a ZIP file. On macOS, that's possible by selecting all the images then control-clicking and selecting "Compress items." On the command line, the command `zip targets.zip *.jpg *.png` will collect all JPG/PNG images into `targets.zip`.

Text targets

This is a valid CSV file that will render textual targets:

```
# file: targets.csv. Zipped into targets.csv.zip and uploaded.
Bode Miller
Lindsey Kildow
Mikaela Shiffrin
<b>Ted Ligety</b>
Paula Moltzan
Jessie Diggins
```

Again, every line here is valid HTML, so the crowdsourcing participant will see bolded text for "**Ted Ligety**." That means we can also render images:

```
# file: targets.csv. Zipped into targets.csv.zip and uploaded.
<img width="300px" src="https://upload.wikimedia.org/wikipedia/commons/3/30/Bode_Miller_
↪at_the_2010_Winter_Olympic_downhill.jpg" />
<img width="300px" src="https://upload.wikimedia.org/wikipedia/commons/8/89/Miller_Bode_
↪2008_002.jpg" />
<img width="300px" src="https://upload.wikimedia.org/wikipedia/commons/5/5e/Lindsey_
↪Kildow_Aspen.jpg" />
<img width="300px" src="https://commons.wikimedia.org/wiki/File:Michael_Sablatnik_Slalom_
↪Spital_am_Semmering_2008.jpg" />
<img width="300px" src="https://upload.wikimedia.org/wikipedia/commons/e/e9/Kjetil_
↪Jansrud_giant_slalom_Norway_2011.jpg" />
```

One rendered target will be this image:

1.3.3 Database dump

The dashboard offers a link to download the experiment on the dashboard (that is, at `http://[url]:8421/dashboard`). This will download a file called `exp-[date]-vX.Y.Z.rdb`. Do not delete the numbers `X.Y.Z`!

Salmon supports the upload of this file to the same version of Salmon. The upload of this file will restore the state of your experiment. After this file is uploaded, the two machines will become indistinguishable from each other (which allows you to download the entire experiment onto your own machine then upload it to a completely new machine a month later and start collecting responses again).

If you run into errors, the FAQ "*Restoring from a backup didn't work*" will likely be relevant.

1.4 Sampler configuration

There are many queries to ask about in triplet embedding tasks. Most of these queries aren't useful; chances are most queries will have obvious answers and won't improve the embedding much.

Choosing the most useful queries to improve the embedding is the task of "active machine learning" aka "adaptive sampling algorithms." These algorithms use all previous responses collected to determine the next query that will help improve the embedding the most.

If quality embeddings are desired, the benchmarks at "*Active sampling*" are likely of interest, as are the FAQs on "*When should I use random/active sampling?*" and "*What active samplers are recommended?*" However, quality embeddings may not always be of interest: sometimes, the goal is not to generate a good embedding, but rather to see how well the crowd agrees with each other, or to make sure participants don't influence each other (and each response is independent of other responses). Here's a rule of thumb:

Note: Want an unbiased estimate of what the crowd thinks? Don't specify `samplers`, or use the default `Random: {}`, which will rely on *Random*

Want to generate a better embedding? Worried about the cost of collecting responses? Use `ARR: {}`, which will rely on *ARR*.

Want to measure how well the crowd agrees with one another? Use `Validation: {}`, which will rely on *Validation*,

---

Now, let's go over how to configure different samplers and an example.

1.4.1 File structure

Part of a `init.yaml` configuration file might look like this:

```yaml
# file: init.yaml
samplers:
  ARR: {random_state: 42}
  Random: {}
sampling:
  probs: {"ARR": 85, "Random": 15}
  samplers_per_user: 0  # use probability above
```

This will create a versions of *ARR* with `random_state=42` would be created alongside the default version of *Random*. When a query is generated, it will be generated from `ARR` 85% of the time and from `Random` the rest of the time. Generally, the keys in `samplers` and `sampling` follow these general rules:

- `samplers`: controls how one specific sampler behaves (i.e., sampler initialization).
  - The type of sampler is specified by each key (e.g., key `ARR` corresponds to *ARR*), and any arguments are initialization parameters for that object.
- `sampling`: controls samplers interact. For example:
  - the `probs` key controls how frequently each sampler in `samplers` is used
  - the `common` key controls sending initialization arguments to *every* sampler.
  - the `samplers_per_user` key controls how many samplers are seen by any one user who visits Salmon.

A good default configuration is mentioned in the FAQ "*What active samplers are recommended?*" The defaults and complete details are in *Config*.

Multiple samplers of the same name

If two samplers with different purposes want to be used, the key `class` is used to specify which type of sampler should be used:

```
# file: init.yaml
samplers:
  testing:
    class: Random
  training:
    class: Random
```

This will generate queries from these two samplers with equal probability:

```
from salmon.triplets.samplers import Random
Random()
Random()
```

Initialization arguments

Arguments inside each key are passed to the sampler. For example,

```
# file: init.yaml
samplers:
  ARR:
    random_state: 42
    module: CKL
    d: 3
    scorer: "uncertainty"
```

would create an instance of *ARR*:

```
from salmon.triplets.samplers import ARR
ARR(random_state=42, module="CKL", d=3, scorer="uncertainty")
```

Note that the argument are documented in *ARR*. Some argument are arguments that *ARR* directly uses (like `module`), and other are passed to *Adaptive* as mentioned in the docstring of *ARR*.

If you have multiple arguments for *every* sampler, you can specify that with the `common` key:

```yaml
# file: init.yaml
samplers:
  arr_tste:
    module: TSTE
  arr_ckl:
    module: CKL
sampling:
  common:
    d: 3
    random_state: 42
```

This would initialize these samplers:

```
from salmon.triplets.samplers import ARR
ARR(module="TSTE", d=3, random_state=42)
ARR(module="CKL",  d=3, random_state=42)
```

The documentation for ARR is available at *ARR*.

1.4.2 Example

Let's start out with a simple `init.yaml` file, one suited for random sampling.

```yaml
targets: ["obj1", "obj2", "foo", "bar", "foobar!"]
samplers:
  Random: {}
  Validation: {"n_queries": 10}
```

By default, `samplers` defaults to `Random: {}`. We have to customize the `samplers` key use adaptive sampling algorithms:

```yaml
targets: ["obj1", "obj2", "foo", "bar", "foobar!"]
samplers:
  ARR: {}
  Random: {}
  Validation: {"n_queries": 10}
sampling:
  probs: {"ARR": 70, "Random": 20, "Validation": 10}
```

When ARR is specified as a key for `samplers`, *salmon.triplets.samplers.ARR* is used for the sampling method. Customization is possible by passing different keyword arguments to *ARR*. For example, this could be a configuration:

```yaml
targets: ["obj1", "obj2", "foo", "bar", "foobar!"]
samplers:
  Random: {}
  ARR:
    module: "TSTE"
```

Validation sampler

Note: generating validation queries will likely require two uploads of your experiment and resetting Salmon

The indices for *Validation* are indices of the target list, which is available at `http://[url]:8421/config`. This code generates the list:

```
>>> import yaml
>>> from pathlib import Path
>>> # config.yaml from [1], copy/pasted into text file named "config.yaml"
>>> # [1]:http://[url]:8421/config
>>> config = yaml.safe_load(Path("config.yaml").open())
>>>
>>> # Items will be selected from this list
>>> config["targets"]
['soccer', 'skiing', 'curling', 'skating', 'hockey']
```

In this case, if you wanted to ask the query "is skating more similar to hockey or curling?", you would specify:

```
Validation(..., queries=[(2, 3, 4)])
```

Let's check that these are the right indices:

```
>>> targets[2]
'curling'
>>> targets[3]
'skating'
>>> targets[4]
'hockey'
```

If the YAML configuration file, indices are specified as below:

```
samplers:
  Validation:
    queries:
      - [2, 3, 4]
      - [1, 0, 3]
  # (if asking the above query and
  # a query with head "skiing" and feet "soccer" and "skating".
```

Sampling detail

Let's say you want to collect data from three samplers: an active sampler for training, a random sampler for testing and a validation sampler to see measure each participant's attention (e.g., "are they blindly clicking answers?").

That sampling is possible through this partial config:

```
samplers:
  ARR: {}  # generating the embedding
  Random: {}  # testing the embedding
  Validation: {}  # measure of human quality

sampling:
  probs: {ARR: 80, Random: 20, Validation: 0}
```

(continues on next page)

(continued from previous page)

```
  details:
    1: {sampler: Validation, query: [0, 1, 2]}
    10: {sampler: Validation, query: [0, 1, 2]}

targets: [zero, one, two, three, four]
# targets are indexed by Python. Each target above is a textual representation
# of the index. For index 0 in sampling.details.query will
# show the user ``targets[0] == "zero"``.

 html:
   max_queries: 10
```

With this `init.yaml`, the crowdsourcing participant will see the same query at the beginning and end (both the 1st and 10th query they see). It will have head `"zero"` and feet `"one"` and `"two"`.

More detail on the target indexing is in *Validation sampler* and *`Validation`*. Another example is in *`Sampling`*.

1.5 Frontend customization

1.5.1 Custom HTML

As mentioned in the *`HTML`*, it's possible to include custom HTML in the query page by customizing the `element_top`, `element_middle`, `element_bottom` and `element_standalone` fields. You'll probably have to look at query_page.html to see exactly the custom HTML elements are inserted.

For example, if you wanted to wanted to include a button with the showing "Skip this question" and didn't want to set `html.skip_button = true`, you could follow this example:

```
html:
  # skip_button: true  # a much easier way to include the HTML code below!
  element_bottom: >
    <div class="d-flex align-items-end flex-column">
      <div id="skip-button" style="padding: 30px; padding-right: 60px;">
        <button type="button" class="btn btn-outline-secondary btn-sm" data-html="true"
                onclick="getquery()" data-toggle="tooltip" data-placement="top"
            title="Skip questions only if you know <em>nothing</em> about the items␣
↪shown."
        >Skip this question</button>
      </div>
    </div>
```

A much easier way to get this is to set `html.skip_button = true` in `init.yml`.

1.5.2 CSS Styling

As mentioned in the *HTML*, it's possible to customize the CSS by including a CSS field in the YAML file. For example, let's say you were asking about the similarity of different colors, and wanted the background to be dark to provide better contrast. This YAML block in `init.yml` would implement that:

```
html:
  instructions: Which of the comparison colors is most similar to the target color?
  debrief: Thanks! Please enter this participant ID in the assignment. max_queries: 100
↪# about 2.94 minutes for 100 queries
  arrow_keys: True
  css: >
    body {
      background-color: #414141;
      color: #ddd;
    }
    .answer {
      background-color: #595959;
      border-color: #bbb;
    }
    .head {
      background-color: #505050;
      border-color: #000;
    }
    .debrief {
      color: #000;
    }
```

1.6 Deploying

Most commonly, I've seen Salmon and interfaces like it deployed to crowdsourcing services (e.g., Amazon's Mechanical Turk). With these services, Salmon is typically deployed to these services by letting crowdsourcing participants click on a single URL and go through various web pages before entering a code into MTurk indicating the participant completed the study.

I have a couple recommendations for this and similar processes:

- **URL redirection.** Do not give Amazon AWS EC2 URLs/DNSs directly to crowdsourcing participants. Instead, give them a URL that redirects to the Amazon EC2 URL/DNS (e.g., via GitHub Pages, `foobar.github.io/mturk.html`).
  - This is a best practice because it avoids debugging production servers. If something goes wrong with your machine (like it's overloaded with too many users), having some redirection scheme allows redirectign crowdsourcing participants away from your machine.
- **Host detailed instructions/etc elsewhere.** This HTML files are typically shown after the crowdsourcing clicks on a link and before they see Salmon. (e.g., for an IRB notice). It is *technically* possible to include these instructions by customizing the Salmon's query page with *Frontend customization*).

Hosting HTML pages is possible and relatively straighforward with GitHub Pages, as are URL Redirections.

In general, I've observed some crowdsourcing trends:

- I've noticed two levels of fraud: crowdsourcing participants who submit bogus codes (like their Amazon MTurk ID) and those who answer responses rather quickly, too quickly for human response time.

- I've generally observed the responses from crowdsourcing participants to be high quality. I have noticed some "bad" or "junk" responses, and throw them out before I start my analysis.

Please file an issue or reach out via email if you have any more deployment questions.

1.7 Experiment monitoring

While participants are answering queries, you can track the state of the experiment by visiting `http://[url]:8421/dashboard`. Initially, this will be fairly barren because users haven't responded.

After users respond, the dashboard will show the following information:

- **Basic information** (start time, number of participants, etc)
- **User response times** (e.g., when did the users respond?)
- **Server side timing** (e.g., how long did it take to process each API endpoint?)
- **Client side timing** (e.g., how long did they spend waiting or thinking about their response?)
- **Algorithm timing** (e.g., how long do model updates take?)
- An **embedding** for each algorithm.
- The list of targets.

1.7.1 Basic information

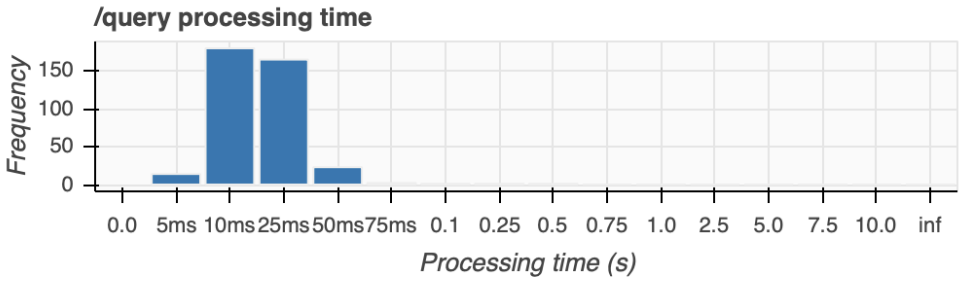
Bolded links are particularly useful. I download all bolded links before stopping or restarting the machine running Salmon (except the query page).

1.7.2 Server side timing

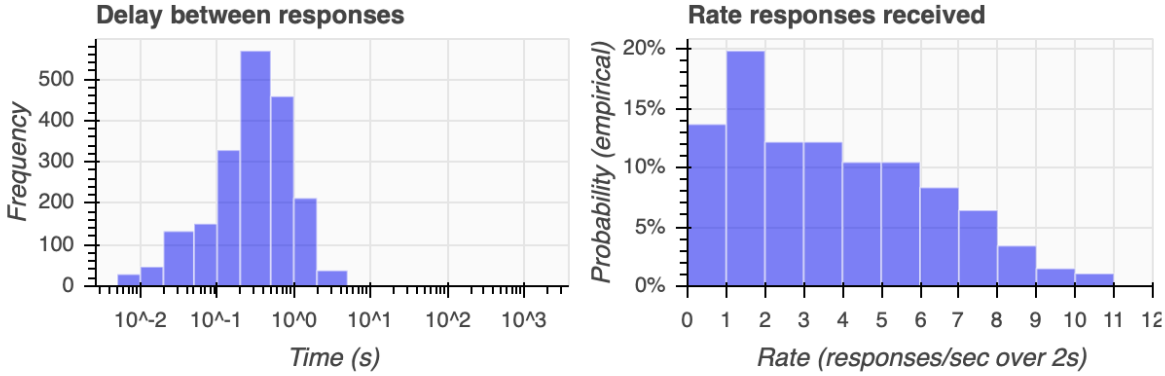
Server-side timing

Here's the computation time required to complete each of the API endpoints on this machine:

/query /dashboard /answer

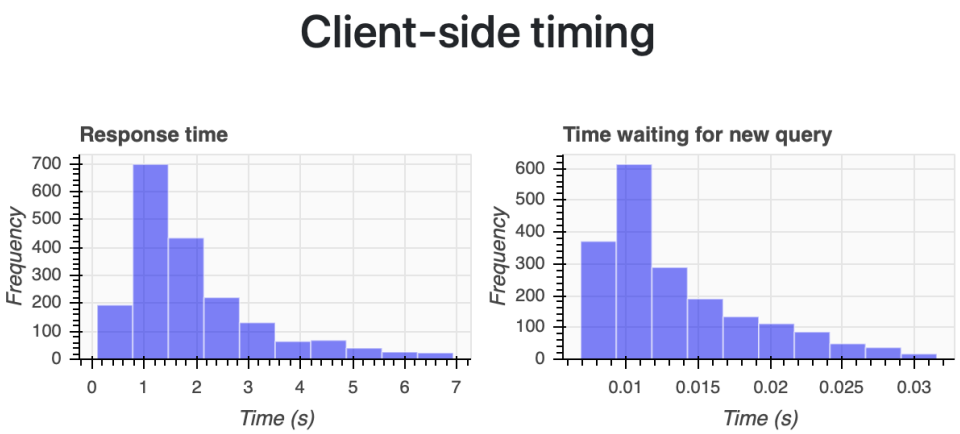


The median delay between responses is 0.32 seconds.



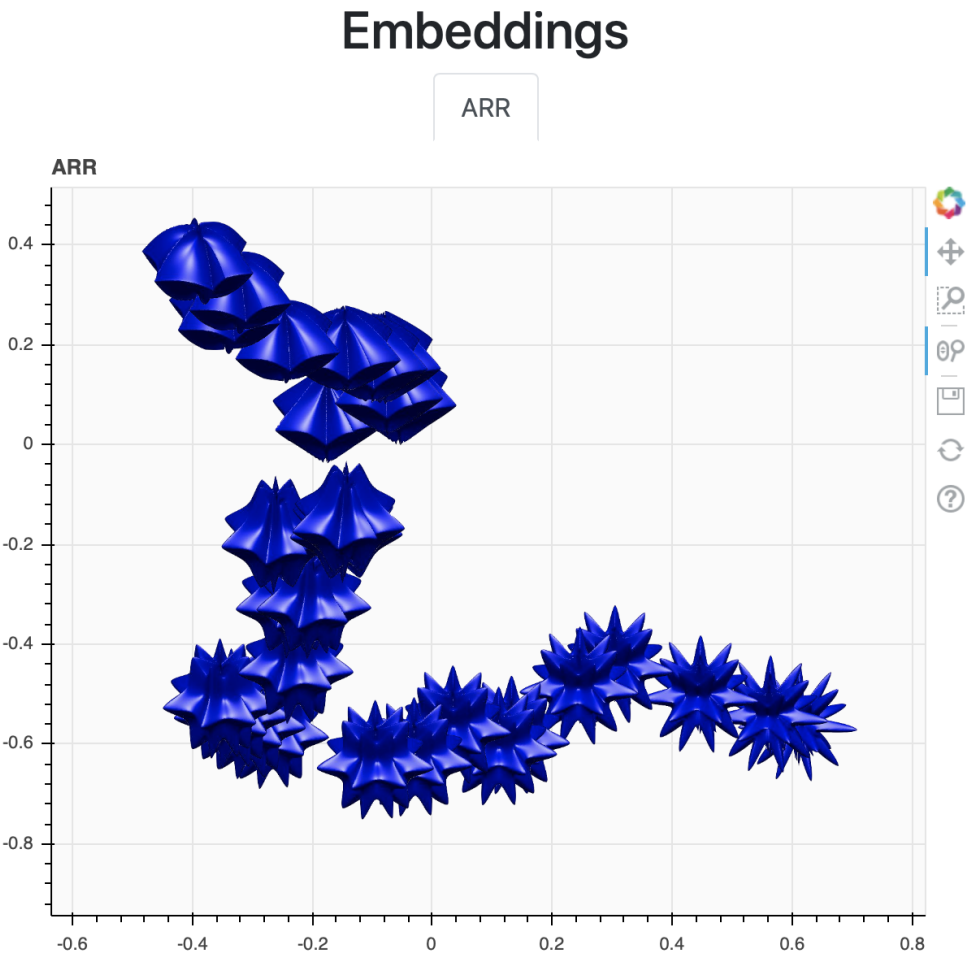
Redis live monitoring.

1.7.3 Client side timing

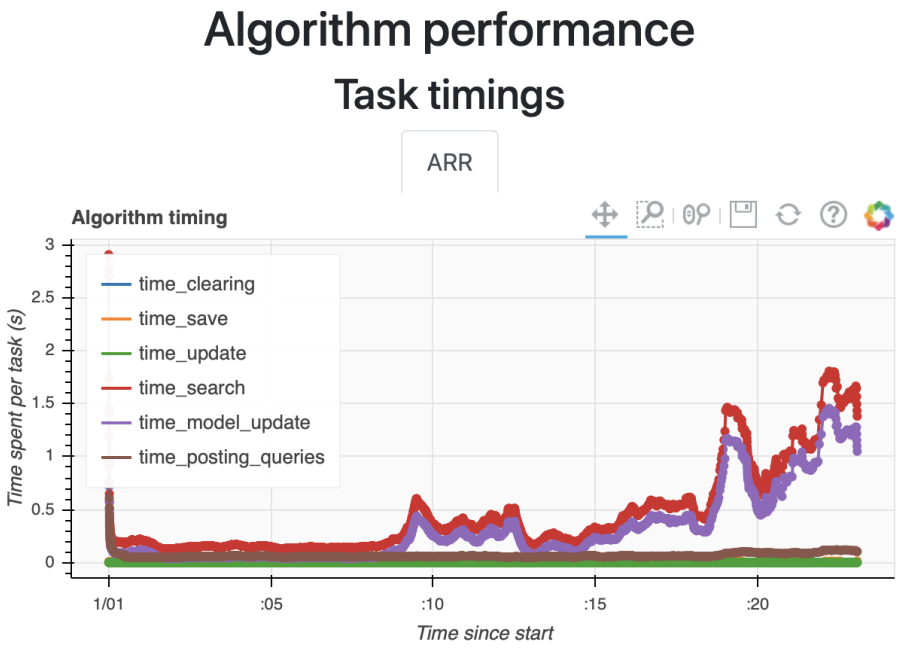


These times are recorded client-side and reported back to Salmon.

1.7.4 Embeddings

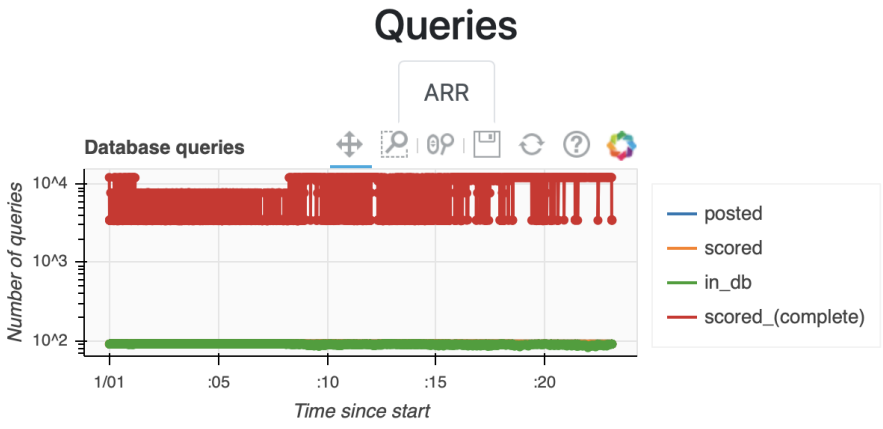


1.7.5 Algorithm timing



Searching queries, posting queries and model updates are performed in parallel on three workers. Each worker has one task. However, the three workers are shared between all active algorithms.

1.7.6 Database



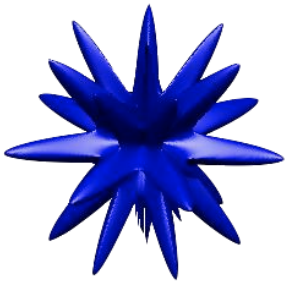
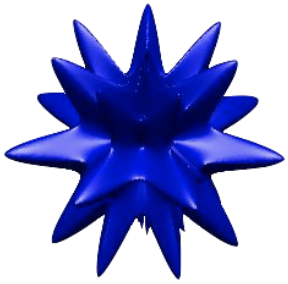
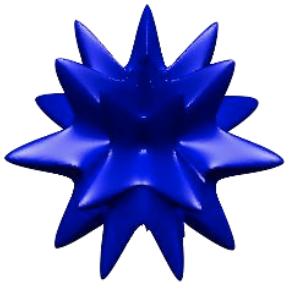
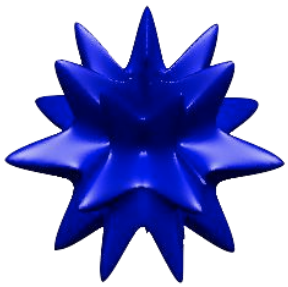
The number of the queries in the database for each algorithm is limited to the number of possible queries, `n*(n-1)*(n-2) / 2`.

Also, algorithm `RR` posts `3*n` of the highest scoring queries to the database when `n` is the number of targets. The label "scored_(complete)" records this value.

1.7.7 Targets

I only show the first 4 targets below; the other targets can be seen in the embedding above.

Targets

Target item	Rendered HTML	Filename/URL/Raw HTML
0		i0022.png
1		i0036.png
2		i0050.png
3		i0074.png
4		i0076.png

1.8 Generating embeddings offline

Salmon is good at asking useful questions to crowdsourcing participants.

These responses should be used to create an embedding *offline*. Yes, Salmon has to generate an embedding to determine which questions are useful... but that shouldn't be the final embedding used for downstream analysis.

This documentation page will step through the process required to generate the embedding:

1. Download responses and experiment
2. Install Salmon on your own machine.
3. Generate embedding.

1.8.1 Downloading responses

Download the responses, either by visiting `http://[url]:8421/responses` or clicking the link on the dashboard (as mentioned in *Experiment monitoring*).

1.8.2 Install Salmon

There are two options to install Salmon for offline embeddings. **Using ``conda`` is preferred** because it installs all the requirements (including Python 3.8, which might not be installed) and has more sophisticated conflict resolution than `pip`.

Using conda

This option is required for a complete installation. This option requires `conda`, a Anaconda's Python package manager. It's available through Anaconda and Miniconda.

1. Download the latest release of Salmon.
2. Unzip/unpack the *.zip* or *.tar.gz* file.
3. Navigate to the directory in the shell/terminal and run these commands:
4. Then run these commands:

```
$ cd ~/Downloads/salmon  # directory just downloaded and unzipped
$ conda env create -f salmon.lock.yml
$ conda activate salmon
(salmon) $ pip install .
```

These commands should be run in your favorite terminal. On macOS, that might be Terminal.app.

Note: The commands above are (*nix) shell commands. The `$` is intended to be your terminal prompt; leave it out when copy and pasting into the terminal.

Using pip

This option is recommended to generate embeddings offline. This option requires `pip`, a Python package manager. It's available through Anaconda and Miniconda.

After you have the Python package manager `pip`, run these commands:

```
$ pip install "salmon-triplets"
$ python -c "from salmon.triplets.offline import OfflineEmbedding"
```

You have successfully installed Salmon if these commands complete successfully.

Note: This package named "salmon-triplets" on PyPI installs a Python package named `salmon`.

1.8.3 Generate embeddings

This Python code will generate an embedding:

```
import pandas as pd
from sklearn.model_selection import train_test_split

from salmon.triplets.offline import OfflineEmbedding

# Read in data
df = pd.read_csv("responses.csv")  # from dashboard
em = pd.read_csv("embedding.csv")  # from dashboard; optional

X = df[["head", "winner", "loser"]].to_numpy()
X_train, X_test = train_test_split(X, random_state=42, test_size=0.2)

n = int(X.max() + 1)  # number of targets
d = 2  # embed into 2 dimensions

# Fit the model
model = OfflineEmbedding(n=n, d=d, max_epochs=500_000)
model.initialize(X_train, embedding=em.to_numpy())  # (optional)

model.fit(X_train, X_test)

# Inspect the model
model.embedding_  # embedding
model.history_  # to view information on how well train/test performed
```

Some customization can be done with `model.history_`; it may not be necessary to train for 500,000 epochs. `model.history_` will include validation and training scores, which might help limit the number of epochs.

Documentation for *`OfflineEmbedding`* is available on *API*.

1.8.4 Embedding visualization

The HTML for each target alongside the embedding coordinates is available from the dashboard by downloading the "embeddings" file (or visiting `[url]:8421/embeddings`. This will give a CSV with the HTML for each target, the embedding coordinates and the name of the embedding that generated the algorithm.

To visualize the embedding, standard plotting tools can be used to visualize the embedding, which might be Matplotlib, the Pandas visualization API, Bokeh or Altair. The Pandas visualization API is likely the easiest to use, but won't support showing HTML (images/video/etc). To do that, Salmon uses Bokeh for it's visualization.

1.9 FAQ

Also relevant is the *Troubleshooting*, which goes over some (blocking) difficulties while launching.

Note: Please include the version in any bug reports or feature requests. The version number should look something like `v0.4.1`. It can be found at `http://[url]:8421/docs` or in the downloaded experiment file (found at `http://[url]:8421/download` which has a filename like `exp-2021-05-20T07:31-salmon-v0.4.1.rdb`).

1.9.1 When should I use random/active sampling?

Rule of thumb:

- Use random sampling for simple problems when not many responses are required (small number of targets and clean responses).
- Use active sampling for anything more complicated (large number of targets or noisy responses) when the crowdsourcing budget and/or embedding quality are relevant.

Specifics are *How many responses will be needed?* and *What active samplers are recommended?*. Random sampling can produce good embeddings, but will require about 3× the number of responses that active sampling requires.

By default, Salmon will produce random embeddings. This is the simplest sampler, and doesn't require any user configuration. Tips on how to use active samplers are in *What active samplers are recommended?*.

1.9.2 How many responses will be needed?

Depends on the targets used, and how humans respond. Let's say there are n targets that are being embedded into d dimensions. At most, we can provide bounds on how many responses you'll need:

- **Lower bound:** at least $nd\log_2(n)$ responses are needed for a *perfect* embedding with noiseless responses. Active triplet algorithms require $\Omega(nd\log_2(n))$ responses (so a constant number of responses more/less).[1]
- **Upper bound:** if random responses are collected, a high quality embedding will likely be generated with $O(nd\log_2(n))$ responses, likely $20nd\log_2(n)$ responses (or possibly $10nd\log_2(n)$).[2]

This suggests that the number of responses required when n and d are changed scaled like $nd\log_2(n)$. i.e, if an embedding below requires 5,000 responses for $n = 30$, scaling to $n = 40$ with $d = 1$ would likely require about $3600 \approx 5000\frac{40\cdot1\cdot\log_2(40)}{30\cdot2\cdot\log_2(30)}$ responses for the same dataset.

[1] "Low-dimensional embedding using adaptively selected ordinal data." Jamieson and Nowak. 2011. Allerton Conference on Communication, Control, and Computing. https://homes.cs.washington.edu/~jamieson/resources/activeMDS.pdf

[2] "Finite sample prediction and recovery bounds for ordinal embedding." Jain, Jamieson and Nowak. 2016. NeurIPS. https://nowak.ece.wisc.edu/ordinal_embedding.pdf

See the *benchmarks on active sampling* for some benchmarks/landmarks on the specific number of responses required for a particular dataset. **If you think your dataset will require too many responses,** see *our recommendations on active samplers*. Active samplers might be able to generate better embeddings with a fixed number of responses.

1.9.3 What active samplers are recommended?

Use of *ARR* is most recommended. See the *benchmarks on active sampling* for an example configuration and the number of responses required for that usage. The defaults of *ARR* have been explored pretty throughly. In the *benchmarks on active sampling*, we used the default parameters for *ARR* after exploring the possible values.

Monitoring performance is difficult with active/adaptive algorithm; random sampling is a lot better. Typically, between 10% and 20% of the sampling is used to monitor and report performance. That means I'd recommend this partial configuration:

```
samplers:
  ARR: {}
  Random: {}
sampling:
  probs: {"ARR": 85, "Random": 15}
```

1.9.4 Can I choose a different machine?

All of our experiments are run with `t3.xlarge` instances. If you want to choose a different machine, ensure that is has the following:

- At least 4GB of RAM
- At least 3 CPU cores.

These are required because Salmon requires 3.2GB of memory and Dask has three tasks per adaptive algorithm: posting queries, searching queries, model updating. Generally, the number of cores should be `3 * n_algs`. This isn't a strict guideline; only 2 out of the 3 tasks take significant amounts of time. Using `2 * n_algs` will work at a small performance hit; we recommend at least 4 cores for two algorithms.

1.9.5 How do I ask specific questions?

It's possible to configure what queries get shown to the crowdsourcing user with two methods:

1. By using *Validation*. You can specify a list of queries, or generate `n_queries` random queries. See *Sampler configuration* and the *Validation sampler* section.
2. By specifying the `query` key in *Sampling*'s `detail` field. This allows showing users specific queries at specific times. e.g., "for the first query the user sees, show them a query with [these objects]."
    - Example: *Sampling detail*
    - Example: the documentation for *Sampling*.

1.9.6 How do I specify when samplers are used?

Controlling when or how often samplers get used is possible with two methods:

1. By setting the `probs` key in *Sampling*. If the YAML specifies `sampling.probs: {"ARR": 80, "Random": 20}`, the `Random` sampler will be used 20% of the time.
2. By setting the `sampler` field in *Sampling*'s `detail` field. This will ensure that the query is generated by a specific sampler (or which sampler will receive the answer if the query is pre-specified in your `init.yaml` per "*How do I ask specific questions?*").

When `sampling.probs` is specified in your `init.yaml`, Salmon serves a query to a user with a certain *probability*. This can pose difficulties if you want to ask *exactly N* questions to each crowdsourcing participant. Specifying `sampling.details` in your `init.yaml` will work around this and allow configuring the details of particular queries.

1.9.7 How do I ask every crowdsourcing user exactly the same questions?

By specifying both the `sampler` key and the `query` key in *Sampling*'s `detail` field. The answers to the questions "*How do I specify when samplers are used?*" and "*How do I ask specific questions?*" are relevant.

An example is in "*Sampling detail*", delegated to the *Sampler configuration* page because the target indexing in "*Validation sampler*" is relevant.

1.9.8 How do I see the Dask dashboard?

Look at port 8787 if you want more information on how jobs are scheduled. If on EC2, this will require some port forwarding to your own machine:

```
ssh -i key.pem -L 7787:localhost:8787 ubuntu@[EC2 public DNS or IP]
# visit http://localhost:7787 in the browser to see Salmon's Dask dashboard
```

If desired, it is possible to open port 8787 on the Amazon EC2 machine. If that action is taken, it is recommended to only allow a specific IP to view that port.

1.9.9 How do I customize the participant unique identifier aka "puid"?

Visiting `http://[url]:8421/?puid=foobar` will set that the participant UID to be `foobar`.

1.9.10 How do I use HTTPS with Salmon?

HTTP is how web servers communicate; HTTPS protects that communication from third parties.

Some crowdsourcing services require HTTPS. There are to ways to provide these crowdsourcing services an HTTPS URL:

1. Redirect to Salmon from an HTTPS page.
2. Set up a TLS termination proxy.

Option (1) is a lot easier because various hosting services support HTTPS (e.g., GitHub Pages and GitLab Pages support HTTPS for custom domains). Hosting a redirect HTML page at one of these services with HTTPS will likely satisfy any requirements you may have.

Option (2) is more complex. A good overview is at FastAPI's page "About HTTPS," available at https://fastapi.tiangolo.com/deployment/https/. This process is beyond scope for this project.[3]

The Docker machines aren't launching

Are you using the command `docker-compose up` to launch Salmon? The command `docker build .` doesn't work.

Salmon requires a Redis docker machine and certain directories/ports being available. Technically, it's possible to build all the Docker machines yourself (but it's not feasible).

1.10 API

1.10.1 Configuration

salmon.triplets.manager.Config	Customization of Salmon sampling and HTML.

salmon.triplets.manager.Config

pydantic settings salmon.triplets.manager.**Config**

Customization of Salmon sampling and HTML.

This class is often specified through uploading an `init.yaml` file as described in *Experiment initialization with YAML file*.

The default configuration is specified completely below, which is rendered into a YAML file in an example.

Some details on specific fields:

- The only required field is targets.
  - `targets` is often specified with the upload of a ZIP file. See *YAML file with ZIP file* for more detail. If not, specify a list of strings (which will be rendered as HTML).
- `sampling` is not relevant until `samplers` is customized.

Here's how to customize the config a bit with a YAML file:

```yaml
samplers:
  testing: {class: Random}
  ARR: {}
  Validation:
    n_queries: 20
sampling:
  probs: {"ARR": 40, "Validation": 20, "testing": 20}
  common:
    d: 3  # dimensions to embed to; passed to every sampler
    random_state: 42  # argument to Adaptive
    initial_batch_size: 128  # argument to Embedding
html:
  instructions: Click buttons, <b><i>or else.</i></b>
```

[3] though the package mkcert might help.

Full documentation is below.

Note: `sampler_per_user` and `detail` only affect the HTML page shown to the user. They do not affect the backend code executed. They are not grouped with *HTML* because they do affect the responses (at least when using Salmon's default config and not writing a custom query page).

```
{
   "title": "Config",
   "description": "Customization of Salmon sampling and HTML.\n\nThis class is␣
↪often specified through uploading an ``init.yaml`` file as\ndescribed in␣
↪:ref:`yamlinitialization`.\n\nThe default configuration is specified completely␣
↪below, which is rendered into a YAML file in `an example`_.\n\n.. _an example:␣
↪https://github.com/stsievert/salmon/blob/master/examples/default.yaml\n\nSome␣
↪details on specific fields:\n\n* The only required field is targets.\n\n    *␣
↪``targets`` is often specified with the upload of a ZIP file. See\n      ␣
↪:ref:`yaml_plus_zip` for more detail. If not, specify a list of strings (which\n ␣
↪    will be rendered as HTML).\n\n* ``sampling`` is not relevant until␣
↪``samplers`` is customized.\n\nHere's how to customize the config a bit with\na␣
↪YAML file:\n\n.. code-block:: yaml\n\n   samplers:\n     testing: {class: Random}\
↪n     ARR: {}\n     Validation:\n       n_queries: 20\n   sampling:\n     probs:
↪{\"ARR\": 40, \"Validation\": 20, \"testing\": 20}\n     common:\n       d: 3  #␣
↪dimensions to embed to; passed to every sampler\n       random_state: 42  #␣
↪argument to Adaptive\n       initial_batch_size: 128  # argument to Embedding\n  ␣
↪html:\n     instructions: Click buttons, <b><i>or else.</i></b>\n\nFull␣
↪documentation is below.\n\n.. note::\n\n   ``sampler_per_user`` and ``detail``␣
↪only affect the HTML page\n   shown to the user. They do not affect the backend␣
↪code executed.\n   They are not grouped with :class:`~salmon.triplets.manager.
↪HTML`\n   because they do affect the responses (at least when using\n   Salmon's␣
↪default config and not writing a custom query page).",
   "type": "object",
   "properties": {
      "targets": {
         "title": "Targets",
         "description": "A list of targets that will be rendered as HTML. If a ZIP␣
↪file is\n            uploaded, this field is populated automatically.\n          ␣
↪ See :ref:`yaml_plus_zip` for more detail.",
         "env_names": "{'targets'}",
         "anyOf": [
            {
               "type": "integer"
            },
            {
               "type": "array",
               "items": {
                  "type": "string"
               }
            }
         ]
      },
      "html": {
         "title": "Html",
         "description": "Stylistic settings to configure the HTML page.\n          ␣
```


(continues on next page)

(continued from previous page)

```
↪ See :class:`~salmon.triplets.manager.HTML` for more detail.",
        "default": {
           "title": "Similarity judgements",
           "instructions": "Please select the <i>comparison</i> item that is most␣
↪similar to the <i>target</i> item.",
           "debrief": "<b>Thanks!</b> Please use the participant ID below.",
           "skip_button": false,
           "css": "",
           "max_queries": 50,
           "arrow_keys": true,
           "query_params": [
              "puid"
           ],
           "js": "",
           "element_top": "",
           "element_middle": "",
           "element_bottom": "",
           "element_standalone": ""
        },
        "env_names": "{'html'}",
        "allOf": [
           {
              "$ref": "#/definitions/HTML"
           }
        ]
     },
     "samplers": {
        "title": "Samplers",
        "description": "Samplers to use, and their initialization parameters. See\
↪n            above or \":ref:`adaptive-config`\" for more detail on\n           ␣
↪customization, and :ref:`experiments` for experiments/benchmarks.",
        "default": {
           "random": {
              "class": "Random"
           }
        },
        "env_names": "{'samplers'}",
        "type": "object"
     },
     "sampling": {
        "title": "Sampling",
        "description": "Settings to configure how more than two samplers are\n    ␣
↪   used. See :class:`~salmon.triplets.manager.Sampling` for more detail.",
        "default": {
           "common": {
              "d": 2
           },
           "probs": null,
           "samplers_per_user": 0,
           "details": {}
        },
        "env_names": "{'sampling'}",
```

(continues on next page)

(continued from previous page)

```
        "allOf": [
           {
              "$ref": "#/definitions/Sampling"
           }
        ]
     }
  },
  "additionalProperties": false,
  "definitions": {
     "HTML": {
        "title": "HTML",
        "description": "Stylistic settings to configure the HTML page.\n\nThe␣
↪default configuration is specified completely below, which is rendered into a␣
↪YAML file in `an example`_.\n\n.. _an example: https://github.com/stsievert/
↪salmon/blob/master/examples/default.yaml\n\nArbitrary keys are allowed in this␣
↪class\n(which might allow for customization on the HTML page).",
        "type": "object",
        "properties": {
           "title": {
              "title": "Title",
              "description": "The title of the HTML page (the text shown in the␣
↪tab bar)",
              "default": "Similarity judgements",
              "env_names": "{'title'}",
              "type": "string"
           },
           "instructions": {
              "title": "Instructions",
              "description": "The instructions the user sees above each query.",
              "default": "Please select the <i>comparison</i> item that is most␣
↪similar to the <i>target</i> item.",
              "env_names": "{'instructions'}",
              "type": "string"
           },
           "debrief": {
              "title": "Debrief",
              "description": "The message that the user sees after ``max_queries``␣
↪responses\n            have been completed. The participant ID (``puid``) is␣
↪shown below\n              this message.",
              "default": "<b>Thanks!</b> Please use the participant ID below.",
              "env_names": "{'debrief'}",
              "type": "string"
           },
           "skip_button": {
              "title": "Skip Button",
              "description": "Wheter to show a button to skip queries. Most\n     ␣
↪  useful when participants are instructed to skip queries only when the\n       ␣
↪know nothing about any object.",
              "default": false,
              "env_names": "{'skip_button'}",
              "type": "boolean"
           },
```

(continues on next page)

(continued from previous page)

```
            "css": {
               "title": "Css",
               "description": "CSS to be included the in the query page.  This CSS␣
↪is\n        inserted just before the ``</style>`` tag in the HTML head.",
               "default": "",
               "env_names": "{'css'}",
               "type": "string"
            },
            "max_queries": {
               "title": "Max Queries",
               "description": "The number of queries that the user will answer␣
↪before\n        seeing the ``debrief`` message). Set ``max_queries=0`` or\n      ␣
↪ ``max_queries=-1`` to ask unlimited queries.",
               "default": 50,
               "env_names": "{'max_queries'}",
               "type": "integer"
            },
            "arrow_keys": {
               "title": "Arrow Keys",
               "description": "Wheter to allow using the arrow keys as input. ␣
↪Specifying\n        ``arrow_keys=True`` might allow bad input (though there is a␣
↪delay of\n        200ms between queries).",
               "default": true,
               "env_names": "{'arrow_keys'}",
               "type": "boolean"
            },
            "query_params": {
               "title": "Query Params",
               "description": "A list of flags to include in the params to /query.\
↪n        The default is ``query_params=['puid']``, which hits the endpoint\n     ␣
↪  ``/query?puid=foo`` if the participant's UID is ``foo``. Developers\n       ␣
↪writing custom query pages might be interested in customizing this\n        value.
↪\n        ",
               "default": [
                  "puid"
               ],
               "env_names": "{'query_params'}",
               "type": "array",
               "items": {
                  "type": "string"
               }
            },
            "js": {
               "title": "Js",
               "description": "JavaScript to include in standalone ``<script>`` tag
↪",
               "default": "",
               "env_names": "{'js'}",
               "type": "string"
            },
            "element_top": {
               "title": "Element Top",
```

(continues on next page)

(continued from previous page)

```
                "description": "Custom HTML to include at top of page",
                "default": "",
                "env_names": "{'element_top'}",
                "type": "string"
            },
            "element_middle": {
                "title": "Element Middle",
                "description": "Custom HTML to include in middle of page (between␣
↪target and comparisons)",
                "default": "",
                "env_names": "{'element_middle'}",
                "type": "string"
            },
            "element_bottom": {
                "title": "Element Bottom",
                "description": "Custom HTML to include at top of page (above footer)
↪",
                "default": "",
                "env_names": "{'element_bottom'}",
                "type": "string"
            },
            "element_standalone": {
                "title": "Element Standalone",
                "description": "Custom HTML to include as a standalone element, only\
↪n        encased in the ``head`` tag.\n        ",
                "default": "",
                "env_names": "{'element_standalone'}",
                "type": "string"
            }
        }
    },
    "Sampling": {
        "title": "Sampling",
        "description": "Settings to configure how more than two samplers are used.\
↪n\nThese settings are used in the HTML but are not stylistic. The\nexception is␣
↪``common``, which is passed to every ``sampler`` during\ninitialization and will␣
↪likely be optional arguments for\n:class:`~salmon.triplets.samplers.Adaptive`.\n\
↪nThe default configuration is specified completely below, which is rendered into␣
↪a YAML file in `an example`_.\n\n.. _an example: https://github.com/stsievert/
↪salmon/blob/master/examples/default.yaml",
        "type": "object",
        "properties": {
            "common": {
                "title": "Common",
                "description": "Arguments to pass to every sampler for␣
↪initialization (likely\n        values for :class:`~salmon.triplets.samplers.
↪Adaptive`; note that\n        values for ``n`` and ``ident`` are already␣
↪specified). Any\n        values specified in this field will be overwritten by\n ␣
↪      sampler-specific arguments.",
                "default": {
                    "d": 2
                },
```

(continues on next page)

(continued from previous page)

```
            "env_names": "{'common'}",
            "type": "object"
         },
         "probs": {
            "title": "Probs",
            "description": "The percentage to sample each ``sampler`` when given␣
↪the\n        opportunity (which depends on ``samplers_per_user``). The␣
↪percentages\n        in this sampler must add up to 100.\n        If not␣
↪specified (default), choose each sampler with equal\n        probability.",
            "env_names": "{'probs'}",
            "type": "object",
            "additionalProperties": {
               "type": "integer"
            }
         },
         "samplers_per_user": {
            "title": "Samplers Per User",
            "description": "The number of samplers to assign to each user.␣
↪Setting\n        ``samplers_per_user=1`` means any user only sees queries␣
↪generated\n        from one sampler, and ``sampler_per_user=0`` means the user␣
↪sees a\n        new sampler every query",
            "default": 0,
            "env_names": "{'samplers_per_user'}",
            "type": "integer"
         },
         "details": {
            "title": "Details",
            "description": "Different options for a deterministic choice of␣
↪samplers.\n\n        This dictionary is of the form ``{query_shown_to_user:␣
↪options}``. The\n        ``options`` is a dictionary with up to two keys:\n\n    ␣
↪   - ``sampler`` (required), which reflects which sampler receives the\n        ␣
↪responses)\n\n        - ``query`` (optional), which is a list of length 3␣
↪indicating the target\n          indices appear in the query.\n\n        For␣
↪example, this YAML will ensure the 1st and 10th query the\n        crowdsourcing␣
↪user sees will be from:\n\n        .. code-block:: yaml\n\n          targets:␣
↪[zero, one, two, three, four, five, six]\n          # ^ list of (textual) targets;
↪ target \"zero\" has index 0 and\n          # is targets[0] in Python\n\n        ␣
↪ samplers:\n            ARR: {}\n            Validation: {}\n            valid2:\
↪n              class: Validation\n              n_queries: 3\n\n        ␣
↪sampling:\n            probs: {ARR: 100, Validation: 0, valid2: 0}\n          ␣
↪details:\n              # Each key \"n\" is the n-th query the user sees\n       ␣
↪      # So here the 1st and 10th queries the user sees is customized\n          ␣
↪  1: {sampler: \"Validation\", query: [0, 2, 3]}\n              10: {sampler: \
↪"valid2\"}\n\n          html:\n            # ask 10 queries according to \
↪"sampling.probs\".\n            # The probabilistic sampling will be overriden by␣
↪sampling.details\n            max_queries: 10\n\n        In this case, the␣
↪crowdsourcing user will see the following:\n\n        * 1st query shown will have␣
↪head \"zero\", and feet \"two\" and \"three\".\n        * Queries 2 and 9 will be␣
↪generated by the :class:`~salmon.triplets.samplers.ARR` sampler.\n        * The␣
↪10th query they see (also the last query): one of three\n          (random) fixed/
↪static queries.\n\n        The sampler ``valid2`` will receive answers to 3 fixed/
↪static queries,\n        and the ``Validation`` sampler will receive answers to␣
```

(continues on next page)

(continued from previous page)

```
↪the query\n        ``[0, 2, 3]``.\n        ",
                "default": {},
                "env_names": "{'details'}",
                "type": "object"
            }
        },
        "additionalProperties": false
    }
}
```

Fields

- *html (salmon.triplets.manager.HTML)*
- *samplers (dict)*
- *sampling (salmon.triplets.manager.Sampling)*
- *targets (Optional[Union[int, List[str]]])*

field html: *HTML* = HTML(title='Similarity judgements', instructions='Please select the comparison item that is most similar to the target item.', debrief='Thanks! Please use the participant ID below.', skip_button=False, css='', max_queries=50, arrow_keys=True, query_params=['puid'], js='', element_top='', element_middle='', element_bottom='', element_standalone='')

Stylistic settings to configure the HTML page. See *HTML* for more detail.

field samplers: dict = {'random': {'class': 'Random'}}

Samplers to use, and their initialization parameters. See above or "*Sampler configuration*" for more detail on customization, and *Active sampling* for experiments/benchmarks.

field sampling: *Sampling* = Sampling(common={'d': 2}, probs=None, samplers_per_user=0, details={})

Settings to configure how more than two samplers are used. See *Sampling* for more detail.

field targets: Optional[Union[int, List[str]]] = None

A list of targets that will be rendered as HTML. If a ZIP file is uploaded, this field is populated automatically. See *YAML file with ZIP file* for more detail.

parse(*user_config*)

This configuration has two optional components:

salmon.triplets.manager.HTML	Stylistic settings to configure the HTML page.
salmon.triplets.manager.Sampling	Settings to configure how more than two samplers are used.

salmon.triplets.manager.HTML

pydantic settings salmon.triplets.manager.**HTML**

Stylistic settings to configure the HTML page.

The default configuration is specified completely below, which is rendered into a YAML file in an example.

Arbitrary keys are allowed in this class (which might allow for customization on the HTML page).


```
{
   "title": "HTML",
   "description": "Stylistic settings to configure the HTML page.\n\nThe default␣
↪configuration is specified completely below, which is rendered into a YAML file␣
↪in `an example`_.\n\n.. _an example: https://github.com/stsievert/salmon/blob/
↪master/examples/default.yaml\n\nArbitrary keys are allowed in this class\n(which␣
↪might allow for customization on the HTML page).",
   "type": "object",
   "properties": {
      "title": {
         "title": "Title",
         "description": "The title of the HTML page (the text shown in the tab bar)
↪",
         "default": "Similarity judgements",
         "env_names": "{'title'}",
         "type": "string"
      },
      "instructions": {
         "title": "Instructions",
         "description": "The instructions the user sees above each query.",
         "default": "Please select the <i>comparison</i> item that is most similar␣
↪to the <i>target</i> item.",
         "env_names": "{'instructions'}",
         "type": "string"
      },
      "debrief": {
         "title": "Debrief",
         "description": "The message that the user sees after ``max_queries``␣
↪responses\n            have been completed. The participant ID (``puid``) is␣
↪shown below\n              this message.",
         "default": "<b>Thanks!</b> Please use the participant ID below.",
         "env_names": "{'debrief'}",
         "type": "string"
      },
      "skip_button": {
         "title": "Skip Button",
         "description": "Wheter to show a button to skip queries. Most\n       ␣
↪useful when participants are instructed to skip queries only when the\n       ␣
↪know nothing about any object.",
         "default": false,
         "env_names": "{'skip_button'}",
         "type": "boolean"
      },
      "css": {
         "title": "Css",
```


(continues on next page)

(continued from previous page)

```
        "description": "CSS to be included the in the query page.  This CSS is\n␣
↪    inserted just before the ``</style>`` tag in the HTML head.",
        "default": "",
        "env_names": "{'css'}",
        "type": "string"
     },
     "max_queries": {
        "title": "Max Queries",
        "description": "The number of queries that the user will answer before\n␣
↪    seeing the ``debrief`` message). Set ``max_queries=0`` or\n       ``max_
↪queries=-1`` to ask unlimited queries.",
        "default": 50,
        "env_names": "{'max_queries'}",
        "type": "integer"
     },
     "arrow_keys": {
        "title": "Arrow Keys",
        "description": "Wheter to allow using the arrow keys as input.  Specifying\
↪n       ``arrow_keys=True`` might allow bad input (though there is a delay of\n␣
↪     200ms between queries).",
        "default": true,
        "env_names": "{'arrow_keys'}",
        "type": "boolean"
     },
     "query_params": {
        "title": "Query Params",
        "description": "A list of flags to include in the params to /query.\n    ␣
↪  The default is ``query_params=['puid']``, which hits the endpoint\n       ``/
↪query?puid=foo`` if the participant's UID is ``foo``. Developers\n       writing␣
↪custom query pages might be interested in customizing this\n       value.\n    ␣
↪  ",
        "default": [
           "puid"
        ],
        "env_names": "{'query_params'}",
        "type": "array",
        "items": {
           "type": "string"
        }
     },
     "js": {
        "title": "Js",
        "description": "JavaScript to include in standalone ``<script>`` tag",
        "default": "",
        "env_names": "{'js'}",
        "type": "string"
     },
     "element_top": {
        "title": "Element Top",
        "description": "Custom HTML to include at top of page",
        "default": "",
        "env_names": "{'element_top'}",
```

(continues on next page)

(continued from previous page)

```
         "type": "string"
      },
      "element_middle": {
         "title": "Element Middle",
         "description": "Custom HTML to include in middle of page (between target␣
↪and comparisons)",
         "default": "",
         "env_names": "{'element_middle'}",
         "type": "string"
      },
      "element_bottom": {
         "title": "Element Bottom",
         "description": "Custom HTML to include at top of page (above footer)",
         "default": "",
         "env_names": "{'element_bottom'}",
         "type": "string"
      },
      "element_standalone": {
         "title": "Element Standalone",
         "description": "Custom HTML to include as a standalone element, only\n    ␣
↪   encased in the ``head`` tag.\n        ",
         "default": "",
         "env_names": "{'element_standalone'}",
         "type": "string"
      }
   }
}
```

Config

- **extra**: *str = allow*

Fields

- *arrow_keys (bool)*
- *css (str)*
- *debrief (str)*
- *element_bottom (str)*
- *element_middle (str)*
- *element_standalone (str)*
- *element_top (str)*
- *instructions (str)*
- *js (str)*
- *max_queries (int)*
- *query_params (List[str])*
- *skip_button (bool)*
- *title (str)*

field arrow_keys: bool = True

Wheter to allow using the arrow keys as input. Specifying `arrow_keys=True` might allow bad input (though there is a delay of 200ms between queries).

field css: str = ''

CSS to be included the in the query page. This CSS is inserted just before the `</style>` tag in the HTML head.

field debrief: str = 'Thanks! Please use the participant ID below.'

The message that the user sees after `max_queries` responses have been completed. The participant ID (`puid`) is shown below this message.

field element_bottom: str = ''

Custom HTML to include at top of page (above footer)

field element_middle: str = ''

Custom HTML to include in middle of page (between target and comparisons)

field element_standalone: str = ''

Custom HTML to include as a standalone element, only encased in the `head` tag.

field element_top: str = ''

Custom HTML to include at top of page

field instructions: str = 'Please select the comparison item that is most similar to the target item.'

The instructions the user sees above each query.

field js: str = ''

JavaScript to include in standalone `<script>` tag

field max_queries: int = 50

The number of queries that the user will answer before seeing the `debrief` message). Set `max_queries=0` or `max_queries=-1` to ask unlimited queries.

field query_params: List[str] = ['puid']

A list of flags to include in the params to /query. The default is `query_params=['puid']`, which hits the endpoint `/query?puid=foo` if the participant's UID is `foo`. Developers writing custom query pages might be interested in customizing this value.

field skip_button: bool = False

Wheter to show a button to skip queries. Most useful when participants are instructed to skip queries only when the know nothing about any object.

field title: str = 'Similarity judgements'

The title of the HTML page (the text shown in the tab bar)

salmon.triplets.manager.Sampling

pydantic settings `salmon.triplets.manager.`**Sampling**

Settings to configure how more than two samplers are used.

These settings are used in the HTML but are not stylistic. The exception is `common`, which is passed to every `sampler` during initialization and will likely be optional arguments for *`Adaptive`*.

The default configuration is specified completely below, which is rendered into a YAML file in an example.

```
{
   "title": "Sampling",
   "description": "Settings to configure how more than two samplers are used.\n\
↪nThese settings are used in the HTML but are not stylistic. The\nexception is␣
↪``common``, which is passed to every ``sampler`` during\ninitialization and will␣
↪likely be optional arguments for\n:class:`~salmon.triplets.samplers.Adaptive`.\n\
↪nThe default configuration is specified completely below, which is rendered into␣
↪a YAML file in `an example`_.\n\n.. _an example: https://github.com/stsievert/
↪salmon/blob/master/examples/default.yaml",
   "type": "object",
   "properties": {
      "common": {
         "title": "Common",
         "description": "Arguments to pass to every sampler for initialization␣
↪(likely\n        values for :class:`~salmon.triplets.samplers.Adaptive`; note␣
↪that\n        values for ``n`` and ``ident`` are already specified). Any\n       ␣
↪values specified in this field will be overwritten by\n        sampler-specific␣
↪arguments.",
         "default": {
            "d": 2
         },
         "env_names": "{'common'}",
         "type": "object"
      },
      "probs": {
         "title": "Probs",
         "description": "The percentage to sample each ``sampler`` when given the\n␣
↪       opportunity (which depends on ``samplers_per_user``). The percentages\n   ␣
↪    in this sampler must add up to 100.\n        If not specified (default),␣
↪choose each sampler with equal\n        probability.",
         "env_names": "{'probs'}",
         "type": "object",
         "additionalProperties": {
            "type": "integer"
         }
      },
      "samplers_per_user": {
         "title": "Samplers Per User",
         "description": "The number of samplers to assign to each user. Setting\n  ␣
↪     ``samplers_per_user=1`` means any user only sees queries generated\n       ␣
↪from one sampler, and ``sampler_per_user=0`` means the user sees a\n        new␣
↪sampler every query",
         "default": 0,
         "env_names": "{'samplers_per_user'}",
         "type": "integer"
      },
      "details": {
         "title": "Details",
         "description": "Different options for a deterministic choice of samplers.\
↪n\n        This dictionary is of the form ``{query_shown_to_user: options}``. The\
↪n        ``options`` is a dictionary with up to two keys:\n\n        -␣
↪``sampler`` (required), which reflects which sampler receives the\n         ␣
↪responses)\n\n        - ``query`` (optional), which is a list of length 3␣
```


(continues on next page)

(continued from previous page)

```
↪indicating the target\n          indices appear in the query.\n\n        For␣
↪example, this YAML will ensure the 1st and 10th query the\n        crowdsourcing␣
↪user sees will be from:\n\n        .. code-block:: yaml\n\n          targets:␣
↪[zero, one, two, three, four, five, six]\n          # ^ list of (textual) targets;
↪ target \"zero\" has index 0 and\n          # is targets[0] in Python\n\n        ␣
↪ samplers:\n            ARR: {}\n            Validation: {}\n            valid2:\
↪n              class: Validation\n              n_queries: 3\n\n        ␣
↪sampling:\n            probs: {ARR: 100, Validation: 0, valid2: 0}\n          ␣
↪details:\n              # Each key \"n\" is the n-th query the user sees\n       ␣
↪      # So here the 1st and 10th queries the user sees is customized\n          ␣
↪  1: {sampler: \"Validation\", query: [0, 2, 3]}\n              10: {sampler: \
↪"valid2\"}\n\n          html:\n            # ask 10 queries according to \
↪"sampling.probs\".\n            # The probabilistic sampling will be overriden by␣
↪sampling.details\n            max_queries: 10\n\n        In this case, the␣
↪crowdsourcing user will see the following:\n\n        * 1st query shown will have␣
↪head \"zero\", and feet \"two\" and \"three\".\n        * Queries 2 and 9 will be␣
↪generated by the :class:`~salmon.triplets.samplers.ARR` sampler.\n        * The␣
↪10th query they see (also the last query): one of three\n          (random) fixed/
↪static queries.\n\n        The sampler ``valid2`` will receive answers to 3 fixed/
↪static queries,\n        and the ``Validation`` sampler will receive answers to␣
↪the query\n        ``[0, 2, 3]``.\n        ",
          "default": {},
          "env_names": "{'details'}",
          "type": "object"
      }
   },
   "additionalProperties": false
}
```

Fields

- *common (Dict[str, Any])*
- *details (Dict[int, Any])*
- *probs (Optional[Dict[str, int]])*
- *samplers_per_user (int)*

field common: Dict[str, Any] = {'d': 2}

Arguments to pass to every sampler for initialization (likely values for *Adaptive*; note that values for `n` and `ident` are already specified). Any values specified in this field will be overwritten by sampler-specific arguments.

field details: Dict[int, Any] = {}

Different options for a deterministic choice of samplers.

This dictionary is of the form `{query_shown_to_user: options}`. The `options` is a dictionary with up to two keys:

- `sampler` (required), which reflects which sampler receives the responses)
- `query` (optional), which is a list of length 3 indicating the target indices appear in the query.

For example, this YAML will ensure the 1st and 10th query the crowdsourcing user sees will be from:

```
targets: [zero, one, two, three, four, five, six]
# ^ list of (textual) targets; target "zero" has index 0 and
# is targets[0] in Python

samplers:
  ARR: {}
  Validation: {}
  valid2:
    class: Validation
    n_queries: 3

sampling:
  probs: {ARR: 100, Validation: 0, valid2: 0}
  details:
    # Each key "n" is the n-th query the user sees
    # So here the 1st and 10th queries the user sees is customized
    1: {sampler: "Validation", query: [0, 2, 3]}
    10: {sampler: "valid2"}

html:
  # ask 10 queries according to "sampling.probs".
  # The probabilistic sampling will be overriden by sampling.details
  max_queries: 10
```

In this case, the crowdsourcing user will see the following:

- 1st query shown will have head "zero", and feet "two" and "three".
- Queries 2 and 9 will be generated by the *ARR* sampler.
- The 10th query they see (also the last query): one of three (random) fixed/static queries.

The sampler `valid2` will receive answers to 3 fixed/static queries, and the `Validation` sampler will receive answers to the query `[0, 2, 3]`.

field probs: Optional[Dict[str, int]] = None

The percentage to sample each `sampler` when given the opportunity (which depends on `samplers_per_user`). The percentages in this sampler must add up to 100. If not specified (default), choose each sampler with equal probability.

field samplers_per_user: int = 0

The number of samplers to assign to each user. Setting `samplers_per_user=1` means any user only sees queries generated from one sampler, and `sampler_per_user=0` means the user sees a new sampler every query

1.10.2 Offline embeddings

This class can be used to create embeddings from a set of downloaded responses from Salmon:

salmon.triplets.offline.OfflineEmbedding([...])	Generate an embedding offline (after responses are downloaded from Salmon).

salmon.triplets.offline.OfflineEmbedding

class salmon.triplets.offline.**OfflineEmbedding**(*n=None*, *d=None*, *max_epochs=400000*, *opt=None*, *verbose=1000*, *ident=''*, *noise_model='CKL'*, *random_state=None*, ***kwargs*)

Generate an embedding offline (after responses are downloaded from Salmon).

Parameters

n
[int] Number of targets.

d
[int] Embedding dimension

max_epochs
[int] Number of epochs or passes through the dataset to run for.

opt
[Optional[Optimizer]]

verbose
[int] Interval at which to score.

random_state
[Optional[int]] Random state for initialization.

noise_model
[str, optional (default: `SOE`)] Noise model to optimize over.

kwargs
[dict, optional, default: `{}`] Arguments for `OGD`. Only used when `opt is None`.

Attributes

*embedding_*
The current embedding.

*history_*
The history that's recorded during `fit`.

*meta_*
Meta-information about this estimator.

Methods

fit(X_train, X_test[, embedding, get_stats])	Fit the embedding with train and validation data.
get_params([deep])	Get parameters for this estimator.
initialize(X_train[, embedding])	Initialize this optimizer.
partial_fit(X_train)	Fit this optimizer for (approximately) one pass through the training data.
score(X)	Score the responses against the current embedding.
set_params(**params)	Set the parameters of this estimator.

property embedding_

The current embedding. If there are `n` objects being embedded into `d` dimensions, then `embedding_.shape == (n, d)`.

fit(*X_train*, *X_test*, *embedding=None*, *get_stats=None*, ***stats_kwargs*)

Fit the embedding with train and validation data.

Parameters

X_train
[array-like] Data to fit the embedding too.

The responses with shape `(n_questions, 3)`. Each question is organized as `[head, winner, loser]`.

X_test
[array-like] Data to score the embedding on

The responses with shape `(n_questions, 3)`. Each question is organized as `[head, winner, loser]`.

embedding
[np.ndarray, optional] The embedding to initialize with.

Note: This is particularly useful when `embedding` is the online embedding from the CSV:

```
import pandas as pd
em = pd.read_csv("embedding.csv")  # from dashboard
df = pd.read_csv("responses.csv")  # from dashboard
X = df[["head", "winner", "loser"]]

from salmon.triplets.offline import OfflineEmbedding
est = OfflineEmbedding(...)
est.initialize(X, embedding=em)
```

---

get_stats
[Callable]

property history_

The history that's recorded during `fit`. Available keys include `score_test` and `loss_test`.

initialize(*X_train*, *embedding=None*)

Initialize this optimizer.

Parameters

X_train
[np.ndarray] Responses organized to be [head, winner, loser].

embedding
[nd.ndarray, optional] If specified, initialize the embedding with the given values.

property meta_

Meta-information about this estimator. Available keys include `score_train` and `loss_train`.

partial_fit(*X_train*)

Fit this optimizer for (approximately) one pass through the training data.

Parameters

X_train
[array-like] The responses with shape `(n_questions, 3)`. Each question is organized as `[head, winner, loser]`.

score(*X*)

Score the responses against the current embedding. Record the loss and accuracy, and return the accuracy.

Parameters

X

[array-like] The responses to score against the current embedding.

The responses with shape `(n_questions, 3)`. Each question is organized as `[head, winner, loser]`.

1.10.3 Samplers

These classes are used to collect responses with Salmon. They can be configured using `init.yaml` as mentioned in *Sampler configuration*. They all conform to the following API:

salmon.backend.sampler.Sampler([ident])	Run a sampling algorithm.

salmon.backend.sampler.Sampler

class salmon.backend.sampler.**Sampler**(*ident: str = ''*)

Run a sampling algorithm. Provides hooks to connect with the database and the Dask cluster.

Parameters

ident

[str] The algorithm idenfifier. This value is used to identify the algorithm in the database.

Attributes

clear

Should the queries be cleared from the database?

Methods

clear_queries(rj)	Clear all queries that this sampler has posted from the database.
get_answers(rj[, clear])	Get all answers the frontend has received.
get_model()	Get the model underlying the algorithm.
post_queries(queries, scores[, rj, done])	Post scored queries to the database.
process_answers(answers)	Process answers.
redis_client([decode_responses])	Get the database (/Redis client)
reset(client, rj[, futures])	Stop the algorithm.
run(client)	Run the algorithm.
save()	Save the sampler's state and current embedding to the database.
serialize_query(q)	Serialize a query (so it can go in the database).

property clear

Should the queries be cleared from the database?

clear_queries(*rj: RedisClient*) → bool

Clear all queries that this sampler has posted from the database.

get_answers(*rj: RedisClient*, *clear: bool = True*) → List[Answer]

Get all answers the frontend has received.

get_model() → Dict[str, Any]

Get the model underlying the algorithm.

Returns

state
[Dict[str, Any]] The state of the algorithm. This can be used for display on the dashboard or with an HTTP get request.

post_queries(*queries: List[Query]*, *scores: List[float]*, *rj: Optional[RedisClient] = None*, *done=None*) → int

Post scored queries to the database.

Parameters

queries
[List[Query]] Queries to post to the database

scores
[List[float]] The scores for each query

rj
[RedisClient, optional] The databaase

Returns

n_queries
[int] The number of queries posted to the database.

process_answers(*answers: List[Answer]*)

Process answers.

Parameters

answers
[List[Answers]] Each answer is a dictionary. Each answer certainly has the keys "head", "left", "right" and "winner", and may have the key "puid" for participant UID.

Returns

data
[dict] An update to self.__dict__.

redis_client(*decode_responses=True*) → RedisClient

Get the database (/Redis client)

reset(*client*, *rj*, *futures=None*)

Stop the algorithm. The algorithm will be deleted shortly after this function is called.

run(*client: DaskClient*)

Run the algorithm.

Parameters

client
[DaskClient] A client to Dask.

rj
[RedisClient] A Redist Client, a rejson.Client

Notes

This function runs the adaptive algorithm. Because it's asynchronous, this function should return if `"reset" in rj.keys() and rj.jsonget("reset")`.

save() → bool

Save the sampler's state and current embedding to the database.

serialize_query(*q: Query*) → str

Serialize a query (so it can go in the database).

This class enables running a triplet embedding algorithm on Salmon: it provides convenient hooks to the database like `get_queries` and `post_answers` if you want to customize the running of the algorithm. By default, the algorithm uses `Sampler.run` to run the algorithm.

Every class below inherits from *Sampler*.

Passive Algorithms

salmon.triplets.samplers.Random(n[, d, ...])	Choose the triplet queries randomly, only ensuring objects are not repeated.
salmon.triplets.samplers.RoundRobin(*args, ...)	Let the head of the triplet query rotate through the available items while choosing the bottom two items randomly.
salmon.triplets.samplers.Validation(n[, d, ...])	Ask about the same queries repeatedly

salmon.triplets.samplers.Random

class salmon.triplets.samplers.**Random**(*n*, *d=2*, *ident=''*, *targets=None*)

Choose the triplet queries randomly, only ensuring objects are not repeated.

Parameters

n
[int] Number of objects

ident
[str] Identifier of the algorithm

__init__(*n*, *d=2*, *ident=''*, *targets=None*)

salmon.triplets.samplers.RoundRobin

class salmon.triplets.samplers.**RoundRobin**(**args*, ***kwargs*)

Let the head of the triplet query rotate through the available items while choosing the bottom two items randomly. This class is user specific if the `/query?puid=foo` endpoint is hit.

__init__(**args*, ***kwargs*)

Parameters

n
[int] Number of objects

ident
[str] Identifier of the algorithm

targets
[Optional[List[int]]] The allowable indexes to ask about.

salmon.triplets.samplers.Validation

class salmon.triplets.samplers.**Validation**(*n*, *d=2*, *n_queries=20*, *queries=None*, *ident=''*)

Ask about the same queries repeatedly

__init__(*n*, *d=2*, *n_queries=20*, *queries=None*, *ident=''*)

This sampler asks the same questions repeatedly, useful to evaluate query difficulty.

Parameters

n
[int] Number of objects

ident
[str] Identifier of the algorithm

n_queries
[int, optional (default=20)] Number of validation queries.

d
[int] Embedding dimension.

queries
[List[Tuple[int, int, int]]] The list of queries to ask about. Each query is `(head, obj1, obj2)` where `obj1` and `obj2` are randomly shown on the left and right. Each item in the tuple is the *index* of the target to ask about. For example:

```
queries=[(0, 1, 2), (3, 4, 5), (6, 7, 8)]
```

will first ask about a query with `head_index=0`, then `head_index=3`, then `head_index=6`.

Active Algorithms

Every active algorithm inherits from one class:

salmon.triplets.samplers.Adaptive(*, n[, d, ...])	The sampler that runs adaptive algorithms.

salmon.triplets.samplers.Adaptive

class salmon.triplets.samplers.**Adaptive**(*, *n: int*, *d: int = 2*, *ident: str = ''*, *module: str = 'TSTE'*, *optimizer: str = 'Embedding'*, *R: float = 10*, *scorer: str = 'infogain'*, *random_state: Optional[int] = None*, ***kwargs*)

The sampler that runs adaptive algorithms.

__init__(*, *n: int*, *d: int = 2*, *ident: str = ''*, *module: str = 'TSTE'*, *optimizer: str = 'Embedding'*, *R: float = 10*, *scorer: str = 'infogain'*, *random_state: Optional[int] = None*, ***kwargs*)

Parameters

n
[int] The number of items to embed.

d
[int (optional, default: `2`)] Embedding dimension.

ident
[str (optional, default: `""`).] The identity of this runner. Must be unique among all adaptive algorithms.

optimizer
[str (optional, default: `Embedding`).] The optimizer underlying the embedding. This method specifies how to change the batch size. Choices are `["Embedding", "PadaDampG", "GeoDamp"]`.

R
[int (optional, default: `1`)] Adaptive sampling after `R * n` responses have been received.

scorer
[str (optional, default: `"infogain"`)] How queries should be scored. Scoring with `scorer='infogain'` tries to link query score and "embedding improvement," and `scorer='uncertainty'` looks at the query that's closest to the decision boundary (or 50% probability).

random_state
[int, None, optional (default: `None`)] The random state to be used for initialization.

kwargs
[dict, optional] Keyword arguments to pass to *Embedding*.

The class *Adaptive* runs the adaptive algorithm and depends on *Embedding* for optimization:

salmon.triplets.samplers.adaptive.Embedding(...)	An optimization algorithm that produces an embedding from human responses of the form `[head, winner, loser]`.

salmon.triplets.samplers.adaptive.Embedding

class salmon.triplets.samplers.adaptive.**Embedding**(*module*, *module__n: int = 85*, *module__d: int = 2*, *optimizer=<class 'torch.optim.adadelta.Adadelta'>*, *warm_start=True*, *max_epochs=100*, *initial_batch_size=512*, ***kwargs*)

An optimization algorithm that produces an embedding from human responses of the form `[head, winner, loser]`.

__init__(*module*, *module__n: int = 85*, *module__d: int = 2*, *optimizer=<class 'torch.optim.adadelta.Adadelta'>*, *warm_start=True*, *max_epochs=100*, *initial_batch_size=512*, ***kwargs*)

Parameters

module
[nn.Module] The noise model to use.

module__n
[int] The number of items to embed.

module__d
[int, optional (default: 2)] The number of dimensions to embed into.

optimizer
[torch.optim.Optimizer] The optimizer to use.

max_epochs
[int, optional (default: 100)] The number of epochs—or passes through the dataset—to perform.

warm_start
[bool, optional (default: True)] Whether to use the existing embedding.

initial_batch_size
[int, optional (default: 512)] The optimizer's (initial) batch size.

kwargs
[dict, optional] Additional keyword arguments to pass the underlying noise model (CKL, TSTE, etc).

To customize the optimization, all extra keyword arguments are passed to the optimizer.

Then, all of these classes inherit from *Adaptive*:

salmon.triplets.samplers.ARR([R, n_top, ...])	An adaptive "round robin" sampler.
salmon.triplets.samplers.TSTE([alpha])	The t-Distributed STE (t-STE) embedding algorithm [1].
salmon.triplets.samplers.SOE(**kwargs)	The soft ordinal embedding detailed by Terada et al. [1].
salmon.triplets.samplers.STE(**kwargs)	The Stochastic Triplet Embedding [1].
salmon.triplets.samplers.CKL([mu])	The crowd kernel embedding.
salmon.triplets.samplers.GNMDS(**kwargs)	The Generalized Non-metric Multidimensional Scaling embedding [1].

salmon.triplets.samplers.ARR

class salmon.triplets.samplers.**ARR**(*R: int = 1*, *n_top: int = 1*, *module: str = 'TSTE'*, *priority: str = 'random'*, ***kwargs*)

An adaptive "round robin" sampler.

Note: This class is usable in it's default configuration. Most of the *active sampling benchmarks* have been run under the default configuration of this class. Please carefully consider any changes to the default parameters.

Notes

This algorithms asks about "high scoring queries" uniformly at random. For each head, the top `n_top` queries are selected. The query shown to the user is a query selected uniformly at random from this set.

If `n_top > 1`, then in practice, this sampling algorithm randomly asks about high scoring queries for each head. Becaues it's asynchronous, it randomly selects a head (instead of doing it a round-robin fashion).

References

[1]

__init__(*R: int = 1*, *n_top: int = 1*, *module: str = 'TSTE'*, *priority: str = 'random'*, ***kwargs*)

Parameters

R
[int (optional, default 1)] Adaptive sampling starts after `R * n` responses have been received.

module
[str, optional (default `"TSTE"`).] The noise model to use.

n_top
[int (optional, default 1)] For each head, the number of top-scoring queries to ask about.

priority
[str, optional (default `"random"`)] Determines how queries should be ordered. Setting `priority="random"` will randomly shuffle queries; setting `priority="original"` will perserve the original scores. Setting `priority="approx"` will add some noise to the original score ranks.

Regardless of the `scores` value, `n_top` queries per head will be perserved. It is likely most relevant when `n_top==1`.

kwargs
[dict] Keyword arguments to pass to *Adaptive*.

salmon.triplets.samplers.TSTE

class salmon.triplets.samplers.**TSTE**(*alpha=1*, ***kwargs*)

The t-Distributed STE (t-STE) embedding algorithm [1].

Notes

This algorithm is proposed for the following reason:

> In STE the value of the corresponding probability rapidly becomes infinitesimal when a triplet constraint is violated. As a result, stronger violations of a constraint do not lead to significantly larger penalties, which reduces the tendency to correct triplet constraints that violate the consensus. This is illustrated by the STE gradient depicted in Figure 1: the STE gradient is nearly zero when a constraint is strongly violated or satisfied. However, it appears that the gradient decays too rapidly, making it hard for STE to fix errors made early in the optimization later on.
>
> To address this problem, we propose to use a heavy-tailed kernel to measure local similarities between data points instead
>
> —Section 4 of [1].

References

[1]

__init__(*alpha=1*, ***kwargs*)

Parameters

alpha
[float, default=1] The parameter that controls how heavily the tails of the probability distribution are.

kwargs
[dict] Keyword arguments to pass to *Adaptive*.

salmon.triplets.samplers.SOE

class salmon.triplets.samplers.**SOE**(***kwargs*)

The soft ordinal embedding detailed by Terada et al. [1]

This is evaluated as "SOE" by Vankadara et al., [2] in which they use the hinge loss on the distances (not squared distances).

References

[1], [2]

__init__(***kwargs*)

Parameters

kwargs
[dict] Keyword arguments to pass to *Adaptive*.

salmon.triplets.samplers.STE

class salmon.triplets.samplers.**STE**(***kwargs*)

The Stochastic Triplet Embedding [1].

References

[1]

__init__(***kwargs*)

Parameters

kwargs
[dict] Keyword arguments to pass to *Adaptive*.

salmon.triplets.samplers.CKL

class salmon.triplets.samplers.**CKL**(*mu=1*, ***kwargs*)

The crowd kernel embedding. Proposed in [1].

References

[1]

__init__(*mu=1*, ***kwargs*)

Parameters

mu
[float] The mu or μ used in the CKL embedding. This is typically small; the default is 10^{-4}.

kwargs
[dict] Keyword arguments to pass to *Adaptive*.

salmon.triplets.samplers.GNMDS

class salmon.triplets.samplers.**GNMDS**(***kwargs*)

The Generalized Non-metric Multidimensional Scaling embedding [1].

References

[1]

__init__(***kwargs*)

Parameters

kwargs
[dict] Keyword arguments to pass to *Adaptive*.

We have tested out the top three algorithms—ARR, TSTE and SOE—in our experiments. We use *ARR* for our adaptive sampling (which defaults to the noise model in *TSTE*) and use *SOE* for the offline embeddings.

These adaptive algorithms are all the same except for the underlying noise model, with the exception of *ARR*. *ARR* introduces some randomness by fixing the head and adding the top `1 * n` triplets to the database. This is useful because the information gain measure used by all of these algorithms (by default) is a rule-of-thumb.

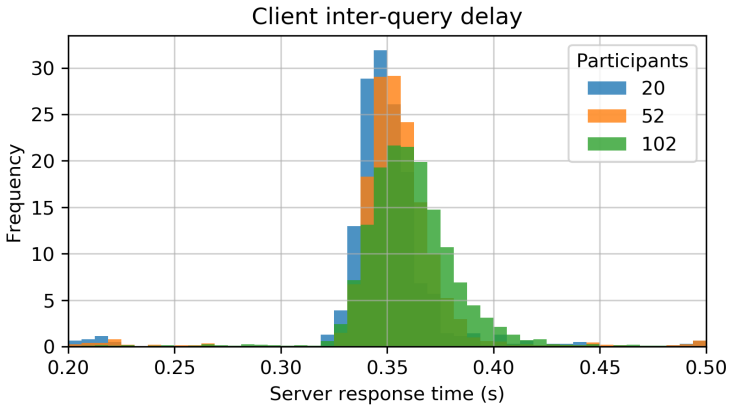
Note: Use of *ARR* is recommended as it performs well in *the experiments we have run*.

1.11 Concurrent Users

Salmon v0.2.3 can handle up to at least 100 simultaneous users interacting with the system and answering questions. Here's the setup:

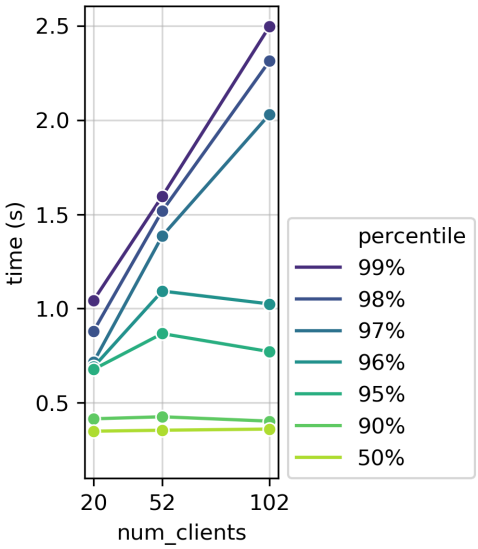
- Launched Salmon on a Amazon EC2 `t3.large` instance.
- Simulated client response time using the most similar task I could find[2]. Response times were a modified Gaussian random variable with mean 750ms and standard deviation 250ms (modified so always greater than 200ms).
- Each client for 50 responses.

Here are the performance results with that setup:



[2] Specifically, a spatial configuration task with 3 elements. I pulled number from Figure 6 of Palmer et. al [palmer].

This performance is pretty good! Most of the responses return in around 400ms. How do the extremes perform?



45 of a users queries will return in less than 500ms if the user answers 50 questions and there are 102 simultaneous clients.

1.12 Active sampling

Experimentalists often ask **"how many responses should I collect with Mechanical Turk?"** It'd be nice to obtain high quality embeddings with few human responses, both to reduce expenses and to ask informative questions for humans.

Active sampling decides which questions to ask about, instead of asking random questions. This can mean that higher accuracies are reached sooner, or that less human responses are required to reach a particular accuracy. This might enable you to ask about more items.

Active sampling algorithms are difficult in this "triplet embedding" context, especially when in the crowdsourcing setting. **However, Salmon enables good performance of active sampling algorithms in crowdsourcing settings.** Specifically, there is evidence that Salmon provides the following features:

1. Salmon's active sampling works well for any (practical) number of targets n.
2. Random sampling requires about **2–3× more human responses** than Salmon's active sampling.
3. Even if **responses are received very quickly,** Salmon's active sampling (almost always) performs no worse than random sampling.

Simulations have been run for each of these points. To show these points, let's first walk through the experimental setup before detailing how Salmon's active sampling performs when compared with previous work.[2] Then, let's investigate how changing the number of targets n and the response rate affect the embeddings.

Warning: All results on this page are for a specific dataset with simulated human responses. Other papers provide evidence of active gains;[1] however, they are more moderate than the results here.

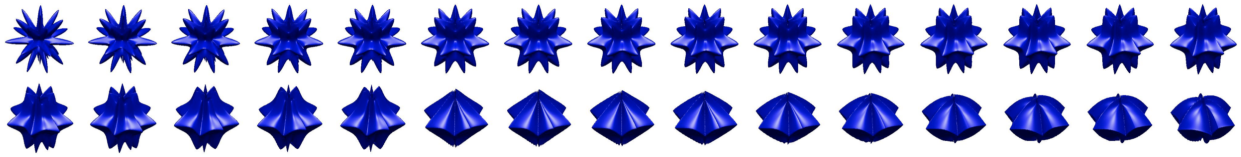
[2] "NEXT: A System for Real-World Development, Evaluation, and Application of Active Learning" by K. Jamieson, L. Jain, C. Fernandez, N. Glattard and R. Nowak. 2017. http://papers.nips.cc/paper/5868-next-a-system-for-real-world-development-evaluation-and-application-of-active-learning.pdf

[1] "Active Perceptual Similarity Modeling with Auxiliary Information" by E. Heim, M. Berger, and L. Seversky, and M. Hauskrecht. 2015. https://arxiv.org/pdf/1511.02254.pdf

Note: Generally, the number of responses required to reach a certain embedding quality scales like $nd\log_2(n)$.[3][4] For more detail, see the FAQ "*How many responses will be needed?*."

1.12.1 Setup

To illustrate the difference between random and active sampling, let's run some experiments with the "alien eggs" dataset. For $n = 30$ objects, that dataset looks like the following:



This dataset is characterized by one parameter, the "smoothness" of each egg, so they have a 1D embedding. However, let's embed into $d = 2$ dimensions to simulate a mistake and to mirror prior work.[Page 56, 2] This page will be concerned with the data scientist workflow, and every experiment below will use the same workflow a data scientists would:

1. Launch Salmon.
2. Simulate human users.[8]
3. Download the human responses from Salmon
4. Generate the embedding offline.

Every graph shows points with this data flow. Each point shown only changes the number of responses available or the sampling method used.[5] Unless explicitly mentioned, let's compare random and active sampling with these `init.yaml` configurations:

```
d: 2
samplers:
  ARR: {random_state: 42}  # active or adaptive sampling
  Random: {}  # random sampling
```

The "ARR" stands for "asynchronous round robin" and creates an instance of *ARR*. For this class, the query head is randomly chosen, and then for each head, the best comparison items are ranked by some measure (information gain by default).

Note: This page shows results of experiments run with Salmon. For complete details, see https://github.com/stsievert/salmon-experiments

[3] "Finite Sample Prediction and Recovery Bounds for Ordinal Embedding." Jain, Jamieson, & Nowak, (2016). https://papers.nips.cc/paper/2016/file/4e0d67e54ad6626e957d15b08ae128a6-Paper.pdf

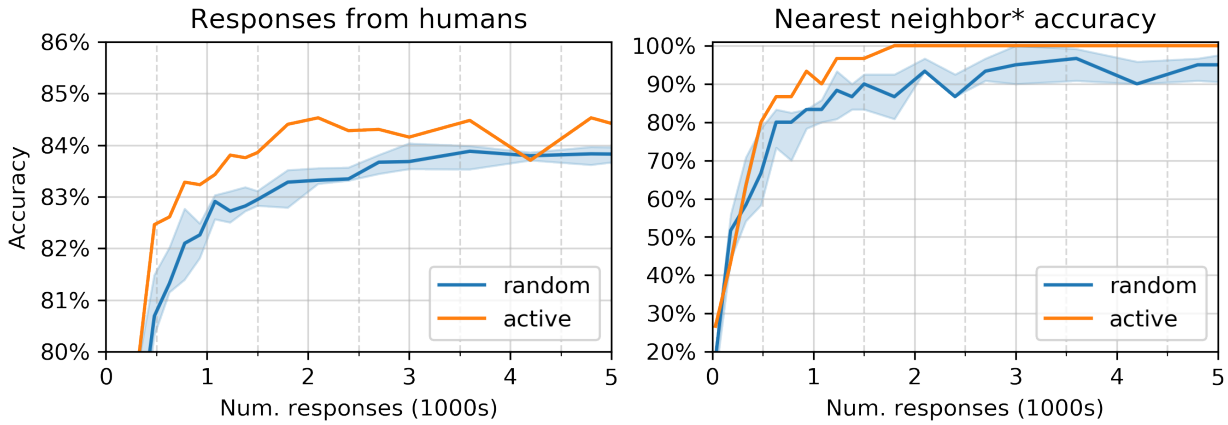
[4] "Low-dimensional embedding using adaptively selected ordinal data." Jamieson, Nowak (2011). https://homes.cs.washington.edu/~jamieson/resources/activeMDS.pdf

[8] Specifically, with a noise model developed the human responses collected for Fig. 3 of the NEXT paper.[Page 56, 2]

[5] For random sampling, the order is also shuffled (not the case for active).

1.12.2 Baseline

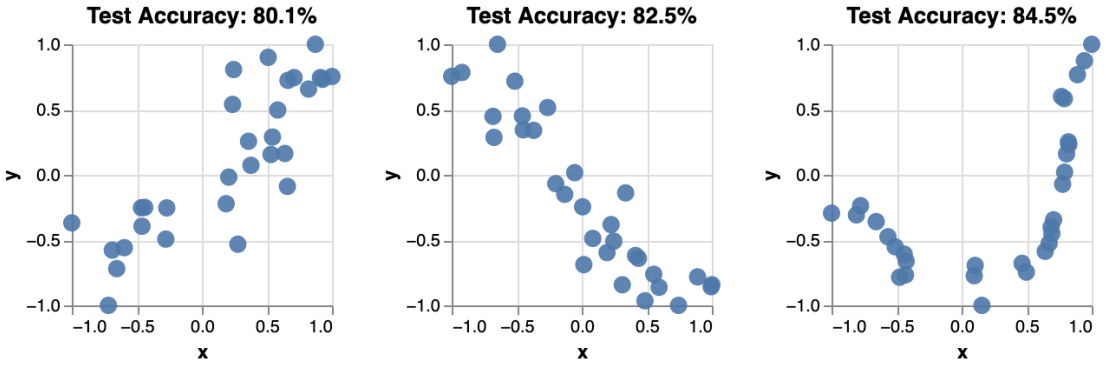
First, let's run a basic experiment, one that will very closely mirror prior work:[Page 56, 2] let's take the $n = 30$ objects above and embed them into $d = 2$ dimensions. To mirror their setup, let's develop a noise model from their collected responses and submit responses at the same time as their responses. Let's do this many times, and generate a graph of how many responses are required to reach a particular accuracy:



This graph uses the same test set as the NEXT paper, which (mis)defines "nearest neighbor accuracy" as "is the true nearest neighbor one of the three closest objects?"[Page 56, 2] (the reason for the * in the title).[7]

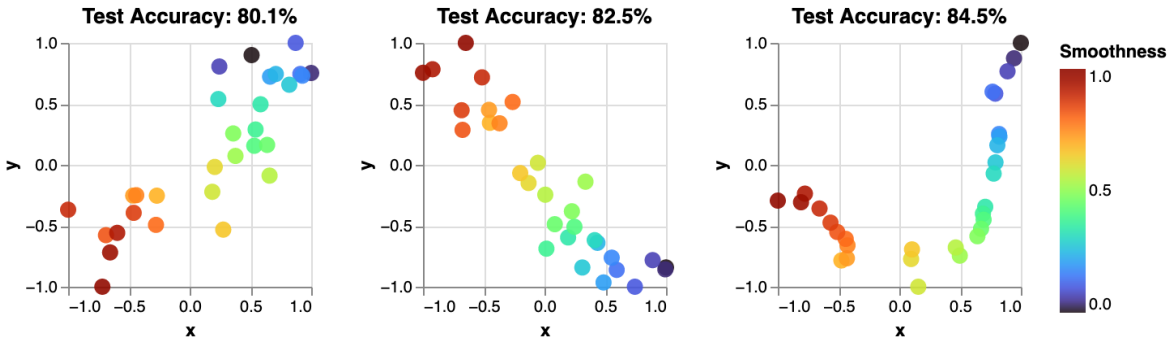
Embedding quality

Experimentalist often cares about the underlying structure more than the accuracy. To start, let's assume that there's no clear relationship between items. Then, this visualization is most appropriate for the embeddings of particular accuracies:



These embeddings are remarkably simple, and have a clear and known relationship. Because of that, let's show the embeddings with colors from now on:

[7] Astute observers might notice that the accuracy doesn't perform well when directly compared with NEXT's results. However, the developed noise model doesn't exactly mirror the human responses; it's about 1.5% less accurate (shown below). However, the nearest neighbor accuracy is a measure of the underlying embedding, and Salmon perform better than the NEXT results.



Note: Only relative distances matter in these embeddings. It doesn't matter how the embedding is rotated, or how the axes are scaled.

1.12.3 Number of targets

Users of Salmon frequently have a variable number of target items. For example, they might be asking about colors – a continuous space, so they can easily change the "number of targets." So, **how does the number of targets influence embedding quality?**

To examine that, let's run the same experiment above, but with 30, 90, 180 and 300 "alien eggs." Here's the number of responses required to reach a particular accuracy to *simulated* human responses:

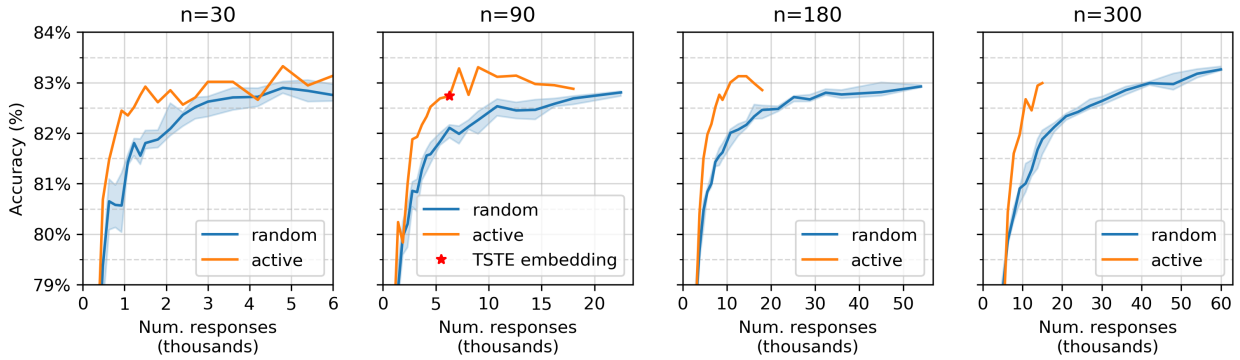
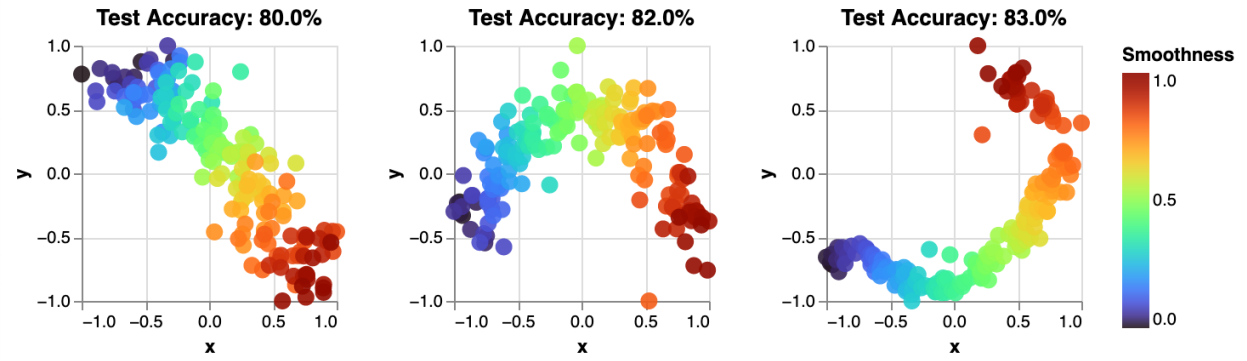


Fig. 1: The accuracy of simulated human responses for various number of responses. The shaded region represents the 25–75% percentile among 10 runs, and the solid line represents the median. The y-axis labels are shared with all plots.

Embedding quality

Here's the underlying embeddings for $n = 180$ for various accuracy levels on *simulated* human responses:



"Test accuracy: 80%" means "80% accurate on simulated human responses not used for training." The local accuracy gets much better as accuracy increases. To visualize the structure of the underlying embedding, let's look at the **average items closer than the true nearest neighbor**. The smaller this value is, the smoother the color gradient is above.

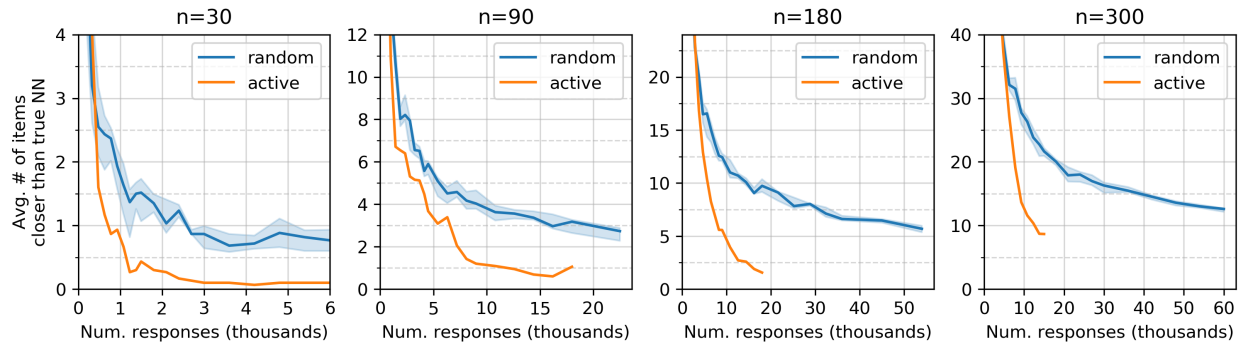
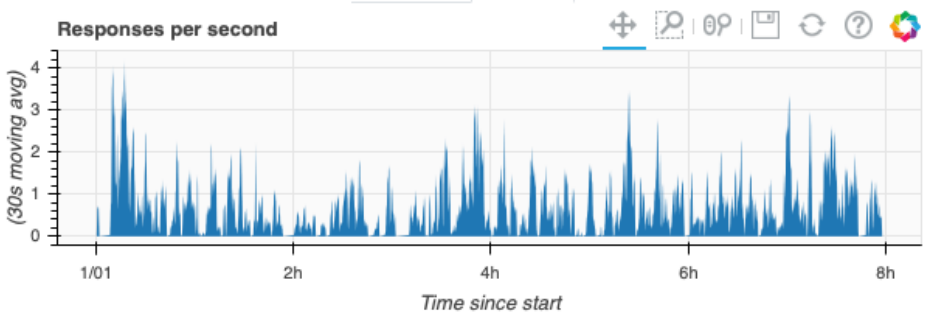


Fig. 2: The average number of items closer than the true nearest neighbor. The upper limit on the y-axis represents a very moderately accurate embedding, slightly worse than the 80% accurate embedding above. The shaded region/solid line has the same meaning as above, the interquartile range and median.

If the embedding were a 1D manifold but not quite perfect,[9] the value on this plot would be 0.5. As with accuracy, there's a clear advantage to active sampling – active sampling requires a lot fewer responses to obtain a high quality embedding in this simulation.

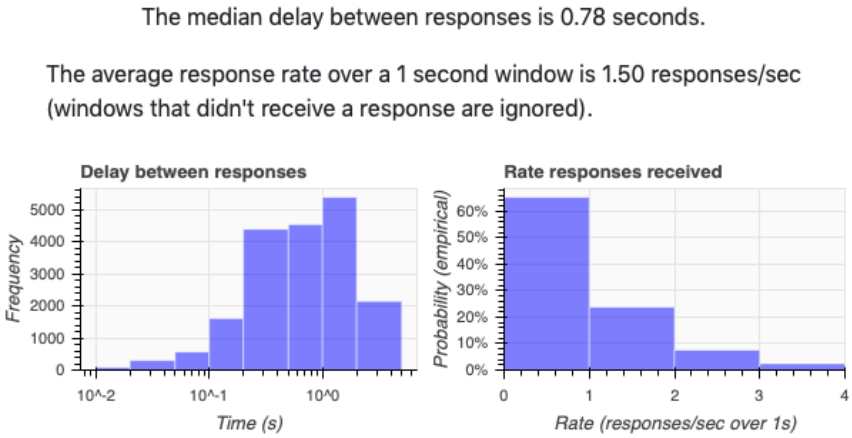
1.12.4 Response rate

One detail has been swept under the rug: the rate at which Salmon received responses. There would be no gain from adaptive algorithms if all 10,000 responses were received in 1 second. In fact, the response rate above is variable:

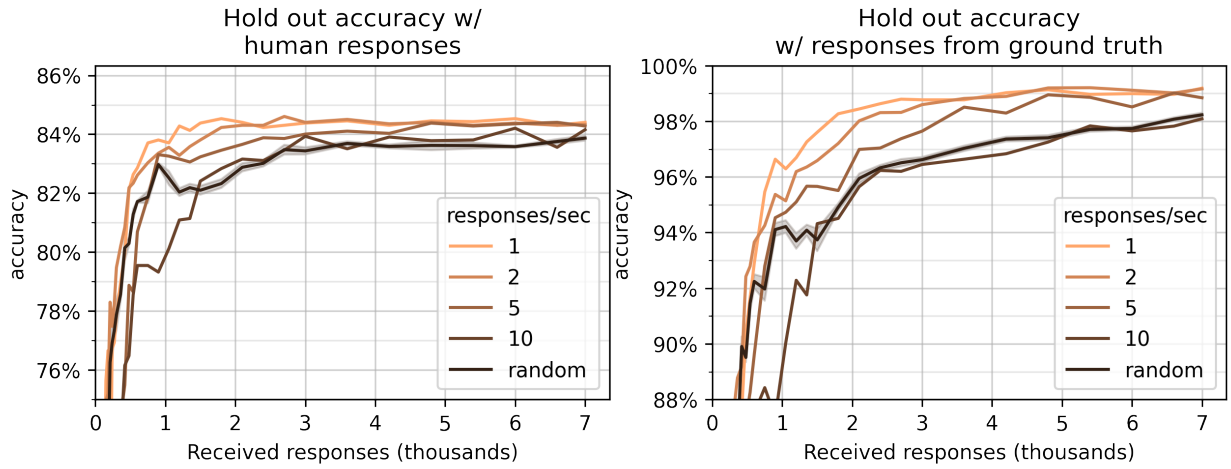


[9] "Not quite perfect" means "1D manifold with a constant distance to the nearest neighbor: an embedding with coordinates `[[1, 0], [2, 0], [3, 0], ..., [n - 1, 0]]`.

Here's a summary of the server side timings:



How does this variable response rate affect adaptive gains? Let's run the same data flow as above, but with a constant response rate and (functionally) Salmon v0.6.0. In this experiment, the number of users varies between 1 concurrent user to 10 concurrent users with a mean response time of 1 second. Here's the performance we see for $n = 30$ alien eggs (the same setup as in *Baseline*).



This graph shows two measures: accuracy on a set of test human responses (left) and responses that are 100% accurate on the ground truth dataset (right). The graph on the right is a measure of quality on the underlying embedding. The graph on the left shows that that this quality is reflected in hold-out performance on human responses.

These experiments provide evidence that the adaptive sampling above works well in crowdsourcing settings. Additionally, they provide evidence that Salmon's adaptive sampling does not perform worse than random sampling.

This measure provides evidence that Salmon's active sampling approach outperforms random sampling. If true, this is an improvement over existing software to deploy triplet queries to crowdsourced audiences: in NEXT's introduction paper,[Page 56, 2] the authors found "no evidence for gains from adaptive sampling" for (nearly) the same problem.[6]

[6] Both experiment use $n = 30$ objects and embed into $d = 2$ dimensions. The human noise model used in the Salmon experiments is generated from the responses collected during NEXT's experiment. The are the same experiment, up to different responses (NEXT actually runs crowdsourcing experiments; Salmon's noise model is generated from those responses).

References

1.13 Adaptive algorithm primer

The API the must conform to below:

salmon.backend.sampler.Sampler([ident])	Run a sampling algorithm.

This API balances the fundamentally serial nature of adaptive algorithms with the parallel context of web servers.

Typically, an adaptive algorithm looks like the following:

```
model = Model(...)
while True:
    q = model.best_query()
    ans = get_human_answer(q)
    model.fit(ans)
```

However, web servers are different. They typically look something like the following:

```
@app.get("/query")
async def query():
    return db.pop("queries")

@app.post("/answer")
async def process_answer(answer):
    db.push(answer)
```

The *Sampler* API balances the two and runs the code to *receive answers* and *process answers* in separate processes. *Processing the received answers* is an optimization that needs to be performed quickly because `model.best_query` depends on the optimization.

Essentially, the following code is run in addition to the web server code above:

```
db = Database(...)

def run_alg():
    model = Model(...)
    while True:
        queries = [model.score(random_query()) for _ in range(10_000)]
        db.push("queries", queries)
        answers = db.pop_all("answers")
        model.partial_fit(answers)
```

This means the web server can scale efficiently because the backend optimization doesn't block the frontend query serving. However, that also means the adaptive algorithm needs to post queries quickly so users aren't waiting. Of course, the computation required to perform the embedding needs to happen quickly too (otherwise adaptive algorithms are meaningless).

Luckily, Salmon should scale sufficiently well for typical use cases (e.g, Mechnical Turk with about $n \approx 100$ targets). The query search is fast enough to search 10,000 queries in 50ms with $n = 85$ targets.

1.14 Developing algorithms

1.14.1 Install

This process is meant for developers. To launch, first download the code. It's possible to download a ZIP file of Salmon's source, or if Git is installed, to run this command:

```
$ git clone https://github.com/stsievert/salmon.git
```

Then, to launch a local version of Salmon you'll need Docker Compose. After that dependency is intalled, run the following code:

```
$ cd salmon
$ docker-compose build
$ docker-compose up
$ # visit http://localhost:8421/init or http://localhost:8421/docs
```

If you make changes to this code, run these commands:

```
$ docker-compose stop
$ docker-compose build
$ docker-compose up
```

If you want to log into the Docker container, execute these commands:

```
$ docker ps  # to get list of running conatiners
CONTAINER ID   IMAGE             ... [more info]  ...  NAMES
08b96fbcc4c3   salmon_server     ... [more info]  ...  salmon_server_1
57cb3b7652d9   redislabs/rejson  ... [more info]  ...  salmon_redis_1
$ docker exec -it 08b96fbcc4c3 /bin/bash
(base) root@08b96fbcc4c3:/salmon# conda activate salmon
(salmon) root@08b96fbcc4c3:/salmon#
```

Note: This is an alternative way to install Salmon's dependencies. If you create a file in the Docker container in `/salmon`, it will also be written to `/path/to/salmon` on your local machine.

If you run the command `export SALMON_DEBUG=1`, the Salmon server will watch for changes in the source and relaunch as necessary. This won't be perfect, but it will reduce the number of times required to run `docker-compose {stop, build, up}`.

If you run the command `export SALMON_NO_AUTH=1`, the Salmon server will not require a username/password.

1.14.2 Basics

First, write an algorithm on your machine. The basic interface requires two functions, one to get queries and one to process answers. Briefly, Salmon expects two functions:

1. `process_answers`, a function to process answers (which might involve updating the model).
2. A function to get queries. There are two choices for this:
    - `get_query`, which returns a single query/score

- `get_queries`, which returns a list of queries and scores. These are saved in the database, and popped when a user requests a query.

Use of `get_queries` is strongly recommended. Then Salmon's backend relies on Dask, which allows for higher throughput (more concurrent users). `get_query` uses a single worker process, so it may get overloaded with a moderate number of concurrent users.

For complete documentation, see *API*. In short, your algorithm should be a class that implement `get_query` and `process_answers`.

After you have developed these functions, look at other algorithms in `salmon/triplets/samplers` (e.g, `_adaptive_runners.py` or `_round_robin.py`) to figure out inheritance details. In short, the following details are important:

- **Inheriting from** `Sampler`, which enables Salmon to work with custom algorithms.
- **Accepting an** `ident: str` keyword argument in `__init__` **and passing that argument to** `super().__init__`. (`ident` is passed to all algorithms and is the unique identifier in the database).

I recommend the following when developing your algorithm. These aren't necessary but are highly encouraged:

- **Have you algorithm be serializable:** `pickle.loads(pickle.dumps(alg))` should work for your algorithm. Otherwise, your algorithm can't be restored on a new machine.
- **Ensure query searches are fast enough.** The user will be waiting if thousands of users come to Salmon and deplete all the searched queries.

1.14.3 Debugging

Let's say you've integrated most of your algorithm into *Sampler*. Now, you'd like to make sure everything is working properly.

This script will help:

```
from salmon.triplets.samplers import STE
from copy import copy
import random

def random_answer(q):
    ans = copy(q)
    winner = random.choice(["left", "right"])
    ans["winner"] = q[winner]
    return ans

params = {
    "optimizer__lr": 0.1,
    "optimizer__momentum": 0.75,
}
alg = STE(n=10, **params)  # or your custom alg
for k in range(1000):
    query, score = alg.get_query()
    if query is None:
        queries, scores = alg.get_queries()
        h, a, b = queries[scores.argmax()]
        query = {"head": h, "left": a, "right": b, "score": scores.max()}
```

(continues on next page)

(continued from previous page)

```
    answer = random_answer(query)
    alg.process_answers([answer])
```

1.15 Dependencies

Salmon is licensed under the BSD License. Details are at LICENSE.txt.

Salmon depends on the libraries below. Below this list, the current license of each library at the time of Salmon v1.0 is included:

- Python (Python's license).
- NumPy (NumPy's license).
- SciPy (SciPy's license).
- Pandas (Pandas's license).
- Cython (Cython's license)
- Scikit-learn (Scikit-learn's license)
- FastAPI (FastAPI's license)
- PyTorch (PyTorch's license)
- Skorch (Skorch's license)
- Altair (Altair's license)
- Bokeh (Bokeh's license)
- Jinja2 (Jinja2's license)
- Redis, ReJSON (Redis's license)
- HTTPX (HTTPX's license)
- Dask (Dask's license).
- Dask-Distributed (Dask-Distributed's license).
- Dask-ML (Dask-ML's license).
- aiofiles (aiofiles's license)
- click (click's license)
- lz4 (lz4's license)
- pythn-blosc (python-block's license)
- cytoolz (cytoolz's license)
- ujson (ujson's license)
- Sphinx (Sphinx's license)
- Sphinx RTD Theme (Sphinx RTD Theme's license)
- PyYAML (PyYAML's license)
- matplotlib (matplotlib's license)
- python-multipart (python-multipart's license)

- numpydoc (numpydoc's license)
- pytest (pytest's license)
- gunicorn (gunicorn's license)
- cloudpickle (cloudpickle's license)
- jupyter-server-proxy (jupyter-server-proxy's license)
- autodoc pydantic (autodoc's license)
- starlette_exporter (starlette_exporter's license)

1.15.1 Python

```
1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and
the Individual or Organization ("Licensee") accessing and otherwise using Python
3.11.0 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby
   grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce,
   analyze, test, perform and/or display publicly, prepare derivative works,
   distribute, and otherwise use Python 3.11.0 alone or in any derivative
   version, provided, however, that PSF's License Agreement and PSF's notice of
   copyright, i.e., "Copyright © 2001-2022 Python Software Foundation; All Rights
   Reserved" are retained in Python 3.11.0 alone or in any derivative version
   prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or
   incorporates Python 3.11.0 or any part thereof, and wants to make the
   derivative work available to others as provided herein, then Licensee hereby
   agrees to include in any such work a brief summary of the changes made to Python
   3.11.0.

4. PSF is making Python 3.11.0 available to Licensee on an "AS IS" basis.
   PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED.  BY WAY OF
   EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR
   WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE
   USE OF PYTHON 3.11.0 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 3.11.0
   FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF
   MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 3.11.0, OR ANY DERIVATIVE
   THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of
   its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship
   of agency, partnership, or joint venture between PSF and Licensee.  This License
   Agreement does not grant permission to use PSF trademarks or trade name in a
   trademark sense to endorse or promote products or services of Licensee, or any
   third party.
```

(continues on next page)

(continued from previous page)

```
8. By copying, installing or otherwise using Python 3.11.0, Licensee agrees
to be bound by the terms and conditions of this License Agreement.
```

1.15.2 NumPy

```
Copyright (c) 2005-2022, NumPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above
       copyright notice, this list of conditions and the following
       disclaimer in the documentation and/or other materials provided
       with the distribution.

    * Neither the name of the NumPy Developers nor the names of any
       contributors may be used to endorse or promote products derived
       from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.3 SciPy

```
Copyright © 2001, 2002 Enthought, Inc.
All rights reserved.

Copyright © 2003-2019 SciPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    Redistributions of source code must retain the above copyright notice,
    this list of conditions and the following disclaimer.
```

(continues on next page)

(continued from previous page)

```
    Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

    Neither the name of Enthought nor the names of the SciPy Developers may
    be used to endorse or promote products derived from this software
    without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS”
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

1.15.4 Pandas

```
BSD 3-Clause License

Copyright (c) 2008-2011, AQR Capital Management, LLC, Lambda Foundry, Inc. and PyData␣
↪Development Team
All rights reserved.

Copyright (c) 2011-2022, Open source contributors.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
```

(continues on next page)

(continued from previous page)

```
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.5 Cython

```
                                 Apache License
                           Version 2.0, January 2004
                        https://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
```

(continues on next page)

(continued from previous page)

```
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
```

(continues on next page)

(continued from previous page)

```
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
```

(continues on next page)

(continued from previous page)

```
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS
```

1.15.6 Scikit-learn

```
BSD 3-Clause License

Copyright (c) 2007-2022 The scikit-learn developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
```

(continues on next page)

(continued from previous page)

```
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.7 FastAPI

```
The MIT License (MIT)

Copyright (c) 2018 Sebastián Ramírez

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.
```

1.15.8 PyTorch

```
From PyTorch:

Copyright (c) 2016-     Facebook, Inc            (Adam Paszke)
Copyright (c) 2014-     Facebook, Inc            (Soumith Chintala)
Copyright (c) 2011-2014 Idiap Research Institute (Ronan Collobert)
Copyright (c) 2012-2014 Deepmind Technologies    (Koray Kavukcuoglu)
Copyright (c) 2011-2012 NEC Laboratories America (Koray Kavukcuoglu)
Copyright (c) 2011-2013 NYU                      (Clement Farabet)
Copyright (c) 2006-2010 NEC Laboratories America (Ronan Collobert, Leon Bottou, Iain␣
↪Melvin, Jason Weston)
```

(continues on next page)

(continued from previous page)

```
Copyright (c) 2006      Idiap Research Institute (Samy Bengio)
Copyright (c) 2001-2004 Idiap Research Institute (Ronan Collobert, Samy Bengio, Johnny␣
↪Mariethoz)

From Caffe2:

Copyright (c) 2016-present, Facebook Inc. All rights reserved.

All contributions by Facebook:
Copyright (c) 2016 Facebook Inc.

All contributions by Google:
Copyright (c) 2015 Google Inc.
All rights reserved.

All contributions by Yangqing Jia:
Copyright (c) 2015 Yangqing Jia
All rights reserved.

All contributions by Kakao Brain:
Copyright 2019-2020 Kakao Brain

All contributions by Cruise LLC:
Copyright (c) 2022 Cruise LLC.
All rights reserved.

All contributions from Caffe:
Copyright(c) 2013, 2014, 2015, the respective contributors
All rights reserved.

All other contributions:
Copyright(c) 2015, 2016 the respective contributors
All rights reserved.

Caffe2 uses a copyright model similar to Caffe: each contributor holds
copyright over their contributions to Caffe2. The project versioning records
all such contribution and copyright details. If a contributor wants to further
mark their specific copyright on a particular contribution, they should
indicate their copyright solely in the commit message of the change when it is
committed.

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
```

(continues on next page)

(continued from previous page)

```
3. Neither the names of Facebook, Deepmind Technologies, NYU, NEC Laboratories America
   and IDIAP Research Institute nor the names of its contributors may be
   used to endorse or promote products derived from this software without
   specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

1.15.9 Skorch

```
BSD 3-Clause License

Copyright (c) 2017, Benjamin Bossan, Daniel Nouri, Marian Tietz
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.10 Altair

```
Copyright (c) 2015-2022, Altair Developers
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of altair nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.11 Bokeh

```
Copyright (c) 2012 - 2022, Anaconda, Inc., and Bokeh Contributors
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

Neither the name of Anaconda nor the names of any contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
```

(continues on next page)

(continued from previous page)

```
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.12 Jinja2

```
Copyright 2007 Pallets

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    Redistributions of source code must retain the above copyright notice,
    this list of conditions and the following disclaimer.

    Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

    Neither the name of the copyright holder nor the names of its
    contributors may be used to endorse or promote products derived from
    this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS”
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

1.15.13 Redis, ReJSON

```
Every file in the Redis distribution, with the exceptions of third party files
specified in the list below, contain the following license:

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    Redistributions of source code must retain the above copyright notice, this
```

(continues on next page)

(continued from previous page)

```
    list of conditions and the following disclaimer.

    Redistributions in binary form must reproduce the above copyright notice,
    this list of conditions and the following disclaimer in the documentation
    and/or other materials provided with the distribution.

    Neither the name of Redis nor the names of its contributors may be used to
    endorse or promote products derived from this software without specific
    prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.14 HTTPX

```
Copyright © 2019, [Encode OSS Ltd](https://www.encode.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its contributors
  may be used to endorse or promote products derived from this software without
  specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.15 Dask

```
BSD 3-Clause License

Copyright (c) 2014, Anaconda, Inc. and contributors
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.16 Dask Distributed

```
BSD 3-Clause License

Copyright (c) 2015, Anaconda, Inc. and contributors
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
```

(continues on next page)

(continued from previous page)

```
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.17 Dask-ML

```
Copyright (c) 2017, Anaconda, Inc. and contributors
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

Neither the name of Anaconda nor the names of any contributors may be used to
endorse or promote products derived from this software without specific prior
written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.18 aiofiles

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
```

(continues on next page)

(continued from previous page)

```
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
```

(continues on next page)

(continued from previous page)

```
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
```

(continues on next page)

(continued from previous page)

```
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "{}"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

1.15.19 click

```
Copyright 2014 Pallets

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1.  Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

2.  Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

3.  Neither the name of the copyright holder nor the names of its
    contributors may be used to endorse or promote products derived from
    this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.20 lz4

```
LZ4 Library
Copyright (c) 2011-2020, Yann Collet
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this
  list of conditions and the following disclaimer in the documentation and/or
  other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
```

(continues on next page)

(continued from previous page)

```
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.21 pythn-blosc

```
BSD License

For python-blosc - A Python wrapper for the Blosc compression library

Copyright (C) 2010-2012 Francesc Alted (faltet@gmail.com)
Copyright (C) 2013-2019 Francesc Alted <francesc@blosc.org>, Valentin Haenel
↪<valentin@haenel.co>
Copyright (C) 2019-present The Blosc Development Team <blosc@blosc.org>

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

 * Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.

 * Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

 * Neither the name Francesc Alted nor the names of its contributors may be used
   to endorse or promote products derived from this software without specific
   prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.22 cytoolz

```
Copyright (c) 2014-2021 Erik Welch

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

  a. Redistributions of source code must retain the above copyright notice,
     this list of conditions and the following disclaimer.
  b. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in the
     documentation and/or other materials provided with the distribution.
  c. Neither the name of cytoolz nor the names of its contributors
     may be used to endorse or promote products derived from this software
     without specific prior written permission.


THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH
DAMAGE.
```

1.15.23 ujson

```
Developed by ESN, an Electronic Arts Inc. studio.
Copyright (c) 2014, Electronic Arts Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
* Neither the name of ESN, Electronic Arts Inc. nor the
names of its contributors may be used to endorse or promote products
derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
```

(continues on next page)

(continued from previous page)

```
DISCLAIMED. IN NO EVENT SHALL ELECTRONIC ARTS INC. BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


Portions of code from MODP_ASCII - Ascii transformations (upper/lower, etc)
https://github.com/client9/stringencoders
Copyright (c) 2007  Nick Galbreath -- nickg [at] modp [dot] com. All rights reserved.

Numeric decoder derived from from TCL library
https://opensource.apple.com/source/tcl/tcl-14/tcl/license.terms
 * Copyright (c) 1988-1993 The Regents of the University of California.
 * Copyright (c) 1994 Sun Microsystems, Inc.
```

1.15.24 Sphinx

```
Unless otherwise indicated, all code in the Sphinx project is licenced under
the two clause BSD licence below.

Copyright (c) 2007-2022 by the Sphinx team (see AUTHORS file).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

* Redistributions of source code must retain the above copyright
  notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright
  notice, this list of conditions and the following disclaimer in the
  documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.25 Sphinx RTD Theme


```
The MIT License (MIT)

Copyright (c) 2013-2018 Dave Snider, Read the Docs, Inc. & contributors

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.
```


1.15.26 PyYAML


```
Copyright (c) 2017-2021 Ingy döt Net
Copyright (c) 2006-2016 Kirill Simonov

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.
```

1.15.27 matplotlib

```
1. This LICENSE AGREEMENT is between the Matplotlib Development Team
("MDT"), and the Individual or Organization ("Licensee") accessing and
otherwise using matplotlib software in source or binary form and its
associated documentation.

2. Subject to the terms and conditions of this License Agreement, MDT
hereby grants Licensee a nonexclusive, royalty-free, world-wide license
to reproduce, analyze, test, perform and/or display publicly, prepare
derivative works, distribute, and otherwise use matplotlib
alone or in any derivative version, provided, however, that MDT's
License Agreement and MDT's notice of copyright, i.e., "Copyright (c)
2012- Matplotlib Development Team; All Rights Reserved" are retained in
matplotlib  alone or in any derivative version prepared by
Licensee.

3. In the event Licensee prepares a derivative work that is based on or
incorporates matplotlib or any part thereof, and wants to
make the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to matplotlib .

4. MDT is making matplotlib available to Licensee on an "AS
IS" basis.  MDT MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, MDT MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF MATPLOTLIB
WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. MDT SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF MATPLOTLIB
 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR
LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING
MATPLOTLIB , OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF
THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between MDT and
Licensee.  This License Agreement does not grant permission to use MDT
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using matplotlib ,
Licensee agrees to be bound by the terms and conditions of this License
Agreement.
```

1.15.28 python-multipart


```
Copyright 2012, Andrew Dunham

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

   https://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```


1.15.29 numpydoc


```
Copyright (C) 2008-2022 Stefan van der Walt <stefan@mentat.za.net>, Pauli Virtanen
↪<pav@iki.fi>

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

 1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.
 2. Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in
    the documentation and/or other materials provided with the
    distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

1.15.30 pytest

```
The MIT License (MIT)

Copyright (c) 2004 Holger Krekel and others

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.
```

1.15.31 gunicorn

```
2009-2018 (c) Benoît Chesneau <benoitc@e-engura.org>
2009-2015 (c) Paul J. Davis <paul.joseph.davis@gmail.com>

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.
```

1.15.32 cloudpickle

```
This module was extracted from the `cloud` package, developed by
PiCloud, Inc.

Copyright (c) 2015, Cloudpickle contributors.
Copyright (c) 2012, Regents of the University of California.
Copyright (c) 2009 PiCloud, Inc. http://www.picloud.com.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of the University of California, Berkeley nor the
      names of its contributors may be used to endorse or promote
      products derived from this software without specific prior written
      permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.33 jupyter server proxy

```
BSD 3-Clause License

Copyright (c) 2017, Data Science 8
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
```

(continues on next page)

(continued from previous page)

```
* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

1.15.34 autodoc pydantic

```
MIT License

Copyright (c) 2021 Franz Wöllert

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.
```

1.15.35 starlette_exporter

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.
```

(continues on next page)

(continued from previous page)

```
"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain
separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
```

(continues on next page)

(continued from previous page)

```
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
```

(continues on next page)

(continued from previous page)

```
      as part of the Derivative Works; within the Source form or
      documentation, if provided along with the Derivative Works; or,
      within a display generated by the Derivative Works, if and
      wherever such third-party notices normally appear. The contents
      of the NOTICE file are for informational purposes only and
      do not modify the License. You may add Your own attribution
      notices within Derivative Works that You distribute, alongside
      or as an addendum to the NOTICE text from the Work, provided
      that such additional attribution notices cannot be construed
      as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.
```

(continues on next page)

(continued from previous page)

```
9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

1.16 Other sources

This documentation is available at these locations:

- https://docs.stsievert.com/salmon/.
- On GitHub as a raw PDF (alternate link).
- On GitHub as a zipped HTML directory, which requires unzipping the directory then opening up `index.html`.

Please file an issue if you can not access the documentation above.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] Heim, Eric, et al. “Active perceptual similarity modeling with auxiliary information.” arXiv preprint arXiv:1511.02254 (2015).

[1] “Stochastic Triplet Embedding”. 2012. http://www.cs.cornell.edu/~kilian/papers/stochastictriplet.pdf van der Maaten, Weinberger.

[1] Terada, Y. & Luxburg, U.. (2014). Local Ordinal Embedding. Proceedings of the 31st International Conference on Machine Learning, in PMLR 32(2):847-855. http://proceedings.mlr.press/v32/terada14.html

[2] Vankadara, L. C., Haghiri, S., Lohaus, M., Wahab, F. U., & von Luxburg, U. (2019). Insights into Ordinal Embedding Algorithms: A Systematic Evaluation. https://arxiv.org/abs/1912.01666

[1] “Stochastic Triplet Embedding”. 2012. http://www.cs.cornell.edu/~kilian/papers/stochastictriplet.pdf van der Maaten, Weinberger.

[1] Tamuz, O., Liu, C., Belongie, S., Shamir, O., & Kalai, A. T. (2011). Adaptively learning the crowd kernel. https://arxiv.org/abs/1105.1033

[1] “Generalized Non-metric Multidimensional Scaling”. 2007. Agarwal, Wills, Cayton, Lanckriet, Kriegman, and Belongie. http://proceedings.mlr.press/v2/agarwal07a/agarwal07a.pdf

[palmer] “What are the shapes of response time distributions in visual search?” Palmer, Horowitz, Torralba, & Wolfe (2011). Journal of experimental psychology. https://doi.org/10.1037/a0020747

INDEX

Symbols

A

C

D

E

F

G

H

I

J

M

O

P

Q

R

S

T

V